Learning-based Self-Collision Avoidance in Retargeting using Body Part-specific Signed Distance Fields

Junwoo Lee[1], Hoimin Kim[1] and Taesoo Kwon[1]

[1]Hanyang University, South Korea

Figure 1: *Self-collisions in the input poses are successfully resolved (gray character: input pose, colorized character: output pose). Our model demonstrates a high generalization ability for unseen characters and motions, effectively resolving challenging self-collisions, including deep collisions and leg-to-leg collisions.*

Abstract

Motion retargeting is a technique for applying the motion of one character to a new character. Differences in shapes and proportions between characters can cause self-collisions during the retargeting process. To address this issue, we propose a new collision resolution strategy comprising three key components: a collision detection module, a self-collision resolution model, and a training strategy for the collision resolution model. The collision detection module generates collision information based on changes in posture. The self-collision resolution model, which is based on a neural network, uses this collision information to resolve self-collisions. The proposed training strategy enhances the performance of the self-collision resolution model. Compared to previous studies, our self-collision resolution process demonstrates superior performance in terms of accuracy and generalization. Our model reduces the average penetration depth across the entire body by 56%, which is 28% better than the previous studies. Additionally, the minimum distance from the end-effectors to the skin averaged 2.65*cm, which is more than* 0.8*cm smaller than in the previous studies. Furthermore, it takes an average of* 7.9*ms to solve one frame, enabling online real-time self-collision resolution.*

CCS Concepts

• ***Computing methodologies*** → ***Motion processing;*** *Collision detection; Neural networks;*

1. Introduction

Motion retargeting refers to the process of processing and applying existing motion data to match a new character's skeleton and body shape. Even when applying the same motion, minor or major edits may be required depending on the character's appearance. For example, if a small character and a large character move their arms at the same angle, the arms might intersect with the body, resulting in an unnatural appearance. However, manually editing or creating motions to fit each character's shape requires significant cost and expertise. By using motion retargeting, existing motion data can be adapted as needed, greatly enhancing the efficiency of character animation production. Additionally, in modern games where users can change the character's body shape in real-time, the need for real-time retargeting increases, as the intervention of expert is not feasible in such cases.

When developing motion retargeting technology, various factors need to be considered, including the quality of the resulting motion, computational speed, scalability to different body shapes, and

DOI: 10.2312/pg.20241288

ease of editing. The criteria for determining the quality of the resulting motion can be broadly divided into two categories. The first criterion is the preservation of motion semantics. Even if different characters take poses at different angles due to motion retargeting, they should be able to preserve the intended meaning of the motion. The second criterion is self-collision avoidance. Generally, larger characters are more prone to penetration phenomena between body parts. Even minor penetrations are noticeable and can convey an impression of unnatural motion, making self-collision avoidance crucial in motion retargeting.

Due to the high demand for such motion retargeting technology, extensive research has been conducted in the field of computer graphics for decades to develop more efficient and high-quality motion retargeting techniques. In recent years, there have been many attempts to research motion retargeting technology that incorporates data-driven approaches [JKY*18, ZWK*23, VCH*21, VYCL18]. However, even in these latest motion retargeting technologies, self-collisions still frequently occur, and over-shooting problems, where poses are excessively altered to prevent collisions, are also observed. This paper proposes a new motion retargeting method to robustly perform self-collision avoidance.

Typical neural motion networks for motion retargeting are designed to learn end-to-end processes that detect collision between body parts and edit avoidance actions simultaneously [ZWK*23, VCH*21]. However, we have observed that this approach makes it difficult to accurately identify self-collisions and reduces generalization performance. Therefore, the new motion retargeting system proposed in this paper is designed to robustly perform self-collision avoidance by dividing it into two detailed modules. The first is the collision detection module, which efficiently and accurately detects self-collisions in the current pose, summarizing collision situations based on geometric computations. The second is the self-collision resolution model, an artificial neural network-based module that determines in which direction to avoid self-collisions while maintaining natural character poses as much as possible.

The collision detection module generates information about self-collision areas that occur when the input pose is applied to the given character structure. This self-collision information is defined in the form of a collision matrix, which divides the character's body structure into six parts and stores the penetration depth between each body part. The dimensions of the collision matrix remain consistent regardless of the character and the matrix have continuously varying values according to sequential posture changes. Therefore, the collision matrix has a form that is easy to use as input by another module proposed in this paper, the self-collision resolution model.

The self-collision resolution model, structured as a neural network, uses the collision matrix as its input, which has been calculated by the previous collision detection module. This module learns and infers how to transform the input posture into a new posture that avoids self-collision based on the given collision matrix.

Finally, we propose two training strategies to enhance the accuracy and generalization ability of the self-collision resolution model. The first strategy is random collision dropout, which filters out some values in the collision matrix with a certain probability during training. This strategy prevents some collision information from being ignored, ensuring that collision information provided by the collision matrix is sufficiently reflected during the collision avoidance process. Another training strategy is random segment padding, which enhances the generalization performance of the inference by training the model using characters with variably transformed body shapes through augmentation.

The motion retargeting method proposed in this paper achieved high performance in avoding self-collisions quickly and accurately through a two-module structure. It was able to accurately detect and avoid collisions in uncommon cases, such as collisions between the head and hands or hands and feet, even when the training data was insufficient. It was also capable of inferring natural motions for new character structures not used in the training. Additionally, we conducted quantitative and qualitative comparative experiments with existing motion retargeting methods, confirming that our method significantly improved self-collision avoidance and preservation of motion semantics.

The new motion retargeting method proposed in this paper has the following contributions:

- It proposes a learning-based motion retargeting model separated into a collision detection module and a self-collision resolution model, instead of the conventional end-to-end learning method.
- It defines a collision matrix that conveys information about depth of self-collisions currently occurring. Calculated through continuous functions, it ensures smooth resulting motions and maintains a constant data volume regardless of the character's posture, making it suitable as input for the learned model.
- It presents two training strategies for performing self-collision avoidance. Through random collision dropout, collisions can be avoided more accurately without ignoring difficult collisions, and through random segment padding, generalization performance for various character body shapes can be enhanced.
- Compared to existing motion retargeting methods, the proposed method naturally resolves self-collision phenomena and generates high-quality resulting motions that preserve the original motion as much as possible.

2. Related Work

Motion retargeting technology significantly enhances the efficiency of generating graphic content. There is substantial industrial demand for motion retargeting, and ongoing research aims to make this technology more universally applicable with improved performance.

Motion Retargeting. To the best of our knowledge, [Gle98] was the first to introduce the concept of motion retargeting. In this study, motion retargeting was achieved by defining the features of motion as spatiotemporal constraints and solving the optimization problem related to these constraints. Around the same time, Lee and Shin proposed a motion retargeting method that adjusts motion sequences based on inverse kinematics (IK) to satisfy constraints [LS99]. Subsequently, research on retargeting has continued in various application areas, including characters with different morphologies, interactions among multiple characters, pose trans-

fer with collision avoidance, video input, and physics-based retargeting [HRE*08, JKL18, BWBM20, KSK21, RWY*23].

Unlike previous studies that primarily relied on energy function optimization, recent attempts have incorporated deep learning approaches to utilize pre-prepared datasets, thereby reducing manual work and improving performance. The traditional optimization-based approach for retargeting had limitations in reflecting the characteristics and meanings of character motion, but integrating deep learning allows for better adaptation to character structures [JKY*18, HZZ*24]. Various studies have also focused on learning retargeting models that incorporate kinematic techniques in characters with skinned meshes [VYCL18, ALL*20, LCC19]. However, these studies did not consider the geometry of the mesh, often resulting in frequent self-collisions and generating actions that are either semantically distant from the original motion or unnatural. Our research is based on deep learning and focuses on avoiding self-collisions that occur during retargeting. Therefore, we will focus our survey on collision avoidance and shape recognition within learning-based methods below.

Collision Avoidance of a Single Character. Many studies on motion retargeting have aimed to resolve the self-collision problem. For instance, [TPM21] used a prediction classifier to predict areas where self-collisions would occur, optimizing the time required to detect collisions and generating collision-avoidance motions. Although this approach demonstrated high generality, applicable to various character mesh structures and motions, it sometimes failed to detect collisions properly and generated unnatural motions occasionally.

PointNet generates boundary points for 3D segmentation [QSMG17]. Villegas et al. used feature vector information obtained through PointNet to embed the shape of the character into the learning model during the motion retargeting process [VCH*21]. Additionally, they performed further optimization in the latent space to maintain self-collision and the semantics of poses. However, this approach had limitations in accurately predicting collision information according to the character's pose, leading to issues such as overshooting and residual collisions during the self-collision resolution process.

Aberman et al. proposed a method training an artificial neural network using motion sequences of characters with different structures to effectively transfer motions [ALL*20]. However, the success rate of retargeting greatly depended on the type of motion to be transferred and the similarity of joint structures, with limitations in preserving the original poses.

R^2ET, proposed by Zhang et al., is a learning-based model for motion retargeting that separates the process of adjusting the pose to preserve the meaning of the motion from the process of self-collision avoidance [ZWK*23]. The collision avoidance module of R^2ET relies on shape recognition of the subset of the character mesh within the bounding box corresponding to each joint. Furthermore, introducing a balancing gate after the collision avoidance module reduced overshooting and achieved better performance than previous studies. Lyard et al. designed heuristic functions individually for resolving self-collisions occurring in the arms and feet and adjusted poses based on these functions [LMT08]. However, this approach had limitations in generally handling complex collision situations, occasionally resulting in increased self-collisions after collision avoidance. The motion retargeting method proposed in this paper implicitly learns how to resolve collisions in the collision avoidance module based on a data-driven approach, without requiring user specifications.

Unlike previous methods, there has also been research on motion retargeting methods for characters without joints [BWBM20, LMHM18]. Both studies performed motion retargeting through optimization of energy functions or solutions of linear systems defined on the mesh surface of the given character. However, this approach has the disadvantage of being difficult to perform self-collision avoidance in real-time. Our method is similar to the aforementioned studies in that it divides the body of a single character into multiple regions and performs self-collision avoidance using collision information of the character's skin within each region.

Shape Awareness for Collision Avoidance. To resolve collisions that occur during the motion generation process, it is essential to have a comprehensive understanding of the mesh shape of the character (or object). A representative method for shape embedding of a character involves utilizing the extent of bounding boxes that enclose the character's body or adapting the existing PointNet method [ZWK*23, VCH*21]. However, since most character shapes are curved and have concave parts, there are inevitably small and large gaps between the actual boundaries of the shape (or skin) and the boundaries of the bounding boxes. Therefore, using bounding box extent information for shape awareness has limitations in precisely capturing the character's shape. Additionally, when using PointNet for shape embedding, complex calculations are needed during the process of inferring collision-free motions, making it challenging to use directly in neural network models, particularly for real-time applications.

When dealing with interactive motions between a character and an object, it is also necessary to understand the spatial relationship between the character and the object. To this end, Karunratanakul et al. proposed a generalized method for maintaining contact states by defining a grasping field that represents mesh surface points as a signed distance field [KYZ*20]. Pirk et al. unified interaction information from motion data obtained through various types of sensors by representing the dynamic interaction paths arising from motion as paths of vertex sets on the object surface [PKH*17]. Zhang et al. represented the occupancy of objects using voxels to express the volume that objects have in space around the hand, and also utilized raycasting and SDF to gather distance information between the hand and objects [ZYSK21]. The Interaction Bisector Surface method identifies the spatial boundaries between two objects [ZWK14], and She et al. used this method to express the spatial relationship between the hand and objects [SHX*22]. While these representation methods can be efficient for the self-collision resolution problem, they have the limitation of being difficult to implement due to their relatively complex nature, especially when the scale of self-collision is large.

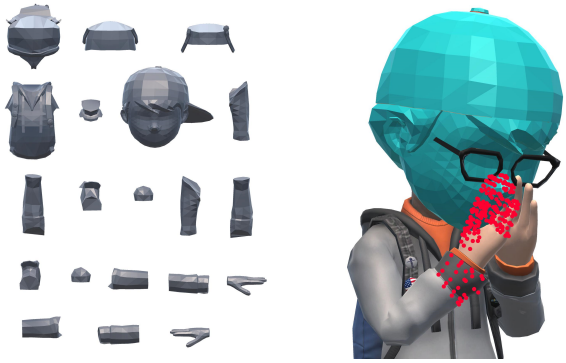
3. Overview

Figure 3 illustrates the process of self-collision avoidance using character and pose as inputs. Specifically, the system takes as inputs the character's current pose $\mathbf{q}$, the character's bind-pose skeleton $\mathbf{s}$, bind-pose vertices $\mathbf{v}$ used for skinning, and skinning weights $\mathbf{W}$. The output is a new pose $\hat{\mathbf{q}}$ in which self-collision has been resolved. The system consists of two components: a collision detection module F_{col} based on geometric computation and a self-collision resolution model F_{res} based on an artificial neural network. The collision detection module F_{col} calculates the depth of self-collisions occurring in the character's current pose $\mathbf{q}$ (Section 4). The self-collision resolution model F_{res} predicts $\hat{\mathbf{q}}$ based on the results of the collision detection module F_{col} and the input pose $\mathbf{q}$:

$$\hat{\mathbf{q}} = F_{\text{res}}(\mathbf{q}, F_{\text{col}}(\mathbf{q})). \tag{1}$$

A detailed explanation of the self-collision resolution model is provided in Section 5.

To enhance the accuracy and generalization ability of the self-collision resolution model, a specific training strategy is required (Section 6). We propose two training strategies: **R**andom **C**ollision **D**ropout (RCD) and **R**andom **S**egment **P**adding (RSP). The RCD improves the collision avoidance accuracy of the self-collision resolution model by randomly removing some of the detected collisions (Section 6.1). The RSP enhances the model's generalization performance to new characters or poses by simulating various body shapes by virtually expanding and contracting some parts of the character's body (Section 6.2).



(a) (b)

Figure 2: *(a) Examples of body segments. Body segments are created by separating the vertices of a given character mesh based on skinning weights. (b) An illustration of the process of calculating the signed distance between two segments. When calculating the signed distance between the character's right hand segment and head segment, the SDF values for the head segment are calculated for each vertex corresponding to the right hand. These vertices are obtained by performing linear blended skinning on the current pose. The smallest distance value is then taken as the signed distance between the right hand segment and the head segment.*

4. Collision Detection Module

The collision detection module calculates the collision matrix $\mathbf{M}_{\text{col}}$, which stores the depths of self-collisions occurring in the character, given the pose $\mathbf{q} \in \mathbb{R}^{J\times 4}$, bind-pose skeleton $\mathbf{s} \in \mathbb{R}^{J\times 3}$, bind-pose vertices $\mathbf{v} \in \mathbb{R}^{V\times 3}$, and skinning weights $\mathbf{W} \in \mathbb{R}^{V\times J}$. Here, J is the number of joints and V is the number of vertices forming the character mesh.

The collision matrix $\mathbf{M}_{\text{col}} \in \mathbb{R}^{6\times 6}$ compactly stores the penetration depths between six segment clusters of the character, rather than between all possible segment combinations. A segment cluster is a set of body segments. When clustering segments, we assume that there is no self-penetration within the same segment cluster. Therefore, segments that frequently experience self-collisions should be placed in different segment clusters. Although our method performs effectively even when each segment cluster contains only one segment, using a small number of segment clusters enhances the generalization ability and learning speed. Consequently, we assign a segment cluster to each of the character's limbs, torso, and head:

$$\mathcal{B}_{c\in[1,6]} \subset \mathcal{S}, \quad \mathcal{S} \equiv \{S_j \mid j \in [1,J]\}, \tag{2}$$

where B_c represents the c-th segment cluster, and segment S_j refers to the part of the character mesh influenced directly by j-th joint.

4.1. Body Segment

A given character's mesh can be viewed as a set of body segments, which are divided based on joints (Figure 2a). We define the j-th segment as the set of mesh vertices that have a skinning weight above a certain threshold for the j-th joint:

$$S_j \equiv \{\mathbf{v}_i | \mathbf{W}_{(i,j)} \geq \eta, \forall i \in [1,V]\}, j \in [1,J]. \tag{3}$$

Here, the coordinates of each segment vertex $\mathbf{v}_i$ are defined in the local coordinate system of the j-th joint. η is the threshold for the skinning weight.

4.2. Signed-Distance Fields

We use signed distance fields for efficient collision detection. To calculate the collision matrix for each pose, $V * J$ SDF queries are generated per character, which constitutes a significant portion of the computational cost of the entire pipeline. Therefore, to speed up SDF querying, we approximate the SDF by voxelizing it, as done in [ZWK*23]. The SDF is defined in the bind-pose space or the local coordinate system of the joint, generated once initially, and subsequent queries are differentiable and executed in constant time.

Character Signed-Distance Field. First, a 3D bounding box enclosing the character's mesh is set, and this space is divided into voxels of a fixed size. The resolution of the voxel grid is determined by the size of each voxel, which is a cube approximately 0.01 times the height of the bounding box. The distance from the center of each voxel to the nearest surface point is stored in the voxel, with the distance being negative for points inside the surface mesh and positive for points outside the shape.

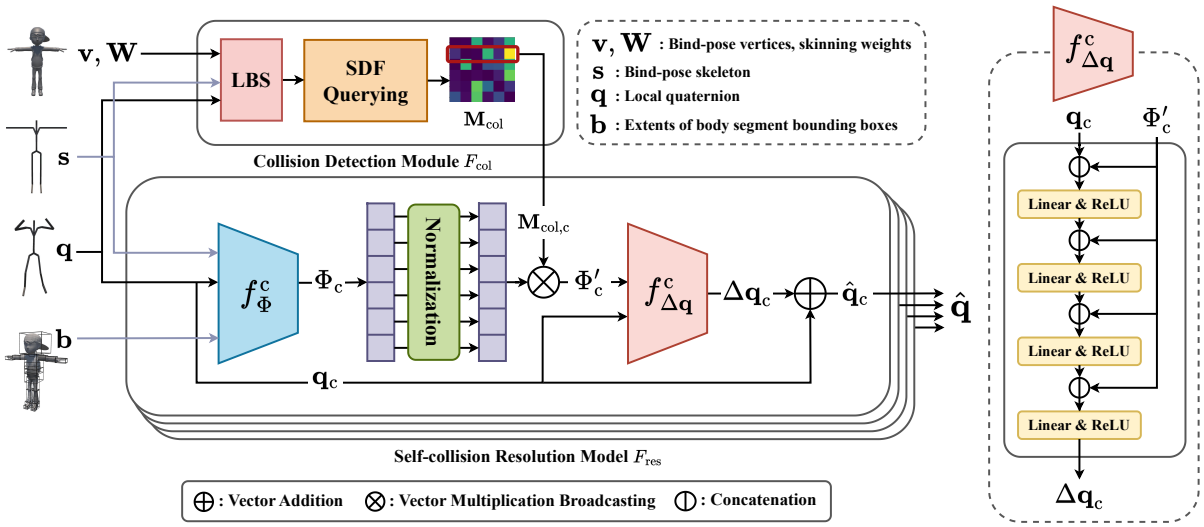


Figure 3: *Structure of the self-collision resolution model.*

Segment Signed-Distance Field. The Segment Signed-Distance Field (Segment SDF) refers to the SDF of the region tightly enclosing the vertices constituting each segment. For memory efficiency, the Segment SDF is defined on two voxel grids: one is a high-precision SDF defined within the bounding box of the segment, $\text{SSDF}_{\text{small}}$, and the other is a coarse SDF that covers the bounding box of the entire character mesh, $\text{SSDF}_{\text{large}}$. Both SDF functions store the signed distance to the segment surface. When calculating the Segment SDF, if the SDF query point is within the bounding box of the segment, the distance sampled from $\text{SSDF}_{\text{small}}$ is used. Otherwise, the distance from $\text{SSDF}_{\text{large}}$ is used.

While the SDF generation process can be automatically conducted through the subdivision of the voxel space within the character, for the sake of explanation, we assume in this paper that the Segment SDFs are pre-generated, and provided as input to the system.

Blended SDF. Using SDF (Signed Distance Field), the closest point on a surface from an arbitrary point can be quickly calculated by moving in the opposite direction of the gradient by the distance. However, as shown in Figure 4a, calculating the penetration depth with the mesh surface using Segment SDF may underestimate the actual penetration depth at the segment boundaries. This can reduce the accuracy of the collision avoidance procedure. One simple way to address this is to use the Character SDF instead of the Segment SDF for the internal regions of the segment where the SDF value is less than zero (Figure 4b). However, this Selective SDF function is discontinuous at the points where the Segment SDF becomes zero near the segment boundaries, causing the collision matrix to change discontinuously over time, and generating noise in the resulting motion. Therefore, we removed the discontinuity of the Selective SDF by linearly interpolating between the Character SDF and the Segment SDF based on the value of the Segment SDF. We named this the Blended SDF (Figure 4c). The Blended SDF can be

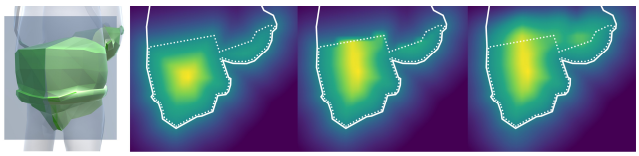


Figure 4: *Visualization of SDF values by location. The SDF values were calculated on a plane intersecting the segment shown on the left. The intersection points between the segment surface and the plane are represented by dotted lines. The solid lines represent the intersection points between the character surface and the plane. (a) In the case of the Segment SDF, the SDF values are zero on the segment surface, regardless of their spatial relationship to the character mesh. This can lead to an underestimation of collisions occurring deep inside the body near the boundary between adjacent segments. (b) The Selective SDF addresses this underestimation by using the Character SDF within the segment. However, it introduces discontinuities in the SDF near the boundary, causing a sharp change in SDF values at the segment's surface. (c) To resolve both issues, the Blended SDF interpolates between the Character SDF and Segment SDF outside the segment, leading to SDF values that vary based on their distance from the segment surface. Unlike the Selective SDF, it does not exhibit discontinuities.*

formulated as follows:

$$\begin{aligned} p &\equiv \min\left(\max\left(SDF_{\text{seg}}\left(\mathbf{v}\right)/\nu, 0.0\right), 1.0\right), \\ SDF_{\text{b}}(\mathbf{v}) &\equiv p * SDF_{\text{seg}}\left(\mathbf{v}\right) + (1-p) * SDF_{\text{char}}\left(\mathbf{v}\right). \end{aligned} \tag{4}$$

Here, ν is a hyperparameter that sets the range of the blended area. Through experiments, we found that the discontinuity of the Selective SDF does not cause problems during the training process and even improves performance. Therefore, we used the Selective

SDF during the training process and the Blended SDF during the inference process for continuous pose output. In the following section, we denote both of these simply as $SDF_j(\cdot)$, the chosen SDF function of the j-th segment S_j.

4.3. Collision Matrices

Before calculating the collision matrix $\mathbf{M}_{col}$, the collision detection module first calculates the pair-wise distance matrix $\mathbf{M}_{dist} \in \mathbb{R}^{J\times J}$ between segments:

$$\mathbf{M}_{dist,ij}(\mathbf{q}) \equiv \min(\{SDF_j(\mathbf{v}) \mid \forall \mathbf{v} \in LBS_i(\mathbf{q})\}) + \Omega_{ij}. \quad (5)$$

This calculation process can be seen in Figure 2b. Here, $LBS_i(q)$ is a function that performs linear blend skinning on the vertices included in the i-th segment S_i based on the current pose $\mathbf{q}$, and $SDF_j(\cdot)$ is the chosen SDF function of the j-th segment. Finally, Ω is a matrix used to ignore collisions between two segments S_i and S_j specified by the user during the overall collision detection process. Ω is a masking matrix where Ω_{ij} is set to ∞ to ignore collisions between segments S_i and S_j and set to 0 otherwise. This prevents collision information from being generated from segment pairs that always collide, such as the pairs between the same segment (diagonal elements) or between parent and child segments.

Based on the calculated $\mathbf{M}_{dist}$, the pair-wise distance $D_B \in \mathbb{R}^{6\times 6}$ between segment clusters can be obtained as follows:

$$D_B(\mathcal{B}_{c_i}, \mathcal{B}_{c_j}, \mathbf{q}) \equiv \begin{cases} \min(\mathbf{M}_{dist,\mathcal{I}_{c_i},\mathcal{I}_{c_j}}(\mathbf{q})) & \text{if } c_i \neq c_j \\ 0 & \text{else} \end{cases}. \quad (6)$$

Here, $\mathcal{I}_{c_i}$ and $\mathcal{I}_{c_j}$ are sets of indices of joints belonging to the c_i-th and c_j-th segment clusters, respectively, and $\mathbf{M}_{dist,\mathcal{I}_{c_i},\mathcal{I}_{c_j}}(\mathbf{q})$ is the submatrix of $\mathbf{M}_{dist}(\mathbf{q})$ corresponding to these indices. $\min(\cdot)$ is a function that selects the smallest element in the given matrix.

Finally, the output of the collision detection module $\mathbf{M}_{col}$ is obtained by extracting the penetrating depth from D_B.

$$\mathbf{M}_{col,c_ic_j}(\mathbf{q}) \equiv \max(-D_B(\mathcal{B}_{c_i}, \mathcal{B}_{c_j}, \mathbf{q}), 0). \quad (7)$$

5. Self-collision Resolution Model

The self-collision resolution model predicts a new pose $\hat{\mathbf{q}} \in \mathbb{R}^{J\times 4}$ that resolves collisions in the given pose $\mathbf{q}$ based on $\mathbf{M}_{col}$. The self-collision resolution model F_{res} operates independently for each of the four segment clusters $\mathcal{B}_c$ corresponding to the character's arms and legs, meaning $F_{res} = \{F^c_{res} \mid c \in [1,4]\}$. These four independent models are trained simultaneously because they need to coordinate the amount of collision avoidance with each other depending on the input pose. As shown in Figure 3, the input to each self-collision resolution model F^c_{res} includes the pose $\mathbf{q}$, the pose $\mathbf{q}_c \in \mathbb{R}^{|\mathcal{B}_c|\times 4}$ of the joints within the segment cluster $\mathcal{B}_c$, the character's bind-pose skeleton $\mathbf{s}$, the extent of the bounding box $\mathbf{b} \in \mathbb{R}^{J\times 3}$ surrounding each segment, same as with [ZWK*23], and the collision vector $\mathbf{M}_{col,c} \in \mathbb{R}^6$, which is the c-th row vector of $\mathbf{M}_{col}$. Based on these inputs, the model predicts a new pose $\hat{\mathbf{q}}_c \in \mathbb{R}^{|\mathcal{B}_c|\times 4}$, which resolves the self-collisions. The predictions $\hat{\mathbf{q}}_c$ from each F^c_{res} are combined to obtain the final result pose $\hat{\mathbf{q}}$.

5.1. Architecture

The self-collision resolution model F^c_{res} consists of two submodules, f^c_Φ and $f^c_{\Delta\mathbf{q}}$ (Figure 3). The intermediate feature predicted by the simple MLP f^c_Φ is called the semantic matrix and is denoted as $\Phi_c \in \mathbb{R}^{6\times d}$. Here, d is a hyper parameter. The current collision information contained in collision vector $\mathbf{M}_{col,c}$ for segment cluster $\mathcal{B}_c$ is embedded into this semantic matrix and used as input to the second submodule $f^c_{\Delta\mathbf{q}}$.

For this embedding, each row vector of Φ_c is first individually normalized to have a mean of 0 and a variance of 1. Subsequently, each column of Φ_c is multiplied element-wise by $\mathbf{M}_{col,c}$ to obtain the semantic collision matrix Φ'_c:

$$\Phi'_{c,ij} = \Phi_{c,ij} * \mathbf{M}_{col,ci}. \quad (8)$$

The resulting semantic collision matrix Φ'_c is flattened and input into the submodule $f^c_{\Delta\mathbf{q}}$.

To sufficiently propagate this semantic collision information to the output layer, we used a neural network in the style of DenseNet [HLvdMW18]. Specifically, we input Φ'_c repeatedly at every layer starting from the intermediate layers of $f^c_{\Delta\mathbf{q}}$ (right side of Figure 3). The performance of the proposed network architecture is verified through comparative experiments in Section 7.1. Finally, $f^c_{\Delta\mathbf{q}}$ predicts the change in pose $\Delta\mathbf{q}_c$. This change is then added to $\mathbf{q}_c$, and the result is normalized to obtain the array of unit quaternions $\hat{\mathbf{q}}_c$.

5.2. Loss Function

The loss function used to train the self-collision resolution model is defined as a weighted sum of Identity loss, Penetration recovery loss, Reconstruction loss, and an L2 weight regularization term:

$$L = \alpha L_{identity} + \beta L_{pen} + \gamma L_{recon} + \zeta|\mathbf{w}|_2. \quad (9)$$

Here, α, β, γ, and ζ are hyper parameters representing the weights of each loss term, and $\mathbf{w}$ are the parameters of the self-collision resolution model.

Identity loss. The self-collision resolution model should output the original pose when there is no collision. Let $\Delta\mathbf{q}_c(\Phi_c, \mathbf{M}_{col,c})$ be the output of the submodule $f^c_{\Delta\mathbf{q}}$ given the semantic matrix Φ_c and the collision vector $\mathbf{M}_{col,c}$. The identity loss is defined as follows:

$$L_{identity} = \sum_{c=1}^{4} |\Delta\mathbf{q}_c(\Phi_c, \mathbf{0})|_2. \quad (10)$$

Penetration recovery loss. The Penetration recovery loss is an objective function that implements the prior that the distance between two segment clusters $\mathcal{B}_i$ and $\mathcal{B}_j$ should be zero after resolving collisions. It is defined as follows:

$$L_{pen} = |u(\mathbf{M}_{col}(\mathbf{q})) * \mathbf{M}_{col}(\hat{\mathbf{q}})|_1. \quad (11)$$

Here, $u(\cdot)$ is a unit step function applied to each element of the given matrix, converting the penetrating depths from the input pose into a boolean mask. The $*$ operator denotes the element-wise multiplication between matrices.

Reconstruction loss. The Reconstruction loss implements the prior that the resolved pose $\hat{\mathbf{q}}$ should be similar to the original pose $\mathbf{q}$. It is defined as follows:

$$\begin{aligned} L_{\text{recon}} = \quad & |\mathbf{q} - \hat{\mathbf{q}}|_2 \\ & + |FK(\mathbf{q})_{\text{p}} - FK(\hat{\mathbf{q}})_{\text{p}}|_2 \\ & + |FK(\mathbf{q})_{\text{q}} - FK(\hat{\mathbf{q}})_{\text{q}}|_2. \end{aligned} \tag{12}$$

Here, $FK(\cdot)_{\text{q}}$ and $FK(\cdot)_{\text{p}}$ denotes the global quaternions and global positions of each joint obtained through forward kinematics, respectively.

6. Training Strategies

The self-collision resolution model described earlier uses a collision matrix $\mathbf{M}_{\text{col}}$, which allows it to resolve collisions between segment clusters in poses of easy difficulty. However, for new characters not used during training, the model often fails to push the segment clusters sufficiently apart or pushes them too far, resulting in poor performance in generalization and accuracy (Figure 6c-2). To improve the accuracy and generalization ability of the self-collision resolution model, we devised two training strategies.

6.1. Random Collision Dropout

The problem of self-collision resolution is an underdetermined problem with many possible solutions. For example, a collision between the left hand and the right hand can be resolved by moving either the left hand or the right hand. This can lead to overfitting during the training process, where a specific limb is predominantly used and overall avoidance fails (Figure 6c-1). We propose a training strategy called **R**andom **C**ollision **D**ropout (RCD) to encourage the appropriate use of various limbs and improve generalization performance.

RCD is a method where, with a certain probability λ, only one element in the input matrix $\mathbf{M}_{\text{col}}$ to the self-collision resolution model is retained, and the remaining elements are set to zero. By masking the elements of $\mathbf{M}_{\text{col}}$, RCD removes the correlation between the elements of $\mathbf{M}_{\text{col}}$ and encourages the model to respond sensitively to each individual element of the collision matrix $\mathbf{M}_{\text{col}}$. Through this, RCD has improved the accuracy of the self-collision resolution model. When the elements of $\mathbf{M}_{\text{col}}$ are masked by RCD, the following Penetration recovery loss is used instead of the original Penetration recovery loss:

$$L_{\text{pen}} = |u(\mathbf{M}_{\text{mask}} * \mathbf{M}_{\text{col}}(\mathbf{q})) * \mathbf{M}_{\text{col}}(\hat{\mathbf{q}}(\mathbf{M}_{\text{mask}} * \mathbf{M}_{\text{col}}(\mathbf{q})))|_1. \tag{13}$$

Here, $\mathbf{M}_{\text{mask}}$ is a matrix with only one random element set to 1 and the rest set to 0, and $\hat{\mathbf{q}}(\mathbf{M}_{\text{col}})$ is the result pose of self-collision resolution model that takes $\mathbf{M}_{\text{col}}$ as input. For notational simplicity, other inputs to $\hat{\mathbf{q}}(\mathbf{M}_{\text{col}})$ are omitted.

6.2. Random Segment Padding

Preparing a sufficient variety of character body shapes for learning is not easy. A commonly used technique to improve the generalization ability of neural networks when data is scarce is data augmentation. **R**andom **S**egment **P**adding (RSP) is a data augmentation method for character appearance.

Given two body segments S_{i} and S_{j}, and a new body segment S'_{j} thickened by an arbitrary real value p, the distance between S_{i} and S'_{j} can be approximated by the following relation:

$$D(S_{\text{i}}, S'_{\text{j}}) \sim D(S_{\text{i}}, S_{\text{j}}) - p. \tag{14}$$

Here, $D(S_{\text{i}}, S_{\text{j}})$ is the distance between the two body segments S_{i} and S_{j}. Based on this relationship, RSP introduces a procedure in the collision detection module to expand or shrink each body segment during the calculation of $\mathbf{M}_{\text{col}}$. Specifically, padding the i-th body segment S_{i} by p can be achieved by adding $-p$ to both the entire i-th row and the i-th column of the distance matrix $\mathbf{M}_{\text{dist}}$ in the collision detection module. We sampled p within a certain range of negative and positive values for each body segment and applied padding accordingly. Padding was also applied to the extent of the bounding box, which is an input to the self-collision resolution model.

7. Experiments and Evaluation

7.1. Experimental Design

Datasets. We trained and evaluated our model using the Mixamo dataset. The Mixamo dataset includes characters with different shapes and skeleton ratios, as well as a variety of motions. For training, we used a total of 152,424 frames of motion data, and for testing, we used motion data with a total length of 40,074 frames. The number of characters used for training is three (AJ, Kaya, and Peasant Man), and testing was conducted on seven characters. All test characters and motion data used in the experiments were not used during training, unless otherwise noted.

Implementation Details. The number of joints in the characters used for training and inference is 22. The hyper parameters α, β, γ, ζ, η, λ, ν, d are 10, 0.5, 0.3, 0.00001, 0.3, 0.5, and 0.1, 8 respectively. The learning rate used for training is 0.0001, the batch size is 32, and the optimizer is Adam. Ω is set to ignore collisions between neighboring body segments on the skeleton and between segments within the same segment cluster. Additionally, collisions between the upper arm and chest segments, which are prone to collisions in a standing position, are also ignored. The padding extent p for RSP is uniformly sampled between values 0.05 greater than the minimum SDF value of each body segment and a maximum of 0.1. The submodule f_{Φ} is a 5-layer MLP with hidden layer dimensions of 64. Each linear module of the submodule $f_{\Delta\mathbf{q}}$ has an output dimension of 16, except for the final linear module. During training, the size of each character is normalized so that the height of the shoulders is 1. We trained our model for approximately 30 minutes on a device equipped with an Intel i7-13700K CPU and an NVIDIA GeForce RTX 3070 Ti. Custom differentiable CUDA kernels were written to increase the parallelism of the SDF querying process.

Baselines. To demonstrate the strengths of our model, we will compare it with two prior studies often considered state-of-the-art: R^{2}ET [ZWK*23] and the methodology by Kim et al. 2020 [KSKK20]. R^{2}ET consists of three modules: a Skeleton-aware module, which maintains the pose semantics by preserving the distances between joints, a Shape-aware module, which minimizes self-collisions while preserving the original contact status,

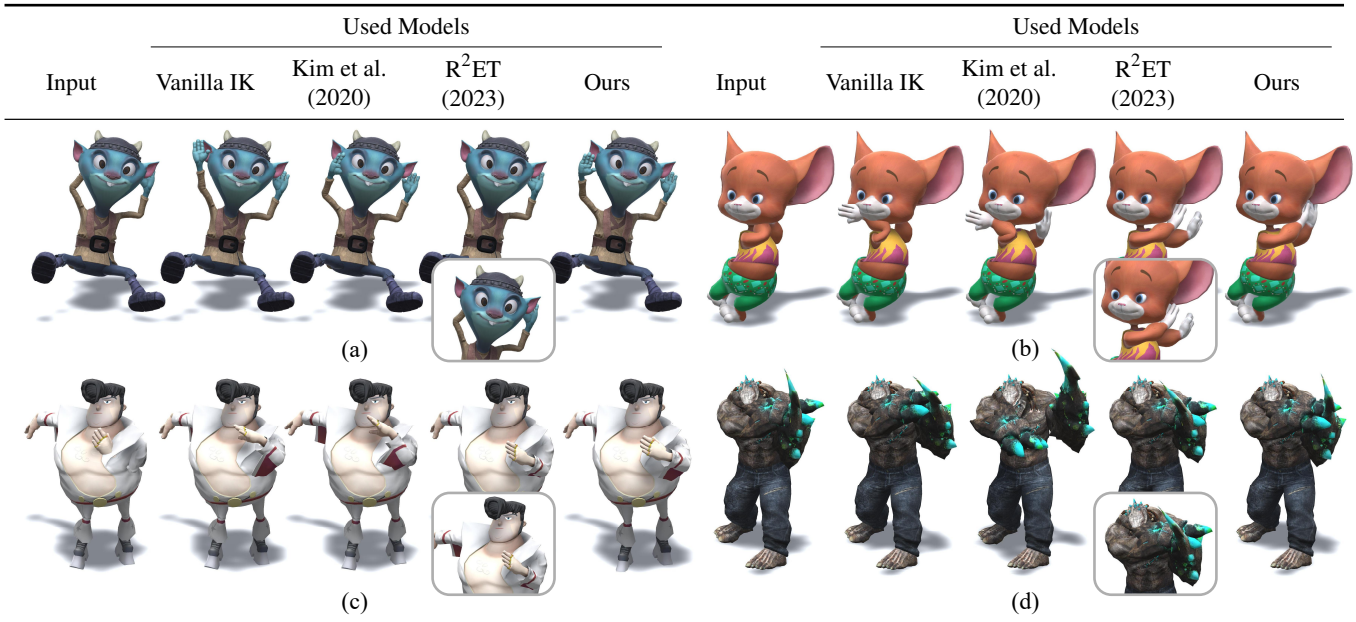


Figure 5: *Illustrations of the prediction results from prior researches and our model. R^2ET: we used a modified R^2ET with the skeleton-aware module turned off. The images within the box show the results of using the original version. While the visual results appear similar, our quantitative metrics indicate unfairly poorer performance when using the original R^2ET.*

and Balancing gate, which adjusts the amount of posture change caused by the Shape-aware module to reduce overshooting. Using the Skeleton-aware module changes the joint angles of the original pose, which can be disadvantageous for the quantitative metrics used in this paper. Therefore, focusing on performance in the self-collision avoidance task, we used only the Shape-aware module and Balancing gate of R^2ET for a fair comparison. Kim's methodology addresses collisions by applying iterative mesh-based IK and filtering, resulting in smooth collision resolution [KSKK20]. Kim's methodology focuses on collision avoidance between the character and the environment, but the same approach can be applied to self-collision. The last baseline method, which we refer to as vanilla IK, is a simplified version of the approach described in [KSKK20], with the filtering turned off. This results in a straightforward mesh-based IK solver that uses half-space constraints for collision avoidance.

Evaluation metrics. We aim to evaluate the model using 6 types of quantitative metrics. First, to measure the distance from the original, we use Mean Per Joint Position Error (MPJPE). Second, there are metrics to measure the penetration depth occurring in the pose: M_c, M_i, and M_o. Mean squared collision depth M_c is the mean of the squared penetration depths occurring in the current pose. Mean square skin distance (In-body) M_i and Mean square skin distance (Out-body) M_o are measured on segment clusters, which have experienced collisions in the original pose, and measure the mean squared distance to the nearest other segment cluster after collision avoidance. M_i measures negative distance, i.e., penetration depth, and M_o measures positive distance. Lastly, to measure the smoothness of the motion, we use the average magnitude of the acceleration Acc_p and Jerk $Jerk_p$ of each joint after Forward Kinematics. M_c, M_i, and M_o are measured using the body segments of the end effectors of both arms and legs and their parent body segments. All metrics are calculated in centimeters, assuming that the height of the Passive Marker Man in the Mixamo dataset is 180 cm.

7.2. Performance comparison

Collision Avoidance. Figure 5 shows the results of self-collision avoidance on the test data. The methods of previous studies either performed less collision avoidance during the process of resolving collisions (a), resolved collisions in the wrong direction (the second and third column of b), or encountered issues with excessive collision avoidance, known as overshooting (the fourth column of b, and the third column of d). In contrast, our model, when compared to existing models, showed visually plausible collision avoidance results for deep collisions.

As shown in Table 1, our model achieved better scores in M_c, a metric measuring penetration depth, compared to existing models. Additionally, our model also showed the best results in M_i, indicating that our model outperforms existing models in collision avoidance. When collisions occur, overshooting can happen as the body is pushed outward, so if M_c or M_i decreases, M_o is likely to increase. However, our model showed values for M_o that were comparable to those of Kim et al.'s model, which had the best performance in the M_o metric, despite having lower M_c and M_i than other models. This demonstrates that our model can perform accurate collision avoidance for current collisions. Figure 7a illustrates the self-collision avoidance for a specific motion, showing the signed distance of the right hand to the head and torso for each frame.

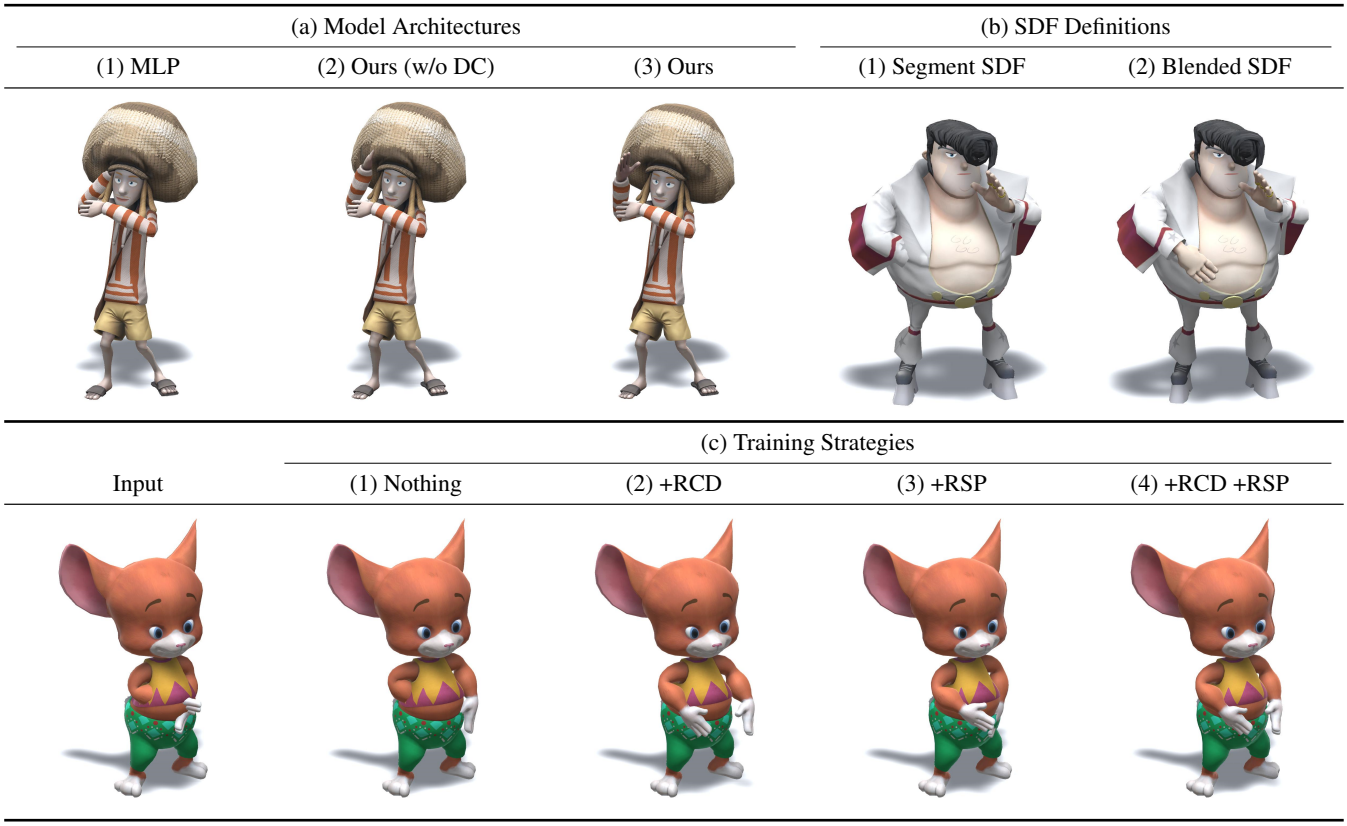


Figure 6: *(a) Comparative experiments on model architecture. (b) Comparative experiments on types of SDF. (c) Ablation studies on training methodologies. (c1) When no training methodology is used, it can be seen that accurate self-collision avoidance for the right arm is not achieved. (c2) When trained using only RCD, self-collisions are resolved, but excessive avoidance occurs. (c3) When using only RSP, collision avoidance is performed, but some self-collisions still remain. (c4) When using both RCD and RSP, accurate self-collision avoidance is achieved.*

Table 1: *Quantitative evaluation of performance comparison with existing models.*

Used Models	MPJPE ↓	M_c ↓	M_i ↓	M_o ↓	Acc_p ↓	$Jerk_p$ ↓
Input	0.0	83.79	83.79	0.0	0.19	0.13
Vanilla IK	4.18	60.52	50.69	5.99	0.46	0.64
Kim et al.	4.00	27.71	22.69	**1.85**	**0.20**	**0.16**
R^2ET	**2.15**	48.67	39.19	3.88	0.21	0.17
Ours	3.37	**16.37**	**11.85**	2.70	0.21	0.18

As seen in the graph, existing models perform collision avoidance when no collision occurs or exhibit overshooting or insufficient collision avoidance when a collision does occur. In contrast, our model follows the original motion when no collision occurs and performs accurate collision avoidance when a collision does occur.

Smoothness. Figure 7b shows the height of the right-hand joint over time in centimeters, measured while performing self-collision avoidance across all frames of a specific motion. This metric demonstrates that our model's results closely follow the original motion, even though many collisions are occurring. This indicates that our model produces sufficiently smooth motion results.

In our methodology, the smoothness of motion is achieved through the continuity of collision information. As seen in Table 5, using selective SDF worsens the smoothness metrics Acc_p and $Jerk_p$, due to the inherent discontinuity in the SDF function definition.

Computational Cost. We measured the average inference speed of both the baseline methods and our model across the entire dataset. Without parallel processing, Vanilla IK took 41.661 ms, Kim et al.'s method took 3.645 ms, R^2ET took 1.75 ms, and our method took 7.9 ms per frame. With parallel processing in both the frame and batch direction, R^2ET took 1.63 ms, while our method took 0.556 ms on average. In our model, linear blend skinning, SDF querying, and neural network inference account for 47%, 13%, and 40% of the total processing time, respectively. Kim et al.'s method converged faster than the Vanilla IK because it reuses the computation results from the previous frame.

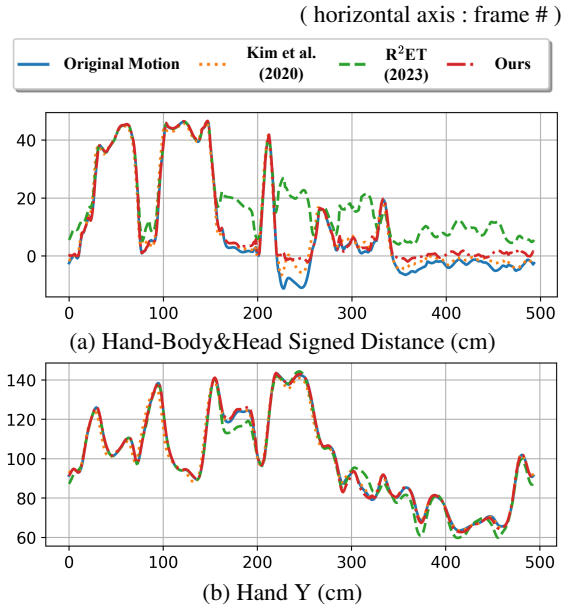


Figure 7: *Graph of each model's prediction results for a specific motion.*

Table 2: *Quantitative evaluation of performance changes according to the model architecture used.*

Used Architectures	MPJPE ↓	M_c ↓	M_i ↓	M_o ↓
Input	0.0	83.79	83.79	0.0
MLP	3.87	65.43	43.15	11.96
Ours (w/o DC)	**2.97**	33.01	23.84	**1.94**
Ours	3.37	**16.37**	**11.85**	2.70

7.3. Ablation study

Model architecture. Figure 6a shows the prediction results according to the model architecture used. All the proposed training strategies were employed when training each model. (1) is a simple MLP consisting of 8 layers, and (2) is a model with the same structure as ours, but with collision information only inputted into the first layer of $f^c_{\Delta \mathbf{q}}$ (without **D**ense **C**onnections). Even though the character shown in Figure 6a is part of the training set, both methods (1) and (2) fail to accurately resolve collisions between the hand and head. In contrast, our model architecture achieves accurate collision avoidance. This is also confirmed in Table 2. As shown in this table, our model architecture produces improved results compared to other architectures, especially in metrics related to penetration depth. This demonstrates that our model architecture effectively utilizes collision information.

Table 3 shows the performance changes according to the method of embedding the collision vector $\mathbf{M}_{\text{col,c}}$ into the semantic matrix Φ_c. The "Concat" embedding method refers to concatenating $\mathbf{M}_{\text{col,c}}$ into Φ_c. Because the dimensionality of $\mathbf{M}_{\text{col,c}}$ ($\mathbb{R}^6$) is much smaller than that of Φ_c ($\mathbb{R}^{6\times d}$), a simple concatenation may lead to reduced accuracy in collision avoidance. Instead, $\mathbf{M}_{\text{col,c}}$ is concatenated into Φ_c by duplicating it to match the dimensionality of the semantic vector (d), resulting in $\Phi'_c \in \mathbb{R}^{6\times 2*d}$. The "Add" method embeds each element of $\mathbf{M}_{\text{col,c}}$ into the corresponding semantic vector by addition, resulting in the mean of the column vectors of Φ_c being $\mathbf{M}_{\text{col,c}}$. As shown in the metrics, the "Concat" method has the smallest M_i metric for collision avoidance but the largest M_o. The "Add" method showed the lowest performance in collision avoidance metrics. In contrast, our "Mul" method achieved similar values in collision avoidance metrics M_c, M_i, and M_o compared to the "Concat" method, while achieving the highest score in the MPJPE metric. This can be intuitively explained as follows. The semantic collision matrix Φ'_c acts as a signal that determines how and in which direction the segment cluster $\mathcal{B}_c$ should move. Because Φ'_c is proportional to the penetration depths $\mathbf{M}_{\text{col,c}}$, our design encourages a proportional relationship between the magnitude of the pose change $\Delta\mathbf{q}_c$ and the penetration depths.

Table 3: *Quantitative evaluation of performance changes according to the methodology for embedding collision information in the semantic matrix Φ'.*

Collision Embedding Methods	MPJPE ↓	M_c ↓	M_i ↓	M_o ↓
Input	0.0	83.79	83.79	0.0
Concat	3.51	16.99	**11.62**	3.20
Add	3.61	29.27	20.87	**2.46**
Mul (Ours)	**3.37**	**16.37**	11.85	2.70

Training strategies. Figure 6c shows the prediction results according to the applied training methodologies. It can be observed that only when both RCD and RSP are used, self-collision avoidance is accurately performed. This trend is also evident in Table 4. For the model that uses no training strategy, the M_i metric is high, and the M_o metric is low, indicating that the model does not adequately perform collision avoidance. For the model that uses RCD, the M_i metric decreases, but at the same time, M_o increases. This indicates that while the model with RCD can perform collision avoidance sufficiently, it also suffers from overshooting. Conversely, for the model that uses only RSP, the M_c value is high, indicating that this model does not sufficiently perform self-collision avoidance. Finally, for the model that uses both RCD and RSP, the M_i value is not significantly different from the methodology that uses only RCD, but it achieves a lower M_o compared to the methodologies that use no training strategy or only RCD. This suggests that the overshooting problem can be mitigated when the model is trained with RSP. Therefore, we argue that data augmentation regarding character body shapes, such as RSP, can be used to enable artificial neural networks to perform accurate collision avoidance for various body types.

Blended SDF. Table 5 shows the prediction results of the model according to the definition of SDF used for collision checking. The model that uses Segment SDF shows poor performance in the M_c

Table 4: *Quantitative evaluation of performance changes according to the training strategy.*

Training Strategies	MPJPE ↓	M_c ↓	M_i ↓	M_o ↓
Input	0.0	83.79	83.79	0.0
Nothing	3.69	31.07	23.92	3.36
+RCD	4.11	**14.80**	**10.28**	4.27
+RSP	**2.06**	28.32	24.28	**0.78**
+RCD +RSP (Ours)	3.37	16.37	11.85	2.70

Table 5: *Quantitative evaluation of performance changes according to the selection of SDF.*

SDF Definitions	MPJPE ↓	M_c ↓	M_i ↓	M_o ↓	Acc_p ↓	$Jerk_p$ ↓
Input	0.0	83.79	83.79	0.0	0.19	0.16
Segment	**2.41**	22.54	17.14	**1.38**	0.22	0.19
Selective	2.91	**16.00**	11.88	2.42	0.23	0.22
Blended	3.37	16.37	**11.85**	2.70	**0.21**	**0.18**

and M_i metrics, indicating insufficient collision avoidance. In contrast, models that use Blended SDF improve each metric, as can also be visually confirmed in Figure 6b.

8. Conclusion

In this study, we propose an artificial neural network-based self-collision resolution model that resolves self-collisions in given motions based on accurate collision information, as well as two training strategies to train this model: **R**andom **C**ollision **D**ropout (RCD) and **R**andom **S**egment **P**adding (RSP). To ensure the self-collision resolution model accurately perceives collision information, we divided the model into two submodules and embedded the collision information into the semantic vector, the output vector of the first submodule. This architecture laid the foundation for the self-collision resolution model to perform accurate collision avoidance. RCD guided the model to respond to each collision event instead of ignoring collisions, and RSP enhanced the model's generalization ability regarding the character's body shape. As a result, the network trained on just three characters outperformed both a model trained on a larger set of characters and a slower approach reliant on geometric computations. In conclusion, our proposed methodology allows for accurate self-collision avoidance for various collisions occurring in motion and exhibits high generalization ability.

Despite our model performing well on various poses, it is noteworthy that some collision details are lost when summarizing all collision information into a compact collision matrix. Although this loss of information could theoretically degrade performance, our manual inspections of the results have not provided sufficient evidence of this. This is likely because the poses and bounding boxes are provided as inputs to the model, and the collision matrix is repeatedly calculated during the learning process. This repetition allows various combinations of collisions to be observed and resolved.

(a) (b)

Figure 8: *Examples of failure cases of our model. The left side shows the input pose, and the right side displays the model's prediction. (a) When the two arms crossed, the collision resolution was not sufficiently performed. (b) When the right arm deeply collided with both the head and the left arm simultaneously, the model's prediction still exhibits self-intersections, specifically between the right arm and the head, as well as between both arms.*

Limitations. This study still has several limitations. Despite being able to generate plausible collision resolution results for many poses, there are cases where our model frequently fails. First, as shown in Figure 8a, when there are multiple possible solutions for self-penetration such as crossed arms, it is challenging for a regression model to find an appropriate solution. Although this issue is common to all collision detection algorithms, when computing penetration depth using Signed Distance Fields (SDF), the SDF value initially decreases as the collision depth increases. However, it eventually approaches zero again as it nears the opposing surface. This causes problems with vanishing SDF gradients; for example, when two arms are deeply overlapped, the optimal avoidance direction becomes ambiguous, hindering effective collision avoidance. More images of our model's success and failure cases are included in Appendix.

Second, as shown in Figure 8b and also in the failure cases in the Appendix, our regression algorithm occasionally falls into local minima in cases of very deep collisions that cannot be resolved with small movements, or in complex situations where an immediate resolution causes another collision. It is possible that the problem of self-collision avoidance solely based on an input pose is inherently ambiguous, because self-collisions occur during the execution of motion and the resolution results must maintain continuity. A global solution requires the context of the motion to resolve collisions effectively.

Next, because our model operates on a per-frame basis, the smoothness of the output motion depends solely on the smoothness of the trained neural network and the temporal smoothness of the input vectors. Consequently, in cases of sudden movements, the animation curve may be clipped to ensure zero collisions, removing the curve where the collision occurs. This ultimately leads to the removal of the motion's intended meaning. Addressing this issue requires using different collision resolution strategies depending on

the motion context, even for the same pose. For example, additional research is needed to allow preemptive motion adjustments through future frame predictions to prevent collisions. Alternatively, assuming an offline retargeting problem, it may be helpful to design a loss function that includes both spatial elements and temporal elements. Developing a collision resolution model that takes motion context into account would be an interesting subject for future research.

Lastly, although this study focuses solely on self-collision avoidance, achieving complete motion retargeting requires predicting root motion and addressing foot sliding and contact semantic preservation. Additionally, for the model to be more practically useful, it should resolve self-penetration for characters with various skeleton structures. Currently, our model is applicable only to a fixed skeleton structure. To use our model on different skeleton structures, retraining is required, or the existing skeleton of a given mesh needs to be replaced with a compatible one using classical or learning-based retargeting algorithms. Research on these issues will be interesting future research topics.

9. Acknowledgement

This work was supported by NC Research under Project Number NCSOFT2023-1918-K and partially supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2020-II201373, Artificial Intelligence Graduate School Program at Hanyang University).

References

[ALL*20] ABERMAN K., LI P., LISCHINSKI D., SORKINE-HORNUNG O., COHEN-OR D., CHEN B.: Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 62–1. 3

[BWBM20] BASSET J., WUHRER S., BOYER E., MULTON F.: Contact preserving shape transfer: Retargeting motion from one shape to another. *Computers and Graphics 89* (2020), 11–23. 3

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, p. 33–42. 2

[HLvdMW18] HUANG G., LIU Z., VAN DER MAATEN L., WEINBERGER K. Q.: Densely connected convolutional networks, 2018. 6

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, Association for Computing Machinery. 3

[HZZ*24] HU L., ZHANG Z., ZHONG C., JIANG B., XIA S.: Pose-aware attention network for flexible motion retargeting by body part. *IEEE Transactions on Visualization and Computer Graphics 30*, 8 (Aug. 2024), 4792–4808. 3

[JKL18] JIN T., KIM M., LEE S.: Aura mesh: Motion retargeting to preserve the spatial relationships between skinned characters. *Computer Graphics Forum 37*, 2 (2018), 311–320. 3

[JKY*18] JANG H., KWON B., YU M., KIM S. U., KIM J.: A variational u-net for motion retargeting. In *SIGGRAPH Asia 2018 Posters*. 2018, pp. 1–2. 2, 3

[KSK21] KIM J., SEOL Y., KWON T.: Interactive multi-character motion retargeting. *Computer Animation and Virtual Worlds 32*, 3-4 (2021), e2015. 3

[KSKK20] KIM J., SEOL Y., KIM H., KWON T.: Interactive character posing with efficient collision handling. *Computer Animation and Virtual Worlds 31*, 3 (2020), e1923. 7, 8

[KYZ*20] KARUNRATANAKUL K., YANG J., ZHANG Y., BLACK M. J., MUANDET K., TANG S.: Grasping field: Learning implicit representations for human grasps. In *2020 International Conference on 3D Vision (3DV)* (2020), IEEE, pp. 333–344. 3

[LCC19] LIM J., CHANG H. J., CHOI J. Y.: Pmnet: Learning of disentangled pose and movement for unsupervised motion retargeting. In *BMVC* (2019), vol. 2, p. 7. 3

[LMHM18] LIU Z., MUCHERINO A., HOYET L., MULTON F.: Surface based motion retargeting by preserving spatial relationship. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games* (New York, NY, USA, 2018), MIG '18, Association for Computing Machinery. 3

[LMT08] LYARD E., MAGNENAT-THALMANN N.: Motion adaptation based on character shape. *Computer Animation and Virtual Worlds 19*, 3-4 (2008), 189–198. 3

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., p. 39–48. 2

[PKH*17] PIRK S., KRS V., HU K., RAJASEKARAN S. D., KANG H., YOSHIYASU Y., BENES B., GUIBAS L. J.: Understanding and exploiting object interaction landscapes. *ACM Transactions On Graphics (TOG) 36*, 3 (2017), 1–14. 3

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 3

[RWY*23] REDA D., WON J., YE Y., VAN DE PANNE M., WINKLER A.: Physics-based motion retargeting from sparse inputs. *Proc. ACM Comput. Graph. Interact. Tech. 6*, 3 (aug 2023). 3

[SHX*22] SHE Q., HU R., XU J., LIU M., XU K., HUANG H.: Learning high-dof reaching-and-grasping via dynamic representation of gripper-object interaction, 2022. 3

[TPM21] TAN Q., PAN Z., MANOCHA D.: Lcollision: Fast generation of collision-free human poses using learned non-penetration constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 3913–3921. 3

[VCH*21] VILLEGAS R., CEYLAN D., HERTZMANN A., YANG J., SAITO J.: Contact-aware retargeting of skinned motion, 2021. 2, 3

[VYCL18] VILLEGAS R., YANG J., CEYLAN D., LEE H.: Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8639–8648. 2, 3

[ZWK14] ZHAO X., WANG H., KOMURA T.: Indexing 3d scenes using the interaction bisector surface. *ACM Transactions on Graphics (TOG) 33*, 3 (2014), 1–14. 3

[ZWK*23] ZHANG J., WENG J., KANG D., ZHAO F., HUANG S., ZHE X., BAO L., SHAN Y., WANG J., TU Z.: Skinned motion retargeting with residual perception of motion semantics & geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 13864–13872. 2, 3, 4, 6, 7

[ZYSK21] ZHANG H., YE Y., SHIRATORI T., KOMURA T.: Manipnet: neural manipulation synthesis with a hand-object spatial representation. *ACM Trans. Graph. 40*, 4 (jul 2021). 3

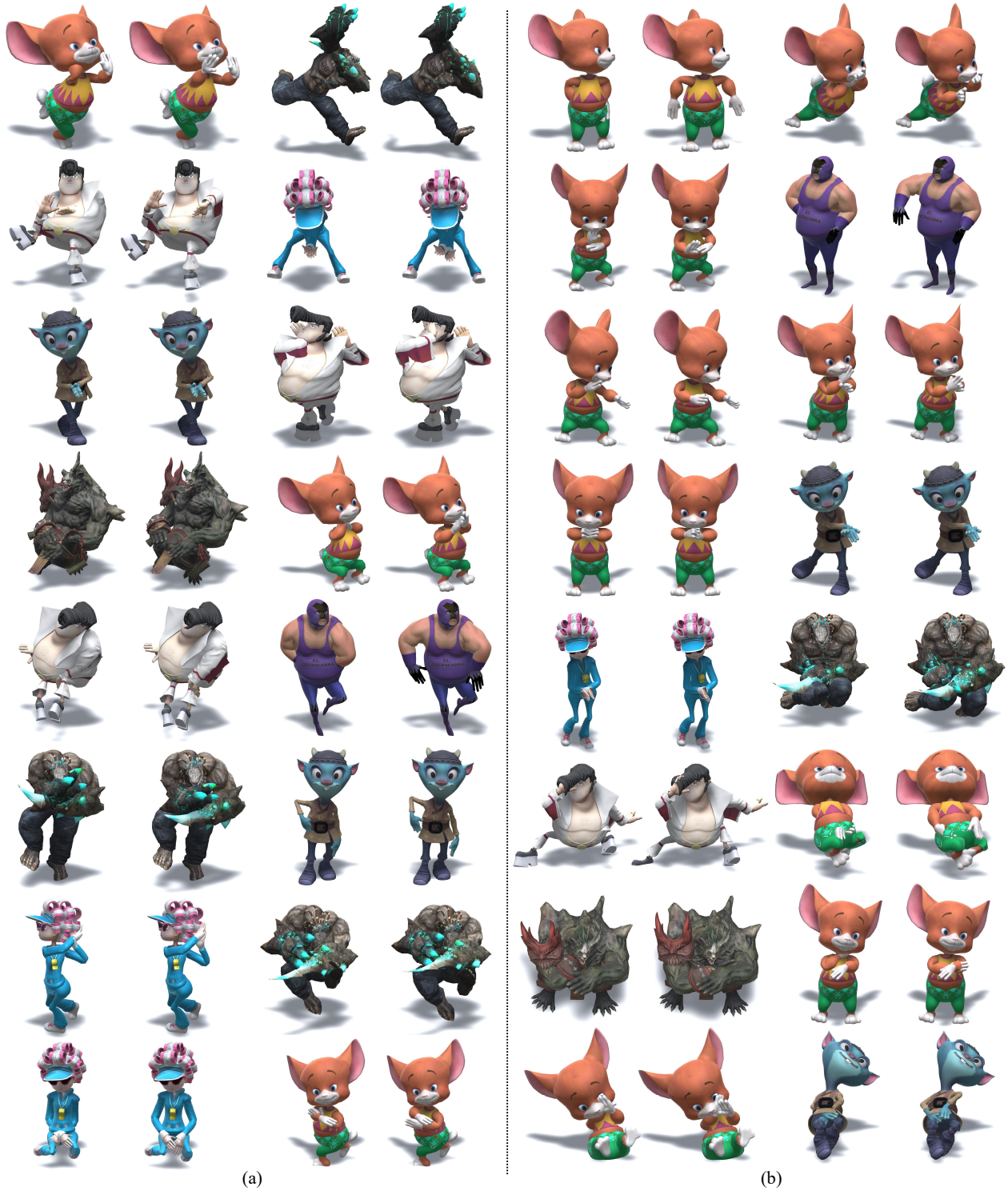


Figure 9: *A list of results of our model resolving self-penetrations in various poses. The odd-numbered columns display the original poses, while the even-numbered columns display the model's predictions. (a) shows cases where self-penetration resolution was successful, while (b) shows cases where it failed. In (b), you can observe instances where overshooting or insufficient collision resolution occurred, and where resolving of one collision led to the occurrence of another self-penetrations.*