ABSTRACT

Title of Dissertation: LEARNING-BASED
PHYSICS SIMULATION
WITH COLLISION HANDLING

Qingyang Tan
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Dinesh Manocha
Department of Computer Science

Numerous physics-based simulation approaches have been proposed to generate realistic and vivid deformations for 3D models. These systems include the mass-spring system, the finite element approach, the thin-shell model, and others. However, previous systems based on analytic and numerical methods tend to be computationally intensive. Achieving an ideal balance between simulation accuracy and efficiency still poses several challenges.

In this dissertation, we present novel learning-based physics simulations and collision-handling algorithms that leverage the benefits of neural networks and optimization techniques. We use neural networks to compress high-dimensional 3D deformable models and accelerate the processing time. We also employ algorithms such as reinforcement learning, active learning, and imitation learning to capture complex physical behaviors that lack closed-form analytic models.

We propose multiple novel approaches for novel learning-based physics simulation. First, we train a learning-based collision detector for 3D deformable models and utilize the detec-

tor as a surrogate constraint in an optimization-based collision handler. Our focus is on collisions between topologically disjoint triangles in triangular meshes. Traditional geometric-based search methods for collision detection are computationally expensive, with costs ranging from $O(n \log n)$ to $O(n^2)$. In comparison, our neural collision detector is $80\times$ faster. To perform stable collision prediction performance in large and unseen spaces, we employ active learning by progressively incorporating new collision data based on network inferences, reaching a collision detection accuracy of up to 98.1%. Second, we present an approach to accelerate collision response computations by incorporating an additional repulsive force unit in the learning-based pipeline. Our experiments demonstrate that backbone networks trained with the repulsive force unit can significantly decrease the number of collisions, boosting collision-free models from 49% to 77%, while maintaining real-time performance, adding only 2 milliseconds to the inference system. Third, we present a neural volumetric deformable object simulator with collision detection and handling based on an actor-critic neural architecture. Our critic network learns to estimate collision penetrations, while our actor network learns to minimize the penalty function through a series of gradient descent steps, resulting in nearly collision-free quasistatic deformable object poses. Finally, we introduce a novel framework for randomly reposing 3D humans to arbitrary poses based on a geometric optimization regularization that incorporates control information into diffusion-based inpainting. Our geometric inpainting algorithm reduces errors by 93% when moving different body parts to random locations.

In practice, our learning-based physics simulation systems can generate realistic 3D models that satisfy various constraints. We have tested our method using several large-scale datasets, including AMASS for humans and TailorNet for garments. Our approach can generate plausible results, and we observe $100 - 300\times$ speedups over numerical or analytic methods.

LEARNING-BASED PHYSICS SIMULATION WITH COLLISION HANDLING

by

Qingyang Tan

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
2023

Advisory Committee:
Professor Dinesh Manocha, Chair
Professor Ming C. Lin
Professor Maria Cameron
Professor Matthias Zwicker
Professor David Mount

Acknowledgments

Woohoo! I have reached a significant milestone in my life by earning my doctoral degree!

The journey toward a Ph.D. is far from easy, especially amidst the global pandemic. I am relieved and proud to have successfully navigated this lengthy academic path at the University of Maryland (UMD) and within the GAMMA research group. Thanks to Professor Dinesh Manocha, whose guidance led me into the fascinating world of physical simulation. I also wish to thank my long-standing collaborator and GAMMA alumni, Dr. Zherong Pan, whose inspiration greatly enriched my understanding of our research area.

I had the privilege of interning at two highly esteemed industry graphics research labs – Meta Reality Labs and Adobe Research. My deepest gratitude goes to my mentors Dr. Takaaki Shiratori, Dr. Yi Zhou, Dr. Noam Aigerman, Dr. Breannan Smith, Dr. Tuanfeng Y. Wang, Dr. Duygu Ceylan, and Dr. Xin Sun. The experiences gained through these internships expanded my understanding of computer graphics beyond academia, instilling in me the aspiration to become a research scientist and contribute to the construction of innovative and captivating virtual worlds using algorithms.

Also, thanks to my undergraduate advisor, Prof. Lin Gao, and my friends at the Institute of Computing Technology, Chinese Academy of Sciences: Dr. Jie Yang, Dr. Shuyu Chen, Dr. Mingze Yuan, Yujie Yuan, and Lingxiao Zhang. They introduced me to the field of computer graphics and inspired me to pursue a Ph.D. degree.

Family has always been my sanctuary during this challenging academic journey. I would like to express my deepest gratitude to my mother, Hongyu Cao, whose emotional and financial support have been indispensable over the past five years. I love you, Mom!

Last but not least, my heartfelt thanks to my friends, old and new, from primary school to graduate school. Thank you for patiently listening to my moments of confusion and frustration, and for not turning away from me during those times. I look forward to reciprocating your support in the future!

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

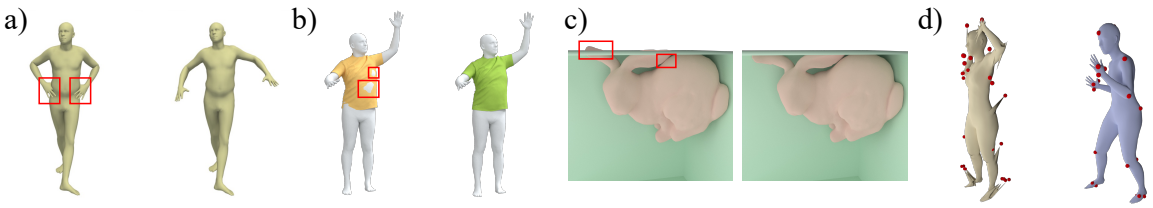


Figure 1.1: A preview of our method's results showcasing: a) Resolving self-collision on human poses, b) Resolving collisions between garments and underlying human bodies in cloth simulation, c) Resolving self and obstacle collisions for volumetric simulation, and d) Reposing humans using geometric diffusion inpaint.

1.1 Motivation and Area Overview

Physics-based simulation and collision handling can generate realistic and vivid deformations for 3D models such as cloth [124] and human bodies [82]. They are fundamental components of different applications such as animation [91], virtual try-on systems [52], immersive audio-visual interactions [195], and multi-robot navigation [17]. Numerous systems have been proposed, including mass-spring methods [12], the finite-element method [170], particle-based systems [31], and the thin-shell model [54]. Physics-based simulation systems need to account for constitutive models of materials and many hard constraints, including collision-free 3D models.

However, previous systems based on analytic and numerical methods tend to be computationally intensive, and achieving an ideal balance between the accuracy and efficacy of simulation still faces several challenges. First, the computational cost grows superlinearly with the dimension of the physical system's configuration space. For example, a high-resolution cloth used in animation or games typically involves tens of thousands of degrees of freedom. As another example, simulating cloth or body tissues with vivid details using the finite element method involves solving large global systems of equations. The complexity can vary from $\mathcal{O}(n^{1.5})$ to $\mathcal{O}(n^3)$ [51]. Second, adding constraints, including being collision-free, brings additional costs. Collision artifacts are easy to notice, and even a single missed collision can considerably affect the accuracy of the overall simulator. Our work focuses on detecting collisions between topologically disjoint triangles on triangular meshes. Such collision detection is a computationally intensive task, often demanding a complexity within the range of $O(n \log n)$ to $O(n^2)$ when using geometric-based search methods. The most accurate physically-based simulators run at 0.5 seconds per frame on commodity GPUs [189], where collision handling can take $50 - 80\%$ of the total simulation time. Finally, it is generally difficult to acquire accurate and complete information or material properties from the real world. For example, human bodies or body parts are reconstructed from incomplete and inaccurate motion capture data [171]. Constructing quality physical models from incomplete data is still an open problem.

Recent advancements in machine learning have presented promising solutions to address many of these challenges. Deep learning models, for instance, have demonstrated the ability to capture details by leveraging learned filters and effectively compressing high-dimensional data into low-dimensional latent vectors using autoencoders [179]. These latent vectors are computationally more efficient to handle, offering a significant advantage. Moreover, modern GPU

architectures and deep learning frameworks such as PyTorch [137] have made neural network models easily scalable to multi objects or denser discretization at relatively low computational costs. This combination of advantages allows for real-time performance, even with tasks that were previously computationally intensive and performed offline. Furthermore, machine learning algorithms like reinforcement learning [112], active learning [133], and imitation learning [74] have demonstrated the ability to capture complex behaviors that lack closed-form, analytic models. These algorithms enable neural network models to learn and replicate intricate behaviors from incomplete observations, opening up new possibilities for accurate simulations. By leveraging the power of machine learning, we can overcome computational limitations, capture complex behaviors, and work with incomplete data, thus paving the way for more efficient and realistic physics-based simulations and collision-handling systems.

1.2 Main Results

In this dissertation, we present innovative learning-based physics simulations and collision-handling algorithms that utilize the advantages of neural networks and optimization techniques. Even though purely learning-based methods have shown substantial potential, various previous works [42, 68, 96, 160, 229] have illustrated how neural networks can employ domain knowledge to significantly boost performance in simulation and collision avoidance. Inspired by these developments, we present novel learning-based physics simulations and collision-handling algorithms to generate collision-free and rational simulation results for human bodies, garments, and general 3D deformable models. Our results include the following:

1. **Learned Constraints for Fast Collision Handling:** We train a learning-based collision

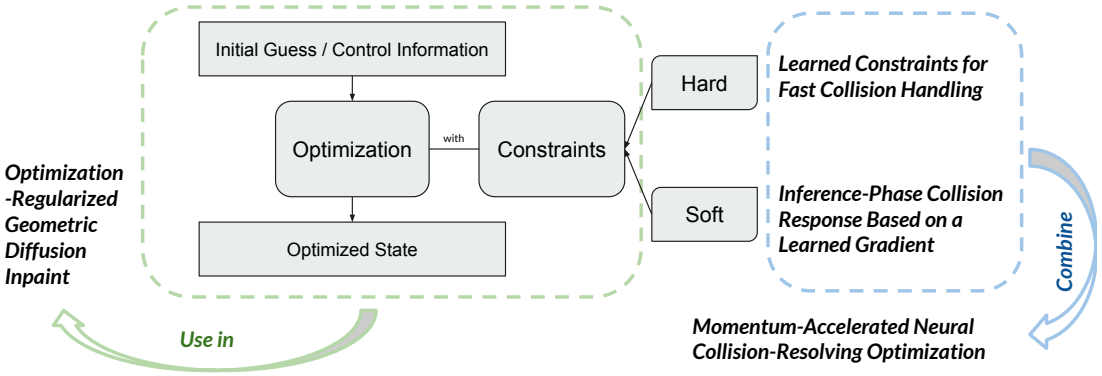


Figure 1.2: In this figure, we illustrate the connections between our works in physics simulation, optimization, and neural networks. These works investigate the integration of neural networks as representations of both hard and soft physical constraints within constrained optimization. Additionally, they introduce a learnable multi-step momentum-accelerated optimizer and combine optimization techniques with a diffusion-based generation framework to enhance control over the inpainting process for 3D deformable materials.

detector for 3D human models and utilize the detector as a surrogate constraint in an optimization-based collision handler. These techniques yield an accuracy of 98.1% when predicting collisions for randomized human poses sampled from widely-used datasets. After learning the feasible domain, solving a constrained optimization for a collision-free human pose with thousands of vertices takes only a fraction of a second.

2. **Inference-Phase Collision Response Based on a Learned Gradient:** We propose a method to speed up collision response computations by introducing an additional repulsive force unit in the learning-based pipeline. Our approach approximates the distance field for inter-object collisions and also predicts the gradient and a suitable step length for reducing penetration. Experiments demonstrate that backbone networks trained with the

repulsive force unit significantly reduce the number of collisions while maintaining real-time performance.

3. **Momentum-Accelerated Neural Collision-Resolving Optimization:** We introduce an end-to-end neural volumetric deformable object simulator with (self-)collision detection and handling. Our approach, designed for real-time applications, relies on an actor-critic neural architecture. The critic network learns to estimate inter-object and self-penetrations for accurate simulation, while the actor network minimizes the penalty function via a series of differentiable momentum-accelerated gradient descent steps, resulting in almost (self-) collision-free quasistatic deformable object poses. Our learned collision handler is $100 - 300\times$ faster than numerical simulators that deal with high-dimensional geometric data.

4. **Optimization-Regularized Geometric Diffusion Inpaint:** We present a novel framework for reposing 3D humans to arbitrary poses based on a geometric optimization regularization that integrates control information into diffusion-based inpainting. Utilizing Laplacian regularization of the 3D mesh, we enable the diffusion generation network to accommodate 3D human deformation with extremely sparse control points on bodies. We showcase results on human models from the AMASS dataset, achieving a reduction of errors by 93% when moving different body parts to random locations.

There is a clear connection between the presented works and the field of optimization, as shown in Fig. 1.2. In physical simulation, optimization techniques are commonly employed to satisfy physical laws and constraints. The first work explores the integration of neural networks as representations of hard physical constraints within constrained optimization, expanding the applicability of neural networks in optimization tasks. The second work investigates the network's

capacity to serve as a soft constraint, leveraging learned gradients and heuristic step lengths to mitigate violations of soft constraints. Building upon the insights from the previous works, the third study introduces a learnable multi-step momentum-accelerated optimizer, effectively reducing collision penetrations recognized by neural networks. Lastly, in the fourth work, optimization techniques are combined with the state-of-the-art diffusion-based generation framework to enhance control over the inpainting process for 3D deformable materials.

In summary, our learning-based physics simulation systems can generate realistic 3D models that satisfy various constraints. We have tested our method using several large-scale datasets, including AMASS for humans [114] and TailorNet [138] for garments. We highlight our method's results in Fig. 1.1. Our approach can generate plausible results and we observe $100 - 300\times$ speedups over numerical or analytic methods.

1.3 Organization

In this dissertation, Chapter 2 is dedicated to reviewing relevant prior works, encompassing physics simulation, collision detection and response, and hard constraints in deep learning.

Chapters 3 to 7 present our work on developing a learning-based physics simulation with collision handling. Chapter 3 and Chapter 4 focus on the learned constraints used in fast collision handling, which includes an active learning system to enhance overall performance. Chapter 5 will spotlight our real-time repulsive force unit that resolves collisions between garments and the underlying human bodies, utilizing a neural-network-based signed distance function. Chapter 6 presents a learned momentum-accelerated optimizer, predicated on the insights from the previous two chapters, to eradicate more collisions occurring in volumetric neural simulation.

In Chapter 7, we delve into the state-of-the-art generation network – diffusion's application on 3D deformable human models – to demonstrate our modified geometric optimization regularizerand enable inpainting that satisfies extremely sparse control points.

Finally, Chapter 8 will summarize the results, discuss the current limitations of our approach, and provide an outline for future research directions.

Chapter 2: Related Works

In this chapter, we discuss previous works on physics simulation, collision detection and response, and hard constraints in deep learning.

2.1 Deformable Object Simulation

Deformation object simulation is a key component in various model-based control algorithms such as virtual surgery [8, 9, 98] and soft robot controller design [39, 75, 134]. However, physics simulators based on the finite element method [90], the boundary-element method [26], or simplified models such as the mass-spring system [30] have a superlinear complexity over the degrees of freedoms (DOF). For most of them, the bottleneck lies in solving the linear systems. An analysis is given in [51], with matrix inversion resulting in $\mathcal{O}(n^{1.5})$ complexity, where n is the number of DOFs. In a high-resolution simulation, n can be tens of thousands, making it impossible to perform fast predictions. As a result, learning-based methods have recently been used to accelerate physics simulations. This can be done by simulating under a low-resolution setting using FEM and then upsampling [212] or by learning the dynamics behaviors of clothes [129] and fluids [208]. In the following research, we want to follow this direction and add domain knowledge to improve the neural simulation accuracy and efficiency.

2.2 Learning-based Mesh Deformations

Most learning-based deformation models are essentially low-dimensional embedding techniques. Early works adopt linear subspaces with bases extracted using principle component analysis (PCA) [6, 125, 232], which can only represent small local deformations, or Gaussian processes [202, 231], which are computationally costly to train and do not scale to large datasets. Recently, deep neural networks have demonstrated far better performances for embedding high-dimensional nonlinear functions [86, 149]. However, these methods rely on regular data structures such as 2D images. To handle meshes with arbitrary topologies, earlier methods [117] represented a mesh as a 3D voxelized grid or reconstructed 3D shapes from 2D images [215] using a projection layer. Recently, methods have been proposed to define networks directly on mesh surfaces such as multi-layer perceptrons on diffusion [164], CNN on parametrized texture space [115], and CNN based on spatial filtering [40]. The latter has been used in the author's prior work [180] to embed large-scale deformations of general meshes. The contribution in this proposal is orthogonal to these techniques and can be used to improve the embedding accuracy for any one of these methods.

2.3 Collision Detection & Response

An important criterion of "correct" 3D models is that they are (self-) collision-free, i.e., elements of the mesh do not penetrate each other. Collision detection and response have been well-studied, with many practical algorithms proposed for large-scale 3D meshes [83, 132]. Collisions can be handled in a discrete or continuous manner. Discrete collision handling [83] assumes that

meshes can occasionally reach an invalid status with penetrations and therefore checks for collisions at fixed time intervals. In contrast, continuous collision detection algorithms estimate the time instance corresponding to the first contact and thereby maintain non-penetration configurations. These continuous collision detection (CCD) methods [27, 28, 144, 185, 187, 188, 200] make some assumptions about the interpolating motion between two time instances and use analytic methods to predict the time of the collision. Many of these methods can be accelerated using GPU parallelism [53]. There is extensive literature on collision response computation based on constraint solvers [131], impulse responses [27], and impact zone methods [62, 144]. These methods have been used to develop robust physics-based simulators that are widely used in animation and VR applications. Several techniques have been proposed to handle collisions in machine learning methods. Many recent works [19–21, 59] use collision loss to penalize penetrated 3D model pairs during training. However, these methods have no component or feature in the network that can resolve these penetrations during inference. To solve the collision problems for the testing set, a simple approach is to perform post-processing on the predicted 3D models by detecting the vertices inside the other models and moving them directly to the nearest point on the surface [68, 159]. However, there are two issues with these approaches: first, computing nearest points on the surface is time-consuming; second, simply moving the penetrated vertices may generate abrupt movements and lose some properties of the original 3D models. In summary, current learning-based methods are significantly faster than these physics-based ones but cannot offer the same level of accuracy or robustness.

2.4 Training Deep Networks with Hard Constraints

State-of-the-art deep learning methods are still less capable of handling general hard constraints. Constraints on neural network parameters [151] are used for regularizing the network training and can be approximately enforced using variants of the projected gradient descent algorithm. On the other hand, constraints on neural network output are used to model application-specific requirements such as collision-free constraints. Prior works [3, 116, 122, 141] use a similar approach to enforce hard constraints: converting the constrained optimization into an unconstrained min-max optimization, which can be solved approximately by updating the primal and dual variables. A special case arises when the hard constraints are convex, where the constrained optimization can be solved efficiently with exact constraint enforcement [3, 141]. However, the collision-free constraints in our applications are neither convex nor smooth.

Chapter 3: LCollision: Fast Generation of Collision-Free Human Poses using Learned Non-Penetration Constraints

3.1 Introduction

There has been considerable work on developing learning algorithms for 3D objects represented as point clouds [145], meshes [61], volumetric grids [204], and physical objects [95]. Because these algorithms are used for different applications, a major challenge is accounting for user requirements and physics-based constraints. Considering these constraints can significantly improve the test-time robustness by preserving some known criteria of "correct" predictions. For example, we need to consider various forces and dynamics constraints for differentiable simulation [148] and cloth embedding [181], and a reliable robot motion planner should preserve a clearance distance from obstacles [141].

In this chapter, we tackle the problem of collision-free human pose generation. Recently, 3D mesh representations have been used for learning-based human pose synthesis [11, 25, 150, 179, 193]. These methods learn a manifold of plausible human poses from a dataset, represented as the latent space of a deep autoencoder. Such autoencoders can be trained for applications including interactive rigging, human pose recognition from images and videos, and VR games. However, current learning-based methods do not account for any physics-based requirements

such as (self-) collision-free constraint, thereby resulting in penetrations or other artifacts [11, 25, 150, 179, 193]. By comparison, non-learning-based methods for character rigging [166] and physics-based simulation [14] can detect and explicitly handle the collisions using numerical methods. Our goal is to equip learned-based methods with similar collision-handling capabilities.

Although 3D data representations explicitly allow the modeling of collision-free constraints, satisfying these hard constraints in an end-to-end learning system is an open problem. Prior works have tried one of three ways to incorporate hard constraints in a learning system. First, classical second-order methods [23] for constrained optimization can enforce exact hard constraints on the parameters of the neural network. Second, variants of the stochastic projected gradient descent [80, 116] have been proposed to approximately satisfy the constraints on the neural network parameters. Finally, differentiable optimization layers [3, 141] can modify the neural network output to satisfy such constraints. However, these methods are either limited to convex constraints, impractical for large networks, or do not provide sufficient accuracy in terms constraint satisfaction.

Main Results: We present LCollision, a new learning algorithm to generate human poses that satisfy collision-free constraints. Our approach incorporates the non-penetration constraints by solving a general constrained optimization during the test time, where the feasible domain corresponding to these hard constraints is learned during the training time. The novel components of our approach include:

- **Constrained Optimization Using Neural Network Function Approximation:** Instead of using exact collision-response, learning the feasible domain using a neural network provides approximate sub-gradients via back-propagation, which is much faster than exact

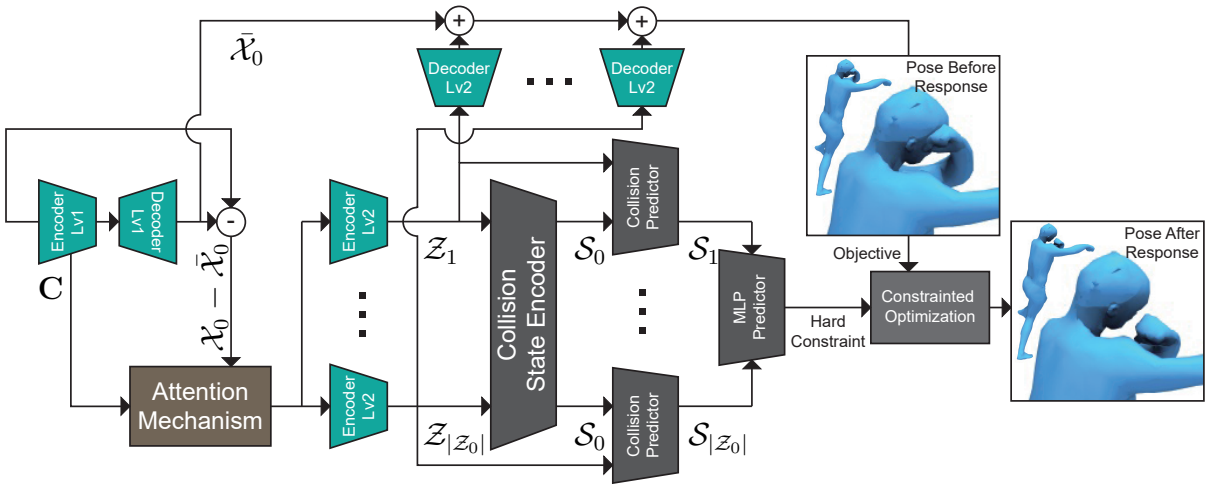


Figure 3.1: Our network architecture combines the domain-decomposed human pose embedding framework (green) and a novel collision state estimator (gray). Given an input pose, we use a weight-shared, level-1 autoencoder to learn a global shape embedding. The error on each domain is further reduced using a set of level-2 autoencoders. Both the level-1 and level-2 autoencoders' latent codes are used to predict a global collision state. Finally, the latent code of each level-2 autoencoder is compared against the global collision state to infer a localized penetration depth. These inferred penetrations are used in hard constraints of a constrained optimization framework for collision handling.

collision-checking algorithms.

- **Collision Decomposition:** A collision only affects local regions of the human body, and we design our collision predictor to respect these local effects. Each point on the human body is softly assigned to a set of local body parts, and the collision loss is decomposed to these local domains, accordingly.

- **Hybrid Ranking, Potential Energy, and Entropy Loss:** Although exact hard constraints correspond to a binary loss (violation or non-violation), this loss should be differentiable so that constrained optimizations can be guided by gradient information. We propose a penetration-depth-based formulation [223] as a collision metric to offer gradient direction, combined with the ranking loss to maintain the relative penetration depth between a pair of samples.

We have evaluated our method on the SCAPE dataset [10], the MIT-Swing dataset [199], and the MIT Jumping dataset [199]. Combining these techniques, we achieve an accuracy of 94.1%, a false positive rate of 6.1%, and a false negative rate of 5.7% when predicting collisions for 2.5×10^6 randomized human poses sampled from these datasets. After learning the feasible domain, solving a constrained optimization for a collision-free human pose with 2161 vertices takes 2.095 iterations and 0.25s on average. Moreover, our learned collision detector is $80\times$ faster than prior exact collision detection methods running on a CPU [132].

3.2 Related Work

Human Pose Estimation & Synthesis: There is considerable work on human pose estimation and synthesis. Earlier methods [92] represent a pedestrian as a bounding box. An improved algorithm was proposed in [1], and this algorithm predicts the 55-D joint angles for a skeletal human pose. More accurate prediction results have been proposed in [153] using random forests and in [192] using convolutional neural networks. Our approach is based on recent learning methods [179, 193] that use 3D meshes to generate detailed human poses. Mesh-based representations are inherently difficult to learn due to the intrinsic high-dimensionality, and the algorithm can produce sub-optimal results with various artifacts such as self-penetrations, noisy mesh surfaces, and flipped meshes. In view of these problems, [198] only computes skeletal poses using learning and then uses skinning to recover the mesh-based representation. However, this approach requires additional skeleton-mesh correspondence information, which is typically unavailable in many datasets, including SCAPE [10].

3.3 Human Pose & Collision-Free Constraints

Recent methods [11, 25, 150, 179, 193] have used neural networks to generate new poses from a small set of examples via shape embedding. In this section, we give an overview of the process of computing the embedding space for human pose generation and highlight the collision-free constraints that LCollision tries to satisfy.

3.3.1 Human Pose Embedding

Our method uses the algorithm in [180,216], which has the ability to extract local deformation components. We represent human models as triangle meshes – a special graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, with $\mathcal{V}$ being a set of vertices and $\mathcal{E}$ being a set of edges. In our datasets, all the models share the same topology, i.e. $\mathcal{E}$ is the same over all the meshes, while $\mathcal{V}$ differs. We transform $\mathcal{V}$ to the as-consistent-as-possible (ACAP) feature space [46], denoted as $\mathcal{X} \in \mathbb{R}^{9\times|\mathcal{V}|}$, to handle large deformations. We use a bilevel autoencoder to embed $\mathcal{X}$ in a latent space. Both levels of the autoencoder involve one graph convolutional layer and one fully connected layer. The fully-connected layer maps the feature to a K-dimensional latent code, with weights denoted as $\mathbf{C} \in \mathbb{R}^{K\times9\times|\mathcal{V}|}$. A sparsity loss is used to ensure that each dimension of $\mathbf{C}$ only accounts for a group of local points.

Domain Decomposition via Attention: We use a bilevel architecture because we want the level-2 autoencoder to learn a decomposed domain of the original mesh, i.e. each level-2 autoencoder only reduces the level-1 residual on a subset of $\mathcal{V}$. The learned domain decomposition not only enhances the reusability and explainability of the neural network but is also used to model the local collisions between body sub-parts, as explained in Sec. 3.3.3.

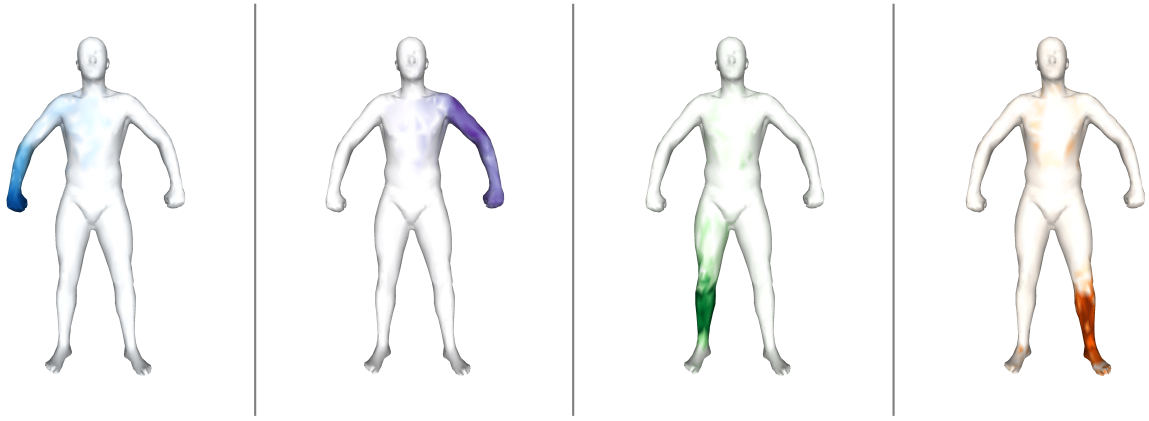


Figure 3.2: We show decomposed domains on the SCAPE dataset using the learned attention mask $\mathcal{M}^{ki}$, highlighted in different colors. The darkness of a given color represents the weight of the soft assignment. These weights are used for localized collision computations.

Each autoencoder maps some input feature $\mathcal{X}$ to a latent code $\mathcal{Z}$ and then reconstructs $\mathcal{Z}$ to feature $\bar{\mathcal{X}}$. We use subscripts to denote the index of an autoencoder, e.g., $\mathcal{X}_0$, $\mathcal{Z}_0$, and $\bar{\mathcal{X}}_0$ are the input, latent code, and output of the level-1 autoencoder, respectively. We assume that each entry of level-1 latent code corresponds to a sub-domain of the human body on which the residual is further reduced using one level-2 autoencoder, so there are altogether $|\mathcal{Z}_0| + 1$ autoencoders. The kth level-2 autoencoder is responsible for representing a subset of residual $\mathcal{X}_0 - \bar{\mathcal{X}}_0$. To determine the subset, an attention mask is computed as: $\mathcal{M}^{ki} = \sum_{j=1}^{9} \mathbf{C}^{kji^2} / \sum_{k=1}^{|\mathcal{Z}_0|} \sum_{j=1}^{9} \mathbf{C}^{kji^2}$. In addition, the input to the kth level-2 autoencoder is $\mathcal{X}_k^i = \mathcal{M}^{ki}(\mathcal{X}_0^i - \bar{\mathcal{X}}_0^i)$. The soft assignment induced by the attention mask conducts the domain decomposition in our network. We illustrate some human body parts decomposed using $\mathcal{M}^{ki}$ in Fig. 3.2.

3.3.2 Collision-Free Constraints

A pivotal requirement of plausible human poses is that they are collision-free, i.e. triangles on the surface mesh do not penetrate each other. However, this constraint is ignored by previous

neural-network-based human pose generation methods. We define a self-collision as an intersection between two topologically disjointed triangles, i.e. two triangles that do not share any edges. We use the following condition to indicate a collision: $\mathbf{t}_p \cap \mathbf{t}_q \neq \emptyset$, where $\mathbf{t}_p$ and $\mathbf{t}_q$ are two triangles. Penetration depth (PD) is a notion that measures the extent of collision constraint violations between two objects. We define the local PD for triangle pair $(\mathbf{t}_p, \mathbf{t}_q)$ as:

$$\text{PD}_{p,q} = \min\{\|\mathbf{d}\|_2 : (\mathbf{t}_p + \mathbf{d}) \cap \mathbf{t}_q = \emptyset\},$$

where $\text{PD}_{p,q}$ is the minimum distance to move $\mathbf{t}_p$ such that $\mathbf{t}_p$ and $\mathbf{t}_q$ have no overlap. The collision-free constraint can be reformulated as the constraint that $\text{PD}_{p,q} = 0$ for any (p, q) pairs. Conceptually, collision constraints can be satisfied by solving the following constrained optimization:

$$\begin{aligned} &\min \quad goal \\ &\text{s.t. } \text{PD}_{p,q} = 0, \quad (p, q) \text{ disjoint}, \end{aligned}$$

where $goal$ is the objective (e.g., as close as possible to a user-desired pose). Prior works solve the constrained optimization by computing $\text{PD}_{p,q}$ for all (p, q) pairs and treating each colliding $(\mathbf{t}_p, \mathbf{t}_q)$ as a standalone constraint, leading to large problem sizes and high computational costs. Instead, we use a neural network to speed up the computation.

3.3.3 Locality of Self-collisions

Our method is inspired by the subspace self-collision culling algorithm (SSCC) [14] and the learning-based collision simplification algorithm [190]. In SSCC, the authors observe that collisions usually occur between pairs of triangles that are originally close to one another on

the template mesh. Pairs of distant triangles penetrate only when the mesh has undergone sufficient deformation. The observation made by SSCC suggests the use of mesh decompositions as described in 3.3.1.

It is worth noting that both works [14, 190] use learned linear subspaces to accelerate collision detection and culling. However, the expressivity of linear subspaces is rather limited, so SSCC can only model deformations that are near the neutral pose and cannot represent larger deformations. Further, it is assumed in [190] that a domain decomposition is provided by users. Our work unifies and extends these ideas into a collision prediction algorithm that works for large deformations and does not require any additional data from users.

3.4 LCollision: Overall Learning Algorithm

Our overall learning architecture is illustrated in Fig. 3.1. Our method augments a normal mesh embedding autoencoder with an additional component to classify the collision status. Given a latent code $\mathcal{Z}_{all}$ defined as:

$$\mathcal{Z}_{all} = \left({\mathcal{Z}_0}^T, {\mathcal{Z}_1}^T, \ldots^T, {\mathcal{Z}_{|\mathcal{Z}_0|}}^T \right)^T,$$

we output a collision probability $\text{MLP}_{classifier}$. We assume that the 0.5 sub-level set of $\text{MLP}_{classifier}$ corresponds to collision-free meshes so that many constraints of the form $\text{PD}_{i,j} = 0$ can be replaced by a single constraint $\text{MLP}_{classifier} < 0.5$, which reduces the computational cost.

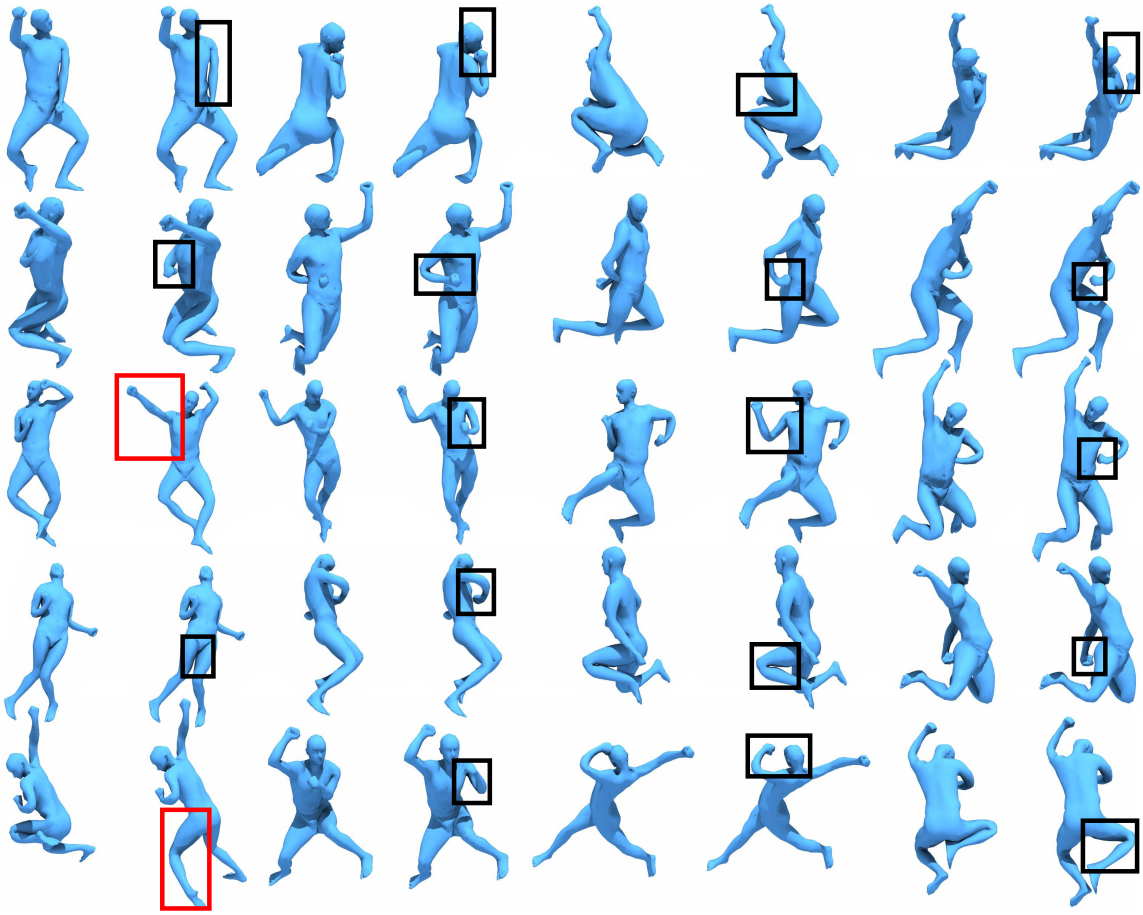


Figure 3.3: We illustrate 20 representative results of collision responses, where the poses on the left are the original poses directly generated using [216], and the poses on the right are the ones after collision responses. We highlight the adjusted body parts using black boxes. In all the examples, our method can successfully avoid penetrations. However, in two cases (red boxes), our adjusted poses drift severely from the original poses.

3.4.1 Collision Detection Architecture

In this subsection, we explain our network architecture to cope with the locality of self-collisions illustrated in gray blocks of Fig. 3.1, including the collision state encoder and the collision predictor.

Naive Subdivision: Our level-2 autoencoders inherently decompose the mesh into $|\mathcal{Z}_0|$ sub-domains. Therefore, if collisions occur within the kth sub-domain, then collisions should

be inferred from $\mathcal{Z}_k$ alone, and we use a collision predictor (CP) in the form of a multilayer perceptron (MLP) to map $\mathcal{Z}_k$ to some collision indicator. If a pair of triangles belongs to two sub-domains, e.g., $\mathcal{Z}_k$ and $\mathcal{Z}_{k'}$, then a possible solution is to use another MLP that takes both $\left({\mathcal{Z}_k}^T, {\mathcal{Z}_{k'}}^T \right)^T$. However, this approach requires $\mathcal{O}(|\mathcal{Z}_0|^2)$ CPs with an excessively large number of weights, and the latent codes of level-2 autoencoders only represent the relative residual $\mathcal{X}_0 - \bar{\mathcal{X}}_0$, while the absolute information $\mathcal{X}_0$ is lost. **Our Method:** To avoid issues with naive subdivision, we propose using a collision state encoder (CSE) that encodes both relative and absolute information over all mesh sub-domains. CSE is an MLP that takes $\mathcal{Z}_{all}$ and brings $\mathcal{Z}_{all}$ through three latent layers with $\left(512, 256, 256 \right)$ neurons and ReLU activation. Finally, CSE outputs a latent code referred to as the global collision state, or $\mathcal{S}_0 = \text{CSE}(\mathcal{Z}_{all})$ for short. $\mathcal{S}_0$ and $\mathcal{Z}_k$ are then fed into a CP to obtain the collision indicator related to the kth sub-domain, i.e. collisions between pairs of triangles where at least one of the triangles belongs to the kth sub-domain. There are altogether $|\mathcal{Z}_0|$ CPs, where the kth CP maps $\left({\mathcal{S}_0}^T, {\mathcal{Z}_k}^T \right)^T$ through four latent layers with $\left(512, 256, 256, 128 \right)$ neurons and ReLU activation. Finally, CP outputs a scalar collision indicator $\mathcal{S}_k$, i.e. $\mathcal{S}_k = \text{CP}(\mathcal{S}_0, \mathcal{Z}_k)$.

3.4.2 Collision Predictor Based on Penetration

We need the collision indicators $\mathcal{S}_k$ and groundtruth labels $\mathcal{S}_i$ to be compatible with numerical optimization. Since we use gradient-based numerical optimization, we need to provide valid gradient information. To this end, $\mathcal{S}_k$ should not only be a collision indicator but also a collision violation metric. In other words, if $\mathcal{S}'_k > \mathcal{S}_k \geq 0$, then we must have $\mathcal{S}'_k$ correspond to a mesh with more collisions than $\mathcal{S}_k$, for which we use the notion of penetration depth. Given a mesh $\mathcal{G}$,

we use the FCL library [132] to compute the squared penetration depth $\text{PD}_{p,q}^2$ of each colliding triangle pair. This colliding pair correlates 6 vertices in $\mathcal{V}$, and we add $\text{PD}_{p,q}^2/6$ to each vertex as the vertex-wise collision violation. After processing all colliding triangle pairs, we have a penetration depth energy vector $\text{PDe} \in \mathbb{R}^{|\mathcal{V}|}$. The overall computation is described by Algorithm 1.

Algorithm 1 Generating Penetration Energy Vector PDe

```
Init PDe = 0⃗ ∈ ℝ^|𝒱|
Run FCL finding the set of all collided disjoint triangle paris as T̂
for (t_p, t_q) ∈ T̂ and the corresponding PD_{p,q} do
    for Vertex i belongs to t_p and t_q do
        PDe_i += PD²_{p,q}/6
    end for
end for
```

After computing the PDe, we use the following domain-decomposed data loss to train $\mathcal{S}_i$:

$$\mathcal{L}_{PD} = \sum_{k=1}^{|\mathcal{Z}|_0} \|\mathcal{S}_k - \sum_{i=1}^{|\mathcal{V}|} \mathcal{M}^{ki}\text{PDe}_i\|^2 + \|\mathcal{S}_{sum} - \text{PDe-sum}\|,$$

where $\text{PDe-sum} = \sum_{i=1}^{|\mathcal{V}|} \text{PDe}_i$ is the ground truth total penetration energy and $\mathcal{S}_{sum} = \sum_{k=1}^{|\mathcal{Z}|_0} \mathcal{S}_k$ is the neural network prediction. Here, we use the same attention mask $\mathcal{M}^{ki}$ defined in Sec. 3.3.1 to decompose the collision energy into body parts. Note that we do not have any loss terms related to $\mathcal{S}_0$. However, a neural network is known to suffer from over-fitting when learning exact distance functions [29, 67], including those corresponding to PD. Further, it is inherently difficult to train a perfect regression model for values like collision penetration depth with a long-tail distribution [206]. We avoid over-fitting by using the marginal ranking loss. Given two meshes, $\mathcal{G}$ and $\hat{\mathcal{G}}$ (with approximated total penetration energy denoted as $\mathcal{S}_{sum}$ and $\widehat{\mathcal{S}_{sum}}$) randomly sampled from the dataset, if $\hat{\mathcal{G}}$ has a higher collision violation than $\mathcal{G}$ in terms of the

total penetration energy, then we define:

$$\mathcal{L}_{rank} = \mathbf{max}(0, \alpha - (\widehat{\mathcal{S}_{sum}} - \mathcal{S}_{sum})),$$

and vice versa. Here, α is used as a margin to enforce ranking strictness. We choose α as the mean energy difference of the given dataset.

With the above training technique, we can predict $\mathcal{S}_1, \mathcal{S}_{|Z_0|}$ and use them as hard constraints by letting $\mathcal{S}_i = 0$, resulting in $|Z_0|$ constraints. We can further reduce the online computational cost by reducing the number of constraints to only one. To perform this computation, we train a single classifier $\text{MLP}_{classifier}(\mathcal{S}_1, \cdots, \mathcal{S}_{|Z_0|})$ to summarize the information and predict whether there are any collisions throughout the human body, i.e. $\text{MLP}_{classifier}$ is an indicator of whether $S_{sum} = 0$. To make sure that the 0.5 sub-level set is the collision-free subset, we use the cross entropy loss:

$$\begin{aligned}\mathcal{L}_{entropy} = &- \mathbb{I}(\text{PDe-sum} > 0)\mathbf{log}(\text{MLP}_{classifier}) \\ &- \mathbb{I}(\text{PDe-sum} = 0)\mathbf{log}(1 - \text{MLP}_{classifier}).\end{aligned}$$

3.4.3 Solving Constrained Optimization

Our collision response solver takes a constrained optimization in the following form:

$$\begin{aligned}&\underset{\mathcal{Z}_{all}}{\mathbf{argmin}} \ \|\mathcal{Z}_{all} - \mathcal{Z}^*_{all}\|^2 \\ &\mathbf{s.t.}\ \text{MLP}_{classifier}\left(\mathcal{S}_1, \cdots, \mathcal{S}_{|\mathcal{Z}_0|}\right) \leq 0.5.\end{aligned} \tag{3.1}$$

The idea is to provide a desired pose $\mathcal{Z}^*_{all}$ for the bilevel decoder, and Eq. 3.1 solves for a collision-free $\mathcal{Z}_{all}$ that is as close to $\mathcal{Z}^*_{all}$ as possible. We solve Eq. 3.1 using the augmented Lagrangian method implemented in LOQO [196], with all the gradient information computed via back-propagation through the neural network. This augmented Lagrangian method can start from an infeasible domain, which means that LOQO allows the hard constraints to be temporarily violated between the iterations. As a result, LOQO uses gradient information to pull the solution back to the feasible sub-manifold.

3.5 Evaluation

We implement our method using PyTorch [136]. All the training and testing are performed on a single desktop machine with a 32-core CPU, 32GB memory, and an NVIDIA GTX 1080Ti GPU. The training is decomposed into two stages. During the first stage, we train the two-level human pose embedding architecture using a set of N meshes. This training would optimize only the $|\mathcal{Z}_0| + 1$ autoencoders and the attention mechanics. After this first stage, we generate a much larger dataset of $M \gg N$ meshes by sampling the latent code $\mathcal{Z}_{all}$ uniformly in the range:

$$[1.2\mathbf{min}(\mathcal{Z}_{all}), 1.2\mathbf{max}(\mathcal{Z}_{all})]^{|\mathcal{Z}_{all}|},$$

where $\mathbf{min}(\mathcal{Z}_{allk}) < 0$, $\mathbf{max}(\mathcal{Z}_{allk}) > 0$, and $\mathbf{min}, \mathbf{max}$ are elementwise over all mesh samples.

We train our collision predictor and classifier on the augmented dataset while fixing the

$|\mathcal{Z}_0| + 1$ autoencoders and the attention mechanics. This stage uses the loss:

$$\mathcal{L} = w_{PD}\mathcal{L}_{PD} + w_{rank}\mathcal{L}_{rank} + w_{entropy}\mathcal{L}_{entropy},$$

which is configured with $w_{PD} = 5$, $w_{rank} = 2$, $w_{entropy} = 2$, and trained using a learning rate of 0.001 and a batch size of 32 over 30 epochs. We evaluate our method on three datasets: the SCAPE dataset [10] with $N = 71$ meshes, the MIT Swing dataset [199] with $N = 150$ meshes, and the MIT Jumping dataset [199] with $N = 150$ meshes. For each dataset, we use all the meshes to train the embedding space during the first stage, where we set $|\mathcal{Z}_0| = 10$ for SCAPE and $|\mathcal{Z}_0| = 12$ for Swing and Jumping. During the second stage, we use $0.7M$ samples of the augmented dataset for training and $0.3M$ samples for validation. We use two settings, one with $M = 5 \times 10^4$ and the other with $M = 2.5 \times 10^6$.

Baseline	MSE	RANK	CLASSIFY
$Ours$	6.72×10^{-4}	6.5×10^{-3}	82.8%
$\mathcal{L}_{entropy} + \mathcal{L}_{PD}$	5.06×10^{-4}	9.4×10^{-3}	81.1%
$\mathcal{L}_{entropy} + \mathcal{L}_{rank}$	-	4.7×10^{-3}	80.7%
$\mathcal{L}_{entropy}$	-	-	80.4%
ND	6.9×10^{-4}	6.7×10^{-3}	80.2%

Table 3.1: We compare our method (Ours) with 4 baselines: $\mathcal{L}_{entropy} + \mathcal{L}_{PD}$, $\mathcal{L}_{entropy} + \mathcal{L}_{rank}$, $\mathcal{L}_{entropy}$, and ND (no collision decomposition). For each method, we train on the smaller dataset with $M = 5 \times 10^4$ meshes, and we compare their accuracy in terms of predicting penetration depth energies (MSE), ranking penetration depth energies (RANK), and classifying collision-free meshes (CLASSIFY). The result shows that our hybrid loss improves the overall accuracy of collision predictions. Especially, the improvement over ND demonstrates the effectiveness of decomposing a collision into body parts.

Accuracy of Collision Prediction: We consider several baselines that are essentially simplified variants of our pipeline in Fig. 3.1. We notice that the constrained optimization Eq. 3.1

only needs the output of $\text{MLP}_{classifier}$ to be correct, which is the goal $\mathcal{L}_{entropy}$. Therefore, we consider retaining only $\mathcal{L}_{entropy}$ while removing $\mathcal{L}_{rank}$ and $\mathcal{L}_{PD}$, leading to three baselines: $\mathcal{L}_{entropy} + \mathcal{L}_{PD}$, $\mathcal{L}_{entropy} + \mathcal{L}_{rank}$, and $\mathcal{L}_{entropy}$, where we use the same weights for the retained terms. In order to demonstrate the power of collision decomposition, we also compare our LCollision with a simplified network architecture that does not decompose the collision into body parts. For this baseline, we simply use S_0 to predict total penetration energy $\mathcal{S}_{sum}$ and classify collision status, and we modify $\mathcal{L}_{PD}$ to only have $\|\mathcal{S}_{sum} - \text{PDe-sum}\|$. The other two losses $\mathcal{L}_{PD}$ and $\mathcal{L}_{entropy}$ remain the same. This baseline is denoted as ND (no decomposition).

In Table 3.1, we compare the accuracy of baselines in terms of predicting penetration depth energies, ranking penetration depth energies, and classifying collision-free meshes. To ensure that our predicted penetration depth energies are accurate, we use the mean squared error (MSE) of total penetration depth energy averaged over the $0.3M$ test meshes. To ensure the accuracy of the ranking penetration depth energies, we randomly formulate a pair for each sample in the $0.3M$ test meshes, and we record the average ranking margin (RANK). To classify collision-free meshes, we use the rate of success (CLASSIFY) over the $0.3M$ test meshes.

From this ablation study, we compare ND and our method to find that penetration decomposition can improve the accuracy of collision predictions, which also suggests that using the two parts of the $\mathcal{L}_{PD}$ could better inform the network of collision locality. Using penetration depth energy in the system not only provides gradient information for optimization but can also boost performance through $\mathcal{L}_{PD}$. $\mathcal{L}_{rank}$ does help improve performance, but its effect is relatively minor compared to $\mathcal{L}_{PD}$.

Our second study inspects the robustness of our network architecture in terms of the size of the dataset. As shown in Table 3.2, we tested our method trained using two different M.

M	Dataset	MSE	RANK	CLASSIFY
5×10^4	SCAPE	6.72×10^{-4}	6.5×10^{-3}	82.8%
	Swing	7.27×10^{-4}	3.38×10^{-3}	91.2%
	Jumping	6.74×10^{-4}	5.29×10^{-3}	91.6%
2.5×10^6	SCAPE	7.80×10^{-4}	2.60×10^{-3}	94.1%
	Swing	2.57×10^{-4}	2.34×10^{-3}	96.2%
	Jumping	6.34×10^{-4}	5.43×10^{-3}	95.4%

Table 3.2: We study the robustness of our method in terms of dataset sizes. Increasing the dataset size M can significantly boost the collision detection accuracy (CLASSIFY). This result implies that learning to predict collisions is challenging, and a larger training dataset can help improve the overall results.

Dataset	Time (FCL)	Time (ours)		Speedup	
		CPU	GPU	CPU	GPU
SCAPE	1min 23s	3.99s	1.02s	21x	81x
Swing	5min 12s	3.78s	0.91s	82x	342x
Jumping	5min 19s	4.23s	1.13s	75x	282x

Table 3.3: We compare LCollision with [132] in terms of the computational cost for collision detection. [132] only supports the CPU version, while we tested both the CPU and the GPU versions of our method. All datasets have 1.5×10^4 samples ($0.3M$ validation samples for $M = 5 \times 10^4$). Meshes in the Swing and Jumping datasets have more vertices (9971 and 10002) than SCAPE (2261), and the complexity of [132] depends on the number of points; thus, exact collision checking [132] takes more time. However, they all share the same level of latent space size with SCAPE, and the running times of our method are almost identical.

Increasing M from 5×10^4 to 2.5×10^6 can significantly boost the collision detection accuracy (CLASSIFY). This result implies that learning to predict collisions is challenging, and a larger training dataset can help improve the overall results.

Speedup Compared with Exact Collision Checking: The goal of our method is to speed up the collision detection process over prior, exact methods that are applied to mesh-based representations. We compare the running time with [132] on the test set of 5×10^4 samples (1.5×10^4 samples) for the SCAPE, Swing, and Jumping datasets. The implementation of [132] only sup-

ports CPU, while LCollision runs on both CPU and GPU. To achieve the best performance for [132], we run their method using 15 threads in parallel and stop when one collision occurs or the mesh is reported to be collision-free. For our method, we feed the network with 500 models at the same time. We optimize the hyper-parameters to obtain optimal performance. We show the results in Table 3.3 and observe two orders of magnitude speedup.

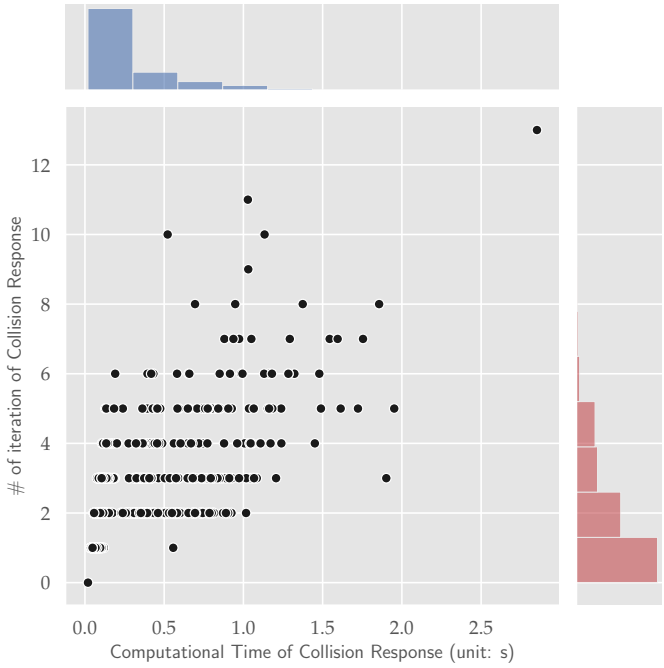


Figure 3.4: The joint histogram of number of iterations (Y-axis) and computational time (X-axis) used for solving the constrained optimization (Eq. 3.1) for the Swing dataset. The average number of iterations is 5.44 and the average computation time is 1.29s.

The Collision Response Solver: In Fig. 3.3, we show 20 results with successful collision responses for the SCAPE dataset. To profile the collision response solver quantitatively, we sample a set of 3000 random human poses by randomizing $\mathcal{Z}_{all}$ for both the SCAPE and Swing datasets. Some of the models have self-collisions and are classified correctly by our learning-based collision detection algorithm. For each of these meshes, we solve Eq. 3.1 and we consider a solution successful if the augmented Lagrangian algorithm returns a feasible solution. On

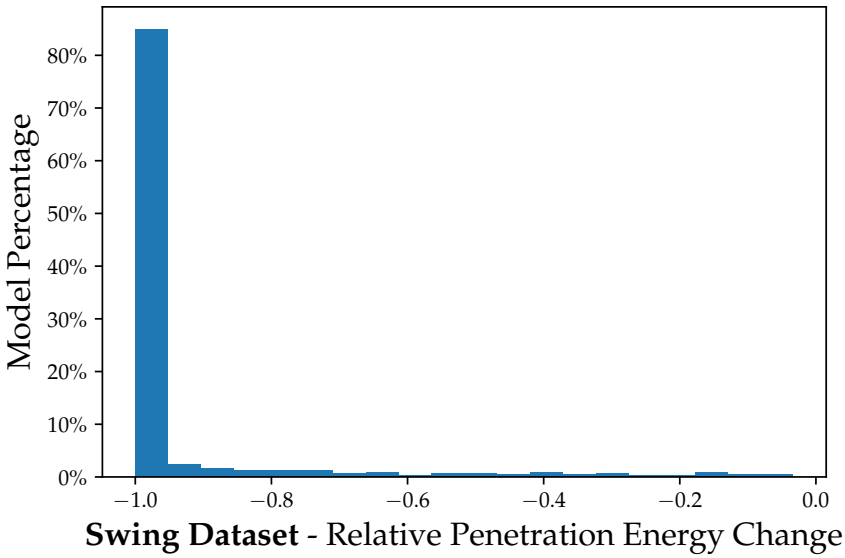


Figure 3.5: The histogram of relative penetration depth energy change for successful examples, after solving Eq. 3.1 for the Swing dataset. We see results with decreased penetration energy as successful ones. Our method achieves a success rate of 85.1%, and we observed an average relative decrease of 94.3% for these models compared to original penetration energy.

the SCAPE dataset, our method achieves a success rate of 85.6%, and we observe a relative decrease of 80.9% in penetration energy. On the Swing dataset, our method achieves a success rate of 85.1%, and we observe a relative decrease of 94.3%. In Fig. 3.4, we plot the number of iterations and computational time used by the constrained optimizer until convergence for the Swing dataset. The average iteration is 5.44 and the average time is 1.29s. For the SCAPE dataset, the average iteration is 2.09 and the average time is 0.25s. In Fig. 3.5, we highlight the distribution of relative penetration energy change for successful collision response models.

3.6 Conclusion, Limitations, and Future Work

We present LCollision, a method for learning the collision-free human pose sub-manifold. We use a mesh embedding autoencoder to learn a full human pose manifold and augment it with additional components to classify the collision-free meshes. Our method decomposes the mesh

into several sub-domains and learns the decision boundary of the collision-free sub-manifold by reusing the decomposed sub-domains. Specifically, we learn to predict the penetration depths aggregated to each sub-domain and then use a binary classifier to predict whether a given mesh has any collisions. When evaluated on the SCAPE dataset, our method achieves a success rate of 94.1% in predicting collisions and a success rate of 85.6% in collision responses.

Our method has some limitations. Being a learning-based method, our collision predictor cannot achieve a 100% success rate, in contrast to exact collision detection algorithms. This could pose a problem when our method is used to generate computer animations, where a few missed collisions can have a considerable impact on the overall simulation accuracy. Moreover, our learning method can only be applied to models with fixed topology and requires additional data collection and training for different mesh topologies. In the future, we would like to consider active learning to collect more data and improve the accuracy of the collision predictor in a self-supervised manner, and thereby reduce the need for large training datasets. A similar approach is used in [65, 133] for rigid objects. A second issue is the use of a continuous constraint optimizer [196] for collision responses. These solvers require twice-differentiable hard constraints, which is not the case in our application because we use non-differentiable ReLU activation units. It is worth exploring new constraint optimization solvers that could work with non-smooth constraints specified by a neural network. There are many issues in terms of incorporating hard constraints into a neural network. If only soft penalties are needed, we can reformulate the hard constraint in Eq. 3.1 as a soft penalty term and solve the unconstrained problem via a Newton-Type method, allowing users to adjust the penetration allowed in the final. We can extend our work by considering other types of hard constraints such as dynamics and accurate collision response models. Finally, since our method uses a hybrid loss, it may compromise the performance

in some metrics, e.g., the regression loss of the penetration depth. Moreover, techniques based on parameter estimation can be used to improve the performance of such learning methods [209].

Chapter 4: N-Penetrate: Active Learning of Neural Collision Handler for Complex 3D Mesh Deformations

4.1 Introduction

Learning to model or simulate deformable meshes is becoming an important topic in computer vision and computer graphics, with rich applications in real-time physics simulation [68], animation synthesis [147], and cross-domain model transformation [33]. Central to these methods are generative models that map from high-dimensional deformed 3D meshes with rich details to low-dimensional latent spaces. These generative models can be trained from high-quality groundtruth datasets, and they infer visually or physically plausible meshes in real time. These 3D datasets can also be generated using physics simulations [124, 186] or reconstructed from the physical world using multi-view capture systems [171]. In general, 3D deformable meshes are more costly to acquire, so 3D mesh datasets typically come in smaller sizes than image or text datasets. Inference models trained using such small datasets can suffer from over-fitting and generate meshes with various visual artifacts. For example, human pose embedding networks [47, 180] can have excessive deformations, and interaction networks [15] can result in non-physically-based object motions.

The goal of our research is to resolve a major source of visual artifacts: self-collisions.

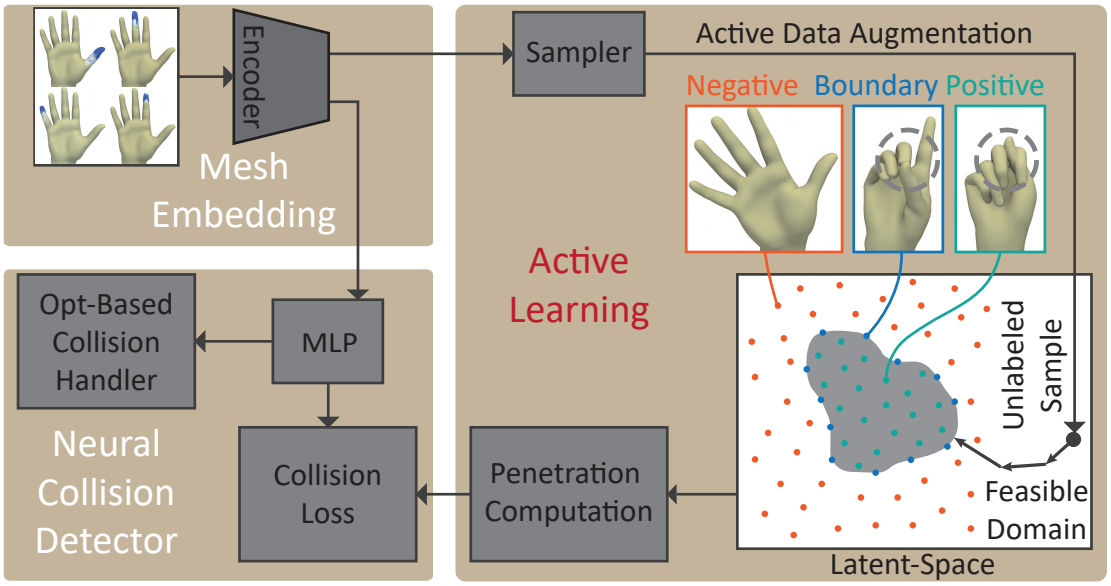


Figure 4.1: Our method (N-Penetrate) consists of a (learned or analytic) latent space, a neural collision detector, and an optimization-based collision handler. We progressively insert data by randomly sampling in the latent space (sampler). We then use a Newton-type method to pull unlabeled samples towards the learned decision boundary (black arrows in the white, feasible domain). The groundtruth collision labels are generated using an analytic collision detector (penetration computation). Finally, we use three different loss functions for samples on the positive (orange), negative side (green), and near (blue) the decision boundaries.

Instead of acquiring more data, we argue that domain-specific knowledge could also be utilized to significantly improve the accuracy of inference models. There have been several prior research works along this line. For example, [217] exploited the fact that near articulated meshes can be divided into multiple components, and they train a recursive autoencoder to stitch the components together. [227] utilized the locality of secondary physics motions to learn re-targetable and scalable real-time dynamics animation. Chapter 3 studied learning-based collision avoidance for 3D meshes corresponding to human poses. We proposed a deep architecture to detect collisions and used numerical optimizations to resolve detected collisions. However, Chapter 3 used a large mesh dataset to obtain stable performance of neural collision detection. Indeed, a deformed 3D mesh typically involves more than 10^4 elements (voxels, points, triangles) where any pair of

two elements can have collisions. Therefore, a huge amount of data is required to present the inference model with enough examples of collisions between all possible element pairs.

Main Results: We present a robust method to train a neural collision handler, *N-Penetrate*, for complex, 3D deformable meshes using active learning. Our key observation is that the distribution of penetrating meshes can have a long tail and active learning is an effective method for modeling the tail [50]. Unfortunately, most 3D mesh datasets do not focus on generating samples in the tail and cannot be used to train stable collision detectors. In order to overcome these issues, our approach combines three main ideas:

- We use active learning to progressively insert new samples into the dataset. Collision labels for the new samples are automatically generated;
- We use a risk-seeking approach to prioritize samples near the decision boundary, so that the inserted samples can best help improve our accuracy;
- We use different loss functions for samples far from and close to the decision boundary.

Our overall approach, N-Penetrate, is shown in Fig. 4.1. We show that our training method is versatile and can learn to detect collisions for meshes encoded in various low-dimensional spaces, such as a learned latent-space [216] and a domain-specific human body deformation representation [107]. For either encoding, we further show that our active learning technique outperforms supervised learning in terms of data efficacy and accuracy. We evaluate our method on three types of complex datasets:

- Dressed and undressed human poses, including SCAPE [10], MIT Swing [199], MIT Jump [199], and AMASS-MPIMosh [114] containing different genders and body shapes with

54387 meshes;

- Cloth simulations from [216] with complex deformation and self-collisions;
- Human hands captured by a multi-view camera system.

Compared to prior supervised learning approaches, our method exhibits much higher data efficacy and accuracy (up to 98.1%) and uses fewer training samples (up to 48.12% less). Given a training dataset of the same size, our method reduces the false negative rate by 14.27% on average, and we are able to resolve more self-colliding meshes. For a test size with 1×10^4 meshes, our method only costs $1.32s$ on the GPU. Overall, ours is the first practical method for neural collision handling of general, 3D complex meshes.

4.2 Related Work

Generative Model of Dense 3D Shapes: Categorized by shape representations, generative models can be based on point clouds [146], volumetric grids [211], multi-charts [56], surface meshes [179], or semantic data structures [61]. We use mesh-based representations with fixed topologies, as most collision detection libraries are designed for meshes. Some generative models can learn to represent general meshes of changing topology, e.g., for modeling meshes of hierarchical structure [220] or modeling scenes with many objects [152]. However, these applications typically involve only static meshes with no need for collision detection. There is a separate research direction on domain-specific mesh deformation representation, e.g., SMPL/STAR human models [107, 130], wrinkle-enhanced cloth meshes [89], and skeletal skinning meshes [213]. Our method is versatile and can be combined with both a learned embedding [179] and a SMPL representation [107, 130].

Active Learning: An active learner alternates between drawing new samples and exploiting existing samples. These samples can be drawn guided by an acquisition function in Bayesian optimization [128] or from an expert algorithm [37]. Active learning has been applied to approximate the boundary of the configuration space [35, 133, 191], where the feasible domain of collision constraints is parameterized using a kernel SVM. However, these methods are limited to rigid or articulated deformations and are not applicable to general 3D deformations. More broadly, active learning has been adopted in various prior works to accelerate data labeling in image classification [43] and object detection [2] tasks. These methods progressively identify unlabeled images to be forwarded to experts for labelling. An alternative method for selecting the samples is identifying a coreset [139], and the authors of [163] propose a practical algorithm for coreset identification via k-center clustering. These methods consider a discrete dataset, while we assume a continuous latent space of samples for training generative models and use a risk-seeking method to identify critical new samples.

4.3 Neural Collision Handler

We introduce our neural collision handling architecture, based on which we build our active learning method. All notations are summarized in the symbol table.

A mesh is represented by the graph $\mathcal{G} = \langle V, E \rangle$, where V is a set of vertices, and E is a set of edges. We assume that all the meshes have the same topology, that is, all the meshes differ in V while the connectivity E stays the same. We further limit ourselves to manifold triangle meshes, i.e., each edge is incident to at most two triangles, and two triangles are adjacent if and only if they share an edge. No other assumptions are made on the mesh deformation and

our method can handle high-resolution meshes consisting of thousands of vertices via efficient embedding. As a result, our method can represent general, complex meshes from various domains of applications. We denote a mesh as self-collision-free if and only if any pair of two non-adjacent triangles are not intersecting each other. Our goal is to design a mesh-based generative neural architecture where we take an input as a coordinate in the latent space and output a 3D mesh without self-collisions. The latent space is defined as a low-dimensional space that can be mapped to high-dimensional meshes injectively using a learned or analytic decoder function. Furthermore, the latent-to-mesh mapping is differentiable and supports multiple downstream applications explained in Sec. 4.3.3. We have experimented with two latent-spaces, a learned bilevel autoencoder [216] and the SMPL human body representation [107], which we briefly review below.

4.3.1 Bilevel Autoencoder

The bilevel autoencoder architecture maps a deformed mesh to two levels of latent codes. We only want to encode intrinsic mesh information such as curvatures instead of extrinsic rigid transformations because mesh shapes are invariant to extrinsic transformation. Therefore, we first use the as-consistent-as-possible (ACAP) feature transformation [46] to factor out rigid transformations. The ACAP feature vector is first brought through the level-1 autoencoder and mapped to a latent code Z_0. We further hypothesize that the error is sparsely distributed throughout the mesh vertices. Therefore, we then use an attention mechanism trained with a sparsity prior to decompose the mesh into sub-domains. The sparsity prior is designed such that each domain can be mapped to a single axis of the latent space, i.e., a single entry of Z_0. Afterwards, a set of

$|Z_0|$ level-2 autoencoders is introduced to further reduce the error, with each autoencoder dedicated to one entry of Z_0. Their latent codes are denoted as $Z_1, \cdots, Z_{|Z_0|}$. The ultimate mesh is reconstructed from Z_{all} by combining level-1 and level-2 latent codes:

$$\begin{aligned} Z_{\text{all}} &= \Big(Z_0, Z_1, \cdots, Z_{|Z_0|} \Big) \\ D(Z_{\text{all}}, \theta_D) &= \sum_{i=0}^{|Z_0|} D_i(Z_i, \theta_{D_i}) \\ V &= \text{ACAP}^{-1}(D(Z_{\text{all}}, \theta_D)), \end{aligned}$$

where D_i is the ith decoder, with θ_{D_i} being the learnable parameters. The mesh vertices V are reconstructed by inverting the ACAP transformation. Correspondingly, we have the encoder defined as $E(\text{ACAP}(V), \theta_E) = Z_{\text{all}}$, which maps the vertices of a mesh to the latent space, with θ_E being the learnable parameters.

Our neural collision detector predicts whether the mesh V is subject to self-collisions using latent information Z_{all}. The extent to which two meshes collide can be measured by the notion of Penetration Depth (PD) [224], defined by the norm of the smallest configuration change needed for a mesh to be self-collision-free, as illustrated in Fig. 4.2. It is well-known that PD is a non-smooth function of V (esp. at the boundaries), thereby making it difficult to resolve collisions by minimizing PD. By choosing appropriate activation functions ($\tanh$ and CELU in our case), we design the neural collision detector to be a differentiable approximation of PD. As a result, gradient information can be propagated to a collision handler to minimize PD.

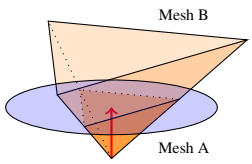


Figure 4.2: PD (red arrow) is the locally minimal translation for mesh B (orange) to be collision-free from mesh A.

Since collisions can happen between any pairs of geometric mesh primitives, a collision detector should consider possible contacts between any pair of sub-domains, leading to a quadratic complexity $\mathcal{O}(|Z_0|^2)$. We use a global-local detection architecture that effectively reduces the number of learnable parameters. Specifically, we introduce a global collision state encoder $S_0 = \text{CSE}(Z_{\text{all}}, \theta_C)$ and a set of local collision predictors $S_i = \text{CP}(S_0, Z_i, \theta_C)$, with $i = 1, \cdots, |Z_0|$, which predicts whether the ith sub-domain is in collision with the rest of the mesh. Finally, the collision information for all local collision predictors is summarized using a classifier network $\text{MLP}_c(S_1, \cdots, S_{|Z_0|})$ to derive a single overall collision classifier:

$$
\begin{aligned}
S_0 &\triangleq \text{CSE}(Z_{\text{all}}, \theta_C) \\
S_i &\triangleq \text{CP}(S_0, Z_i, \theta_C) \\
I_c(Z_{\text{all}}) &\triangleq \mathbb{I}(\text{MLP}_c(S_1, \cdots, S_{|Z_0|}, \theta_C) \geq 0.5),
\end{aligned}
$$

where θ_C is the learnable parameters. The feasible space boundary of collision-free constraints corresponds to the 0.5-levelset of MLP_c.

4.3.2 SMPL Human Pose Representation

The SMPL model represents a human body shape using two kinds of parameters: body pose and body shape. In order to determine mesh vertices V, SMPL uses a template mesh of a reference body shape and pose V^0. The variation in body shape is addressed by adding a linear perturbation: $V_s^0 = V^0 + \sum_i Z_{\beta_i} B_i$, where B_i is a set of shape variation bases and Z_{β_i} are the corresponding shape coefficients. From V_s^0, SMPL derives the final V using standard linear blend skinning: $V = D(Z_\theta, V_s^0)$, where we unify the notations for both mesh embedding methods and

reuse the symbol D for the linear blend skinning function. Here Z_θ is the pose parameters. The overall latent information for the SMPL model is $Z_{\text{all}} = (Z_\beta \; Z_\theta)$. We refer readers to [107] for more details. We use the collision detector offered by [121] on the bodies generated by the SMPL model to neglect natural intersections around nearby tissues. Nevertheless, we still use PD to represent the energy measuring the collision extent. Since no domain decomposition is involved, we predict a single collision value S and use a standard MLP as our collision detector taking S as input, i.e.,:

$$S \triangleq \text{CP}(Z_{\text{all}}, \theta_C)$$

$$I_c(Z_{\text{all}}) \triangleq \mathbb{I}(\text{MLP}_c(S, \theta_C) \geq 0.5).$$

4.3.3 Optimization-Based Collision Response

Following Chapter 3, we design our collision detector MLP_c to be differentiable. Suppose we take as input a randomly sampled latent code $Z_{\text{all}}^{\text{user}}$, which might not satisfy the collision-free constraints, we project that latent code back to the feasible domain by solving the following optimization problem under neural collision-free constraints using the Augmented Lagrangian Method (ALM):

$$\underset{Z_{\text{all}}}{\textbf{argmin}} \;\; E(Z_{\text{all}}) \quad \textbf{s.t.} \;\; \text{MLP}_c(S_1, \cdots, S_{|Z_0|}, \theta_C) \leq 0.5, \tag{4.1}$$

where $E(\bullet)$ is some objective function, which can take multiple forms, as specified by down-stream applications. In the simplest case, we take as input a desired $Z_{\text{all}}^{\text{user}}$, and we can define $E(Z_{\text{all}}) = \|Z_{\text{all}} - Z_{\text{all}}^{\text{user}}\|^2/2$, which is only related to latent space variables. As a more intuitive

interface, the user might want to change meshes in the Cartesian space instead of the of latent space. For example, if the user wants a human hand to be at a certain position V^{user}, we could define $E(Z_{\text{all}}) = \|D(Z_{\text{all}}) - V^{\text{user}}\|^2/2$. A desirable feature of Eq. 4.1 is an invariant problem size. However many vertices a mesh has, there is only one constraint, which guarantees high test-time performance. Moreover, it has been shown in Theorem 10.4.3 of [178] that ALM either finds a feasible solution or returns an infeasible solution that is closest to the boundary of the feasible domain. In other words, ALM always makes a best effort to resolve collisions, even if feasible solutions are not available.

4.4 Active Learning Algorithm

The goal of active learning is to iteratively improve the accuracy of the neural collision detector. We assume the availability of an existing dataset $\mathcal{D}$ of "high-quality" meshes with deformed vertices $\mathcal{D} = \{V^{1,2,\cdots,|\mathcal{D}|}\}$, which is used to train the mesh embedding component. (Note that SMPL also requires a dataset $\mathcal{D}$ to learn the linear blend skinning weights). We assume that meshes in $\mathcal{D}$ are collision-free but the meshes reconstructed using function D can still suffer from collisions due to embedding error after training, and users might explore the latent space in regions that are not well covered by the training dataset. As a result, our neural collision detector cannot be trained with $\mathcal{D}$ alone. This is because $\mathcal{D}$ only contains negative (collision-free) samples, while the neural collision detector must learn the decision boundary between positive and negative samples. In other words, the neural detector must be presented with enough samples to cover all possible latent codes with both self-penetrating and collision-free meshes. We denote the training dataset of neural collision detectors as another set: $\mathcal{D}_c =$

$\{\langle Z^i_{\text{all}}, I^*_c(Z^i_{\text{all}})\rangle \,|i = 1, 2, \cdots, |\mathcal{D}_c|\}$, where $I^*_c(\bullet)$ is the groundtruth $0-1$ collision state label.

The groundtruth collision state label can be generated automatically using a robust algorithm such as [132], to compute PD, where a positive PD indicates self-collisions, so we can define $I^*_c(Z_{\text{all}}) \triangleq \mathbb{I}(\text{PD}(Z_{\text{all}}) > 0)$, where $\text{PD}(Z_{\text{all}}) > 0$ means first recovering the mesh V from Z_{all} and then compute PD via [132]. However, the cost to compute penetration depth, $\text{PD}(\bullet)$, is superlinear in the number of mesh vertices, and computing PD for an entire dataset can still be a computational bottleneck. Moreover, we are considering a continuous space of possible training data that cannot be enumerated. To alleviate the computational burden, we design a three-stage method, as illustrated in Fig. 4.1. During the first stage of bootstrap, we sample an initial boundary set, by which we train MLP_c to approximate the true decision boundary. At the second stage of data augmentation, new training data is selected and progressively injected into a dataset. Finally, for the third stage, our neural collision detector is updated to fit the augmented dataset. The criterion for selecting the subset is critical to the performance of active learning. Since our neural collision predictor provides constraints for a nonlinear optimization method, we observe that samples far from the boundary are not used by the optimizer and only the boundary of the feasible domain (gray area in Fig. 4.1 right) is useful. Therefore, we propose using a Newton-type risk-seeking method to push the samples towards the decision boundary. We provide more details for each step below.

4.4.1 Bootstrap

Active learning would progressively populate $\mathcal{D}_c$, so prior work [2] simply initializes the dataset to an empty set. However, we find that a good initial guess can significantly improve the

convergence of training. This is because we select new data by moving (randomly sampled) latent codes towards the decision boundary of the PD function using a risk-seeking method. However, the true boundary of the collision-free constraints corresponds to the boundary of C-Obstacles, which is high-dimensional and unknown to us (PD is a non-smooth function, so we cannot even use gradient information to project a mesh to the zero level-set of PD). Instead, we propose using the learned neural decision boundary, i.e., the 0.5-levelset of MLP_c, as an approximation. If we initialize $\mathcal{D}_c = \emptyset$, the surrogate decision boundary is undefined, and the training might diverge or suffer from slow convergence. For our bootstrap training, we uniformly sample a small set of N_{init} latent codes $\mathcal{Z}_{\text{all}}$ at random positions from the latent space and compute PD for each of them. We define a valid space of sampling by mapping all the data $Z^i_{\text{all}} \in \mathcal{D}$ to their latent codes and compute a bounded box in the latent space:

$$\mathcal{Z} = \prod_{j=0}^{|Z_0|} \left[\min_{i=1,\cdots,|\mathcal{D}|} \left[e_j^T Z^i_{\text{all}}\right], \max_{i=1,\cdots,|\mathcal{D}|} \left[e_j^T Z^i_{\text{all}}\right] \right].$$

We hypothesize that all the meshes can be embedded using our autoencoder or the SMPL model with small error corresponding to latent codes in $\mathcal{Z}$, so we can initialize $\mathcal{D}_c = \{Z^{1,\cdots,N_{\text{init}}}_{\text{all}} | Z^i_{\text{all}} \sim U(\mathcal{Z})\}$. We then divide the data points into three subsets ($\mathcal{D}_c = \mathcal{D}_p \bigcup \mathcal{D}_n \bigcup \mathcal{D}_b$, illustrated in Fig. 4.1 right):

$$\begin{aligned}
\mathcal{D}_p &\triangleq \{\left\langle Z^i_{\text{all}}, I_c(Z^i_{\text{all}})\right\rangle | \text{PD}(Z_{\text{all}}) > \epsilon\} \\
\mathcal{D}_n &\triangleq \{\left\langle Z^i_{\text{all}}, I_c(Z^i_{\text{all}})\right\rangle | \text{PD}(Z_{\text{all}}) = 0\} \\
\mathcal{D}_b &\triangleq \{\left\langle Z^i_{\text{all}}, I_c(Z^i_{\text{all}})\right\rangle | \text{PD}(Z_{\text{all}}) \in (0, \epsilon]\}.
\end{aligned} \tag{4.2}$$

Here, $\mathcal{D}_p$ is the positive set consisting of samples with penetrations deeper than a threshold ϵ, $\mathcal{D}_n$ is the negative set consisting of collision-free samples, and $\mathcal{D}_b$ is a boundary set where samples are nearly collision-free and lie on the decision boundary. We will introduce our new loss term for the boundary set in Sec. 4.4.3.

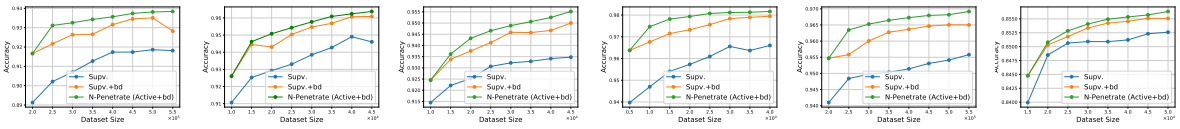


Figure 4.3: We plot the accuracy of the neural collision detector against the dataset size. The baselines are trained using the same amount of data. On average, ours achieves 1.62% higher accuracy than ***Supv***. From left to right: SCAPE, Swing, Jump, Skirt, Hand, and AMASS.

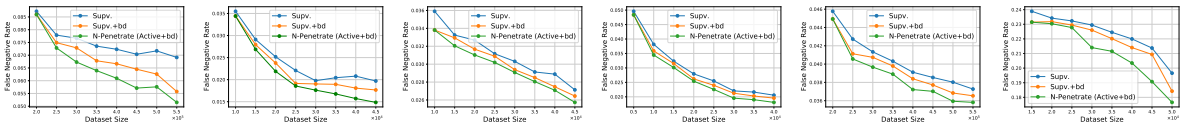


Figure 4.4: We plot the false negative rate against the dataset size. The baselines are trained using the same amount of data. On average, ours achieves a 13.59% lower false negative rate than ***Supv***. From left to right: SCAPE, Swing, Jump, Skirt, Hand, and AMASS.

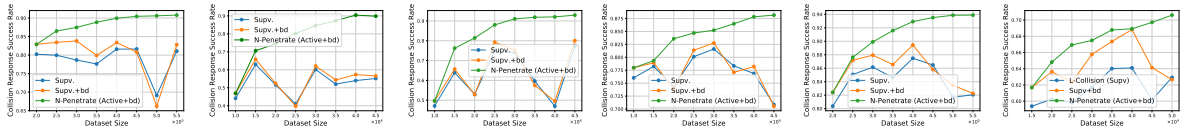


Figure 4.5: We plot the success rate of the neural collision handler against the dataset size. Our method resolves 22.73% more collisions than ***Supv+bd***. From left to right: SCAPE, Swing, Jump, Skirt, Hand, and AMASS.

4.4.2 Data Aggregation

The accuracy of our neural collision detector can be measured by the discrepancy between the surrogate decision boundary deemed by MLP_c and the true decision boundary of PD, formu-

lated as:

$$E_{\{Z_{\text{all}} \sim \mathcal{Z} | \text{PD}(Z_{\text{all}})=0\}} \left[\text{CE}(I_c(Z_{\text{all}}), I_c^*(Z_{\text{all}})) \right],$$

which is an expectation over the true decision boundary. Here CE is the cross-entropy loss. However, it is very difficult to derive a sampled approximation of the above metric because PD is a non-smooth function whose level-set is measure-zero, which corresponds to the boundaries of C-obstacles. Instead, we propose to take the expectation over the surrogate decision boundary:

$$E_{\{Z_{\text{all}} \sim \mathcal{Z} | \text{MLP}_c(Z_{\text{all}}, \theta_C)=0.5\}} \left[\text{CE}(I_c(Z_{\text{all}}), I_c^*(Z_{\text{all}})) \right].$$

Generally speaking, the 0.5-level-set of MLP_c can also be measure-zero, but we have designed our neural networks $D, \text{CSE}, \text{CP}, \text{MLP}_c$ to be differentiable functions. As a result, we could always project samples onto the 0.5-level-set by solving the following risk-seeking unconstrained optimization:

$$\underset{Z_{\text{all}}}{\textbf{argmin}} \ \frac{1}{2} \| \text{MLP}_c(Z_{\text{all}}, \theta_C) - 0.5 \|^2.$$

We adopt the quasi-Newton method and update Z_{all} using the following recursion:

$$Z_{\text{all}} - \bar{H}^{-1} \nabla \text{MLP}_c (\text{MLP}_c - 0.5), \tag{4.3}$$

where $\bar{H}$ is some first-order approximation of the Hessian matrix, which is much faster to compute than the exact Hessian, which requires the second-order term $\nabla^2 \text{MLP}_c$. In summary, we

would sample a new set of size $N_{\text{aug}}/2$ from previous $\mathcal{D}_c$ during each iteration of data augmentation. For each sampled Z_{all}, we project Z_{all} to the surrogate decision boundary using Eq. 4.3 recursively until the relative change within Z_{all} is smaller than ϵ_z between consecutive iterations. For datasets based on the SMPL model, we randomly choose to fix Z_β or not. We also sample $N_{\text{aug}}/2$ directly from $U(\mathcal{Z})$, using random samples to discover uncovered regions, which achieves a balance between exploitation and exploration. Finally, we classify Z_{all} into either one of $\mathcal{D}_{p,n,b}$, according to Eq. 4.2 using the penetration depth.

4.4.3 Model Update

After $\mathcal{D}_c$ has been updated, we fine-tune $\text{CSE}, \text{CP}, \text{MLP}_c$ by updating θ_C using the following loss functions:

$$\mathcal{L} = w_{\text{PD}}\mathcal{L}_{\text{PD}} + w_r\mathcal{L}_r + w_{\text{ce}}\mathcal{L}_{\text{ce}} + w_b\mathcal{L}_b,$$

where $w_\bullet$ are weights corresponding to each type of loss. Our first term $\mathcal{L}_{\text{PD}}$ is a regularization that enforces consistency between S_i and true PD, defined as:

$$\mathcal{L}_{\text{PD}} = E_{\{Z_{\text{all}} \in \mathcal{D}_C\}} \left[\sum_{i=1}^{|Z_0|} \|S_i - \text{PD}_i\|^2 + w_{\text{PDsum}} \| \sum_{i=1}^{|Z_0|} S_i - \text{PD}\|^2 \right],$$

where we penalize both the domain-decomposed penetration depth PD_i defined in Chapter 3 and the total penetration depth with weight w_{PDsum}. Our second term $\mathcal{L}_r$ is a marginal ranking loss that enforces the correct ordering of penetration depth to avoid over-fitting, defined as:

$$\mathcal{L}_r = E_{\{Z_{\text{all}}^{a,b} \in \mathcal{D}_C | \text{PD}^a < \text{PD}^b\}} \left[\max(0, \alpha - (\sum_{i=1}^{|Z_0|} S_i^a - \sum_{i=1}^{|Z_0|} S_i^b) \right],$$

where α is the maximal allowable order violation. We use superscripts to distinguish two samples drawn from $\mathcal{D}_C$. Note that the domain-decomposition is only used for bilevel autoencoders and omitted in the SMPL representation. Therefore, for SMPL, we only use the overall collision value S in the regression loss $\mathcal{L}_{\text{PD}}$ and the marginal ranking loss $\mathcal{L}_r$. There are no other terms for sub-domains. Our third term measures the discrepancy between MLP_c and PD over the entire latent space:

$$\mathcal{L}_{\text{ce}} = E_{\{Z_{\text{all}} \in \mathcal{D}_p \bigcup \mathcal{D}_n\}} \left[\text{CE}(I_c(Z_{\text{all}}), I_c^*(Z_{\text{all}})) \right].$$

The last term is performed on the new boundary set $\mathcal{D}_b$. We propose using the l_1-loss function for $\mathcal{D}_b$ to approximate the decision boundary:

$$\mathcal{L}_b = E_{\{Z_{\text{all}} \in \mathcal{D}_b\}} \left[\| \text{MLP}_c(Z_{\text{all}}, \theta_C) - 0.5 \| \right]. \tag{4.4}$$

We update our neural collision detector with objective function $\mathcal{L}$ by running a fixed number of training epochs, denoted as N_{epoch}, with θ_C warm-started from the last iteration of the model update.

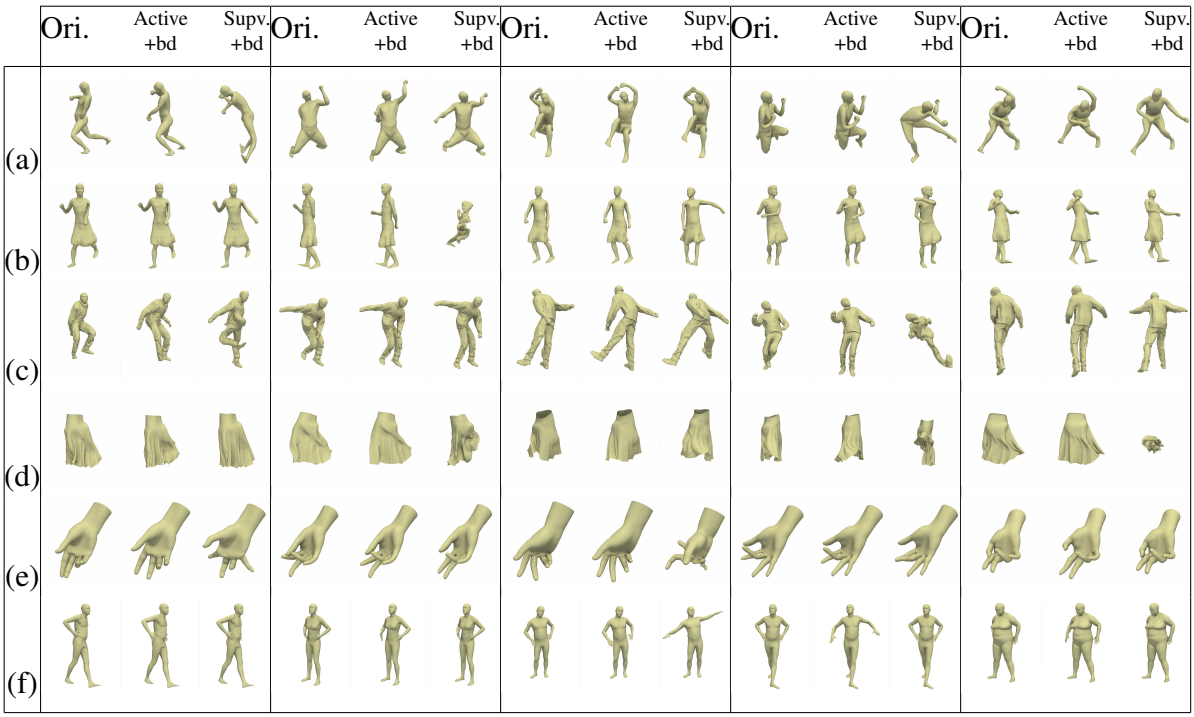


Figure 4.6: Representative examples of collision handling on the five datasets: (a) SCAPE; (b) MIT Swing; (c) MIT Jump; (d) Skirt; (e) Hand; (f) AMASS. For each example, we show the given self-penetrating mesh (left), our result (middle, ***Active+bd***), and ***Supv+bd*** (right). In comparison, our method can resolve more collisions, while we keep the output model closer to the input and reduce unnecessary deformations in ***Supv+bd***.

4.5 Evaluation

Datasets: We evaluate our method on six datasets. The first three (SCAPE [10] with $N = 71$ meshes each having 2161 vertices, MIT Swing [199] with $N = 150$ meshes each having 9971 vertices, and MIT Jump [199] with $N = 150$ meshes each having 10002 vertices) contain human bodies with different sets of actions and poses. The forth one is a skirt dataset introduced by [216] that contains $N = 201$ simulated skirt meshes synthesized by NVIDIA clothing tools, each of which has 2830 vertices. The skirt is deformable everywhere, and the dataset is rather small. Obtaining stable performance in this case is challenging and we observe reasonably good results using active learning. Our fifth one is a custom dataset of human hand poses. We captured

various hand poses and transitions between the poses in a multi-view capture system. We ran 3D reconstruction [45] and 3D keypoint detection [167] for the captured images and registered a linear blend skinning model consisting of 2825 vertices for each frame of the data [44], resulting in $N = 7314$ meshes. The first five datasets use the bilevel autoencoder for embedding. Our sixth dataset comes from the captured motion model repository AMASS [114], which uses SMPL to embed human body shapes. We choose one subset MPIMosh covering 19 subjects with different genders and body shapes. Suggested by [121], we sample at half of its original frame rate and get $N = 54387$ meshes each having 6890 vertices.

	SCAPE	Swing	Jump	Skirt	Hand	AMASS
N_{init}	200000	10000	10000	5000	200000	15000
N_{aug}	50000	5000	5000	5000	50000	5000

Table 4.1: N_{init} and N_{aug} used by each dataset.

Implementation: We implement our method using PyTorch and perform experiments on a desktop machine with an NVIDIA RTX 2080Ti GPU. If the bilevel autoencoder is used, we begin by training $\langle \theta_E, \theta_D \rangle$ using Adam with a learning rate of 0.01 and a batch size of 128 over 3000 epochs. When we train the embedding network, we empirically use 10 sub-domains for the SCAPE and Skirt datasets, and 12 for the Swing, Jump, and Hand datasets. For neural collision detector training, we choose the following hyper-parameters: $\epsilon = 1 \times 10^{-4}, w_{\text{PD}} = 5, w_{\text{PDsum}} = 0.2, w_r = 2, w_{ce} = 2, w_b = 0.5$. We bootstrap by supervised learning θ_C on N_{init} data points. We progressively inject data points into N_{init} until the "elbow point" of accuracy vs. sample size is reached, which is detected using [161]. For each experiment, we train θ_C using Adam with a learning rate of 0.001 and a batch size of 512 over $N_{\text{epoch}} = 100$ epochs. We choose suitable N_{aug} according to N_{init}. The N_{init} and N_{aug} used for each dataset are summarized in Table 4.1. During

data aggregation, we terminate Newton's method when the relative changes of Z_{all} are less than $\epsilon_z = 10^{-7}$. For each subsequent iteration of active data augmentation, we fine-tune θ_C using Adam and adjust the learning rate, batch size, and N_{epoch} using the performance on the validation set. For collision handling, we run ALM until Eq. 4.1 is satisfied.

Collision Detection: We compare our method with two baseline algorithms. The first one (as Chapter 3 shows, denoted as ***Supv***) trains θ_C using supervised learning, where $\mathcal{D}_c$ is constructed by randomly sampled poses from $U(\mathcal{Z})$ and boundary set D_b where the associated loss Eq. 4.4 mentioned in Sec. 4.4.1 is not used, i.e., $w_b = 0$. Our second baseline (denoted as ***Supv + bd***) also uses supervised learning, but the boundary set and loss in Eq. 4.4 are used. Our proposed method, N-Penetrate, is denoted as ***Active + bd***. After k iterations of active data augmentations, we have a dataset with $N_{init} + kN_{aug}$ points for training θ_C. For fairness, we train our two baselines using $N_{init} + kN_{aug}$ points randomly sampled from $U(\mathcal{Z})$ for each iteration. For all the methods, we use 80% of the data for training and the rest are used as a validation set for hyperparameter tuning (learning rate, batch size, and N_{epoch}). For each dataset, we create a test set with 7.5×10^5 samples from $U(\mathcal{Z})$, which is unseen in the training stage, to evaluate the performances. The performances of neural collision detectors are evaluated based on two metrics: the fraction of successful predicates (accuracy) and the fraction of times a self-penetrating mesh is erroneously predicted as collision-free (false negative rate). False negatives are more detrimental to our applications than false positives as we want to detect and eliminate colliding samples using our collision handler, while we can tolerate a few false positive samples. As illustrated in Fig. 4.3 and Fig. 4.4, our method effectively improves both metrics. The performance after active learning is summarized in Table 4.2. We reach an accuracy of $85.6 - 98.1\%$ compared with the groundtruth generated by the exact method [121, 132].

metric	SCAPE	Swing	Jump	Skirt	Hand	AMASS
final dataset size	5.5×10^5	4.5×10^4	4.5×10^4	4×10^4	5.5×10^5	5×10^4
accuracy (Ours (***Active+bd***))	0.9383	0.9638	0.9552	0.9817	0.9692	0.8563
accuracy (***Supv+bd***)	0.9282	0.9609	0.9500	0.9795	0.9650	0.8551
accuracy (***Supv***)	0.9181	0.9460	0.9347	0.9660	0.9558	0.8526
false neg. (Ours (***Active+bd***))	0.05151	0.01485	0.02573	0.01808	0.03582	0.17656
false neg. (***Supv+bd***)	0.05576	0.01766	0.02644	0.01956	0.03652	0.18422
false neg. (***Supv***)	0.06914	0.01969	0.02713	0.02056	0.03727	0.19654
equi. dataset size (***Supv+bd***)	6.63×10^5	7.64×10^4	7.02×10^4	7.71×10^4	7.30×10^5	7.40×10^4

Table 4.2: We summarize the accuracy and false negative rate of three methods under comparison. We also include the equivalent dataset size for the baseline to reach the same performance as our method.

On average, our method achieves 1.62% higher accuracy and a 13.59% lower false negative rate than ***Supv***. In the last row of Table 4.2, we measure an equivalent dataset size, which is defined as the size of the dataset needed by ***Supv+bd*** to achieve the same accuracy as our method. We derive this number by interpolating on experimental results of ***Supv+bd***. The results show that our method achieves a similar accuracy using a 34.6% smaller dataset than ***Supv+bd*** on average.

Cross-dataset validation: We can use a pre-trained collision detector module, if the representation is compatible with our mesh embedding module (i.e. use the same autoencoder architecture or SMPL/LBS models). For example, we use the AMASS-MPIMosh subset to initialize the sampling range for our augmented dataset and train the collision predictor for SMPL. We evaluate the performance of the trained predictor on the AMASS-HDM05 dataset for cross-dataset validation and summarize the results in Table 4.3. This new test set has a total of 520276 models, of which 49879 have self-collisions. This is $10\times$ larger than the AMASS-MPIMosh dataset. Overall, our method can explore the entire latent space and obtain good prediction accuracy on challenging unseen datasets through active learning.

Metric	***Supv***	***Supv+bd***	Ours (***Active+bd***)
Accuracy	0.7235	0.7622	0.7950
False Negative Rate	0.3018	0.2803	0.2706

Table 4.3: Cross-dataset Validation on the AMASS-HDM05 dataset using pre-trained model for AMASS-MPIMosh.

Running Time: We have compared our neural collision detector with the BVH-based exact method provided as part of the FCL library [132] on a dataset with 1×10^4 meshes, which is unseen by our learning method. We compared the CPU-based implementations on an Intel Xeon Silver 4208 CPU (32 cores). Our neural method is implemented using PyTorch and we evaluate both the CPU and GPU versions. We compare to a parallel and tuned version of the BVH-based algorithm in FCL using 15 threads. We can achieve a $29 - 124\times$ speedup over the BVH-based collision detection (Table 4.4).

	SCAPE	Swing	Jump	Skirt	Hand
FCL (parallel BVH-based collision checker)	37.84s	134.1s	135.7s	43.82s	42.7s
Ours (CPU)	1.05s	1.20s	1.16s	1.13s	1.46s
Ours (GPU)	0.922s	1.14s	1.09s	0.956s	1.32s
Speedup (CPU)	36.03	111.75	116.98	38.78	29.25
Speedup (GPU)	41.04	117.62	124.50	45.84	32.35

Table 4.4: Comparing our neural method with exact collision checking in FCL. Our method achieves up to a $124\times$ speedup.

Collision Handling: We plug the trained neural collision detectors into ALM and compare our method, ***Supv***, against ***Supv+bd*** in terms of resolving self-penetrating meshes. To this end, we randomly sample 10000 self-penetrating, unseen Z_{all}^{user} from $U(\mathcal{Z})$, and use Eq. 4.1 to derive Z_{all}. For the SMPL model, we will fix Z_β in the optimization to maintain the body shape for each sample. We compare the performance based on relative PD reduction defined as:

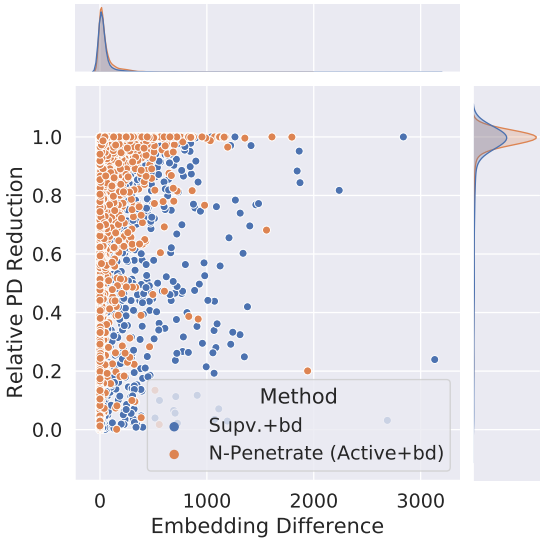


Figure 4.7: We plot the joint distribution of relative PD reduction (Y-axis) and embedding difference (X-axis) over successfully collision-handled test meshes in the SCAPE dataset for our method and ***Supv+bd***. Our method resolves more collisions (average PD reduction 94.81% vs. 88.76%) while remaining closer to the input (average embedding difference 56.17 vs. 66.11) as compared with ***Supv+bd***.

$$\frac{\mathrm{PD}(\mathrm{ACAP}^{-1}(Z_{\mathrm{all}}^{user}, \theta_D)) - \mathrm{PD}(\mathrm{ACAP}^{-1}(Z_{\mathrm{all}}, \theta_D))}{\mathrm{PD}(\mathrm{ACAP}^{-1}(Z_{\mathrm{all}}^{user}, \theta_D))}.$$

Collision resolution is completely successful if this value equals one, which may not always happen because ALM uses soft penalties to relax hard constraints. Thus, we consider a solution successful if the value is greater than 0. We plot the success rate against the dataset size in Fig. 4.5, which shows that our method resolves 22.73% more collisions than ***Supv+bd***. Thanks to our risk-seeking data aggregation method, our method monotonically improves the collision handling success rate when more data points are injected, while ***Supv+bd*** exhibits unstable performance. Since ***Supv*** uses the same randomly sampled dataset as ***Supv+bd***, the performance exhibits similar instability. Meanwhile, our novel boundary loss improves the results for ***Supv+bd***,

since it can better approximate the decision boundary. Another criterion for good collision handling is the embedding difference – the objective function in Eq. 4.1. We want the output to be as close to the input as possible. We plot the relative PD reduction vs. embedding difference over successfully collision-handled test meshes in the SCAPE dataset for our method and ***Supv+bd*** in Fig. 4.7. The mean relative PD reduction for our method is 94.81% and the mean embedding difference is 56.17, compared to 88.76% and 66.11, respectively, for ***Supv+bd***. The results show that our method resolves more collisions while the outputs stay closer to the input latent codes. Some exemplary results are shown in Fig. 4.6.

4.6 Conclusion & Limitations

We present an active learning method for training a neural collision detector in which training data are progressively sampled from the learned latent space using a risk-seeking approach. Our approach is designed for general 3D deformable meshes, and we highlight its benefits on many complex datasets. In practice, our method outperforms supervised learning in terms of accuracy, false negative rate, and stability. As a major limitation, our collision handler does not consider physics models. Physics models can be incorporated in the future via a learning-based physics simulation approach such as [34, 227]. The design of an appropriate mesh embedding module for collision handling is another challenge. Currently, we empirically choose network architectures and parameters, e.g., the number of sub-domains for different mesh types. It would be useful to extend our method to handle collisions between multi-objects by concatenating the representations of all the objects. For example, we can predict the collisions for human-environment interactions [63, 207]. We are also interested in extending these ideas to other formulations, e.g.,

part-based occupancy networks [119] or SDF networks [7]. It would also be useful to combine these optimizations with learning methods [87] and design a combined formulation for predicting high-fidelity human models, where our learned collision constraints could be directly used to handle self-collisions.

Chapter 5: A Repulsive Force Unit for Garment Collision Handling in Neural Networks

5.1 Introduction

Predicting how a 3D garment deforms in response to the underlying 3D body motion is essential for many applications, including realistically dressed human body reconstruction [22], interactive garment design [205], virtual try-on [159], and robotics control [181]. To generate accurate cloth deformations, most techniques are based on physically-based simulation (PBS). Common physically-based models include the mass-spring system [12, 30], the finite element approach [90, 124], the thin-shell model [54], etc. However, these methods tend to be computationally intensive since they typically involve solving large linear systems and handling collisions. In particular, robust collision handling based on collision detection and response computation is

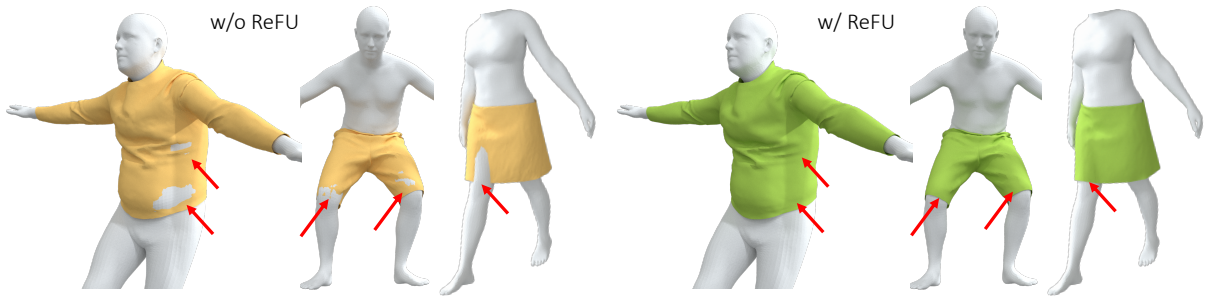


Figure 5.1: Collisions solved by applying ReFU in garment prediction neural networks. Our approach reduces the number of artifacts.

a critical component of cloth or garment simulation. Even a single missed collision can considerably affect the accuracy of the overall simulator [27, 53]. The most accurate physically-based simulators run at 0.5 seconds per frame on commodity GPUs [189], where collision handling can take 50-80$\%$ of total simulation time. As a result, these simulators are unable to provide real-time performance for interactive applications such as gaming and virtual try-on.

Machine learning methods provide a promising direction to dramatically reduce the computational cost of cloth simulators. Hence, in recent years, various neural network methods have been proposed to predict 3D cloth deformations. However, a common setback of such methods is the lack of efficient handling of collisions between the garments and the body surface as shown in the yellow garments in Fig. 5.1. In our experiments (Sec. 5.4), we observe that only 49% of garments predicted from TailorNet [138], a state-of-the-art neural network based 3D garment prediction method, are collision-free. For some tight clothes like shirts, only 12% of models are collision-free. Thus, the resulting state of the cloth mesh can collide with the body mesh, which affects the reliability and usefulness of these methods for many applications related to rendering, simulation, and animation [28, 144, 200]. As a result, it is important to design learning methods that can significantly reduce or eliminate such collisions.

One option to address the body-cloth collision problem is to perform post processing optimizations [58]. However, these optimization approaches can take considerable CPU time (around 0.6-0.8s per frame), which can be too expensive for interactive applications. A more common practice is to apply specialized collision loss functions during training [19–21, 59, 159]. However, this only provides a soft constraint to avoid collisions for network training, and the network still cannot handle the penetrated vertices when collisions happen during inference.

Main Results: To let the network learn to solve the collisions through inference, we propose

a novel neural network layer called Repulsive Force Unit (ReFU). ReFU is fully differentiable and can be plugged into different garment prediction backbone networks, trained either through fine-tuning or from scratch.

Our design of ReFU is inspired by physically-based simulators [41, 113, 190], which use a scheme that collects repulsive forces, friction forces, and adhesion forces as part of time integration. Our goal is to design a learning scheme that can model the effects of repulsive forces and can easily cope with existing 3D garment prediction networks. We compute the force based on the implicit field of the body geometry to quickly detect the set of penetrated garment vertices and the repulsive direction. The repulsive strength is predicted by the neural network inside the ReFU layer. Instead of simply pushing the problematic garment vertices to the body surface, ReFU applies a flexible offset to move them. This improves the overall collision handling performance, avoids additional Edge-Edge (EE) collisions that normally cannot be detected by the signed distance of the vertices, and overcomes the artifacts in the estimated implicit functions of the human body. To achieve real-time performance for the whole garment prediction system, we leverage the power of neural implicit surface representation and use a neural network to quickly estimate the approximate Signed Distance Function (SDF) of the human body under different poses [55].

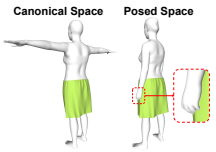
To evaluate ReFU with different backbones, we train it with TailorNet [138], the state-of-the-art 3D garment prediction network, and a 3D mesh convolutional neural network [230]. Our experiments show that backbone networks trained with ReFU can significantly reduce the number of body-cloth collisions while achieving real-time performance for the whole garment prediction system during testing. The ReFU layer and SDF network will only add 2 milliseconds of inference time to the overall system. Overall, our method achieves much better results in terms

of the number of interpenetrating vertices, the number of collision-free 3D garment models, and the reconstruction error of the generated garments. Compared to prior learning-based methods, the use of ReFU results in garment meshes with fewer artifacts and higher visual quality.

5.2 Related Work

Cloth Prediction using Machine Learning: Many fast techniques have been used to predict cloth deformation in 3D. These include simple linear models such as [36, 58] and motion graph methods [81]. More recent works use neural networks [19–21, 59, 138, 159, 160]. Many of these learning methods have been designed for SMPL-based [107] parametric obstacle models, including human shapes. Other methods are designed for general triangle mesh-based obstacles [68]. The GarNet network architecture [59] can predict the cloth deformation from the target posture with DQS pre-processing. However, these learning methods do not explicitly account for cloth-obstacle collisions.

Learning-Based Collision Handling: Several techniques have been proposed to handle collisions in machine learning methods. In Chapter 3, we estimated a collision-free subspace for 3D human models, but it is not practical to compute a similar subspace for deforming garments as they have a lot of degrees of freedom. Its performance can be improved using active learning, as shown in Chapter 4, but still, this approach is limited by the use of numerical optimization algorithm. These methods can't be combined with general garment prediction backbone networks. Many recent works [19–21, 59] use collision loss to penalize penetrated garment-body pairs during training. However, these methods have no component or feature in the network that can resolve these penetrations during inference.



Recently, Santesteban et al. [160] proposed a self-supervised collision handling method, but it requires strict additional restrictions on the training data, i.e., that both garments in the global space and the canonical space must be collision-free. However, most public garment datasets cannot satisfy this restriction. We show an example on the left from TailorNet dataset which is collision-free in canonical space, but not in the posed space.

To solve the collision problems for the testing set, a simple approach is to perform post-processing on the predicted cloth by detecting the vertices inside the human body and moving them directly to the nearest point on the body surface [68, 159]. However, there are two issues with these approaches: first, computing nearest points on the body surface is time-consuming; second, simply moving the garment vertices may generate abrupt cloth movements and lose some properties of the original garment.

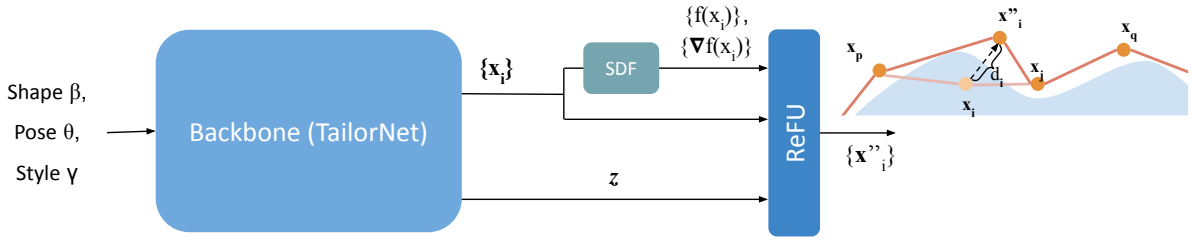


Figure 5.2: A garment inference model with ReFU: Given body shape $\vec{\beta}$, pose $\vec{\theta}$, and garment style $\vec{\gamma}$ parameters, the backbone network is used generate a deformed garment with potential body-cloth collisions. Our ReFU layer is attached to the backbone and processes every point $\{\boldsymbol{x}_i\}_{i=1}^{N}$ in the garment. This layer first checks whether the point is inside the human body or not based on the SDF value $f(\boldsymbol{x}_i)$. If it is outside, ReFU directly outputs the point. Otherwise, ReFU will apply a repulsive force along the direction of the gradient of the SDF $\nabla f(\boldsymbol{x}_i)$ with a predicted amount of movement d_i. Finally, we collect all the vertices passed through the ReFU layer and obtain a 3D garment mesh with fewer collisions.

5.3 Collision Handling using ReFU

In this section, we will first explain the formulation of the Repulsive Force Unit (ReFU) (Sec. 5.3.1) then describe how to apply and train it in the garment prediction backbone network as an additional network layer using the example of TailorNet [138] (Sec. 5.3.2). Finally, we will present how we train a neural network to quickly estimate the SDF for the human body (Sec. 5.3.3). Our overall learning-based pipeline is shown in Fig. 5.2. With a neural network-based SDF approximation, our system can achieve real-time performance for garment prediction given the body pose and shape parameters.

5.3.1 ReFU: Repulsive Force Unit

The goal of body-cloth collision handling is to find the penetrating garment vertices and move them to the proper positions to resolve the collision while preserving original wrinkles and other details on the garments. Let N_c be the number of vertices that need to be moved. The degrees of freedom of this set of vertices result in a large solution space of dimension $3N_c$. Our goal is to reduce the dimension of the solution space. Inspired by prior work in physically-based simulation [41, 190], we design the Repulsive Force Unit (ReFU) to move the vertices only along a *repulsion* direction, which is toward the closest point on the body surface. Our formulation of ReFU can be seen as applying a virtual repulsive force to move the penetrated vertex outside the human body.

To find the repulsion direction, ReFU uses the implicit representation of the body, i.e., the signed distance function (SDF). Given a query point $\boldsymbol{x}$, the SDF function f returns its distance to the closest point on the corresponding surface, and its sign is associated with whether the point

is inside (negative) or outside (positive) the surface:

$$f(\boldsymbol{x}) = s, \quad \boldsymbol{x} \in \mathbb{R}^3, s \in \mathbb{R}. \tag{5.1}$$

The zero-level set of $f(\boldsymbol{x})$ indicates the surface.

Given the nature of SDF, we can quickly determine whether a vertex $\boldsymbol{x}_i$ on the garment mesh is inside or outside the body. For $\boldsymbol{x}_i$ with negative SDF value, the gradient of the SDF at $\boldsymbol{x}_i$ is pointing towards the nearest point on the surface along the normal direction. Thus, we formulate ReFU as

$$\text{ReFU}(\boldsymbol{x}_i) = \begin{cases} \boldsymbol{x}_i - d_i \widehat{\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}_i)}, & f(\boldsymbol{x}_i) < 0; \\ \boldsymbol{x}_i, & otherwise, \end{cases} \tag{5.2}$$

where d_i is a predicted offset scalar indicating the amount of movement, and $\widehat{\nabla_{\boldsymbol{x}} f(\cdot)}$ is the normalized gradient of f at $\boldsymbol{x}$, indicating the direction of movement, calculated as below:

$$\widehat{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})} = \frac{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})}{\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2} \tag{5.3}$$

Although the gradient of accurate SDF should be a unit vector [32, 55], the approximated SDF by neural networks [111, 169] might not strictly satisfy this property. Thus, we need to normalize the gradient in practice.

5.3.1.1 A Learned Moving Offset

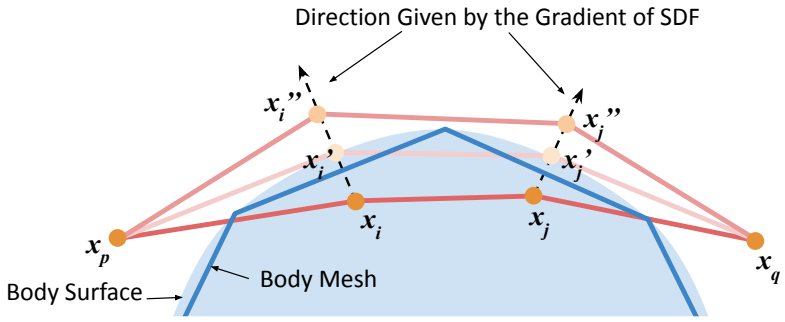


Figure 5.3: We show an example of resolving vertex-face (VF) and edge-edge (EE) collisions. Assume the blue polyline is a part of the human body mesh, and the light blue region is the body represented by the SDF estimator f. We show four garment vertices $\boldsymbol{x}_{i,j,p,q}$, where $\boldsymbol{x}_{i,j}$ are inside the human body and $\boldsymbol{x}_{p,q}$ are outside of it. The dotted arrows are the repulsive force directions given by the gradient of f. If we set $\alpha_{i,j} = 1$ and the moving offset to be $|f(\boldsymbol{x}_{i,j})|$, the vertices will moved to $\boldsymbol{x}'_{i,j}$. While the collision along the edges $\overline{\boldsymbol{x}_p\boldsymbol{x}'_i}$ and $\overline{\boldsymbol{x}_q\boldsymbol{x}'_j}$ are resolved, the edge $\overline{\boldsymbol{x}'_i\boldsymbol{x}'_j}$ will still induce a collision. By allowing $\alpha_{i,j} >= 1$, we can move the vertices to $\boldsymbol{x}''_{i,j}$, resolving all the VF and EE collisions.

A straightforward way to decide the moving offset is to use the SDF value directly. However, as pointed out in the context of physically-based simulators [189], this is only guaranteed to solve the Vertex-Face (VF) collisions, but not the Edge-Edge (EE) collisions. As shown in Figure 5.3, we push the two neighboring garment vertices further outside to resolve the EE collisions. To compute this extra offset based on the mesh representation, we need to use a global optimizer in an iterative manner. However, this computation can be time consuming and the resulting configuration may not match the groundtruth. Instead, we use neural networks to predict α_i, the scale of movement, and multiply it with the SDF value to compute the final offset as:

$$d_i = \alpha_i f(\boldsymbol{x}_i), \alpha \geq 0. \tag{5.4}$$

α_i is predicted based on the global latent feature $\boldsymbol{z}$ of the whole garment and the SDF value of vertex $\boldsymbol{x}_i$.

$$\alpha_i = g(k(\boldsymbol{z})_i, f(\boldsymbol{x}_i)), \boldsymbol{z} \in \mathbb{R}^M, \tag{5.5}$$

where $k : \mathbb{R}^M \rightarrow \mathbb{R}^{N \times D}$ is a topology-dependent Multilayer Perception (MLP) network that infers the latent vector for every vertex from the global feature $\boldsymbol{z}$, and $k(\boldsymbol{z})_i \in \mathbb{R}^D$ is for i-th vertex $\boldsymbol{x}_i$. g is another MLP that outputs the movement scale for $\boldsymbol{x}_i$. Both $g(\cdot, \cdot)$ and $k(\cdot)$ are jointly trained with the backbone network in an end-to-end manner. We choose $M = 1024$ and $D = 10$ for our experiments.

The global garment latent vector $\boldsymbol{z}$ can be obtained from the backbone network. In terms of using TailorNet as the backbone network, $\boldsymbol{z}$ is computed from the predefined body pose and shape parameters $\vec{\beta}, \vec{\theta}$ and the garment style parameter $\vec{\gamma}$ (as formulated in Tailornet [138] for the definition of $\vec{\beta}$, $\vec{\theta}$, and $\vec{\gamma}$) with an MLP function h.

$$\boldsymbol{z} = h(\vec{\beta}, \vec{\theta}, \vec{\gamma}). \tag{5.6}$$

When the accurate groundtruth SDF is given, the output range of g can be set as $[1, +\infty)$ so it ensures all penetrated vertices will be pushed outside. When using a neural network based approximate SDF, we relax the range to be $[0, +\infty)$ to account for inaccuracies in the SDF prediction. Implementation details of the MLPs are given in Sec. 5.4.2.

Using the learned offset has several benefits. First, a flexible extent can handle the collision more naturally and let the moved vertices blend more smoothly with neighboring vertices. Second, it can help resolve additional Edge-Edge (EE) collisions. Finally, the flexible offset can cope with the inaccuracies of the neural network approximated SDF. For example, when the absolute SDF value predicted by $f(x)$ is smaller than the ground truth, directly using the $f(x)$ as the offset could leave the penetrated vertex inside the human body. On the other hand, if $f(x)$ predicts larger distance than the ground truth, the resulting vertex may be pushed too far away from the human body, resulting in a "pump out" artifact, as shown in Fig. 5.5. However, when using ReFU during training, the backbone network and the MLPs in the ReFU layer will foresee the quality of the SDF function. As a result, it will learn to adjust the prediction of both x_i and α_i and thereby result in a more accurate final output garment with fewer collisions.

5.3.2 Train Backbone with ReFU

ReFU can be easily plugged into current neural network frameworks for garment prediction. Here we show how to train the backbone network with ReFU with the example of TailorNet [138]. When using ReFU, we assume that the colliding vertices are not far from the body surface so that the repulsive force can be estimated through the SDF. Thus, we train the ReFU layer with the backbone network in the fine-tuning stage. One could also train the ReFU with the garment network from scratch.

The original TailorNet trains different frequency focused components separately to allow the different components realize the difference between low-frequency and high-frequency deformations. However, the groundtruth representation of the garment, obtained after the frequency

division, may not satisfy the collision-free condition, i.e., the high-frequency pieces may have deeper penetrations inside the human body. If we plug in ReFU after computing the high-frequency output, the new vertices will all be moved outside the body, and their coordinates will be different from their high-frequency groundtruth. Thus, we propose training ReFU as a finetuning process together with all the components of TailorNet, which have the summation of all the frequencies with no collisions in the groundtruth data.

We attached a ReFU layer to the end of the pre-trained TailorNet so that it receives the raw output of $\{\boldsymbol{x}_i\}_{i=1}^N$. Assuming the predicted garment vertex positions after the ReFU layer is $\{\boldsymbol{x}'_i\}_{i=1}^N$ and the corresponding groundtruth is $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^N$, we use the following loss terms to train the backbone network and the ReFU layer:

$$\mathcal{L} = \lambda_1 \mathcal{L}_r + \lambda_2 \mathcal{L}_c, \tag{5.7}$$

$$\mathcal{L}_r = \sum_{i=1}^{N} \|\boldsymbol{x}'_i - \tilde{\boldsymbol{x}}_i\|_2^2, \tag{5.8}$$

$$\mathcal{L}_c = \sum_{i=1}^{N} |\max(-f(\boldsymbol{x}'_i), 0)|, \tag{5.9}$$

where $\mathcal{L}_r$ is the reconstruction loss and $\mathcal{L}_c$ is the collision loss to cover missed penetrated vertices. $\lambda_{1,2}$ are weights to balance the loss terms.

We train the network with groundtruth collision-free garment data so the reconstruction loss will guide the prediction of the $\boldsymbol{x}_i$ and α_i to move $\boldsymbol{x}'_i$ to the position with no EE collisions. In this manner, it better preserves the local smoothness and details.

It turns out that the post-processing methods in previous works [68, 159] can only move

the vertex along the gradient direction of the SDF of $\boldsymbol{x}_i$. With our method, however, although the adjustment space of each collided vertex is also one degree-of-freedom (DOF) during inference, through the training, both the backbone and the ReFU layer are fine-tuned so the adjustment space extends to 4 DOFs, i.e., the movement scale α_i and the original network output $\boldsymbol{x}_i$. With higher adjustment capability, our method achieves performance on par with complex optimization-based post-processing such as [58]. Detailed comparisons with previous methods are described in Sec. 5.4.5.

5.3.3 Neural Network for Body SDF

In physically-based simulation methods, the SDF values are computed using analytical methods and accelerated using spatial data structures like KD-trees. Even with these acceleration data structures, the SDF computation is far from real-time. Recent works [55, 135] show that the implicit function of a 3D geometry can be approximated by a neural network. As a result, one can use the trained network to quickly estimate the SDF values of a 3D point set. We design the network to predict SDF conditioned on the SMPL [107] parameters. To train a generalized SDF network that can predict the implicit function of human bodies with different shapes and poses in real-time, we design the network to predict SDF conditioned on the SMPL [107] parameters.

$$f(\boldsymbol{x}, \vec{\beta}, \vec{\theta}) \approx \text{SDF}^{M(\vec{\beta},\vec{\theta})}(x). \tag{5.10}$$

SMPL is a PCA model computed from a large human shape data. $\vec{\beta}$ and $\vec{\theta}$ are its shape and pose parameters. $M(\vec{\beta}, \vec{\theta})$ is the human shape reconstructed from $\vec{\beta}$ and $\vec{\theta}$.

To train the SDF network, we combine both the regression loss on sampled points in the

space and the geometric regularization loss on the gradient as proposed by Park et al. and Gropp et al. [55, 135]. For each garment-body pair in the TailorNet dataset, we collect three categories of SDF value samples:

1. Randomly sampled points from the body surface, with or without Gaussian disturbance. For samples right on the body surface, we also collect their normals. Note that, we can only get correct SDF gradients for the surface points which are their normals. For other points, we can estimate their gradients through analytic methods.

2. Randomly sampled points from the garment surface, with or without Gaussian disturbance.

3. Randomly sampled points inside the bounding box of the body. We use a general bounding box for all the samples with size $4m \times 4m \times 4m$, centering at $[0, 0, 0]$.

For points from the body surface without disturbance, we denote them as $\{\boldsymbol{x}_i\}_{i \in I_S}$, their normals as $\{\boldsymbol{n}_i\}_{i \in I_S}$. For other points, we denote them as $\{\boldsymbol{x}_j\}_{j \in I_E}$. The ground truth SDF values for all the points are $\{s_i\}_{i \in I_S \cup I_E}$. We compute the loss for training SDF as:

$$\mathcal{L}_{SDF} = \lambda_a \mathcal{L}_v + \lambda_b \mathcal{L}_{sg} + \lambda_c \mathcal{L}_{se} \tag{5.11}$$

$$\mathcal{L}_v = \mathbb{E}_{i \in I_S \cup I_E}(|f(\boldsymbol{x}_i) - s_i|) \tag{5.12}$$

$$\mathcal{L}_{sg} = \mathbb{E}_{i \in I_S}(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i) - \boldsymbol{n}_i\|) \tag{5.13}$$

$$\mathcal{L}_{se} = \mathbb{E}_{i \in I_E}(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i)\| - 1)^2, \tag{5.14}$$

where $\mathcal{L}_v$ is a regression loss for the values [135], $\mathcal{L}_{sg}$ and $\mathcal{L}_{se}$ are losses for the gradients [55]. Specifically, $\mathcal{L}_{se}$ is based on the Eikonal equation [32]. We set the weights to balance each term as $\lambda_a = 2, \lambda_b = 1, \lambda_c = 0.1$.

We include the performance for the approximated SDF on the datasets we used in Table 5.1. We use two metrics:

Mean Absolute Error defined in Eq. 5.12;

Mean Relative Error defined as

$$\mathbb{E}_{i \in I_S \cup I_E} \left(\left| \frac{f(\boldsymbol{x}_i) - s_i}{s_i} \right| \cdot 100\% \right). \tag{5.15}$$

Using those loss functions, we can have supervision on the absolute values for the SDF samples, but no supervision on the norm of the gradient for vertices that are not on the body surfaces. Consequently, the mean relative error is much worse than the mean absolute error. Thus, in the main paper, we use the predicted offset scale to help ReFU improve its collision handling ability using the approximated SDF.

Dataset	Mean Absolute Error	Mean Relative Error
Shirt Male	2.38mm	28.22%
T-shirt Male	2.37mm	25.85%
Short-pant Male	2.46mm	31.66%
Skirt Female	3.10mm	32.64%

Table 5.1: Mean absolute error and mean relative error of the SDF network.

5.4 Experimental Results

In this section, we will quantitatively and qualitatively evaluate our method. Given our goal of achieving real-time performance for the whole garment prediction system, we use an approximated neural network predicted SDF (abbreviated as Approx. SDF) and combine it with ReFU. We also want to see the upper bound of our method, so we combine with an approach that

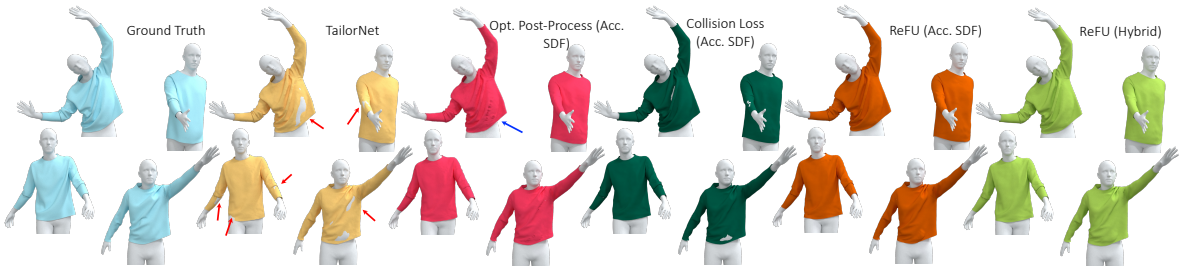


Figure 5.4: Benefits of ReFU: The baseline models from TailorNet have collisions (red arrow), optimization-based post-processing results in non-smooth regions (blue arrow); collision loss cannot resolve the collisions well; our method based on ReFU resolves the collisions and preserves the original shape with plausible wrinkles in both offline (using accurate SDF) and real-time ("Hybrid") modes. The results obtained using approximated SDF ("Hybrid") are similar to those obtained using accurate SDF.

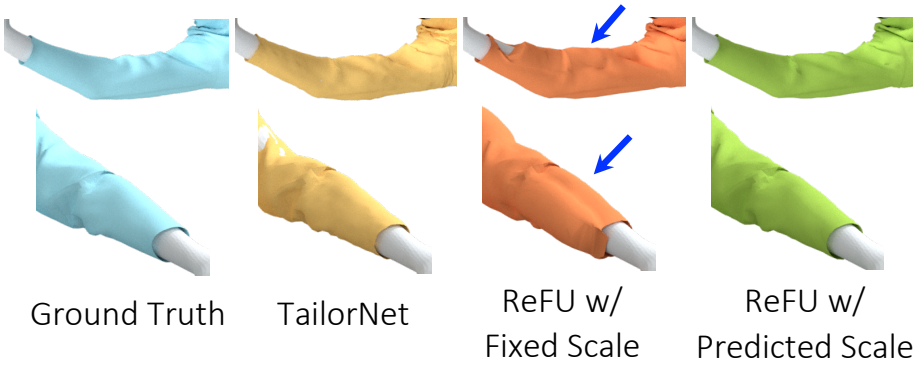


Figure 5.5: We zoom in to highlight the "pump out" artifacts with fixed moving scale and approximated SDF. When $\alpha_i = 1$, the ReFU layer may wrongly push out some regions (blue arrow) with higher approximated SDF values than the ground truth. However, with predicted scale, the network learns how to cope with the inaccuracies of the approximated SDF and generates smooth results. This also reduces the number of VF and EE collisions (Table 3).

computes accurate SDF (abbreviated as Acc. SDF).

5.4.1 Datasets, Metrics, and Settings

We utilize the datasets from TailorNet [138] to evaluate our method. We use an exact collision detection algorithm available as part of FCL [132] to select a subset of the garments from different genders and garment types that have no garment-body collisions or any self-penetrations in the 3D garment mesh. Except for this modification, we use the same train-test split as in

Dataset	# of vertices	# of Training Models	# of Testing Models
Shirt Male	9723	28360	3197
T-shirt Male	7702	21097	2688
Short-pant Male	2710	14555	2691
Skirt Female	7130	15664	3525

Table 5.2: We highlight the details of the datasets used to evaluate our approach. We have selected collision-free subsets from the datasets in [138]. The resulting datasets include different genders and garment types.

[138] for both garment networks and SDF networks on different garment types to perform a fair comparison. The resulting dataset is summarized in Table 5.2. "All garments" in Table 5.4 correspond to the weighted-average performance for four types of datasets.

We use the following metrics in our comparisons with prior methods:

MPVE [140] (Mean per-vertex error): Euclidean distance between the ground-truth and predicted garment vertices. It indicates the reconstruction error of the predicted garments. We use millimeters as the underlying unit.

VFCP [21] (Vertex-face collision percentage): The percentage of vertices on the garment that are inside the body surface.

CFMP (Collision-free models percentage): The percentage of garment models that are body-cloth collision-free in both types of VF and EE collisions.

For training and testing, we have three settings:

Approx. SDF: Always use the neural network approximated SDF for training, testing and post-processing.

Acc. SDF: Always use the accurate SDF for training, testing and post-processing. For discretized 3D models such as the human body represented by the SMPL parameters, we compute the SDF value for one query point. This accurate formulation can compute closest point/face and

returns the distance on the normal direction, thought it is time consuming.

Hybrid: Use the accurate SDF for computing the collision loss during training but the approximate SDF for ReFU during both training and testing.

5.4.2 Implementation

We implement our method using PyTorch [137]. All the training and testing are performed on a server machine with a 96-core CPU, 740GB memory, and 4 NVIDIA V100 GPU with 32GB memory. To estimate the body surface SDF, we design the neural network f with nine hidden layers each with 1024 neurons. Between each layer, we use a Softplus activation layer [226] with $\beta = 100$. We include a skip connection in the fourth layer, i.e., concatenating the query point coordinate with the hidden vector. To train f, we feed a batch with 32 human body models each with SDF value of 4000 random sampled points. For the network predicting the scale α_i, we use three 1024-dimension layer for h, one $N \times 10$-dimension layers for k, and two 10-dimension layers for g. They all use ReLU as the activation layer in between. We set the weights of the loss terms as $\lambda_1 = 1.5$ and $\lambda_2 = 0.5$. For all the training, we use an Adam optimizer with a learning rate of 1×10^{-5}.

5.4.3 Performance

We first evaluate how ReFU performs when using the accurate body SDF during both training and testing. As listed in Table 5.4, only 49.09% of the garments generated by the original TailorNet are collision-free. After fine-tuning TailorNet with ReFU, the collision-free models increase significantly to 76.77%. For difficult cases like men's shirts, which have dense meshes and

are tight fitting on the body, collision free models increase significantly by $5.3\times$, from 11.92% to 63.06%. Since we set $\alpha >= 1$, all the interpenetrating vertices are pushed outside, and the VFCP drops from 0.6% to 0%. In the meantime, for MPVE, the reconstruction error also decreases from 8.89 to 8.64. However, querying the SDF values is computationally expensive, adding around 0.106 seconds of time cost to each garment inference.

Then we train and test how ReFU performs when using the neural network approximated body SDF. In other words, we train and test ReFU with imperfect but much faster SDF. The result shows that although this strategy cannot achieve the same level of collision handling as using accurate SDF, it still improves the collision and reconstruction accuracy of the original TailorNet a considerable level. The collision-free models increase to 58.52%, MPVE drops to 8.66, and VFCP decreases by half as shown in the third to last column in Table 5.4. The inference time for the ReFU layer with approximated SDF is 2.00 milliseconds, and the inference time for the whole garment prediction system is 22.57 milliseconds.

To improve the collision handling capability while maintaining the real-time performance during testing, we experiment with using accurate SDF values when computing the collision loss during training while feeding the approximate SDF into the ReFU layer during both training and testing. The network is guided with accurate collision loss during training while learning to accommodate the errors in the approximated SDF, which will be provided at test time. Now the result is much closer to the case of using accurate SDF. We call this a "hybrid" mode for simplicity. As shown in the last column in Table 5.4, the collision-free models achieve 68.71%, VFCP improves to 0.24%, and the reconstruction error is nearly the same.

From the last two columns in Fig. 5.4, we can visually observe how using ReFU improves the body-cloth collision problems in the original TailorNet. When trained and tested with Accu-

Metric	Method					
	TailorNet	w/ Fixed Scale		w/ Predicted Scale		
		Approx. SDF	Acc. SDF	Approx. SDF	Acc. SDF	Hybrid
MPVE	11.27	10.84	10.60	10.59	10.56	10.57
VFCP	1.18%	0.58%	0.00%	0.62%	0.00%	0.51%
CFMP	11.92%	4.88%	37.22%	26.9%	63.06%	49.32%
Avg. VF	180.36	287.85	1.94	110.12	1.45	78.21
Avg. EE	11.56	19.60	8.89	8.6	6.98	7.39

Table 5.3: Ablation study on the predicted scale in ReFU. As we mentioned in Sec. 5.3.1.1, we use networks to predict the scale α_i to determine the moving offset of collided vertices. Here, we show that if we use only fixed scale (α_i always equals to 1), the performances are worse than the predicted ones with either approximated or accurate SDF. These experimental results show that our approach can also reduce the number of EE collisions.

rate SDF, the body penetrations are almost solved and the predicted garments look much more like the groundtruth. In the hybrid mode, although the network cannot achieve the same quantitative results as in the accurate SDF mode, the visual quality is quite similar. This is impressive for a real-time garment prediction system.

5.4.4 Ablation Study

In ReFU, we design a network-predicted scale α_i to determine the offset length d_i for each vertex. Theoretically it can help solve EE collisions and improve the reconstruction quality. Here we perform the ablation study to evaluate the effect of using predicted scale. We report the experimental results for male shirt in Table 5.3. As listed in the third and fourth columns, when only using $f(\boldsymbol{x}_i)$ without scaling, the quantitative results for all metrics are worse than with the predicted scaling. The percentage of collision-free models drops by two times in the case of using accurate SDF and over 5 times in the case of using the approximate SDF. Figure 5.5 further

shows that when using the approximate SDF resulted in some pump out artifacts. With the fixed scale, the predicted cloth covers the pump out region of the approximate body surface (where approximated SDF values are larger than the accurate ones) in an awkward way. However, when trained with the predicted offset scale, the network learns to handle the imperfectness in the SDF and generates a smoother and more plausible result.

We also include the average number of colliding triangles on the garment, based on VF and EE contacts, denoted as "Avg. VF" and "Avg. EE". The results in Table 5.3 shows that our predicted α_i can further reduce the number of EE collisions, as compared with a fixed value.

5.4.5 Comparisons

As reported in Table 5.4, we compared our method with other collision handling approaches. A typical approach is to employ a post-processing step. A naïve method is to move the penetrating vertices directly to the body surface along the SDF gradient direction using $f(\boldsymbol{x}_i)\widehat{\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}_i)}$ [68, 159] to eliminate the VF-collision. As shown in Figure 5.3, this method does not handle EE-collisions and might introduce new artifacts. Our experiments verify that this naïve approach increases the collision-free models only marginally from 49.0% to 59.39% . When tested with approximate SDF, the collision-free models drop to 30.89% and MPVE increases to 9.67, compared with 8.89 in TailorNet.

Guan et. al. [58] propose a more advanced technique to post-process the collisions using optimization. While this technique is more effective than the naïve approach, it cannot beat the performance of our ReFU-based method in terms of both quantitative results and computational speed.

Dataset	Metric	Method									
		TailorNet	w/ Naïve Post-Process		w/ Opt. Post-Process		w/Collision Loss		w/ ReFU		Hybrid
			Approx. SDF	Acc. SDF	Approx. SDF	Acc. SDF	Approx. SDF	Acc. SDF	Approx. SDF	Acc. SDF	
Shirt Male	MPVE	11.27	11.45	11.26	11.30	11.26	10.85	10.61	10.59	**10.56**	10.57
	VFCP	1.18%	0.69%	0.00%	0.41%	0.00%	1.28%	0.68%	0.62%	**0.00%**	0.51%
	CFMP	11.92%	3.1%	27.5%	25.21%	50.36%	20.21%	39.35%	26.9%	**63.06%**	49.32%
T-Shirt Male	MPVE	10.77	10.81	10.75	10.76	10.75	10.60	10.59	10.58	**10.57**	10.56
	VFCP	0.95%	0.56%	0.00%	0.39%	0.00%	0.85%	0.86%	0.53%	**0.00%**	0.34%
	CFMP	23.96%	12.01%	38.02%	36.57%	51.60%	28.98%	28.76%	32.18%	**58.77%**	47.25%
Short-pant Male	MPVE	6.81	6.87	6.83	6.82	6.81	6.79	6.80	6.79	**6.76**	6.77
	VFCP	0.27%	0.83%	0.07%	0.11%	0.00%	0.19%	0.21%	0.13%	**0.00%**	0.13%
	CFMP	60.35%	20.07%	73.47%	68.60%	81.42%	68.38%	66.78%	73.28%	**83.05%**	76.89%
Skirt Female	MPVE	6.90	9.35	6.90	6.98	6.90	6.92	6.90	6.90	**6.87**	6.89
	VFCP	0.067%	0.04%	0.00%	0.015%	0.000%	0.059%	0.047%	0.028%	**0.00%**	0.027%
	CFMP	93.36%	78.78%	93.87%	94.01%	96.45%	93.99%	94.44%	96.03%	**98.16%**	96.43%
All Garments	MPVE	8.89	9.67	8.89	8.92	8.88	8.74	8.67	8.66	**8.64**	8.65
	VFCP	0.60%	0.50%	0.01%	0.22%	0.00%	0.58%	0.43%	0.31%	**0.00%**	0.24%
	CFMP	49.09%	30.89%	59.39%	57.42%	70.97%	54.36%	59.15%	58.52%	**76.77%**	68.71%
Trend	MPVE	-	+8.77%	0.00%	+0.34%	−0.11%	−1.69%	−2.47%	−2.59%	**−2.81%**	−2.70%
	VFCP	-	−16.67%	−99.90%	−97.72%	−100.00%	−94.00%	−95.55%	−96.79%	**−100.00%**	−97.52%
	CFMP	-	−37.07%	+20.98%	+16.97%	+44.57%	+10.74%	+20.49%	+19.21%	**+56.39%**	+39.97%

Table 5.4: Comparison with baseline and different collision handling methods, including naïve post-processing [68, 159], optimization-based post-processing [58], and soft collision loss [19–21, 59]. We include an additional row called "Trend" to show the rate of change for each method compared to TailorNet, the baseline model. The report shows all methods perform better with accurate SDF than with approximate SDF. In either situation, ReFU has the best results among all the methods. Overall, ReFU with "hybrid" SDF works better than approximate SDF, and ReFU with accurate SDF achieves the best results.

Given the accurate SDF value, it eliminates the VF collisions and increases the ratio of collision-free models to 70.97%. Examples are shown in the third column in Fig. 5.4. Although this approach attempts to preserve the original details of the garments, it still results in higher reconstruction error and the resulting garments are not smooth, as shown by the blue arrow in Fig. 5.4 (a close-up view of the corresponding area is shown in the inset figure). When tested with approximate SDF, the accuracy with respect to the reconstruction error and collision handling drops significantly. Furthermore, the optimization requires $0.6 - 0.8$s per frame even with approximated SDF.

Instead of performing post-processing optimization, some techniques [19–21,59] apply the collision loss in Eq. 5.9 in the garment prediction models. The collision loss provides a soft constraint during network training. According to our benchmarks in Table 5.4, adding collision loss to TailorNet can reduce the overall collision artifacts and improve the reconstruction accuracy for garment samples close to the training set, however, it introduces even more collisions for testing samples that are farther away from the training set. For each sample in the test set, we compute the minimal Euclidean distance to the training set samples in the parameter space (pose, shape, and style). In Fig. 5.6, we show the mean collision error for "Shirt Male" grouped by the distance. The soft constraint can only reduce collisions for samples near the training set and even introduces more errors for samples far away. In contrast, ReFU can still resolve some collisions for samples with great differences from the seen training ones. After fine-tuning the TailorNet network with the collision loss using accurate SDF values, the collision-free garment models are around 56% for both approximate SDF and accurate SDF. Moreover, the reconstruction errors are around 8.7. The fourth column in Fig. 5.4 shows visible collisions in the predicted garments. Overall, our approach based on ReFU offers improved accuracy (i.e., fewer collisions), real-time

performance, and higher visual quality.

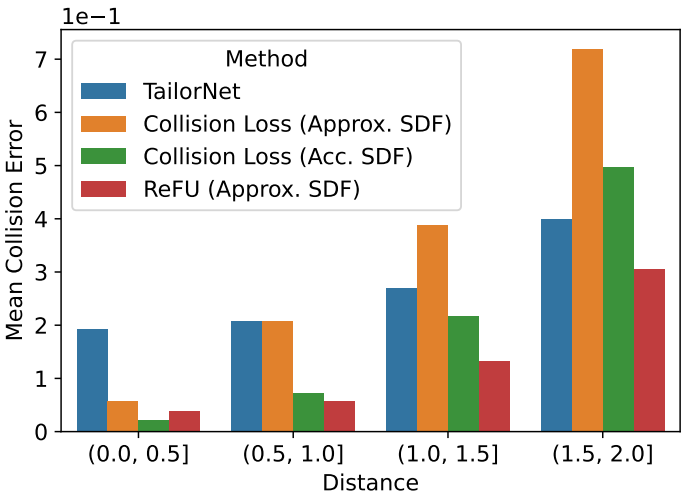


Figure 5.6: We highlight the benefits of our approach on samples distinct from the training set over methods based on collision loss. Collision loss can only help reduce collisions for samples close to the training set.

5.4.6 Running Time

We include the running time for SDF, ReFU layer, and the backbone network TailorNet in Table 5.5.

Dataset	Component			
	Approx. SDF	Acc. SDF	ReFU	Backbone
Shirt Male	1.97ms	121.96ms	0.29ms	22.14ms
T-Shirt Male	1.77ms	99.27ms	0.28ms	21.51ms
Short-pant Male	1.58ms	89.69ms	0.21ms	18.82ms
Skirt Female	1.67ms	107.38ms	0.23ms	19.76ms
All Garments	1.75ms	105.50ms	0.25ms	20.57ms

Table 5.5: Per-frame running time, including approximated SDF query, accurate SDF query computed using spatial data structures, ReFU layer inference, and the backbone network based on TailorNet inference [138].

5.5 Conclusion, Limitations, and Future Work

We propose ReFU for handling body-cloth collisions in neural networks for 3D garment prediction. Our method is inspired by the use of repulsive forces in physics-based simulators. Specifically, ReFU predicts the repulsion direction and the magnitude based on the SDF of the collided vertices and the global latent feature of the garment. ReFU can be combined with different backbone networks, and we highlight the benefits with state-of-the-art learning methods.

While our experiments show that using ReFU for training can significantly reduce the body-cloth collisions and improve the reconstruction accuracy, our approach has some limitations. To achieve live performance for the whole system, we must use a neural network approximated human body SDF for real-time computation. However, the accuracies in the SDF network prediction affects ReFU's capability of collision handling. But using more advanced neural SDF methods such as the articulated occupancy networks [7,38,120] can improve ReFU's overall performance. Furthermore, our computation of the moving offset may not fully resolve all collisions, especially all EE collisions.

Chapter 6: NPC-Net: Neural Physics-Based Collision-Aware Volumetric Deformation

6.1 Introduction

The simulation of volumetric deformable objects arises in several applications, including physics-based animation [170], multi-view capture system of human tissues [171], medical simulation [118], reinforcement learning for soft robots [148], etc. For volumetric objects composed of homogeneous materials, these simulations are driven by two main forces: internal potential forces and internal/external (self-)collision forces. The internal potential forces hold the object together, while the (self-)collision forces prevent the objects from penetrating. In this paper, we mainly deal with the problem of quasistatic deformation prediction, where the goal is to predict a deformable object pose corresponding to a force equilibrium configuration, where the two driving forces cancel each other out. Quasistatic simulation is known to be more challenging than dynamic simulation due to its highly non-convex objective function in the underlying optimization problem, while dynamic simulation utilizes the positive definite kinetic energy term to regularize the optimization.

There is considerable prior work on simulating or predicting volumetric deformations. The most well-studied methods are based on numerical physics simulations, e.g., the finite-element

methods [170], the mass-spring methods [104], and particle-based simulations [31]. However, these methods need to segment or decompose a continuous object into discrete elements and formulate a joint system of equations to solve for the deformation of each element. In practice, complex volumetric objects are represented using large meshes consisting of tens of thousands of elements. The numerical algorithms may take tens of seconds to predict a single quasistatic pose [93, 171] for such objects and cannot achieve real-time performance.

Several methods have been proposed that leverage machine learning techniques to accelerate various applications involving deformable object simulation, including volumetric objects [42] and human tissues [227]. While these learning methods can achieve orders of magnitude faster computation than numerical methods, they do not explicitly model the collisions and response forces. In general, reliable handling of collisions remains a major challenge with respect to neural simulators [222, 225]. Current learning algorithms can predict a collision-free latent representation of deformable objects and can be orders of magnitude faster than numerical algorithms. However, these neural collision handlers do not account for many characteristics or behaviors of physics-based models, so their predicted objects can exhibit unnecessarily large deformations. Other learning-based methods [154] assume that the global shape of the object is given while using some neural adjustments to resolve local collisions. In practice, these methods cannot predict large, collision-induced global deformations. Some learning-based simulation methods [158] are designed for thin shells such as deformable clothes and papers and do not generalize to volumetric objects.

Main Results: We present NPC-Net, the first end-to-end learning-based approach to predict physics-based, (self-)collision-free, volumetric object deformations at real-time frame rates. To this end, our neural architecture is trained using physics-informed loss functions and consists

of neural collision detection and handling modules. Compared with state-of-the-art neural architectures for simulation [154], our method is not restricted to local collision handling. As a result, our approach can predict large, global deformations induced by collision forces. Moreover, NPC-Net uses the low-dimensional latent representation of the object throughout its prediction, so our method is $100\times$ faster than a pure numerical simulator, hitting real-time performance. In order to perform accurate prediction, our approach is based on the actor-critic reinforcement learning algorithm [97]. Specifically, we use a critic network to learn the internal potential energy model and then use an actor network to approach the local minima of the learned energy functions, which corresponds to a force equilibrium pose. The novel components of our approach include the following:

- Although only the outer surface of a deformable object is visible, we utilize the volumetric mesh dataset to formulate a loss that penalizes the volumetric deformation errors, leading to higher embedding accuracy of the surface mesh.
- Our actor-critic architecture learns to predict quasistatic deformable poses in the latent space, where we use separate networks to learn both internal elastic and (self-)collision penalty functions.
- We propose a differentiable neural conjugate gradient optimizer as our actor network. This method uses momentum-accelerated gradient descent with neural-predicted step sizes to efficiently minimize the critic's approximated energies.
- We propose an active learning method to stabilize the training, where the critic network continually improves the potential energy estimation while the actor network explores the latent space to find force equilibrium poses.

We evaluate our method on three different datasets, where the primary geometric objects are bunny, torus, and cross undergoing large, global deformations. These objects have 3296, 321, and 439 vertices, respectively. Prior techniques based on numerical methods take $4 - 9$s to predict each pose, while our learning algorithm (NPC-Net) takes $25 - 30$ms on GPU to simulate each frame. In practice, our method can predict the stable deformable poses of complex objects, the visual quality of which is comparable to ground truth simulators, while being $100\times$ faster. Our ablation study further shows that, among the predicted deformable object poses, the number of collision-free poses is increased by $2 - 6\times$ using NPC-Net, as compared with a fully connected network baseline. Overall, NPC-Net provides significant improvements in terms of handling collisions in neural simulators.

6.2 Related Work

Physics-Based Deformation: The prediction of physically correct deformable objects, including volumetric solids [170], shells [54], and strings [18], is a well-studied problem. Our approach is designed to predict the poses of volumetric objects. In case of such objects, the quasistatic and dynamic simulation problem can be cast as an unconstrained numerical optimization problem [49], making them inherently amenable to learning-based methods. A major drawback of the numerical simulation of volumetric solids lies in the high computational cost. Some well-known methods [49,171] take seconds to minutes to predict even a single frame of high-resolution deformable objects with thousands of vertices. Recently, several works [105, 123] have proposed advanced numerical techniques, e.g., the ADMM method, to accelerate the computation, but their performance on high-resolution meshes is not fast enough for real-time applications. The main

computational bottleneck arises due to the high-dimensional configuration space, which is composed of many vertices, leading to a large system of nonlinear equations. In addition, several works have focused on the handling of various hyperelastic models [105, 123, 171]. Among these models, a widely used candidate is the Neo-Hookean model [170], which is used in this work. The Neo-Hookean model has the remarkable property of being invariant to rigid transformation and strictly avoids element inversion. Computationally, however, the Neo-Hookean model can lead to a non-Lipschitz, non-convex objective function formulation, posing a significant challenge to the underlying optimization algorithm.

Learning-Based Physics and Collision: Due to the amenability of learning-based methods, several approximations have been proposed as a computationally inexpensive replacement for an analytic collision handler. Some methods [13, 42] learn a low-dimensional linear or nonlinear embedded subspace of deformations. Within such a subspace, simulation can be orders of magnitude faster than in full space simulations. However, these methods need to resort to the high-dimensional configuration space to detect and handle collisions [73]. We proposed learning-based collision handling methods in Chapter 3 and Chapter 4. These methods use a neural architecture to embed deformations in low-dimensional space and learn to detect collisions from only the latent code. The resulting neural network is differentiable, so optimization-based methods can be used to resolve collisions. However, these methods ignore the physics-based model during collision handling, so their predicted pose might not satisfy the laws of physics. In a parallel effort, a learning-based physics model was proposed in [95, 96]. These methods can learn subtle dynamic behaviors from data, but their computational cost is still too high for real-time applications. Compared to prior methods, our learning-based approach is designed for volumetric objects and can perform integrated collision detection and response computation using an

end-to-end learning approach for (self-)collision-free volumetric deformations.

6.3 Problem Statement

In this section, we mathematically formulate our physics-based volumetric deformable object simulation problem. We assume our volumetric object is represented as a tetrahedral Lagrangian mesh $\mathcal{M} \triangleq < \mathcal{V}, \mathcal{T} >$, where $\mathcal{V}$ is a set of volumetric vertices and $\mathcal{T}$ is a set of tetrahedral elements. We use V to denote the concatenated vertex position vector, and V_0 denotes the position vector at rest pose, i.e., the equilibrium pose without any external forces. The driving force behind object deformations is the internal potential energy [110], which models the material property of an object and penalizes deviation from the rest pose. Assuming a Lagrangian formulation, the potential energy takes the following form:

$$E_{ee}(V) \triangleq \int_{M(V_0)} \rho(F(v, V))dv,$$

where we slightly abuse notation and use $M(V)$ to denote the volume taken up by the object at its pose V. $F(v, V)$ is the deformation gradient at continuous location v caused by deformation from V_0 to V, and ρ is the energy density function. We assume the object is made of neo-Hookean material, which is an isometric elastic material. We refer readers to [171] for the definition of F, ρ, and discretization of integral. In addition to internal forces, the object is undergoing external control forces such as gravitational forces or user dragging forces, which can be summarized as an external potential energy term $E_c(V, q)$. Here q is the controllable parameters of the external forces. For example, q could be the gravitational acceleration coefficients or the direction and strength of user dragging forces.

A fundamental requirement for physically plausible object deformation is that it is collision-free, which requires that there is no overlapping between the object and static obstacles or between different parts of the object (i.e., no self-collisions). Although the collision-free constraints are mathematically well-defined, these constraints are non-differentiable, and constructing a computationally efficient constraint formulation is non-trivial. In this work, we adopt the penalty-based formulation [84], which defines each object and obstacle as a deformed signed distance field and adopts the necessary condition that all the vertices have positive distance values. This treatment leads to an energy term $E_{sc}(V)$ that penalizes self-collisions and a second term $E_{oc}(V)$ that penalizes collisions with static obstacles. Since all the physical models and constraints use a conservative formulation, we can combine them into the following unconstrained optimization:

$$V^*(q) = \underset{V}{\textbf{argmin}}\ E_{ee}(V) + E_{sc}(V) + E_{oc}(V) + E_c(V, q), \tag{6.1}$$

with the solution corresponding to the (nearly) collision-free force-equilibrium object pose $V^*(q)$ under user control parameter q. We consider $V^*(q)$ as the physics-based ground truth object pose. Several numerical algorithms have been proposed to solve Eq. 6.1, including [84, 170], but their performances are far from real-time. For example, the state-of-the-art algorithm [170] takes several seconds to predict a single deformation. We propose a learning-based solution to approximately solve Eq. 6.1 and predict $\tilde{V}(q)$ from the control signal q in real-time.

6.4 Real-Time Collision-Aware Physical Deformation

In this section, we present our novel architecture and training method for our NPC-Net architecture. It consists of four novel components, and our trained NPC-Net can predict nearly

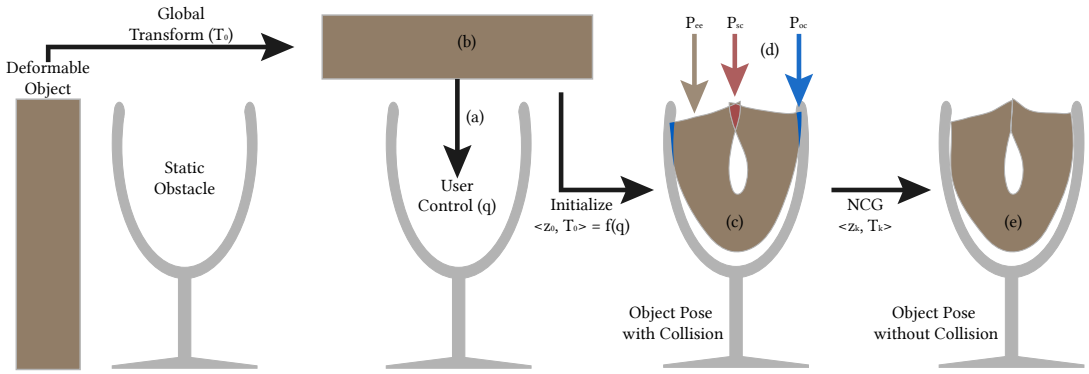


Figure 6.1: Given a deformable object and static obstacle, NPC-Net takes the user control signal q (a) as input and predicts a collision-free quasistatic pose. Our initializer will first predict a global transformation T_0 and initial latent-space pose z_0 (b). The initially predicted result (c) can have obstacle-collisions (blue), self-collisions (red), or is not in force equilibrium. We solve this problem by using 3 critic networks to learn the potential energies $P_{ee,sc,oc}$ (d). We then use several steps of differentiable NCG to revise the pose to be (self-)collision-free and force-balanced (e).

collision-free object poses undergoing global deformations. During the first step (Sec. 6.4.1), we train a mesh embedding network that maps the surface of a deformable pose to a latent code. Next, we train several critic networks to estimate various energy terms from the latent code (Sec. 6.4.2). As a result, the gradient of the critic networks can be utilized to approximately solve Eq. 6.1 and yield the force equilibrium poses, similar to the actor-critic reinforcement learning [88]. To this end, we design a differentiable neural conjugate gradient (NCG) network that serves as our actor (Sec. 6.4.3) to minimize the learned energy terms via several iterations of neural momentum-accelerated gradient descent steps. Finally, we propose an active learning method in Sec. 6.4.4 to enhance the robustness of unseen control parameters. The pipeline of NPC-Net is illustrated in Fig. 6.1.

6.4.1 Physics-Aware Shape Embedding

The high computational cost of conventional numerical simulators is due to the large number of vertices in $\mathcal{V}$, leading to high-dimensional linear system solving. In view of this, all

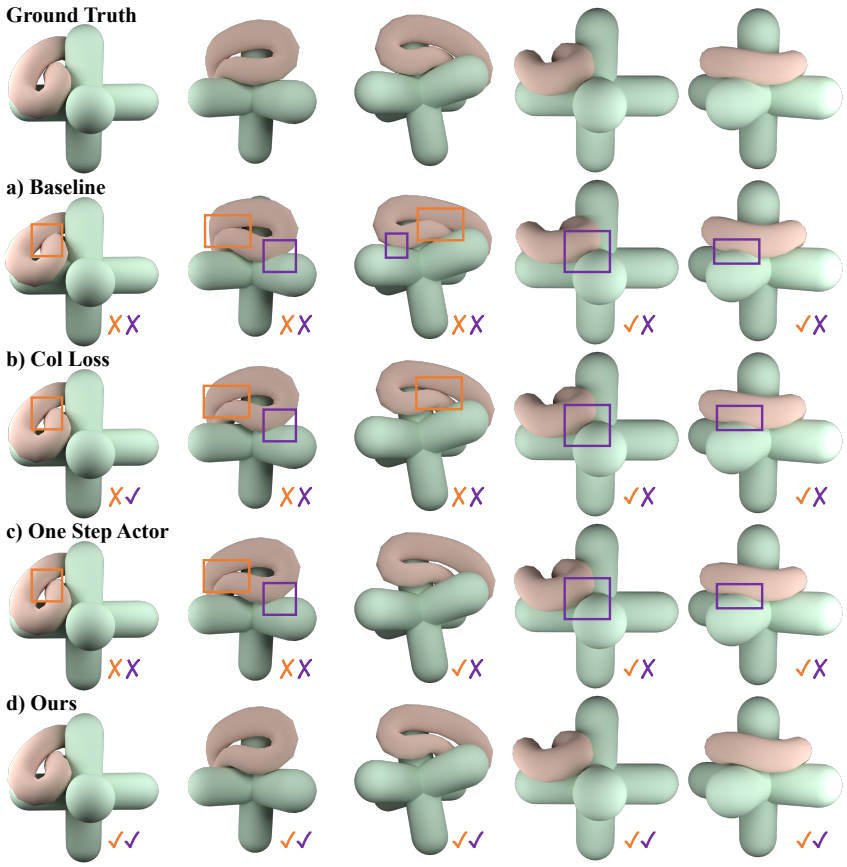


Figure 6.2: Results for Torus: We compare numerically simulated results (Ground Truth) with different learning-based methods and mark self-collisions (orange) and obstacle-collisions (purple). The Baseline uses the fully-connected initializer to directly predict output poses; the Col-Loss uses the same network architecture as Baseline but uses $\mathcal{L}_{\text{actor}}$ for fine-tuning; the One-Step-Actor only runs NCG for one iteration.

prior learning approaches [68, 154] use shape embedding to map salient object poses to a low-dimensional manifold. We also adopt this approach, with the important difference that we train our embedding module to represent only the surface mesh. We denote the subset of surface vertices by $\mathcal{V}^s \subset \mathcal{V}$, with the concatenated position vector being V^s. Mapping only surface vertices can significantly reduce the size of the dataset by discarding many internal vertices. We follow [160, 181] and use a standard autoencoder to parameterize our embedding module. In order to handle the unstructured mesh input data, we adopt a convolutional autoencoder using a graph

CNN layer [230]. Put together, our autoencoder architecture is denoted as:

$$z = \text{Enc}(T^{-1}V^s) \quad V^{s\prime} = \text{Dec}(z),$$

where z is our latent code, $V^{s\prime}$ denotes the reconstructed surface mesh, and Enc, Dec denote the encoder and decoder, respectively. Here T represents the extrinsic, global rigid transformation of the given input pose V^s. This is a widely adopted approach proposed in [46] to factor out the extrinsic transformations and has the autoencoder only represent intrinsic deformations, leading to better data efficacy. In practice, we could compute T from a global alignment between V^s and V_0^s (surface mesh of rest pose) using [176], which is also a differentiable procedure.

The conventional loss for training the autoencoder is the mean-squared reconstruction error $\text{MSE}(TV^{s\prime}, V^s)$. However, considering that we have a volumetric mesh V associated with each surface mesh, we devise a novel volume-aware reconstruction error, denoted as $\mathcal{L}_{\text{vol}}$, which utilizes the volumetric mesh to improve the embedding quality. Specifically, for each ground truth volumetric mesh V, we keep its internal vertices and replace the surface vertices with our reconstructed version, denoted as: $\bar{V} \triangleq (V - V^s) \oplus (TV^{s\prime})$. Next, we introduce the following novel loss term to penalize deviations from the ground truth over the entire volume:

$$\mathcal{L}_{\text{vol}} \triangleq \int_{M(V_0)} \|F(v, \tilde{V}) - F(v, \bar{V})\|_F^2 dv - \int_{M(V_0)} \min(|F(v, \bar{V})|, 0) dv.$$

The first term above penalizes the difference in terms of deformation gradient, and the second term penalizes the negative singular values of the deformation gradient corresponding to inverted elements [162], i.e., a form of local self-collision. All the integrals are discretized using the

same finite-element method as the internal potential energy. Note that we only need to evaluate the integral in $\mathcal{L}_{\text{vol}}$ over a small subset of $M(V_0)$ consisting of tetrahedra connected to surface vertices because fully internal tetrahedra are not affected by $TV^{s\prime}$ and contribute constant values to the energy. Our final loss function for training the embedding autoencoder takes the following weighted combination:

$$\mathcal{L}_{\text{emb}} = \lambda_{\text{vert}}\text{MSE}(TV^{s\prime}, V^s) + \lambda_{\text{vol}}\mathcal{L}_{\text{vol}},$$

with $\lambda_\bullet$ being the weight of corresponding terms.

6.4.2 Latent-Space Critic Network

Our embedding module maps each deformable pose to a latent code z. However, not all latent codes are physically plausible, i.e., correspond to the minima of Eq. 6.1. We need to use additional projection operators to map z onto the latent physically correct manifold. As the ground truth, such a manifold can be defined by solving Eq. 6.1 restricted to the latent space, which is formulated as:

$$z^*(q), T^*(q) = \underset{z,T}{\textbf{argmin}}\ E_{ee}(\bar{V}) + E_{sc}(\bar{V}) + E_{oc}(\bar{V}) + E_c(\bar{V}, q). \tag{6.2}$$

Unfortunately, solving Eq. 6.2 exactly would require decoding z and evaluating the energy terms for each tetrahedral mesh element, which compromises the efficacy of using latent space. Therefore, we propose training several networks to learn the various energy terms in latent space, i.e., the critic networks. A similar idea has been adopted in off-policy reinforcement learning [97] and

neural collision detectors as in Chapter 3. Our main point of departure from these prior works lies in our loss function for training these critic networks. Since we need to use gradient information to guide us to solve Eq. 6.2, we propose matching the ground truth energies in terms of both function value and gradient, leading to the following value- and tangent-matching loss:

$$\mathcal{L}_{\text{critic}}(z) \triangleq \sum_{\bullet \in \{ee,sc,oc\}} \lambda_{\text{val}}\text{MSE}(P_\bullet(z,T), E_\bullet(\bar{V})) + \lambda_{\text{tan}}\text{MSE}(\nabla P_\bullet(z,T), \nabla E_\bullet(\bar{V}),$$

where $P_\bullet$ represents the critic networks learning to mimic the potential energy term $\bullet$. These critic networks allow us to solve Eq. 6.2 approximately and robustly without resorting to the high-dimensional mesh.

Algorithm 2 Differentiable NCG(q, z_0, T_0)

Require: User control parameter q

$< z_0, T_0 >= f(q)$ ▷ Compute initial guess

$s_e^{-1} \leftarrow 0$

for $k = 0, \cdots, K-1$ **do**

  $d_e^k \leftarrow \nabla_{z_k,T_k} g_e \quad e \in \{ee, sc, oc\}$

  $\beta_e^k \leftarrow \begin{cases} 0 & k = 0; \\ \max\left(0, \frac{{d_e^k}^T (d_e^k - d_e^{k-1})}{{d_e^{k-1}}^T d_e^{k-1}}\right) & k \geq 1. \end{cases} \quad e \in \{ee, sc, oc\}$

  ▷ We maintain and update momentum per energy term

  $s_e^k \leftarrow d_e^k + \beta_e^k s_e^{k-1} \quad e \in \{ee, sc, oc\}$

  ▷ We use differentiable neural network to predict per-energy step sizes

  $\alpha_{ee}^k, \alpha_{sc}^k, \alpha_{oc}^k \leftarrow S(q, z_k, T_k, P_{ee}(z_k, T_k), P_{sc}(z_k, T_k), P_{oc}(z_k, T_k), k)$

  $< z_{k+1}, T_{k+1} > \leftarrow < z_k, T_k > - \sum_e \alpha_e^k s_e^k$

end for

Return $< z_K, T_K >$

6.4.3 Latent-Space Actor Network

Given the potential energies approximated by our critic networks, $P_\bullet$, our actor networks use two steps to approximately solve Eq. 6.2 guided by the gradients $\nabla P_\bullet$. First, we use an

initializer network denoted by $< z_0, T_0 >= f(q)$ to propose an initial guess. We train our initializer network using the following MSE loss:

$$\mathcal{L}_{\text{init}} = \text{MSE}(z_0, \text{Enc}(T^{-1}V^s)) + \text{MSE}(T_0, T),$$

with $< V^s, T >$ being the ground truth surface mesh and global transformation. As our second step, we use several neural gradient descent steps to locally optimize $P_\bullet$. For faster convergence, we adopt the momentum-accelerated nonlinear conjugate gradient (NCG) algorithm (we refer readers to [60] for more details). The K iterations of the NCG algorithm can be denoted as a function $< z_K, T_K >= \text{NCG}(q, z_0, T_0)$, and we consider $< z_K, T_K >$ as the final, latent-space deformable object pose predicted by NPC-Net. We can reconstruct the corresponding high-dimensional mesh vertex positions by:

$$V_K^{s\prime} \triangleq \text{Dec}(z_K) \quad \tilde{V} \triangleq (V - V^s) \oplus (T_K V_K^{s\prime}),$$

where we have also combined the ground truth interior vertices to yield the volumetric mesh vertex positions, denoted as $\tilde{V}$. We then train the actor network in an end-to-end manner. The predicted pose should have a locally minimal energy value, and the physical energy terms can be used as loss functions. We also add an MSE term between $V_K^{s\prime}$ and ground truth V^s for regularization. The combined actor loss takes the following form:

$$\mathcal{L}_{\text{actor}} \triangleq \sum_{\bullet \in \{ee, sc, oc\}} \lambda_\bullet E_\bullet(\tilde{V}) + \lambda_{reg}\text{MSE}(V_K^{s\prime}, V^s).$$

Note that we use the ground truth energy $E_\bullet$ for training the actor instead of the learned energy $P_\bullet$. Our learned energy is only used inside the NCG algorithm to approximately solve Eq. 6.2. As Chapter 5 suggested, this hybrid energy setting can boost the final performance while maintaining the real-time running benefits of the learned critic.

The end-to-end training of the actor network involves the NCG algorithm. However, this algorithm is non-differentiable due to its discrete line-search step deciding the step sizes. To resolve this problem, we borrow an idea from Chapter 5 and use neural networks to predict step sizes. These step size networks are defined as the following function S:

$$\alpha^k_{ee,sc,oc} \triangleq S(q, z_k, T_k, P_{ee}(z_k, T_k), P_{sc}(z_k, T_k), P_{oc}(z_k, T_k), k).$$

Note that we allow different step sizes for different energy terms to accelerate convergence. In a similar fashion, we also maintain separate momentums for different energy terms, using the Polak-Ribière formula with automatic direction reset to compute the mixing parameter β [142, 165]. Combined, we derive a differentiable NCG outlined in Algorithm 2.

6.4.4 Training Algorithm

The key to the success of actor-critic algorithms lies in the interleaved update to the actor and critic networks. Following this idea, we design a four-stage interleaved training algorithm using a dataset of groundtruth $< V^s, T >$. During the first stage, we train the embedding module by reducing $\mathcal{L}_{\text{emb}}$. We then train the critic network by reducing $\mathcal{L}_{\text{critic}}$. For this step, we use an augmented dataset containing all intermediary poses during each iteration of optimization performed by the groundtruth numerical simulator. Next, we train the NCG initializer network

by reducing $\mathcal{L}_{\text{init}}$. Our first three stages initialize NPC-Net via supervised learning on a fixed dataset. Networks trained this way have limited generalization ability to unseen object poses.

To improve the robustness, we introduce a fourth stage of interleaved actor-critic training via active learning. When updating the actor network, we fix the three critic networks and minimize $\mathcal{L}_{\text{actor}}$. Note that, to evaluate $\mathcal{L}_{\text{actor}}$, we need to query the ground truth energy $E_\bullet$ on the unseen predicted pose $\tilde{V}$. Similarly, when updating the critic network, we fix the initializer and step size networks and reduce:

$$\mathcal{L}_{\text{active}} = \sum_{e \in \{ee, sc, oc\}} \text{MSE}(P_\bullet(z_K, T_K), E_\bullet(\tilde{V})).$$

Again, this requires querying the ground truth energy $E_\bullet$ on the unseen pose $\tilde{V}$. We obtain this ground truth energy using the numerical simulator. Note that during the last stage of active learning, we only use value matching without tangent matching to avoid a much more costly evaluation of ground truth energy gradient on each predicted pose $\tilde{V}$.

6.5 Results

We implement NPC-Net using pytorch [136], and all experiments are performed on a server with a 16-core AMD EPYC CPU and NVIDIA A5000 GPU. We evaluate our algorithm on three datasets: Bunny, Torus, and Cross. Our ground truth numerical simulator is a C++ implementation of [84].

As illustrated in the left part of Fig. 6.3, the first dataset has a bunny-shaped object with 3296 vertices inside a box. The user is able to grasp a certain area of the bunny, fixing the vertices in the area to a specified position and orientation (represented as a 3×3 matrix). The bunny can

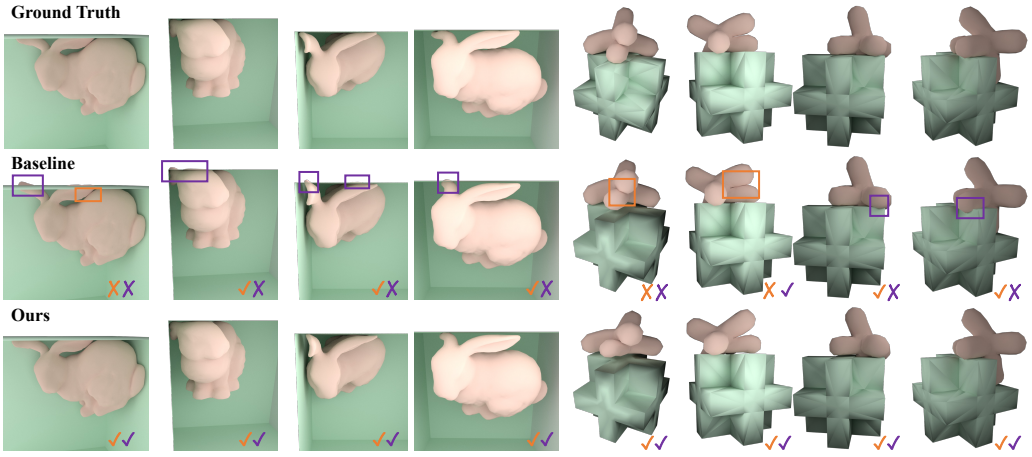


Figure 6.3: Results for Bunny and Cross under the same setting as in Fig. 6.2.

collide with the boundary of the box, inducing (self-)collisions. Our second dataset is illustrated in Fig. 6.2; it has a torus with 321 vertices colliding with three capsule-shaped obstacles. In this case, the user is allowed to specify an approach direction q (a 3-dimensional vector), and our numerical optimizer will have the torus approach the center of capsule-shaped obstacles along the specified direction, inducing (self-)collisions. Our third dataset is illustrated in the right part of Fig. 6.3. This dataset has the same setting as Torus; the only differences are that the object is a cross-shaped capsule with 439 vertices and the obstacle is three boxes.

We use our numerical simulator to generate $45586/77791/54996$ ground truth poses $< V^s, T >$ for the Bunny, Torus, and Cross datasets, respectively, and we use 30% data for testing. For each ground truth pose, the simulator solves the optimization of form Eq. 6.1. We not only store the result of optimization, i.e., the force equilibrium pose, but also the intermediary poses after each iteration of optimization.

For the ablation study, we compare our method with three alternative algorithms. The first algorithm (Baseline) uses the supervised-learned initializer network to output the ultimate pose, i.e., skipping the actor-critic network using $K = 0$. Our second algorithm (Col-Loss)

Dataset	Metric	Method			
		Baseline	Col Loss	One Step Actor	Ours
Bunny	L2-V	0.0973	0.00969	0.00952	**0.00948**
	E_{oc}	0.403	0.263	0.230	**0.197**
	CFP	70.6%/29.4%/27.3%	72.8%/37.3%/33.9%	74.5%/41.6%/37.9%	**77.1%/46.5%/41.8%**
Torus	L2-V	0.103	0.0889	0.0887	**0.0886**
	E_{oc}	1.45	0.822	0.622	**0.532**
	CFP	98.2%/9.59%/9.55%	98.9%/23.2%/23.1%	99.0%/26.9%/26.8%	**99.1%/29.3%/29.1%**
Cross	L2-V	0.227	0.224	0.215	**0.204**
	E_{oc}	1.31	0.88	0.807	**0.547**
	CFP	84.1%/4.72%/4.15%	85.5%/12.4%/11.1%	88.9%/13.7%/12.3%	**90.8%/25.7%/23.0%**

Table 6.1: For the test set of each dataset, we profile the averaged vertex-wise position error compared with ground truth quasistatic pose (L2-V), the energy level of obstacle-collision penalty E_{oc}, and the percentage of self-collision-free/obstacle-collision-free/both-collision-free poses (CFP).

Dataset	Numerical Simulator	Ours			
		Initizer (f)	Actor (NCG)	Decoder (Dec)	Total
Bunny	3.88s	4.84ms	15.50ms	5.14ms	25.48ms
Torus	6.16s	4.20ms	20.33ms	5.42ms	29.95ms
Cross	9.02s	4.19ms	20.05ms	5.49ms	29.73ms

Table 6.2: Running Time Comparison: The average per frame cost for predicting 100 quasistatic poses, comparing our method with the ground truth numerical simulator. We also show the breakdown of our computation cost into different neural components.

is similar to Baseline, but uses $\mathcal{L}_{\text{actor}}$ to fine-tune the initializer, which queries the ground truth collision energy $E_\bullet$. This approach is also adopted by several prior works for learned cloth draping [19, 21]. Our third algorithm (One-Step-Actor) uses our full NPC-Net architecture but only uses a single NCG step. This approach is like that in Chapter 5. As summarized in Table 6.1, our method achieves the best rate of (self-)collision reduction. Further, we profile the running time of different components of our network and compare it with the ground truth simulator in Table 6.2. We can achieve real-time performance, and it is $100-300\times$ faster than the numerical simulator.

6.6 Conclusion

We present NPC-Net, a real-time actor-critic (self-)collision-free deformable object pose predictor. Our main idea is to train critic networks to memorize the landscape of various potential energies and then use actor networks to find local minima in the landscape. We further propose training algorithms allowing NPC-Net to generalize to unseen user controls and object poses. By evaluating on three complex datasets, we show that our algorithm outperforms previous state-of-the-art neural architectures in terms of eliminating the (self-)collisions. The major limitation of our method lies in the need for retraining for each new object or obstacle settings. This could increase the overhead of simulation setup. Further, our method does not consider kinetic energy, so we do not support dynamic object simulations, which will be our future work.

Chapter 7: Deformation Images

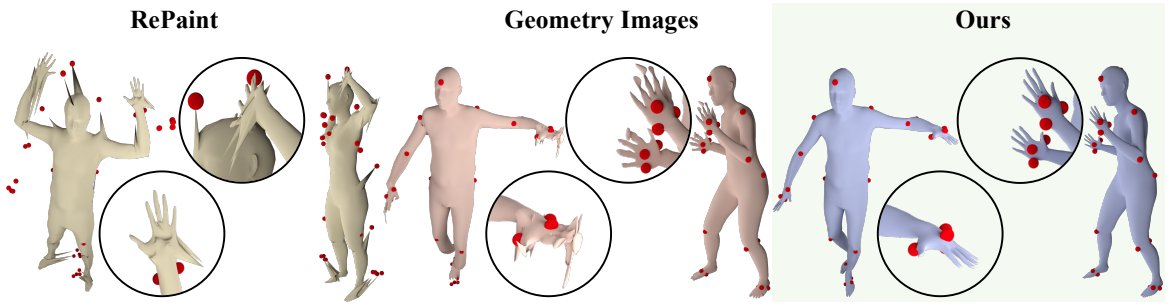


Figure 7.1: Our method, which predicts deformation gradients over a 2D image in UV spaces, enables us to predict plausible deformations of humans. The red spheres are control points constrained to specific target locations, which are enforced during the diffusion process through inpainting. While our method (right) respects the constraints, RePaint [108] (left) cannot handle their sparse nature, thus leading to a different deformation that leaves the inpainted constraints as outliers. Directly predicting the output signal (the coordinates) in the image domain (middle) as proposed in Geometry Images [57, 115, 168] leads to loss of detail in regions that are undersampled in the image space such as the hands. Since our method predicts the deformation's gradients in image space, the geometric details are preserved regardless of undersampling.

7.1 Introduction

Deformation is one of the most fundamental and integral tools in geometry processing and physically-based modeling. Its uses are vast, including simulating physical phenomena such as cloth [68, 181] and elasticity [174], modifying geometric shapes through edits [175], and registration [24]. These tasks are in turn important for many applications in graphics, vision, engineering, medicine, and physics.

As such, many works in the last few years have focused on applying machine learning techniques for predicting 3D deformations. These are in turn tailor-made for the specific task they aim to perform, using various representations such as having the network induce and manipulate a rig controlling the deformation through cages [219] or control points that induce skinning weights [72, 103], displacement fields [76], and deformation gradient fields [4].

These tailor-made, specific representations are common for 3D-geometry machine learning tasks, but their non-standard nature usually prohibits their direct use within general machine-learning frameworks. Consider a generative framework guided by a diffusion process [66] as a concrete example of this shortcoming. At the most general level, the diffusion framework learns to receive and denoise a noisy signal. This, in essence, requires an encoder that can analyze the noisy input and a decoder that can output the denoised signal. When applying diffusion to 2D images, both the encoder and the decoder can be implemented as part of one standard image-based neural architecture, e.g., a standard UNet [155]. However, when tackling a mesh deformation problem, attempting to apply the diffusion framework to one of the previously-mentioned representations requires a custom encoder/decoder pair, e.g., by considering the signal defined over the vertices of a non-regular graph and using some type of graph neural network [210].

This paper's core observation is that 2D image-based techniques can be used to define a deformation of a 3D mesh directly, despite the mesh having arbitrarily high resolution details (i.e., the mesh's resolution can be far beyond that of the 2D image). This observation follows recent works [4], which show that the gradients of a deformation make up a smooth, low-frequency signal and that retrieving the deformation from the gradients can be easily achieved within a deep-learning framework. Thus, given a UV map of the source mesh to be deformed, the deformation gradients can be rasterized to a 2D image without significant degradation caused to the gradients

in the rasterization process. Conversely, the gradients can be predicted as a raster image and then, using the UV map, be interpolated to the 3D mesh and used to deform it without ruining the details on the mesh. This is in contrast to previous methods that use 2D images to define 3D geometry by directly interpreting each pixel RGB value as an XYZ coordinate [57,157,168,218], which would restrict their predicted meshes to the exact resolution and connectivity of the pixel grid. See their result in Fig. 7.1, middle, compared to our method, Fig. 7.1, right.

In sum, our main contributions include:

- We show that by working in the gradient domain, deformations of 3D meshes can be represented as images without loss of detail of the underlying geometry.
- We adopt this representation within a diffusion framework, enabling us to learn a generative space of deformations.
- We propose a technique to *restrict* and control the generated deformations by modifying the regular inpainting process usingLlaplacian regularization, which enables inpainting based on an extremely sparse set of points (e.g., the equivalent of 2 pixels).

We train a diffusion process to generate arbitrary human poses to evaluate our method. Using inpainting, we show that we can repose arbitrary human models into poses that respect given arbitrary given constraints. We highlight the results on human models from the AMASS dataset and compare the performance with other methods. Our method using Jacobian-based deformation images with Laplacian regularized inpaint achieves the best performance in terms of L2-V and L2-N metrics [4].

7.2 Related Work

Mesh Deformation: Deformations are a core task in geometry processing and graphics, with many methods devised to represent and generate them, including rigs that control the deformed shape [42, 70, 71, 77, 79, 100] and variational formulations involving optimization of energies [5, 101, 174]. These representations have been recently adopted to machine learning frameworks for predicting deformations [4, 69, 94, 103, 179, 213, 214, 219]. These in turn enable data-driven methods to automate deformation-related tasks such as deformation transfer [47, 177], object reconstruction or template deformation from images [78, 203], deformable object retrieval [194], deformable mesh production [48], and physically-plausible deformation prediction of cloth [68, 102]. All these representations require defining a non-standard graph structure or completely abstracting the representation as an implicit neural field, thereby prohibiting the use of simple, out-of-the-box tools such as an image CNN or UNet.

3D Shapes as Images: The idea of UV-mapping meshes and then representing their XYZ coordinates as colors on the pixel grid of an image, called "Geometry Images" was proposed by [57] and was later extended to accommodate for better representation of the geometry [157, 218]. Subsequent works explored applying image-based CNNs to learn over geometry images [115, 168] and use them to generate meshes [16]. However, none of these can represent geometry in a higher resolution or different connectivity than that of the image's or adhere to an input mesh's connectivity. Thus, if used directly to represent a deformation of a model, they would lead to a loss of detail and modification to the underlying triangulation.

Diffusion: Diffusion has been gaining immense popularity as the most expressive generative technique [66, 172, 173]. Applied to 2D images, many works have investigated inpainting:

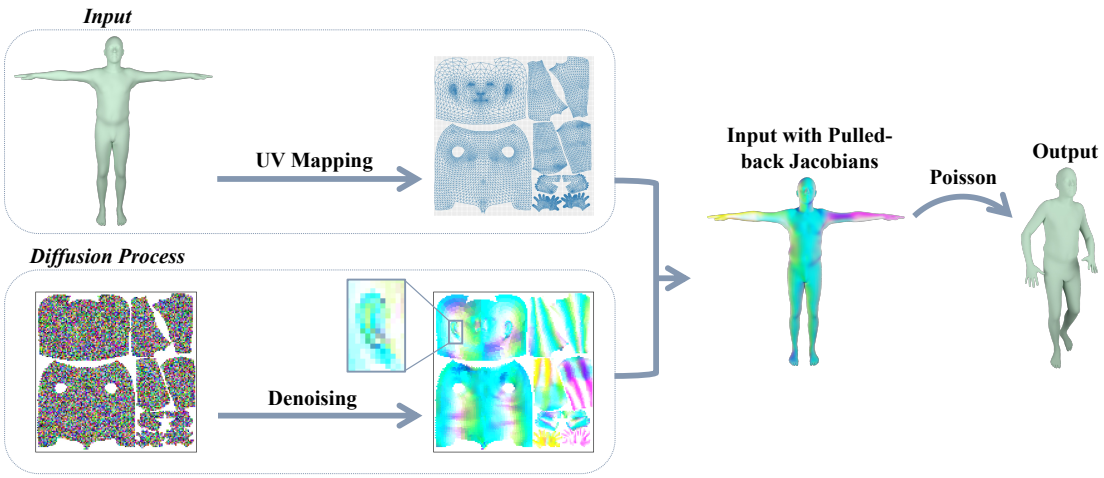


Figure 7.2: Our *input* is one mesh with a UV map. A *diffusion process* generates a 2D image, assigning a 3×3 matrix to each pixel (visualized here via RGB colors for illustration). We then pullback these matrices to each triangle on the input mesh to get a Jacobian for each triangle, placing us in the setting of [4] and outputting the final deformation of the mesh by solving Poisson's equation.

how to restrict specific areas of the image to have specific predesignated colors [108, 126, 156]. With its rise in popularity, it has been applied to generate 3D points clouds [109, 221, 228]; however, with the lack of 3D data to train on, many works resort to aggregating 2D diffusion processes to create 3D representations such as NeRFs [99, 143, 201].

7.3 Method

7.3.1 Preliminaries

Jacobians. We operate over triangular meshes, with each mesh having vertices $V_0 \in \mathbb{R}^{n\times 3}$ and triangle faces $T \in \mathbb{R}^{m\times 3}$. A deformation of a mesh is defined by assigning new positions to each of its vertices, $V \in \mathbb{R}^{n\times 3}$.

Each triangle is thus mapped via an affine map. The Jacobian $J_i \in \mathbb{R}^{3\times 3}$ of that triangle is

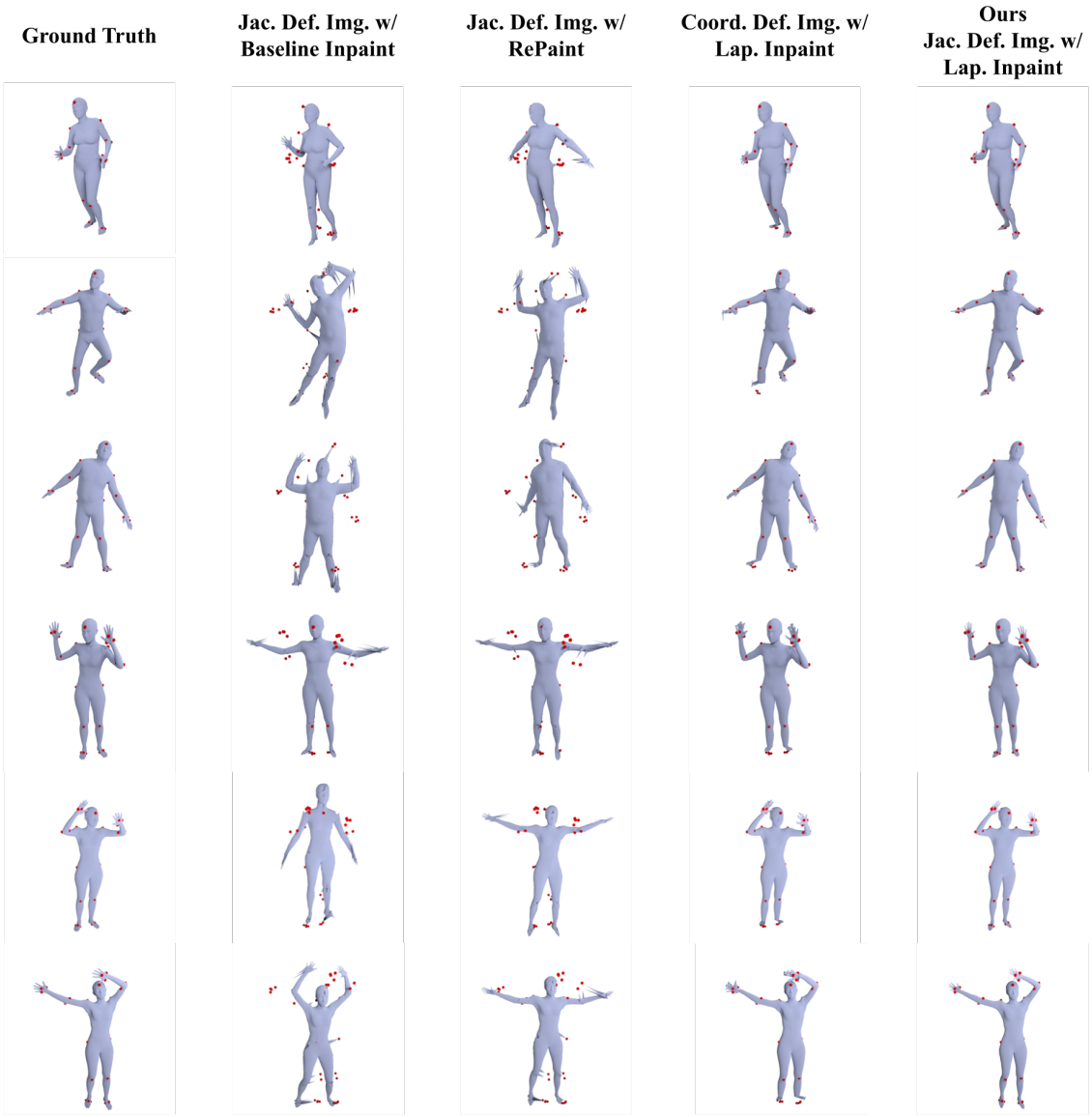


Figure 7.3: We validate the efficacy of our approach by choosing 30 control points, placing them based on their location in a ground truth deformation (left column), and then generating deformations that comply with these control points via diffusion and inpainting. Our predicted results, closely matching the groundtruth, are shown in the rightmost column. We also show the prediction when we remove our custom inpainting approach (**Jac. Def. Img. w/ Baseline Inpaint**) as well as a state-of-the-art inpainting approach, RePaint [108] (**Jac. Def. Img. w/ RePaint**). Both these baselines fail to produce global movement and create singularities around the control points, manifesting as long spikes. We also compare to direct prediction of vertex coordinates in image space (**Coord. Def. Img. w/ Lap. Inpaint**), similar to [16, 57, 115, 168]. This representation cannot handle areas where several triangles are assigned to the same pixel, such as the hands and feet, leading to severe artifacts and loss of detail in these regions.

a $3x3$ real matrix, which is the linear part of the affine map,

$$J_i = V\nabla_i^T, \tag{7.1}$$

where ∇_i is the gradient operator of triangle t_i. Poisson's equation is given by:

$$V = L^{-1}A\nabla^T J, \tag{7.2}$$

where L is the mesh's cotangent Laplacian, A is the mesh's mass matrix, and J is the stacking of each triangle's Jacobian. It returns the deformation with the gradient closest to the predicted gradient; see [4] for the full details.

Diffusion Models. We quickly summarize the necessary technical details of a diffusion generative process. A diffusion model is a probabilistic generative framework [66, 172, 173]. It learns to denoise a signal with added gaussian noise, thereby learning to map the gaussian distribution to the distribution of the training set. The diffusion training process starts from a sample $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, gradually adds noise, and produces $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, which can be described by a Markov chain:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I}).., \tag{7.3}$$

where $\beta_1, \ldots, \beta_T$ is the variance schedule. The diffusion model learns the reverse process as:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\mu_\theta(\boldsymbol{x}_t), \Sigma_\theta(x_t)), \tag{7.4}$$

to reduce the noise from a pure noise sample $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, generating a new sequence $\boldsymbol{x}_{T-1}, \ldots, \boldsymbol{x}_0$.

Training of the posterior model q_θ can be performed by optimizing the variational lower bound; however, we follow the approach of [66], which shows that optimizing a surrogate loss and training a model $\boldsymbol{\epsilon}_\theta$ to predict the ground truth added noise $\boldsymbol{\epsilon}$ for $\boldsymbol{x}_t$ compared to $\boldsymbol{x}_0$ could lead to better results. The simplified loss is:

$$L_{\text{simple}} = \mathop{\mathbb{E}}_{t\sim[1,T],\boldsymbol{x}_0\sim q(\boldsymbol{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|]. \tag{7.5}$$

We also adopt the techniques shown in [127] to learn the posterior variance Σ_θ instead of fixing it as in [66].

Inpainting. For 2D images, it is common to force the model to *inpaint*, i.e., have some pixels be restricted to ones from a given image $\boldsymbol{y}$: let m be a binary mask, with $\mathrm{m}_i = 1$ denoting that pixel i is to be inpainted with the corresponding pixel from $\boldsymbol{y}$, and $\mathrm{m}_i = 0$ denoting pixel i is to be generated freely.

The standard diffusion denoising step samples $\boldsymbol{x}_{t-1}$ via:

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \tag{7.6}$$

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\tilde{\boldsymbol{x}}_t + \Sigma_\theta(\boldsymbol{x}_t, t)\boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{7.7}$$

where β_t is the forward process variance, $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \Pi_{s=1}^{t}\alpha_s$, $\boldsymbol{\epsilon}_\theta$ predicts the noise introduced to $\boldsymbol{x}_t$, $\tilde{\boldsymbol{x}}_t$ is the predicted denoised version of $\boldsymbol{x}_t$, and $\Sigma_\theta(\boldsymbol{x}_t, t)$ is the learned posterior variance.

The standard inpainting approach thus replaces known regions of $\tilde{\boldsymbol{x}}_t$ from Eq. (7.6) with the same regions from the given image $\boldsymbol{y}$ as :

$$\tilde{\boldsymbol{x}}_t = \mathrm{m} \odot \boldsymbol{y} + (1 - \mathrm{m}) \odot \tilde{\boldsymbol{x}}_t, \tag{7.8}$$

where $\odot$ stands for element-wise multiplication. This inpainted $\tilde{\boldsymbol{x}}_t$ is then plugged into Eq. (7.7).

7.3.2 Representing 3D Deformations as 2D Images

We assume that we have a dataset of mesh deformations, with each sample in the dataset consisting of a quadruplet V_0, T, V, U of a mesh (V_0, T), its deformation V, and a UV mapping into 2D, represented by assigning a 2D coordinate to each vertex, $U \in \mathbb{R}^{n \times 2}$. We assume this UV map is semantically consistent between all samples in the dataset, i.e., each mesh of a human has its hand mapped to the same region in UV space.

Using the UV map U, we generate a 2D raster image $\boldsymbol{x}$ such that each pixel represents the mean of all Jacobians of triangles occupying that pixel:

$$\boldsymbol{x}_{h,w} = \begin{cases} \mathbf{0} & |T_{h,w}| = 0, \\ \frac{\sum_{k \in T_{h,w}} J_k}{|T_{h,w}|} & \text{else.} \end{cases} \tag{7.9}$$

where $T_{h,w}$ are all the triangles with centroids inside pixel (h, w) or with that pixel's centroid within them. We denote the process of receiving the deformed vertices V and producing the

deformation image $\boldsymbol{x}$ via f:

$$\boldsymbol{x} = f(V). \tag{7.10}$$

Given such a predicted image $\boldsymbol{x}$, we can extend the predicted Jacobians from the image back to the 3D mesh, again using the UV map, assigning the Jacobian J_i to the triangle t_i via:

$$J_i' = \boldsymbol{x}_{h_i w_i}, \tag{7.11}$$

where the pixel $\boldsymbol{x}_{h_i w_i}$ is the one that contains the centroid of triangle T_i in UV space. Finally, we can use the predicted Jacobians J_i' in a fashion similar to [4] to define a deformation of the mesh by solving Poisson's equation. We denote the process of receiving a deformation image $\boldsymbol{x}$ and outputting the deformed vertices V as:

$$V = g(\boldsymbol{x}). \tag{7.12}$$

With this conversion of operations between images and deformations of a 3D mesh, we can use any standard image-based neural framework. As the main example of this approach, we use diffusion to learn a generative space of deformations.

7.3.3 Inpainting Geometric Constraints

Thus far, we have devised a method that enables training a generative diffusion model to generate plausible deformations of a given mesh. To control this diffusion process, at run time we

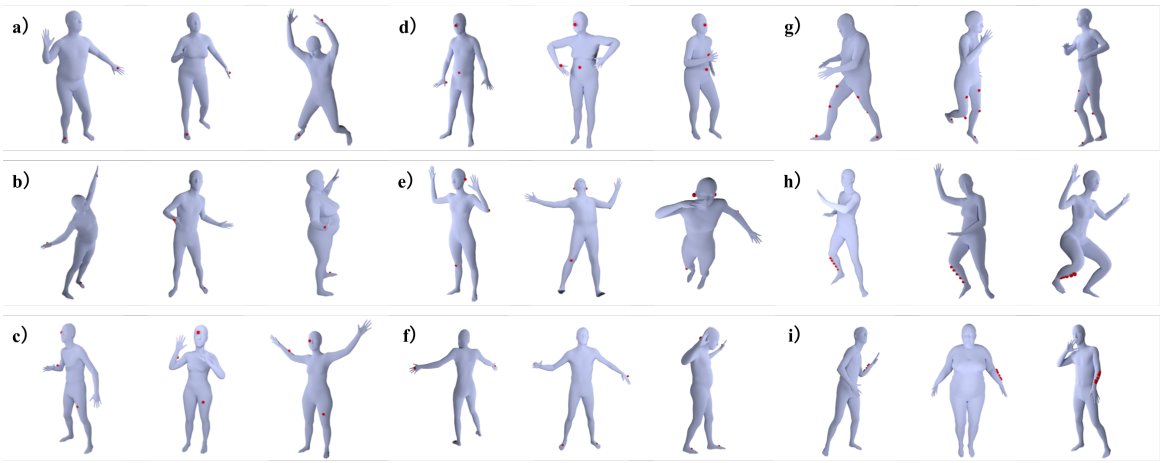


Figure 7.4: We exhibit the ability of our framework to handle arbitrary constraints by choosing arbitrary control point locations and generating deformations concerning them. Our method produces plausible deformations that respect the control points' target positions in all cases, showing that the diffusion process and our custom geometric inpainting approach are effective.

wish to be able to choose specific points on the mesh, $c_1, ..., c_k \in V$, and designate them to lie at specific target positions $P_1, ..., P_k \in \mathbb{R}^3$ (we visualize these constrained points as red spheres in all figures). Thus, we revise the regular image inpainting process (detailed in Sec. 7.3.1) for the 3D mesh case: In this case, the mask m is a vector assigning a binary value to each vertex such that $\mathrm{m}_i = 1$ iff vertex i is one of the constrained vertices c_j, and $\boldsymbol{y} \in \mathbb{R}^{n\times 3}$ holds new positions for each vertex w.r.t $P_1, ...P_k$.

To apply inpainting to our setting, we need to overcome two limitations: 1) The signal the image represents is the deformation *gradients*. However, directly inpainting gradients in specific regions does not allow for controlling the coordinates of the vertices on the mesh. 2) The constraints are extremely sparse (each constraint equivalent to one inpainted pixel), which is proven to be ineffective when applying standard inpainting; see the left column in Fig. 7.3.

We approach these challenges by designing a custom *optimization problem*. This optimization problem introduces an offset to each vertex of the mesh, $\boldsymbol{r} \in \mathbb{R}^{n\times 3}$, and requires that once

the deformation is reconstructed from the given gradient image $\tilde{\boldsymbol{x}}_t$, the offsets $\boldsymbol{r}$ move the target vertex positions exactly to where they should be:

$$\mathrm{m} \odot (g(\tilde{\boldsymbol{x}}_t) + \boldsymbol{r}) = \mathrm{m} \odot \boldsymbol{y}. \tag{7.13}$$

The objective of the optimization is thus to achieve the smoothest (measured through the Dirichlet energy of the deformation via the mesh's cotangent Laplacian L) offsets, which have minimal norm:

$$O(\boldsymbol{r}) = \boldsymbol{r}^T L \boldsymbol{r} + \mu \boldsymbol{r}^T \boldsymbol{r}, \tag{7.14}$$

where μ is a scalar parameter; we set $\mu = 1 \times 10^{-5}$ in all experiments. By minimizing Eq. 7.14 under the constraints of Eq. 7.13, we ensure the resulting deformation correctly "inpaints" the desired vertices while avoiding artifacts that occur due to interpolating from pixels to vertices or due to sparse inpainting constraints (see Figure 7.3).

To define the global translation of the deformation well, we also require that the mesh is centered at zero:

$$\sum_{i=1}^{n} \boldsymbol{r}_i = \mathbf{0}. \tag{7.15}$$

The optimization problem is solved in a standard fashion by a simple linear solve. We build the Lagrangian $\mathcal{L}(\boldsymbol{r})$:

$$O(\boldsymbol{r}) + \lambda_1 \sum_{i=1}^{k} (g(\tilde{\boldsymbol{x}}_t) + \boldsymbol{r} - \boldsymbol{y})_{c_k} + \lambda_2 \sum_{i=1}^{n} \boldsymbol{r}_i, \tag{7.16}$$

which leads to the linear KKT system:

$$\begin{bmatrix} 2(L+\mathbf{I}) & A^T \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ [\boldsymbol{d}_{c_1}, \ldots, \boldsymbol{d}_{c_k}, 0]^T \end{bmatrix}, \tag{7.17}$$

where $\boldsymbol{d}_i = (\boldsymbol{y} - g(\tilde{\boldsymbol{x}}_t))_i$ and $A \in \mathbb{R}^{(k+1)\times n}$ is a matrix representation of the constraints, where in the first k rows only the entries $(1, c_1), (2, c_2), \ldots, (k, c_k)$ are 1 and all the others are 0; the last row all entries are 1.

Solving this KKT system yields the desired offsets $\boldsymbol{r}$, which we add to the original deformation and convert back to gradients in the image domain to obtain

$$\tilde{\boldsymbol{x}}'_t = f(g(\tilde{\boldsymbol{x}}_t) + \boldsymbol{r}), \tag{7.18}$$

which we plug into Eq. 7.7 before continuing as in the regular diffusion process.

7.3.4 Implementation Details

We utilize the framework proposed in [127] to train the diffusion network based on a UNet model [66] combining multi-head attention [197] with four attention heads. We set $T = 1000$ for the diffusion process. For our 128×128 input, we stack five steps of downsampling in the UNet, each with three residual blocks [64]. From the highest to lowest resolution, we use $[C, 2C, 2C, 4C, 4C]$ channels, where $C = 128$. The upsampling has the exact inverse structure to the downsampling, with skip connections from corresponding layers. The output is a 128×128 image with 9 channels per pixel which we interpret as the 3×3 Jacobian. We use Adam [85] with a learning rate of 1×10^{-4} to optimize the network with a batch size of 256. We run the

Metric	Method					
	Coord. Def. Image			Jac. Def. Image		
	w/ Baseline Inpaint	w/ RePaint	w/ Lap. Inpaint	w/ Baseline Inpaint	w/ RePaint	w/ Lap. Inpaint
L2-V	0.243	0.191	0.0188	0.235	0.217	0.0159
L2-N	36.997°	33.497°	18.459°	35.291°	33.494°	14.160°

Table 7.1: The prediction error w.r.t the ground truth, using two metrics: 1) **L2-V**, measuring the mean Euclidean distance for each vertex in meters; 2) **L2-N**, measuring the mean angular difference for each face's normal in degrees. We compare representing the deformation via deformation gradients (**Jac. Def. Image**) vs. direct representation of the 3D coordinates (**Coord. Def. Image**), and for standard baseline inpainting, a current inpainting technique, RePaint [108], and our inpainting method (**Lap. Inpaint**). Our full method (right column) achieves the best performance on both metrics.

training for 50000 epochs, which takes 43 hours on $4\times$ NVIDIA A5000 GPUs. The timing for the generation of 32 samples with our inpainting method is 20 minutes for our approach run on one NVIDIA A5000 GPU. This is mostly due to the diffusion process itself (19 minutes); while our custom inpainting method adds one minute. In comparison, RePaint [108] takes 46 minutes to run.

7.4 Experimental Results

7.4.1 Datasets

We use the AMASS [114] dataset which has many human subjects performing various different sequences. We choose one subset from MPI-MoSh [106] that has 19 subjects with different genders and body shapes. We select 70% of the sequences as the training set, which results in 75749 samples, with each sample representing the deformation of the human from its T-pose. From the remaining data, we randomly select 4096 samples as our test set. We then generate the 128×128 deformation raster image, with each pixel holding a 3×3 matrix

representing the Jacobian at the triangles matching that pixel via f, as described in Eq. (7.10).

7.4.2 Comparison and Ablation

We begin by comparing our method to alternatives and ablating the different components of it showing quantitative and qualitative results in Table 7.1 and Figure 7.3. Since the diffusion process is non-deterministic, to compare it to ground truths we ensure its output is well-defined w.r.t a ground-truth deformation: we carefully choose 30 constrained points on the mesh such that these constraints uniquely define the deformation of the human and then generate the constrained deformation of the human, comparing it to the ground truth.

We compare our representation of deformations via Jacobian-based deformation images with a baseline that uses coordinate-based geometry images. For each of these two representations, we also test the results combined with two different inpainting methods: 1) a baseline method that does not use our inpainting regularizer (Eq. 7.18) but instead uses the standard inpainting method (Eq. 7.8) used in other image diffusion papers:

$$\tilde{\boldsymbol{x}}'_t = f(\mathrm{m} \odot \boldsymbol{y} + (1 - \mathrm{m}) \odot g(\tilde{\boldsymbol{x}}_t)). \tag{7.19}$$

We also compare our method to a recent method, RePaint [108] which proposes mixing the diffusion step (Eq. 7.3) and the reverse step in the sampling process to incorporate the semantic information from the known region over the entire denoising process and improves inpainting. For RePaint, we choose parameters as follows: $T' = 250$ with respacing during sampling and 10 times resampling with jump size 10. For more details, please see [108].

For each method, we compare its prediction with the ground truth deformation from which

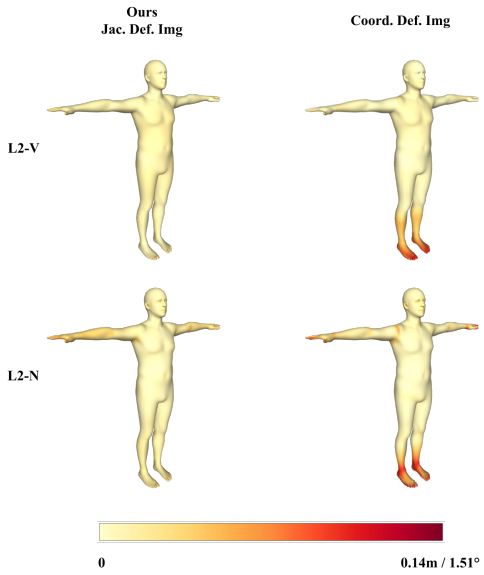


Figure 7.5: Heat map of the average error. The baseline method, using vertex coordinates as in Geometry Images [16, 57, 115, 168], exhibits significantly higher error in the areas that have a dense set of triangles occupying one pixel (hands and feet).

the control points' locations were taken. We use two metrics: **L2-V** compares the mean Euclidean distance for each vertex; **L2-N** compares the mean angular difference for each face's normal, in degrees. The results are reported in Table 7.1, with qualitative visualization in Fig. 7.3. Additionally, Fig. 7.5 shows the error distribution of the predictions over the T-posed template via a heatmap.

As is verified by the quantitative comparisons, our method is the most accurate. The qualitative results further reveal that our method (Fig. 7.3, right col., **Jac. Def. Img. w/ Lap. Inpaint**) can better recover the correct deformation while retaining geometric details than others. Using our full approach outputs smoother and more reasonable surfaces in areas with denser triangulations, i.e., hands and feet (in which a cluster of triangles occupies one pixel), in comparison to coordinate-based geometry images (Fig. 7.3, **Vert. Def. Img. w/ Lap. Inpaint**). Neither the baseline inpainting (Fig. 7.3, **Jac. Def. Img. w/ Baseline Inpaint**) nor RePaint (Fig. 7.3, **Jac.**

Def. Img. w/ RePaint) can account for the desired constraints, instead ignoring the inpainted vertices, except for a local neighborhood around them, producing spikes.

7.4.3 Deformation from Arbitrary Constraints

Approaching deformations via diffusion and inpainting enables us to choose any subset of vertices and assign them new positions to get a deformation that respects those constraints. We test our method's ability to deform human models w.r.t arbitrary control point combinations and show the results in Fig. 7.4. We show extreme cases with only two points (**a** and **b**) as well as non-joint locations such as forearm, nose, ears, and belly button (**c**, **d**, and **e**). In **f**, we use the extremities to roughly define a human pose. In **g**, we test our method's ability to generate various lower body poses. In **h** and **i**, we use dense constraints to provide fine-grained control.

Notice that, by only modifying the sampling process, our method does not require any pre-training process for any specific case, nor do we need to carefully design possible inpainting masks to use in pretraining as is done in [126, 156].

7.4.4 Invariance to Mesh Resolution/Triangulation

Due to the representation of deformations via Jacobians, at test time, our framework can handle arbitrary triangulations, even if they have high resolution that leads to multiple triangles assigned to the same pixel in the image, if it has a consistent UV map as the training set. We show two examples of higher resolution meshes and their deformations in Fig. 7.6, validating that our method can deform them accurately in spite of their much higher resolution.

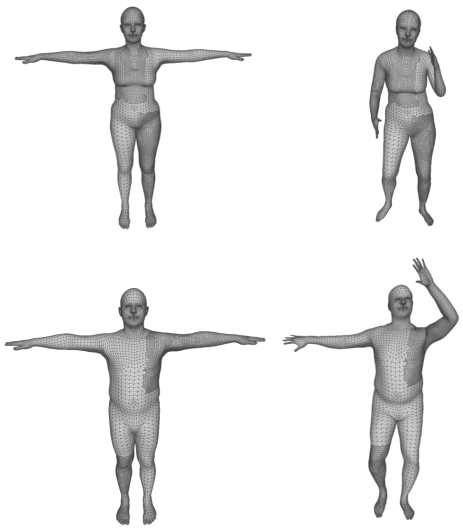


Figure 7.6: Our trained framework is readily applicable to unseen, higher-resolution triangulations (left) and deforms them accurately (right).

7.5 Conclusion, Limitations, and Future Work

Deformation Images are a novel representation allowing for representing deformation of high-resolution meshes without loss of detail through 2D images. The image-based representation directly leverages standard image-based techniques, such as diffusion, which we show can be used to generate deformations of humans into arbitrary poses. As a secondary contribution, we propose a method to enable inpainting in this setting, even in cases where the inpainted regions are extremely sparse.

Since diffusion processes require massive amounts of data, we focused on one of the only large datasets of 3D deformations for human models. However, this is in essence a didactic experiment, designed to validate the capability of our framework for generative deformation tasks without loss of detail or accuracy. Considering the many compact representations for human poses and wealth of research on this, there are many ways to generate human poses, e.g., by controlling a low-dimension skeletal rig.

Once additional large datasets of deformations of other classes are authored, we believe our method can be used in impactful applications. One interesting extension is to consider temporal motion as a sequence of deformations. Given a large collection of arbitrary objects in arbitrary motion (e.g., assets from an animated film), our method can learn to generate motion for new objects at test time, where the ability to inpaint and control the motion will further give artists granular control over the generated motion. In different contexts, our framework can be trained on physical systems such as objects in elastic equilibrium, then used to predict the equilibrium for new objects under different constraints.

The limitations of our method are first inherited from the image diffusion process, which requires massive amounts of data as well as extensive training times and is slow to produce predictions. Because the research into diffusion is only at its early stages, and that our representation enables plugging arbitrary image-based diffusion architectures and training them without any changes to the framework or code, our method will immediately benefit from faster image-based diffusion approaches once they are devised. A second limitation of our method is that while the resolution of the image does not ruin geometric detail or prevent us from representing low-frequency deformations such as articulations, it does prohibit us from representing more fine-grained ones such as local edits to add fine details to the models. Applying current image-based techniques for super-resolution may be an efficient way to overcome this barrier, and we mark it as important future work.

Chapter 8: Discussion

This dissertation presents novel methodologies for learning-based physics simulations and collision-handling algorithms that capitalize on the strengths of neural networks and optimization techniques. By overcoming the computational limitations of traditional analytic and numerical methods, we have successfully incorporated machine learning approaches, achieving enhanced efficiency without significant loss in accuracy. Our work promises high-quality visual results in real-time applications such as online gaming and virtual reality, where simulation speed is crucial.

8.1 Summary of Results

In Chapter 3 and Chapter 4, we introduce our work on LCollision [182] and N-Penetrate [183], which involves the development of a learning-based collision detector for 3D deformable models. Our system has demonstrated a promising collision detection accuracy of up to 98.1%. By leveraging active learning, our system robustly predicts collisions in large and previously unseen spaces, illustrating the effectiveness of our learning-based approach. Furthermore, our optimization-based collision handler successfully resolves collisions involving randomly generated 3D deformable materials in a fraction of a second with a success rate exceeding 90%. This demonstrates the substantial benefits of integrating neural networks into conventional optimiza-

tion techniques.

In Chapter 5, we introduce a technique to expedite collision response computations by incorporating an additional repulsive force unit, ReFU [184], into the learning-based pipeline. Our experiments reveal that this inclusion significantly reduces collisions between the garments and the underlying bodies, elevating collision-free models from 49% to 77%. This novel method thus represents a substantial improvement in our simulation's real-time performance, highlighting the power of a learning-based approach.

In Chapter 6, we showcase our third contribution, a neural volumetric deformable object simulator, NPC-Net, which utilizes an actor-critic neural architecture for efficient collision detection and handling. This system's learning capabilities result in almost collision-free quasistatic deformable object poses, further enhancing the realism and accuracy of our simulations. Significantly, this approach offers a $100-300\times$ speed improvement over traditional numerical methods.

Finally, in Chapter 7, we introduce a pioneering framework for reposing 3D human models to arbitrary poses. This framework, based on a geometric optimization regularization, successfully integrates control information into diffusion-based inpainting. Our novel algorithm leads to a 93% reduction in errors when reposing various body parts, thereby opening new avenues for applications requiring precise and realistic 3D simulations.

Through rigorous testing on large-scale datasets such as AMASS [114] for human models and TailorNet [138] for garments, we have confirmed the practical utility and accuracy of our learning-based physics simulation systems. By generating realistic and plausible results at speeds significantly exceeding those of numerical and analytic methods, our techniques open new possibilities in physics-based simulation, particularly for high-dimensional, real-time applications.

8.2 Liminations and Future Work

Despite these promising advancements, certain limitations remain, providing a clear direction for future research.

Sequence Generation: All our work focuses on static simulations. However, many real-world applications such as cloth animation and robot control require dynamic simulations. Therefore, we plan to extend our work to dynamic simulations. One challenge for dynamic simulation using learning methods is to limit the accumulation of errors. For example, in our work, we use a neural network to predict the collision-free pose. However, the predicted pose may be not exactly collision-free. If we use this pose as the initial pose for the next frame, the error will accumulate. One possible solution is leveraging the generation power of diffusion networks, which start the process from random signals. These networks may recover some errors from the imperfect approximations of previous frames and avoid disastrous error explosions.

Neural Numerical Hybrid Simulation: Our works improve the performance of learning-based physical simulators. However, they can easily lose accuracy when the inputs are significantly different from the training set. Although better learning methods aim to match the accuracy of exact analytic and numerical methods, they will not be the same. Therefore, we propose combining the learning-based and traditional methods to achieve the best of both worlds. For example, we can use the learning-based method to predict a rough solution and reduce the search space, and the numerical method works on that to find the exact solution. As we continue to integrate machine learning with traditional physics-based simulation techniques, we are optimistic that our work will lead to significant progress in the field.

Bibliography

[1] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):44–58, 2005.

[2] Hamed H Aghdam, Abel Gonzalez-Garcia, Joost van de Weijer, and Antonio M López. Active learning for deep detection neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3672–3680, 2019.

[3] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pages 9558–9570, 2019.

[4] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *SIGGRAPH*, 2022.

[5] Noam Aigerman and Yaron Lipman. Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics (TOG)*, 32(4):1–14, 2013.

[6] Marc Alexa and Wolfgang Müller. Representing animations by principal components. *Comp. Graph. Forum*, 19(3):411–418, 2000.

[7] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imghum: Implicit generative models of 3d human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5461–5470, 2021.

[8] Ron Alterovitz, Michael Branicky, and Ken Goldberg. Motion planning under uncertainty for image-guided medical needle steering. *IJRR*, 27(11-12):1361–1374, 2008.

[9] Ron Alterovitz, Kenneth Y Goldberg, Jean Pouliot, and I-Chow Hsu. Sensorless motion planning for medical needle insertion in deformable tissues. *IEEE Transactions on Information Technology in Biomedicine*, 13(2):217–225, 2009.

[10] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH*, pages 408–416. 2005.

[11] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3877–3886, 2018.

[12] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[13] Jernej Barbič and Doug L James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM transactions on graphics (TOG)*, 24(3):982–990, 2005.

[14] Jernej Barbič and Doug L. James. Subspace self-collision culling. *ACM Trans. Graph.*, 29(4):81:1–81:9, 2010.

[15] Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

[16] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[17] Jur van den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[18] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*, pages 1–12. 2008.

[19] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: Clothed 3d humans. In *European Conference on Computer Vision*, pages 344–359. Springer, 2020.

[20] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: Physically based neural simulator for unsupervised garment pose space deformation. *arXiv preprint arXiv:2012.11310*, 2020.

[21] Hugo Bertiche, Meysam Madadi, Emilio Tylson, and Sergio Escalera. Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5471–5480, 2021.

[22] 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/CVF International Conference on Computer Vision*, pages 5420–5430, 2019.

[23] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[24] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*, 2014.

[25] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7213–7222, 2019.

[26] C. A. Brebbia and M. H. Aliabadi, editors. *Industrial Applications of the Boundary Element Method*. 1993.

[27] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3):594–603, 2002.

[28] Tyson Brochu, Essex Edwards, and Robert Bridson. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph.*, 31(4):96:1–96:7, 2012.

[29] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[30] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):604–611, July 2002.

[31] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, 2005.

[32] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

[33] Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael J Black. Capture, learning, and synthesis of 3d speaking styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10101–10111, 2019.

[34] Y D. Li, Min Tang, Yun Yang, Zi Huang, R F. Tong, SC Yang, Yao Li, and Dinesh Manocha. N-cloth: Predicting 3d cloth deformation with mesh-based networks. In *Computer Graphics Forum*, volume 41, pages 547–558. Wiley Online Library, 2022.

[35] Nikhil Das, Naman Gupta, and Michael Yip. Fastron: An online learning-based model and active learning strategy for proxy collision detection. In *Conference on Robot Learning*, pages 496–504. PMLR, 2017.

[36] 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.

[37] Luc De Raedt, Andrea Passerini, and Stefano Teso. Learning constraints from examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[38] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *The European Conference on Computer Vision (ECCV)*. Springer, August 2020.

[39] Christian Duriez. Control of Elastic Soft Robots based on Real-Time Finite Element Method. In *IEEE ICRA*, Karlsruhe, France, 2013.

[40] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, pages 2224–2232, 2015.

[41] Susan Fisher and Ming C Lin. Fast penetration depth estimation for elastic bodies using deformed distance fields. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 330–336. IEEE, 2001.

[42] Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. *Computer Graphics Forum*, 2019.

[43] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[44] Juergen Gall, Carsten Stoll, Edilson De Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In

Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1746–1753. IEEE, 2009.

[45] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, 2015.

[46] Lin Gao, Yu-Kun Lai, Jie Yang, Zhang Ling-Xiao, Shihong Xia, and Leif Kobbelt. Sparse data driven mesh deformation. *IEEE transactions on visualization and computer graphics*, 2019.

[47] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[48] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[49] Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. Optimization integrator for large time steps. *IEEE TVCG*, 21(10):1103–1115, 2015.

[50] Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*, 2017.

[51] A. George and E. Ng. On the complexity of sparse qr and lu factorization of finite-element matrices. *SIAM Journal on Scientific and Statistical Computing*, 9(5):849–861, 1988.

[52] 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.

[53] Naga K. Govindaraju, Ming C. Lin, and Dinesh Manocha. Quick-CULLIDE: Fast inter- and intra-object collision culling using graphics hardware. In *IEEE Virtual Reality Conference 2005, VR 2005, Bonn, Germany, March 12-16, 2005*, pages 59–66, 2005.

[54] Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67. Citeseer, 2003.

[55] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[56] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[57] Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361, 2002.

[58] Peng Guan, Loretta Reiss, David A Hirshberg, Alexander Weiss, and Michael J Black. Drape: Dressing any person. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012.

[59] 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/CVF International Conference on Computer Vision*, pages 8739–8748, 2019.

[60] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[61] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[62] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Trans. Graph.*, 27(3):23:1–23:4, 2008.

[63] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2282–2292, 2019.

[64] 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.

[65] Liang He, Jia Pan, Danwei Li, and Dinesh Manocha. Efficient penetration depth computation between rigid models using contact space propagation sampling. *IEEE Robotics and Automation Letters*, 1(1):10–17, 2015.

[66] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[67] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[68] 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.

[69] Daniel Holden, Jun Saito, and Taku Komura. Learning an inverse rig mapping for character animation. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 165–173, New York, NY, USA, 2015. Association for Computing Machinery.

[70] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011.

[71] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014.

[72] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12783–12792, 2021.

[73] Doug L James and Dinesh K Pai. Bd-tree: Output-sensitive collision detection for reduced deformable models. In *ACM SIGGRAPH 2004 Papers*, pages 393–398. 2004.

[74] Biao Jia, Zherong Pan, Zhe Hu, Jia Pan, and Dinesh Manocha. Cloth manipulation using random-forest-based imitation learning. *IEEE Robotics and Automation Letters*, 4(2):2086–2093, 2019.

[75] Biao Jia, Zherong Pan, and Dinesh Manocha. Fast motion planning for high-dof robot systems using hierarchical system identification, 2018.

[76] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, Leonidas Guibas, et al. Shapeflow: Learnable deformations among 3d shapes. *arXiv preprint arXiv:2006.07982*, 2020.

[77] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *SIGGRAPH*, 2005.

[78] Angjoo Kanazawa, Shahar Kovalsky, Ronen Basri, and David Jacobs. Learning 3d deformation of animals from 2d images. In *Computer Graphics Forum*, volume 35, pages 365–374. Wiley Online Library, 2016.

[79] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008.

[80] Hoel Kervadec, Jose Dolz, Meng Tang, Eric Granger, Yuri Boykov, and Ismail Ben Ayed. Constrained-cnn losses for weakly supervised segmentation. *Medical image analysis*, 54:88–99, 2019.

[81] 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*, 32(4):87:1–7, July 2013. Proceedings of ACM SIGGRAPH 2013, Anaheim.

[82] Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. Data-driven physics for human soft tissue animation. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[83] Y Kim, M Lin, and D Manocha. Collision and proximity queries. *Handbook of Discrete and Computational Geometry*, 2018.

[84] Young J Kim, Miguel A Otaduy, Ming C Lin, and Dinesh Manocha. Fast penetration depth computation for physically-based animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 23–31, 2002.

[85] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[86] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[87] 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/CVF International Conference on Computer Vision*, pages 2252–2261, 2019.

[88] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[89] 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.

[90] Mats G. Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*. Springer Publishing Company, Incorporated, 2013.

[91] John Lasseter. Tricks to animating characters with a computer. *ACM Siggraph Computer Graphics*, 35(2):45–47, 2001.

[92] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 878–885. IEEE, 2005.

[93] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. Incremental potential contact: Intersection- and inversion-free large deformation dynamics. *ACM Trans. Graph. (SIGGRAPH)*, 39(4), 2020.

[94] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)*, 40(4):1, 2021.

[95] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *ICLR*, 2019.

[96] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *ICRA*, 2019.

[97] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[98] Yi-Je Lim and Suvranu De. Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres. *Computer Methods in Applied Mechanics and Engineering*, 196(31-32):3011–3024, 2007.

[99] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022.

[100] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008.

[101] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, page 1495–1504, Goslar, DEU, 2008. Eurographics Association.

[102] Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[103] Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12–21, 2021.

[104] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Trans. Graph.*, 32(6):209:1–7, Nov. 2013.

[105] Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph.*, 36(3), May 2017.

[106] Matthew Loper, Naureen Mahmood, and Michael J. Black. MoSh: Motion and Shape Capture from Sparse Markers. *ACM Trans. Graph.*, 33(6), Nov. 2014.

[107] 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.

[108] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

[109] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[110] Anatolii Isakovich Lurie. *Theory of elasticity*. Springer Science & Business Media, 2010.

[111] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. *arXiv preprint arXiv:2011.13495*, 2020.

[112] Pingchuan Ma, Yunsheng Tian, Zherong Pan, Bo Ren, and Dinesh Manocha. Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

[113] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse. Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1):1–17, 2020.

[114] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[115] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017.

[116] Pablo Marquez Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. 2017.

[117] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ IROS*, pages 922–928, Sept 2015.

[118] Ullrich Meier, Oscar López, Carlos Monserrat, Mari C Juan, and M Alcaniz. Real-time deformable models for surgery simulation: a survey. *Computer methods and programs in biomedicine*, 77(3):183–197, 2005.

[119] Marko Mihajlovic, Yan Zhang, Michael J Black, and Siyu Tang. Leap: Learning articulated occupancy of people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10461–10471, 2021.

[120] Marko Mihajlovic, Yan Zhang, Michael J Black, and Siyu Tang. LEAP: Learning articulated occupancy of people. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[121] Lea Muller, Ahmed A. A. Osman, Siyu Tang, Chun-Hao P. Huang, and Michael J. Black. On self-contact and human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9990–9999, June 2021.

[122] Yatin Nandwani, Abhishek Pathak, Parag Singla, et al. A primal dual formulation for deep learning with constraints. In *Advances in Neural Information Processing Systems*, pages 12157–12168, 2019.

[123] Rahul Narain, Matthew Overby, and George E Brown. Admm projective dynamics: fast simulation of general constitutive models. In *Symposium on Computer Animation*, volume 1, page 2016, 2016.

[124] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6):147:1–10, Nov. 2012.

[125] Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. Sparse localized deformation components. *ACM Trans. Graph.*, 32(6):179:1–179:10, Nov. 2013.

[126] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[127] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[128] Radu Stefan Niculescu, Tom M Mitchell, R Bharat Rao, Kristin P Bennett, and Emilio Parrado-Hernández. Bayesian network learning with parameter constraints. *Journal of machine learning research*, 7(7), 2006.

[129] Young Jin Oh, Tae Min Lee, and In-Kwon Lee. Hierarchical cloth simulation using deep neural networks. *arXiv:1802.03168*, 2018.

[130] Ahmed AA Osman, Timo Bolkart, and Michael J Black. Star: Sparse trained articulated human body regressor. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 598–613. Springer, 2020.

[131] Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum*, 28(2):559–568, 2009.

[132] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012.

[133] Jia Pan, Xinyu Zhang, and Dinesh Manocha. Efficient penetration depth approximation using active learning. *ACM Transactions on Graphics (TOG)*, 32(6):1–12, 2013.

[134] Z. Pan and D. Manocha. Realtime planning for high-dof deformable bodies using two-stage learning. In *IEEE ICRA*, pages 1–8, May 2018.

[135] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[136] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[137] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

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

[139] Rohan Paul, Dan Feldman, Daniela Rus, and Paul Newman. Visual precis generation using coresets. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1304–1311. IEEE, 2014.

[140] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *CVPR*, pages 459–468, 2018.

[141] T. Pham, G. De Magistris, and R. Tachibana. Optlayer - practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243, 2018.

[142] Elijah Polak and Gerard Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 3(16):35–43, 1969.

[143] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

[144] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics Interface*, pages 177–189, 1997.

[145] 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.

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

[147] Yi-Ling Qiao, Yu-Kun Lai, Hongbo Fu, and Lin Gao. Synthesizing mesh deformation sequences with bidirectional lstm. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2020.

[148] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168*, 2020.

[149] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[150] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[151] Sathya N Ravi, Tuan Dinh, Vishnu Suresh Lokhande, and Vikas Singh. Explicitly imposing constraints in deep networks via conditional gradients gives improved generalization and faster convergence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4772–4779, 2019.

[152] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6182–6190, 2019.

[153] Grégory Rogez, Jonathan Rihan, Srikumar Ramalingam, Carlos Orrite, and Philip HS Torr. Randomized trees for human pose detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[154] Cristian Romero, Dan Casas, Maurizio M. Chiaramonte, and Miguel A. Otaduy. Contact-centric deformation learning. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 41(4), 2022.

[155] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[156] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[157] Pedro V Sander, Zoë J Wood, Steven Gortler, John Snyder, and Hugues Hoppe. Multi-chart geometry images. 2003.

[158] Igor Santesteban, Miguel Otaduy, Nils Thuerey, and Dan Casas. Ulnef: Untangled layered neural fields for mix-and-match virtual try-on. *Advances in Neural Information Processing Systems*, 35:12110–12125, 2022.

[159] Igor Santesteban, Miguel A Otaduy, and Dan Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, volume 38, pages 355–366. Wiley Online Library, 2019.

[160] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-supervised collision handling via generative 3d garment models for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[161] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[162] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. In *Computer Graphics Forum*, volume 32, pages 125–135. Wiley Online Library, 2013.

[163] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[164] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

[165] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[166] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. In *ACM SIGGRAPH 2007 papers*, pages 81–es. 2007.

[167] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[168] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 223–240. Springer, 2016.

[169] Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020.

[170] Breannan Smith, Fernando De Goes, and Theodore Kim. Stable neo-hookean flesh simulation. *ACM Transactions on Graphics (TOG)*, 37(2):1–15, 2018.

[171] Breannan Smith, Chenglei Wu, He Wen, Patrick Peluse, Yaser Sheikh, Jessica K Hodgins, and Takaaki Shiratori. Constraining dense hand surface tracking with elasticity. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[172] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[173] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[174] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116, 2007.

[175] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[176] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.

[177] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[178] Wenyu Sun and Ya-Xiang Yuan. *Optimization theory and methods: nonlinear programming*, volume 1. Springer Science & Business Media, 2006.

[179] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5841–5850, 2018.

[180] Qingyang Tan, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia. Mesh-based autoencoders for localized deformation component analysis. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[181] 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.

[182] Qingyang Tan, Zherong Pan, and Dinesh Manocha. Lcollision: Fast generation of collision-free human poses using learned non-penetration constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3913–3921, 2021.

[183] Qingyang Tan, Zherong Pan, Breannan Smith, Takaaki Shiratori, and Dinesh Manocha. Npenetrate: Active learning of neural collision handler for complex 3d mesh deformations. In *International Conference on Machine Learning*, pages 21037–21049. PMLR, 2022.

[184] Qingyang Tan, Yi Zhou, Tuanfeng Wang, Duygu Ceylan, Xin Sun, and Dinesh Manocha. A repulsive force unit for garment collision handling in neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 451–467. Springer, 2022.

[185] Min Tang, Sean Curtis, Sung-Eui Yoon, and Dinesh Manocha. Iccd: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):544–557, 2009.

[186] Min Tang, Dinesh Manocha, Miguel A Otaduy, and Ruofeng Tong. Continuous penalty forces. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.

[187] Min Tang, Dinesh Manocha, and Ruofeng Tong. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 7–13, 2010.

[188] Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. Fast and exact continuous collision detection with Bernstein sign classification. *ACM Trans. Graph. (SIGGRAPH Asia)*, 33:186:1–186:8, November 2014.

[189] Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. I-Cloth: Incremental collision handling for GPU-based interactive cloth simulation. *ACM Trans. Graph.*, 37(6):204:1–10, November 2018.

[190] Yun Teng, Miguel A Otaduy, and Theodore Kim. Simulating articulated subspace self-contact. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.

[191] Hao Tian, Xinyu Zhang, Changbo Wang, Jia Pan, and Dinesh Manocha. Efficient global penetration depth computation for articulated models. *Computer-Aided Design*, 70:116–125, 2016.

[192] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[193] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. *European Conference on Computer Vision (ECCV)*, 2020.

[194] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-aware 3d model embedding and retrieval. In *European Conference on Computer Vision*, pages 397–413. Springer, 2020.

[195] Vesa Välimäki and Tapio Takala. Virtual musical instruments—natural sound using physical models. *Organised Sound*, 1(2):75–86, 1996.

[196] Robert J Vanderbei. Loqo user's manual—version 3.10. *Optimization methods and software*, 11(1-4):485–514, 1999.

[197] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[198] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8639–8648, 2018.

[199] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008.

[200] Huamin Wang. Defending continuous collision detection against errors. *ACM Trans. Graph. (SIGGRAPH)*, 33(4):122:1–122:10, July 2014.

[201] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022.

[202] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE TPAMI*, 2008.

[203] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[204] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based CNNs with output-guided skip connections for 3D shape and scene completion. 2020.

[205] Tuanfeng Y. Wang, Duygu Ceylan, Jovan Popovic, and Niloy J. Mitra. Learning a shared shape space for multimodal garment design. *ACM Trans. Graph.*, 37(6):1:1–1:14, 2018.

[206] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. *Advances in Neural Information Processing Systems*, 30:7029–7039, 2017.

[207] Zhe Wanga, Liyan Chena, Shaurya Rathorea, Daeyun Shina, and Charless Fowlkesa. Geometric pose affordance: Monocular 3d human pose estimation with scene constraints. *arXiv preprint arXiv:1905.07718*, 2019.

[208] Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent-space physics: Towards learning the temporal evolution of fluid flow. *arXiv:1802.10123*, 2018.

[209] David Wolinski, S J. Guy, A-H Olivier, Ming Lin, Dinesh Manocha, and Julien Pettré. Parameter estimation and comparative evaluation of crowd simulations. In *Computer Graphics Forum*, volume 33, pages 303–312. Wiley Online Library, 2014.

[210] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[211] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[212] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Trans. Graph.*, 37(4):95, 2018.

[213] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020.

[214] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 International Conference on 3D Vision (3DV)*, 2019.

[215] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *arXiv:1612.00814*, 2016.

[216] Jie Yang, Lin Gao, Qingyang Tan, Yihua Huang, Shihong Xia, and Yu-Kun Lai. Multiscale mesh deformation component analysis with attention-based autoencoders, 2020.

[217] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Lin Gao. Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry. *arXiv preprint arXiv:2008.05440*, 2(3), 2020.

[218] Chih-Yuan Yao and Tong-Yee Lee. Adaptive geometry image. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):948–960, 2008.

[219] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020.

[220] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9491–9500, 2019.

[221] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[222] Ryan S Zesch, Bethany R Witemeyer, Ziyan Xiong, David IW Levin, and Shinjiro Sueda. Neural collision detection for deformable objects. *arXiv preprint arXiv:2202.02309*, 2022.

[223] Liangjun Zhang, Young J Kim, Gokul Varadhan, and Dinesh Manocha. Generalized penetration depth computation. *Computer-Aided Design*, 39(8):625–638, 2007.

[224] Xinyu Zhang, Young J Kim, and Dinesh Manocha. Continuous penetration depth. *Computer-Aided Design*, 46:3–13, 2014.

[225] Zhijun Zhang, Lunan Zheng, Zhuoming Chen, Lingdong Kong, and Hamid Reza Karimi. Mutual-collision-avoidance scheme synthesized by neural networks for dual redundant robot manipulators executing cooperative tasks. *IEEE transactions on neural networks and learning systems*, 32(3):1052–1066, 2020.

[226] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4, 2015.

[227] Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbič. A deep emulator for secondary motion of 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5932–5940, 2021.

[228] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5826–5835, October 2021.

[229] Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *International Conference on Learning Representations*, 2018.

[230] Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *arXiv preprint arXiv:2006.04325*, 2020.

[231] J. Zhu, S. C. H. Hoi, and M. R. Lyu. Nonrigid shape recovery by gaussian process regression. In *IEEE CVPR*, pages 1319–1326, June 2009.

[232] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *J. Comp. Graph. Statistics*, 15:2006, 2004.