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

Qingyang Tan[1], **Zherong Pan**[2], **Dinesh Manocha**[1]

[1] Department of Computer Science, University of Maryland at College Park
[2] Department of Computer Science, University of Illinois at Urbana-Champaign
qytan@umd.edu, zherong@illinois.edu, dmanocha@umd.edu

Abstract

We present LCollision, a learning-based method that synthesizes collision-free 3D human poses. At the crux of our approach is a novel deep architecture that simultaneously decodes new human poses from the latent space and predicts colliding body parts. These two components of our architecture are used as the objective function and surrogate hard constraints in a constrained optimization for collision-free human pose generation. A novel aspect of our approach is the use of a bilevel autoencoder that decomposes whole-body collisions into groups of collisions between localized body parts. By solving the constrained optimizations, we show that a significant amount of collision artifacts can be resolved. Furthermore, in a large test set of 2.5×10^6 randomized poses from SCAPE, our architecture achieves a collision-prediction accuracy of 94.1% with $80\times$ speedup over exact collision detection algorithms. To the best of our knowledge, LCollision is the first approach that accelerates collision detection and resolves penetrations using a neural network.

1 Introduction

There has been considerable work on developing learning algorithms for 3D objects represented as point clouds (Qi et al. 2017), meshes (Hanocka et al. 2019), volumetric grids (Wang, Liu, and Tong 2020), and physical objects (Li et al. 2019). 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 (Qiao et al. 2020) and cloth embedding (Tan et al. 2020), and a reliable robot motion planner should preserve a clearance distance from obstacles (Pham, De Magistris, and Tachibana 2018).

In this paper, we tackle the problem of collision-free human pose generation. Recently, 3D mesh representations have been used for learning-based human pose synthesis (Tretschk et al. 2020; Bouritsas et al. 2019; Ranjan et al. 2018; Bagautdinov et al. 2018; Tan et al. 2018a). 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 (Tretschk et al. 2020; Bouritsas et al. 2019; Ranjan et al. 2018; Bagautdinov et al. 2018; Tan et al. 2018a). By comparison, non-learning-based methods for character rigging (Shi et al. 2007) and physics-based simulation (Barbič and James 2010) 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 (Boggs and Tolle 1995) for constrained optimization can enforce exact hard constraints on the parameters of the neural network. Second, variants of the stochastic projected gradient descend (Márquez-Neila, Salzmann, and Fua 2017; Kervadec et al. 2019) have been proposed to approximately satisfy the constraints on the neural network parameters. Finally, differentiable optimization layers (Pham, De Magistris, and Tachibana 2018; Agrawal et al. 2019) 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 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 (Zhang et al. 2007) 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 (Anguelov et al. 2005), the MIT-Swing dataset (Vlasic et al. 2008), and the MIT Jumping dataset (Vlasic et al. 2008). 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 (Pan, Chitta, and Manocha 2012).

2 Related Work

We review related works on human pose estimation and synthesis, collision detection and response, and deep network training with hard constraints.

Human Pose Estimation & Synthesis: There is considerable work on human pose estimation and synthesis. Earlier methods (Leibe, Seemann, and Schiele 2005) represent a pedestrian as a bounding box. An improved algorithm was proposed in (Agarwal and Triggs 2005), and this algorithm predicts the 55-D joint angles for a skeletal human pose. More accurate prediction results have been proposed in (Rogez et al. 2008) using random forests and in (Toshev and Szegedy 2014) using convolutional neural networks. Our approach is based on recent learning methods (Tan et al. 2018a; Tretschk et al. 2020) 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, (Villegas et al. 2018) 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 (Anguelov et al. 2005).

Collision Detection & Response: An important criterion of "correct" human body shapes is that they are (self-) collision-free, i.e. elements of the mesh do not penetrate each other. Collision detection and response computations have been well-studied, with many practical algorithms proposed for large-scale 3D meshes (Pan, Chitta, and Manocha 2012; Kim, Lin, and Manocha 2018) that can be used to resolve penetrations. Collisions can be handled in a discrete or continuous manner. Discrete collision handling (Kim, Lin, and Manocha 2018) 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 (Tang et al. 2009; Bridson, Fedkiw, and Anderson 2002; Tang, Manocha, and Tong 2010) 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 (Govindaraju, Lin, and Manocha 2005). In theory, we can use different collision handling methods to avoid penetrations in a 3D mesh of a human pose. However, there are two practical challenges. First, collision response is a physical behavior tightly coupled with a physics-based model of the human body. However, modeling the physical deformations of a human body can be computationally expensive. The running time for simulating one timestep of a human body can be more than 20 seconds (Smith, Goes, and Kim 2018), which is intractable for interactive applications. Second, collision handling algorithms require a volumetric mesh, while many applications of human pose synthesis rely on surface meshes. Techniques have also been proposed to estimate the extent of penetrations between complex 3D geometric models, e.g., penetration depth (Zhang et al. 2007; Kim et al. 2002; Burgard, Brock, and Stachniss 2008). However, these formulations can be non-smooth and expensive to compute.

Training Deep Networks with Hard Constraints: An additional layer of challenge is to incorporate collision handling into a deep learning framework. In particular, state-of-the-art deep learning methods are unable to handle such hard constraints. Constraints on neural network parameters (Ravi et al. 2019) are used for regularizing the network training and can be approximately enforced using variants of the projected gradient descend algorithm. On the other hand, constraints on neural network output are used to model application-specific requirements such as collision-free constraints. Prior works (Pham, De Magistris, and Tachibana 2018; Agrawal et al. 2019; Márquez-Neila, Salzmann, and Fua 2017; Nandwani et al. 2019) 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; then the constrained optimization can be solved efficiently with exact constraint enforcement (Pham, De Magistris, and Tachibana 2018; Agrawal et al. 2019). However, the collision-free constraints in our applications are neither convex nor smooth.

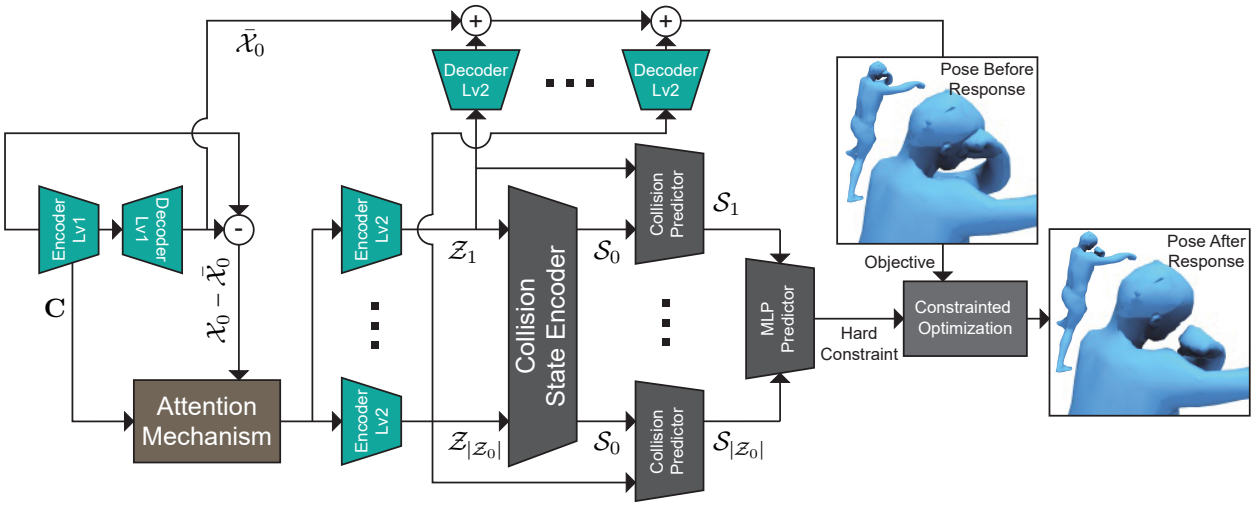

Figure 1: Our network architecture combines the domain-decomposed human pose embedding framework (green) and a novel collision state estimator (gray). Given an input pose, we use a weight-shared, level-1 autoencoder to learn a global shape embedding. The error on each domain is further reduced using 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.

3 Human Pose & Collision-Free Constraints

Recent methods (Tretschk et al. 2020; Bouritsas et al. 2019; Ranjan et al. 2018; Bagautdinov et al. 2018; Tan et al. 2018a) 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.

3.1 Human Pose Embedding

Our method uses the algorithm in (Tan et al. 2018b; Yang et al. 2020), which has the ability to extract local deformation components (more details given in the appendix). 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 (Gao et al. 2019), 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 $\mathbf{C} \in \mathbb{R}^{K\times 9\times|\mathcal{V}|}$. 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 Section 3.3.

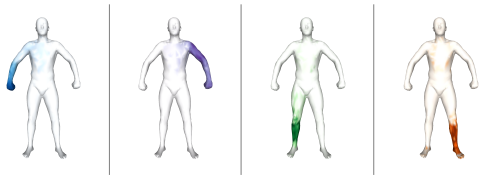

Figure 2: We show decomposed domains on the SCAPE dataset using the learned attention mask $\mathcal{M}^{ki}$, highlighted in different colors. The darkness of a given color represents the weight of the soft assignment. These weights are used for localized collision computations.

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 Figure 2.

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: $\mathbf{t}_p \cap \mathbf{t}_q \neq \varnothing$, 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:

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

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:

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

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 Locality of Self-collisions

Our method is inspired by the subspace self-collision culling algorithm (SSCC) (Barbič and James 2010) and the learning-based collision simplification algorithm (Teng, Otaduy, and Kim 2014). 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 3.1.

It is worth noting that both works (Barbič and James 2010; Teng, Otaduy, and Kim 2014) 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 (Teng, Otaduy, and Kim 2014) 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.

4 LCollision: Overall Learning Algorithm

Our overall learning architecture is illustrated in Figure 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:

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

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.

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 Figure 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 $(512, 256, 256)$ 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 $(512, 256, 256, 128)$ neurons and ReLU activation. Finally, CP outputs a scalar collision indicator $\mathcal{S}_k$, i.e. $\mathcal{S}_k = \text{CP}(\mathcal{S}_0, \mathcal{Z}_k)$.

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 (Pan, Chitta, and Manocha 2012) to compute the squared penetration depth $\text{PD}^2_{p,q}$ of each colliding triangle pair. This colliding pair correlates 6 vertices in $\mathcal{V}$, and we add $\text{PD}^2_{p,q}/6$ 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.

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 $\mathcal{M}^{ki}$ de-

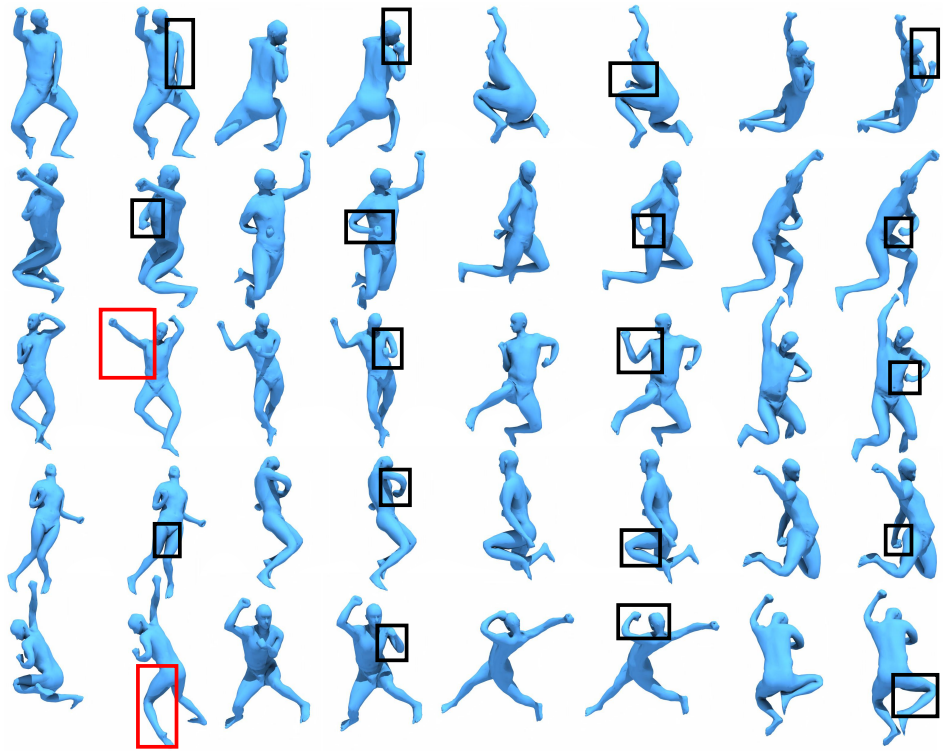

Figure 3: We illustrate 20 representative results of collision responses, where the poses on the left are the original poses directly generated using (Yang et al. 2020), and the poses on the right are the ones after collision responses. We highlight the adjusted body parts using black boxes. In all the examples, our method can successfully avoid penetrations. However, in two cases (red boxes), our adjusted poses drift severely from the original poses.

Algorithm 1 Generating Penetration Energy Vector PDe

Init PDe = $\vec{0} \in \mathbb{R}^{|\mathcal{V}|}$
Run FCL finding the set of all collided disjoint triangle paris as $\hat{T}$
for $(\mathbf{t}_p, \mathbf{t}_q) \in \hat{T}$ and the corresponding $\text{PD}_{p,q}$ **do**
  for Vertex i belongs to $\mathbf{t}_p$ and $\mathbf{t}_q$ **do**
    PDe_i += $\text{PD}^2_{p,q}/6$
  end for
end for

fined in Section 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 (Hoffer and Ailon 2015; Burges et al. 2005), 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 (Wang, Ramanan, and Hebert 2017). 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:

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

4.3 Solving Constrained Optimization

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

$$\begin{aligned}&\underset{\mathcal{Z}_{all}}{\mathbf{argmin}} \ \|\mathcal{Z}_{all} - \mathcal{Z}^*_{all}\|^2 \\ &\mathbf{s.t.}\ \text{MLP}_{classifier}\left(\mathcal{S}_1, \cdots, \mathcal{S}_{|Z_0|}\right) \leq 0.5.\end{aligned} \tag{1}$$

The idea is to provide a desired pose $\mathcal{Z}^*_{all}$ for the bilevel decoder, and Equation 1 solves for a collision-free $\mathcal{Z}_{all}$ that is as close to $\mathcal{Z}^*_{all}$ as possible. We solve Equation 1 using the augmented Lagrangian method implemented in LOQO (Vanderbei 1999), with all the gradient information computed via back-propagation through the neural network. This augmented Lagrangian method can start from an infeasi-

ble 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.

5 Evaluation

We implement our method using PyTorch (Paszke et al. 2017). All the training and testing are performed on a single desktop machine with a 4-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 $\mathbf{min}(\mathcal{Z}_{all_k}) < 0$, $\mathbf{max}(\mathcal{Z}_{all_k}) > 0$, 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 (Anguelov et al. 2005) with $N = 71$ meshes, the MIT Swing dataset (Vlasic et al. 2008) with $N = 150$ meshes, and the MIT Jumping dataset (Vlasic et al. 2008) 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 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 Figure 1. We notice that the constrained optimization Equation 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 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 $\mathcal{L}_{PD}$.

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 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.

Our second study inspects the robustness of our network architecture in terms of the size of the dataset. As shown in Table 2, we tested our method trained using two different M. 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.

Dataset	Time (Pan et al.)	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: We compare LCollision with (Pan, Chitta, and Manocha 2012) in terms of the computational cost for collision detection. (Pan, Chitta, and Manocha 2012) 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 (Pan, Chitta, and Manocha 2012) depends on the number of points; thus, exact collision checking (Pan, Chitta, and Manocha 2012) 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.

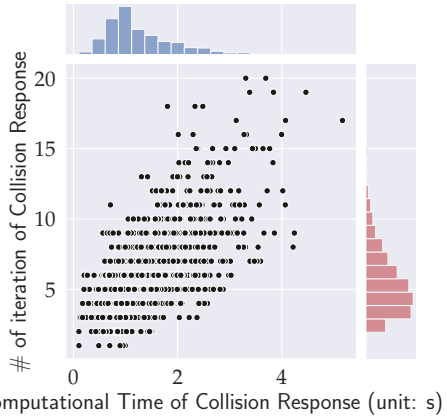

Figure 4: The joint histogram of the number of iterations (Y-axis) and the computational time (X-axis) used for solving the constrained optimization (Equation 1) for the Swing dataset. The average number of iterations is 5.44 and the average computation time is 1.29s.

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 (Pan, Chitta, and Manocha 2012) on the test set of 5×10^4 samples (1.5×10^4 samples) for the SCAPE, Swing, and Jumping datasets. The implementation of (Pan, Chitta, and Manocha 2012) only supports CPU, while LCollision runs on both CPU and GPU. To achieve the best performance for (Pan, Chitta, and Manocha 2012), 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 and observe two orders of magnitude speedup.

The Collision Response Solver: In Figure 3, we show 20

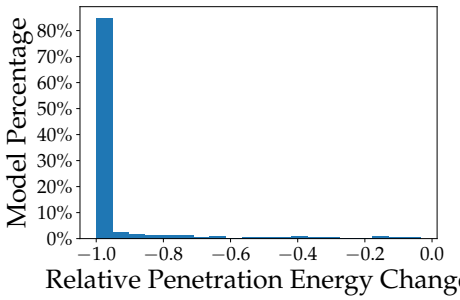

Figure 5: The histogram of relative penetration energy change for successful examples in the Swing dataset. Our method achieves a success rate of 85.1%, and we observed an average relative decrease of 94.3% in penetration energy.

results with successful collision responses for the SCAPE dataset (more results on the other datasets given in the appendix). 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 Equation 1 and we consider a solution successful if the augmented Lagrangian algorithm returns a feasible solution. On 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 Figure 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 Figure 5, we highlight the distribution of relative penetration energy change for successful collision response models.

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 (Pan, Zhang, and Manocha 2013; He et al. 2015) for rigid objects. A second issue is the use of a continuous constraint optimizer (Vanderbei 1999) 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 Equation 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 (Wolinski et al. 2014).

Acknowledgements

This work was supported in part by ARO under Grants W911NF1810313 and W911NF1910315, and in part by Intel.

References

Agarwal, A.; and Triggs, B. 2005. Recovering 3D human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence* 28(1): 44–58.

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 9558–9570.

Anguelov, D.; Srinivasan, P.; Koller, D.; Thrun, S.; Rodgers, J.; and Davis, J. 2005. SCAPE: shape completion and animation of people. In *ACM SIGGRAPH*, 408–416.

Bagautdinov, T.; Wu, C.; Saragih, J.; Fua, P.; and Sheikh, Y. 2018. Modeling facial geometry using compositional vaes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3877–3886.

Barbič, J.; and James, D. L. 2010. Subspace Self-Collision Culling. *ACM Trans. on Graphics (SIGGRAPH 2010)* 29(4): 81:1–81:9.

Boggs, P. T.; and Tolle, J. W. 1995. Sequential quadratic programming. *Acta numerica* 4: 1–51.

Bouritsas, G.; Bokhnyak, S.; Ploumpis, S.; Bronstein, M.; and Zafeiriou, S. 2019. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE International Conference on Computer Vision*, 7213–7222.

Bridson, R.; Fedkiw, R.; and Anderson, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 594–603.

Burgard, W.; Brock, O.; and Stachniss, C. 2008. *A Fast and Practical Algorithm for Generalized Penetration Depth Computation*, 265–272.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, 89–96.

Gao, L.; Lai, Y.-K.; Yang, J.; Ling-Xiao, Z.; Xia, S.; and Kobbelt, L. 2019. Sparse data driven mesh deformation. *IEEE TVCG* .

Govindaraju, N. K.; Lin, M. C.; and Manocha, D. 2005. Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, 59–66. IEEE.

Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; and Cohen-Or, D. 2019. MeshCNN: A network with an edge. *ACM Transactions on Graphics* 38(4): 90. ISSN 15577368. doi:10.1145/3306346.3322959.

He, L.; Pan, J.; Li, D.; and Manocha, D. 2015. Efficient penetration depth computation between rigid models using contact space propagation sampling. *IEEE Robotics and Automation Letters* 1(1): 10–17.

Hoffer, E.; and Ailon, N. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, 84–92. Springer.

Kervadec, H.; Dolz, J.; Tang, M.; Granger, E.; Boykov, Y.; and Ayed, I. B. 2019. Constrained-CNN losses for weakly supervised segmentation. *Medical image analysis* 54: 88–99.

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

Kim, Y. J.; Otaduy, M. A.; Lin, M. C.; and Manocha, D. 2002. Fast penetration depth computation for physically-based animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 23–31.

Leibe, B.; Seemann, E.; and Schiele, B. 2005. Pedestrian detection in crowded scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, 878–885. IEEE.

Li, Y.; Wu, J.; Tedrake, R.; Tenenbaum, J. B.; and Torralba, A. 2019. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *ICLR*.

Márquez-Neila, P.; Salzmann, M.; and Fua, P. 2017. Imposing Hard Constraints on Deep Networks: Promises and Limitations. *CoRR* abs/1706.02025.

Nandwani, Y.; Pathak, A.; Singla, P.; et al. 2019. A Primal Dual Formulation For Deep Learning With Constraints. In *Advances in Neural Information Processing Systems*, 12157–12168.

Pan, J.; Chitta, S.; and Manocha, D. 2012. FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, 3859–3866. IEEE.

Pan, J.; Zhang, X.; and Manocha, D. 2013. Efficient penetration depth approximation using active learning. *ACM Transactions on Graphics (TOG)* 32(6): 1–12.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch .

Pham, T.; De Magistris, G.; and Tachibana, R. 2018. OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6236–6243.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qiao, Y.-L.; Liang, J.; Koltun, V.; and Lin, M. C. 2020. Scalable Differentiable Physics for Learning and Control. *arXiv preprint arXiv:2007.02168* .

Ranjan, A.; Bolkart, T.; Sanyal, S.; and Black, M. J. 2018. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 704–720.

Ravi, S. N.; Dinh, T.; Lokhande, V. S.; and Singh, V. 2019. 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, 4772–4779.

Rogez, G.; Rihan, J.; Ramalingam, S.; Orrite, C.; and Torr, P. H. 2008. Randomized trees for human pose detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE.

Shi, X.; Zhou, K.; Tong, Y.; Desbrun, M.; Bao, H.; and Guo, B. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. In *ACM SIGGRAPH 2007 papers*, 81–es.

Smith, B.; Goes, F. D.; and Kim, T. 2018. Stable neohookean flesh simulation. *ACM Transactions on Graphics (TOG)* 37(2): 1–15.

Tan, Q.; Gao, L.; Lai, Y.-K.; and Xia, S. 2018a. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5841–5850.

Tan, Q.; Gao, L.; Lai, Y.-K.; Yang, J.; and Xia, S. 2018b. Mesh-based autoencoders for localized deformation component analysis. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tan, Q.; Pan, Z.; Gao, L.; and Manocha, D. 2020. Realtime Simulation of Thin-Shell Deformable Materials Using CNN-Based Mesh Embedding. *IEEE Robotics and Automation Letters* 5(2): 2325–2332. doi:10.1109/LRA.2020.2970624.

Tang, M.; Curtis, S.; Yoon, S.-E.; and Manocha, D. 2009. ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics* 15(4): 544–557.

Tang, M.; Manocha, D.; and Tong, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 7–13.

Teng, Y.; Otaduy, M. A.; and Kim, T. 2014. Simulating Articulated Subspace Self-Contact. *ACM Trans. Graph.* 33(4). ISSN 0730-0301. doi:10.1145/2601097.2601181. URL https://doi.org/10.1145/2601097.2601181.

Toshev, A.; and Szegedy, C. 2014. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1653–1660.

Tretschk, E.; Tewari, A.; Zollhöfer, M.; Golyanik, V.; and Theobalt, C. 2020. DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects. *European Conference on Computer Vision (ECCV)* .

Vanderbei, R. J. 1999. LOQO user's manual—version 3.10. *Optimization methods and software* 11(1-4): 485–514.

Villegas, R.; Yang, J.; Ceylan, D.; and Lee, H. 2018. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8639–8648.

Vlasic, D.; Baran, I.; Matusik, W.; and Popović, J. 2008. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, 1–9.

Wang, P.-S.; Liu, Y.; and Tong, X. 2020. Deep Octree-based CNNs with Output-Guided Skip Connections for 3D Shape and Scene Completion.

Wang, Y.-X.; Ramanan, D.; and Hebert, M. 2017. Learning to model the tail. *Advances in Neural Information Processing Systems* 30: 7029–7039.

Wolinski, D.; J. Guy, S.; Olivier, A.-H.; Lin, M.; Manocha, D.; and Pettré, J. 2014. Parameter estimation and comparative evaluation of crowd simulations. In *Computer Graphics Forum*, volume 33, 303–312. Wiley Online Library.

Yang, J.; Gao, L.; Tan, Q.; Huang, Y.; Xia, S.; and Lai, Y.-K. 2020. Multiscale Mesh Deformation Component Analysis with Attention-based Autoencoders.

Zhang, L.; Kim, Y. J.; Varadhan, G.; and Manocha, D. 2007. Generalized penetration depth computation. *Computer-Aided Design* 39(8): 625–638.