Zhu *et al.*
Visual Computing for Industry, Biomedicine, and Art (2022) 5:15
https://doi.org/10.1186/s42492-022-00113-4

Visual Computing for Industry, Biomedicine, and Art

ORIGINAL ARTICLE Open Access

Collision-aware interactive simulation using graph neural networks

Xin Zhu[1,2], Yinling Qian[2], Qiong Wang[2*], Ziliang Feng[1] and Pheng-Ann Heng[2,3]

Abstract

Deep simulations have gained widespread attention owing to their excellent acceleration performances. However, these methods cannot provide effective collision detection and response strategies. We propose a deep interactive physical simulation framework that can effectively address tool-object collisions. The framework can predict the dynamic information by considering the collision state. In particular, the graph neural network is chosen as the base model, and a collision-aware recursive regression module is introduced to update the network parameters recursively using interpenetration distances calculated from the vertex-face and edge-edge tests. Additionally, a novel self-supervised collision term is introduced to provide a more compact collision response. This study extensively evaluates the proposed method and shows that it effectively reduces interpenetration artifacts while ensuring high simulation efficiency.

Keywords: Deep physical simulation, Collision-aware, Continuous collision detection, Graph neural network

Introduction

Many computer graphics applications, such as computer games, movie production, and fashion design, require physical simulation. Traditional numerical calculation methods produce physically accurate and visually excellent results. However, these methods are time consuming. Consequently, they cannot satisfy the performance requirements for interactive applications.

Deep simulation methods have emerged as popular alternatives to traditional numerical calculation methods owing to the rapid development of deep learning techniques. These methods [1–4] use the ability of neural networks to learn nonlinear functions to propose differentiable models that output deformable objects as functions of the target shape, pose, motion, and other design parameters. However, these methods perform poorly in collision detection and response (CDR), which has a significant impact on visual realism and simulation accuracy. To avoid interpenetration, these methods manually set a relatively large collision threshold in the training data generation process [1]. However, a manually set threshold cannot meet the requirements for an accurate CDR.

This study proposes a framework for collision-aware interactive physical simulation using a graph neural network (GNN), which can achieve a CDR function similar to continuous collision detection (CCD), which is the most effective method for solving the CDR problem in traditional physical simulation. The GNN was used as the base model because it can provide complete vertex-edge-face information, which can be used intuitively in basic geometric primitive collision tests. Additionally, a novel collision-aware recursive regression module is introduced to update the network parameters recursively using the interpenetration distances calculated from the vertex-face and edge-edge tests.

Using a regression module, our model detects collisions. Finally, to provide a compact collision response, a novel self-supervised term is introduced. In summary, our main contributions are as follows: (1) propose a GNN with a collision-aware recursive regression module

*Correspondence: wangqiong@siat.ac.cn

[2] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
Full list of author information is available at the end of the article

to effectively sense and respond to tool-object collisions; (2) a novel self-supervised collision term is introduced to reduce the interpenetration errors in unseen (that is, test) sequences and provide a more compact collision response; and (3) the proposed method was extensively evaluated in several common interactive simulation scenes with vertex-face, edge-face, and face-face collisions.

Related works

This section reviews three main areas: deep simulation, CCD, and GNNs.

Deep simulation

Neural networks can be used as effective function approximators in physical systems. For linear elastic deformation, Luo et al. [3] proposed a highly reusable and efficient neural network-based nonlinear deformable simulation framework, which partially restores the force-displacement relationship by warping the simulated nodal displacement, and used a simplistic constitutive model to infer the linear elasticity. For nonlinear elastic deformation, Holden et al. [1] combined subspace simulation techniques with machine learning to support interactions with external objects. Romero et al. [4] used a model formula with nonlinear corrections applied to the local undeformable setting and decoupled internal and external contact-driven corrections.

The collision process is one of the physical simulation difficulties associated with deep learning techniques. The most basic method is to learn the implicit collision relations between collision objects. Teng et al. [5] managed self-collisions by applying forces on a sparse set of de-projected simulation points. They supported external collisions by allowing partial, albeit costly, full-space simulations in collision-prone mesh areas. Additionally, Tan et al. [6] presented a learning-based method that synthesizes collision-free 3D human poses. They decomposed whole-body collisions into groups of collisions between localized body parts using a bi-level autoencoder. Pfaff et al. [7] took advantage of the excellent explanatory capability of GNNs for graph datasets (mesh-based datasets), and their model can learn the dynamics of a wide range of physical systems, from cloth simulation over structural mechanics to fluid dynamics directly. In this study, a GNN is used as a base model because it can learn complete vertex-edge-face information.

CCD

CCD is widely applied in many areas, including physical-based simulation, computer-aided design/computer-aided manufacturings, and robot motion planning. Its main purpose is to use some form of the interpolating trajectory to check for collisions between two discrete positions of objects or primitives. A common method of CCD is to simply enclose the bounding volumes (BVs) at the beginning and end of a motion step using a swept volume. Axis-aligned bounding boxes are usually chosen for this method. Coming and Staadt [8] proposed a velocity-aligned discrete oriented polytopes as a type of swept volume for underlying spheres as BVs. Additionally, Redon et al. [9] proposed an oriented bounding boxes algorithm. Penetration depth-based detection is another method of CCD. The minimum distance is not a good measure for defining repelling forces, and computing the exact impact time using CCD is too time-consuming for real-time applications. Redon and Lin [10] estimated the local penetration depth on the graphics processing unit using the local penetration direction computed for these regions. Tang et al. [11] traced the contact features along their deforming trajectories and accumulated penalty forces along the penetration time intervals between overlapping feature pairs.

Choi et al. [12] presented a framework for the CCD of composite quadric models with piecewise linear or quadric surface patches as boundary surfaces and conic curves or line segments as boundary curves. Although these methods can effectively provide CDR in traditional physical simulations, deep simulations remain an open problem.

GNNs

GNNs have been shown to be effective representations for learning large-scale tasks [13]. A GNN can effectively learn knowledge representation [14], message passing [15], and encode long-range dependencies (video processing). GNNs also perform well in dynamic physical systems, such as for climate prediction [16], with an emphasis on individual objects [17] and their relations [18], partially observable systems [19], prevalent interactions within physical systems [14], hierarchically organized particle systems [20], or more generally physical simulation [7, 21, 22].

Methods

The objective of the study is to design a deep interactive physical simulation framework that can effectively address tool-object collisions. A GNN-based encoder-processor-decoder architecture was chosen as the baseline, which can provide complete vertex-edge-face information. To detect tool-object collisions, a collision-aware recursive regression module that uses interpenetration distances calculated from vertex-face and edge-edge tests to recursively update network parameters is introduced. Furthermore, a novel self-supervised collision term to provide a more compact collision response

is introduced to reduce the interpenetration errors in unseen (that is, test) sequences. Figure 1 shows an overview of the proposed method.

GNN-based architecture

A GNN-based encoder-process-decoder architecture is used to learn the dynamic information. GNNs, compared with other networks, can provide complete vertex-edge-face information. In particular, the dynamic information is encoded to a mesh graph, passed the messages on the mesh graph, and adapted the mesh discretization during the forward simulation. The mesh discretization information of the latent space contains the dynamic information of the system, and the mesh discretization information can be decoded to learn the dynamic information of the system. Figure 2 shows the network-specific configuration.

A combined mesh $M = \left(M_X^t, M_Y^{t+1}\right)$ is the model input, where $M_X^t = (V_X, E_X)$is the simulation mesh of the deformable object at time t, nodes V_X connected by mesh edges E_X an $M_Y^{t+1}\left(M_Y^{t'}, P_{t+1}\right) = (V_Y, E_Y)$ is the simulation mesh of tool at time $t+1$, nodes V_Y connected by mesh edges E_Y . Additionally, $P_{t+1}(p_{t+1}, o_{t+1})$ is the pose of the tool (position p_{t+1} and orientation o_{t+1}) at time $t+1$ and $M_Y^{t'}$ is the base tool mesh at time t'. We used P_{t+1} and $M_Y^{t'}$ to calculate the tool mesh M_Y^{t+1} at time $t+1$. Each node $i \in M$ is associated with coordinates u_i, additional dynamic information q_i.

Encoder: The combined mesh is encoded into a multi-graph $G=(V,E)$. The nodes in the mesh correspond to the nodes in the graph, and the edges in the mesh correspond to the E in the graph. They are used to calculate the dynamic information inside the system. E handles dynamic information external to the system, such as collisions and contacts, which are the overall information of the system. An edge feature is defined as follows: $e_{ij}\left(e_{ij}^W, e_{ij}^M\right)$, where $e_{ij}^M \in E$ and $e_{ij}^W = u_i - u_j$, if $|u_i - u_j| < r_W$, r_W is the collision radius. If the world distance between two nodes is less than r_W, the two nodes may collide. The node feature v_i is represented as a dynamic feature a_i and a one-hot vector of node types. Finally, the node and edge features are encoded by two latent layers multilayer perceptrons (MLPs) ϵ^E, ϵ^V into a 128 dimensional hidden vector.

Processor: Message-passing blocks were used to pass messages on the mesh graph and adapt the mesh discretization during forward simulation. The processor consists of L identical message passing blocks, which

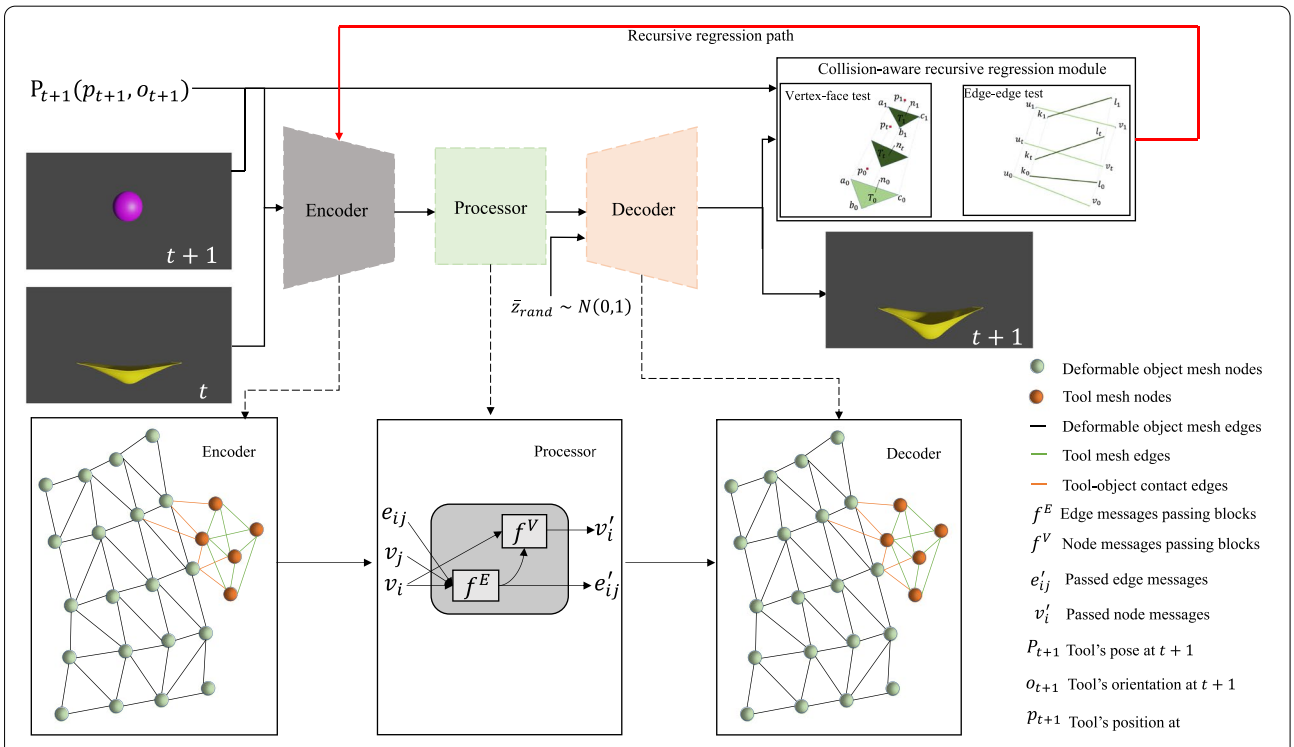

Fig. 1 Overview of the proposed method. GNN is the base model. A collision-aware recursive regression module updates the network parameters recursively using interpenetration distances calculated from vertex-face and edge-edge tests. A novel self-supervised collision term (random latent space vector $\bar{z}_{rand} \sim (0,1)$) provides a more compact collision response

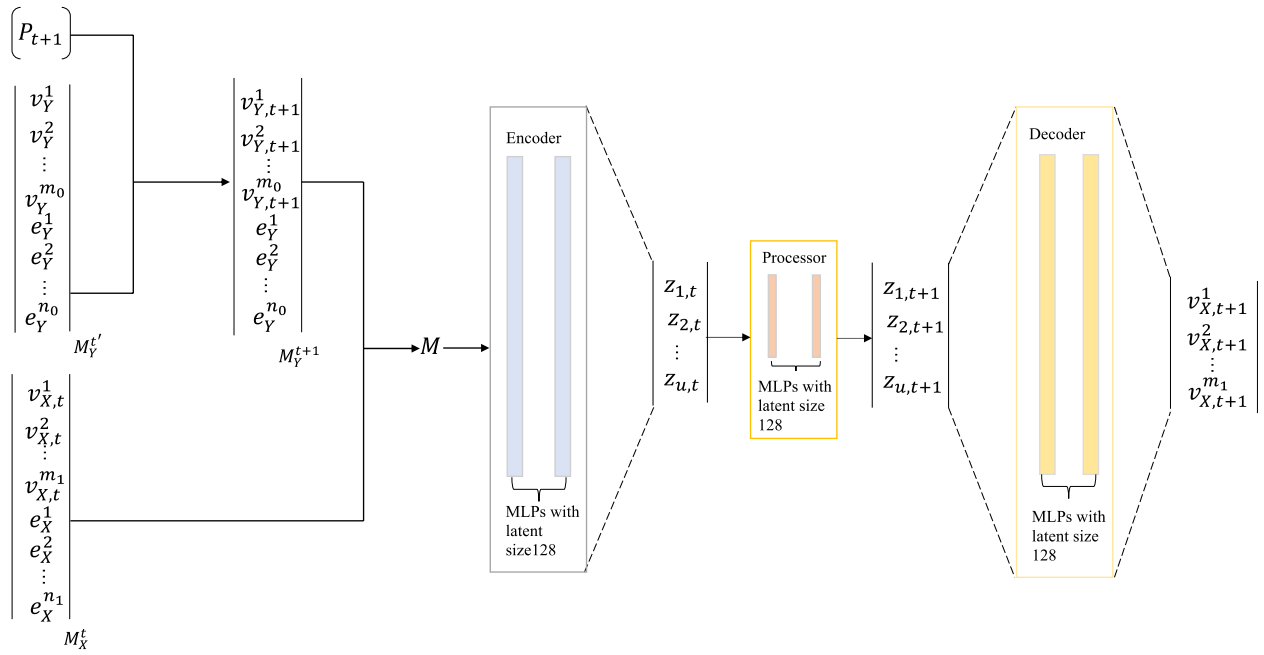

Fig. 2 The proposed network configuration. m_0, m_1, n_0, and n_1 are the numbers of nodes and edges of the deformable object and tool. The encoder, processor and decoder have two latent layers with a latent size 128

generalizes GraphNet blocks to multiple edge sets, and $L=2$ by default. Each block contains a separate set of network parameters and is sequentially applied to the output of the previous block, updating the edge e_{ij}, and node v_i embeddings to e'_{ij} and v'_i, respectively, by the following:

$$e'_{ij} \leftarrow f^E\left(e_{ij}, v_i, v_j\right), v'_i \leftarrow f^V\left(v_i, \sum\nolimits_j e'_{ij}\right) \tag{1}$$

where f^E and f^V are implemented using two latent-layer MLPs with residual connections. Then, the proposed model learns the dynamic information latent space at time $t+1$, the key to which is to decode the latent dynamic information to the real physical space.

Decoder: To transform the latent dynamic information space into real physical dynamic information, a two latent-layer MLP, δ^V, was used as a decoder to update the dynamic information of the nodes in the mesh by converting the latent node feature v^i at time t to the dynamic feature a^i of a deformable object at time t. The dynamic feature a^i is the derivative of the dynamic information q^i at time t. The forward Euler integration can be used to calculate the dynamic information q^i_{t+1} at time $t+1$. For first-order systems, $q^i_{t+1} = a^i + q^i_t$, whereas for second-order systems, $q^i_{t+1} = a_i + 2q^i_t - q^i_{t-1}$.

Furthermore, to train a collision-aware model that learns dynamic information and tool-object collisions, the proposed GNN-based model loss is defined as follows:

$$L_{GNN} = L_q + L_{ccd} + L_{compact} \tag{2}$$

where L_q is the dynamic information loss defined as follows:

$$L_q = |q - \overline{q}|_2 \tag{3}$$

L_{ccd} is the collision-aware recursive regression module's continuous collision-detection loss, which is explained in detail in collision-aware recursive regression module section. $L_{compact}$ is the self-supervised term loss that provides a compact collision response, which is explained in detail in self-supervised term section.

Collision-aware recursive regression module

The message-passing architecture learns dynamic information. However, it is difficult to detect the collision information in the system. To address this problem, the architecture outputs are used as the inputs to the collision-aware recursive regression module to calculate the interpenetration distance and update the network parameters recursively. Additionally, a novel self-supervised collision term is introduced to provide a more compact collision response.

To calculate the interpenetration distance, a non-penetration continuous collision-detection filter that filters vertex-face collision and edge-edge collision pairs is used. The interpenetration distance is defined as the continuous collision-detection loss of the module as follows:

$$L_{ccd} = \xi_{VF} + \xi_{EE} \tag{4}$$

where ξ_{VF} and ξ_{EE} are the distances between vertex-face and edge-edge collision pairs, respectively.

Traditional iterative continuous collision-detection algorithms are difficult to integrate into networks; therefore, a fast non-penetration continuous collision-detection filter [23] is chosen as the collision-aware recursive regression module to calculate the interpenetration distance. Furthermore, because of the high computational cost, a culling strategy based on the signed distance field (SDF) values is provided. The collision-detection module contains two terms: the vertex-face and edge-edge tests. Figures 3(a) and 4(a) show the vertex-face and edge-edge tests, respectively.

Vertex-face test: For a triangle T_t and a vertex P_t defined by the start and end positions during the interval $[0,1]$, these positions are linearly interpolated in the interval with respect to the time variable t. If the

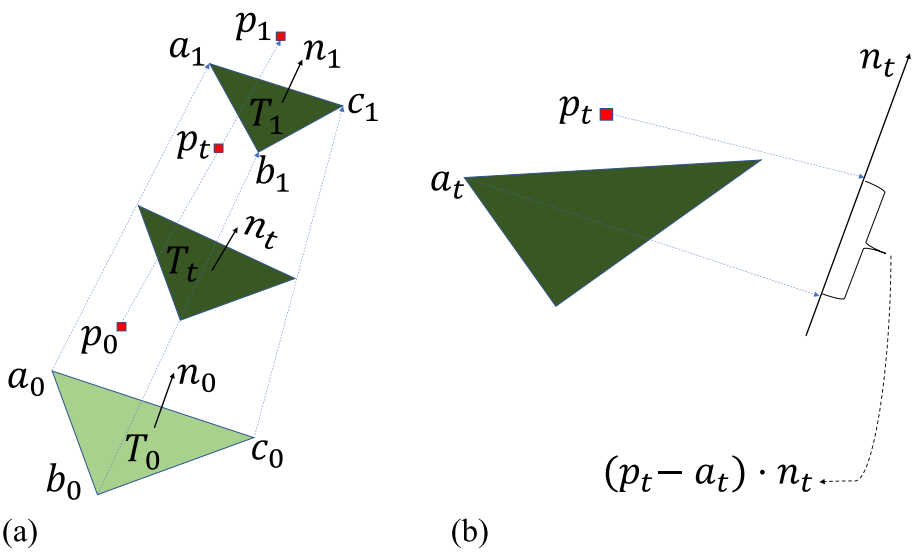

Fig. 3 Vertex-face test: To perform a vertex-face test between a deforming triangle (defined by a_0, b_0, and c_0 at $t=0$, and a_1, b_1, and c_1 at $t=1$) and a moving vertex (defined by p_0 at $t=0$ and p_1 at $t=1$), coplanarity between the vertex and the triangle by finding a t ($t \in [0, 1]$) when the projected distance along the normal vector of the triangle is equal to zero, that is, $(p_t - a_t) \cdot n_t = 0$ is checked. **a**: Deforming triangle T and deforming vertex p; **b**: Projected distance between p_t and T_t

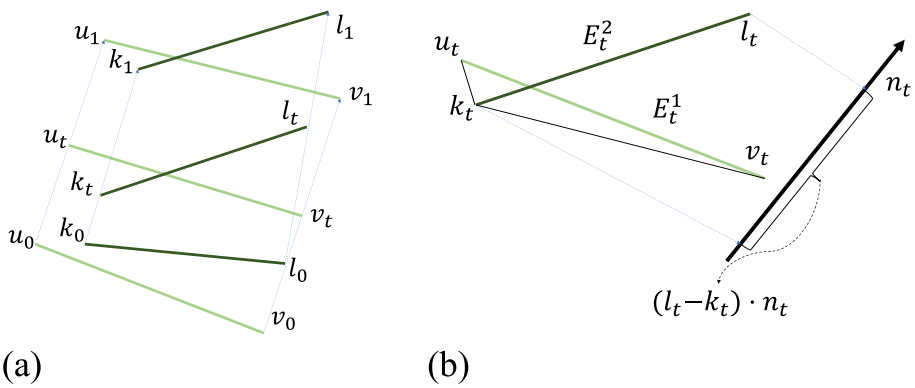

Fig. 4 Edge-edge test: To perform an edge-edge test between the two edges E^1 and E^2 (defined by u_0, v_0, and k_0, l_0 at $t=0$, and u_1, v_1, and k_1, l_1 at $t=1$), the coplanarity conditions of these vertices by finding a t ($t \in [0, 1]$) when the projected distance between l_t and the triangle defined by k_t, u_t and v_t is equal to zero, that is, $(l_t - k_t) \cdot n_t = 0$ is checked

following four scalar values, A, B, $\frac{2*C+F}{3}$, and $\frac{2*D+E}{3}$ have the same sign, T_t and P_t will not be coplanar during the interval:

$$\begin{aligned} A &= (p_0 - a_0)\ n_0 \\ B &= (p_1 - a_1)\ n_1 \\ C &= (p_0 - a_0)\ \overline{n} \\ D &= (p_1 - a_1)\ \overline{n} \\ E &= (p_0 - a_0)\ n_1 \\ F &= (p_1 - a_1)\ n_0 \end{aligned}$$

where n_0 is the normal of $\triangle a_0 b_0 c_0$, n_1 is the normal of $\triangle a_1 b_1 c_1$, and $\overline{n} = \frac{n_0 + n_1 - \left(\overrightarrow{v_b} - \overrightarrow{v_a}\right) \times \left(\overrightarrow{v_c} - \overrightarrow{v_a}\right)}{2}$, $\overrightarrow{v_a}$ is the vector of $a_0 a_1$, $\overrightarrow{v_b}$ is the vector of $b_0 b_1$, $\overrightarrow{v_c}$ is the vector of $c_0 c_1$.

Edge-edge test: For two edges E^1 and E^2 defined by the start and positions during the interval $[0,1]$, these positions are linearly interpolated in the interval with respect to the time variable t. If the following four scalar values: $A^{'}$, $B^{'}$, $\frac{2*C'+F'}{3}$, and $\frac{2*D'+E'}{3}$ have the same sign, E^1 and E^2 will not be coplanar during the interval.

$$\begin{aligned} A^{'} &= (l_0 - k_0)\ n_0^{'} \\ B^{'} &= (l_1 - k_1)\ n_1^{'} \\ C^{'} &= (l_0 - k_0)\ \overline{n}^{'} \\ D^{'} &= (l_1 - k_1)\ \overline{n}^{'} \\ E^{'} &= (l_0 - k_0)\ n_1^{'} \\ F^{'} &= (l_1 - k_1)\ n_0 \end{aligned}$$

where n_0' is the normal of $\triangle u_0 k_0 v_0$, n_1' is the normal of $\triangle u_1 k_1 v_1$, and $\overline{n}' = \frac{n_0' + n_1' - \left(\overrightarrow{v_u} - \overrightarrow{v_k}\right) \times \left(\overrightarrow{v_v} - \overrightarrow{v_k}\right)}{2}$, $\overrightarrow{v_k}$ is the vector of $k_0 k_1$, $\overrightarrow{v_u}$ is the vector of $u_0 u_1$, $\overrightarrow{v_v}$ is the vector of $v_0 v_1$.

The computation cost of every vertex-face and edge-edge pair is very large; therefore, the vertexes which SDF values that are smaller than a certain value before filtering are culled. The filtered vertex-face and edge-edge pairs that did not collide and the rest were defined as the collision pairs.

For vertex-face collision pairs, γ^{t_0,t_0+1} are vertex-face pairs during the interval frame $[t_0, t_0+1]$, and the vertex-face pairs collide at time t_0+1. Figure 3(b) shows the vertex-face distance at time interval $[0,1]$. To reduce the number of vertex-face collision pairs, the distance between vertex-face pairs is reduced. $D_{vf}^{t_0,t_0+1}$ is defined as the distance between vertex-face pairs during the interval frame interval $[t_0, t_0+1]$. Therefore, the vertex-face loss is defined as follows:

$$\xi_{VF} = \max\left(D_{vf}\right) \tag{5}$$

where

$$D_{vf} = \left[D_{vf}^{0,1}, D_{vf}^{1,2}, \ldots, D_{vf}^{t,t+1}\right] \tag{6}$$

For edge-edge collision pairs, η^{t_0,t_0+1} are edge-edge pairs during the interval frame $[t_0, t_0+1]$, and the edge-edge pairs collide at time t_0+1. Figure 4(b) shows the edge distance at time interval $[0,1]$. To reduce the number of edge-edge collision pairs, the distance between edge-edge pairs is reduced. $D_{ee}^{t_0,t_0+1}$ is defined as the distance between edge-edge pairs during the interval frame interval $[t_0, t_0+1]$. Therefore, the edge-edge loss is defined as follows:

$$\xi_{EE} = \max(D_{ee}) \tag{7}$$

where

$$D_{ee} = \left[D_{ee}^{0,1}, D_{ee}^{1,2}, \cdots, D_{ee}^{t,t+1}\right] \tag{8}$$

Self-supervised term

Using the learned dynamic information defined in GNN-based architecture section and the tool-object collision detection module in collision-aware recursive regression module section, a collision-aware model to learn the dynamic information and tool-object collision can be trained. However, there were interpenetration errors in the unseen (that is, test) sequence. This challenge is addressed by learning a compact collision response that reliably solves tool-object interpretations. To provide a compact collision response, the following self-supervised collision term is proposed:

$$Lcompact = \xi Random + LKL \tag{9}$$

and

$$\xi_{Random} = \max(\Delta - SDF(D(\overline{z}_{rand}), P_{t+1}), 0) \tag{10}$$

where $\overline{z}_{rand} \sim N(0,1)$, Δ is the collision-free constraint threshold, $SDF()$ is the signed distance field of the tool, $D()$ is the decoder of our model, and P_{t+1} is the pose of the tool at time $t+1$. The self-supervised term samples the latent space and checks collisions against a constraint tool mesh using a self-supervised strategy (that is, ground truth positions are not needed for this term). This key ingredient allows for thorough sampling of the latent space and the learning of a compact collision response that reliably solves the tool-object interpenetration problem.

The self-supervised loss is derived from ref. [24], which requires a consistent distribution of the sampled latent space and training data. To enforce a normal distribution in the latent space, an additional term L_{KL} is included.

Datasets

Generally, most mesh-based simulation methods are suitable for acquiring data for the proposed method. The inputs to the training procedure were a raw time series of frame-by-frame vertex positions and face indices. More details about the exact data acquisition process used in our results are provided.

All simulations were performed using the incremental potential contact (IPC) simulation library [25] and captured data at 25 fps. The datasets used are shown in Fig. 5. The IPC library can provide accurate CDR simulation results. The datasets used in this study involve vertex-face collisions (cone-bunny), edge-face collisions (knife-torus), and face-face collisions (sphere-mat and cylinder-banana). All datasets contain dynamic information (velocity), SDF values of the tools, vertex positions, and face information. The vertex-face collision datasets used in this study are cone-bunnies, which simulate a cone stabbing a bunny. The edge-face collision datasets used in this study are knife-torus, which simulate a knife cutting a rubber torus. The face-face collision datasets used in this study were a sphere-mat and cylinder-banana. The sphere-mat datasets simulated a rigid sphere falling onto a rubber mat, whereas the cylinder-banana datasets simulated a rigid cylinder pressing a banana. Table 1 shows the model complexity of the datasets.

Training

This section introduces the training software environment, normalization strategies, training noise, and optimization procedures used in this study.

Software: All models were implemented using TensorFlow1, Sonnet1, and the "Graph Nets" library.

Normalization: All input and target vectors elementwise were normalized to zero mean and unit variance, using statistics computed online during training. Normalization can lead to faster training and better performance. Preliminary experiments showed that normalization led to faster training, although the converged performance did not improve significantly.

Training noise: Modeling a complex and chaotic simulation system requires a model to mitigate error accumulation over long rollouts. Because the models in this study were trained on ground-truth one-step data, they were never presented with input data corrupted by this

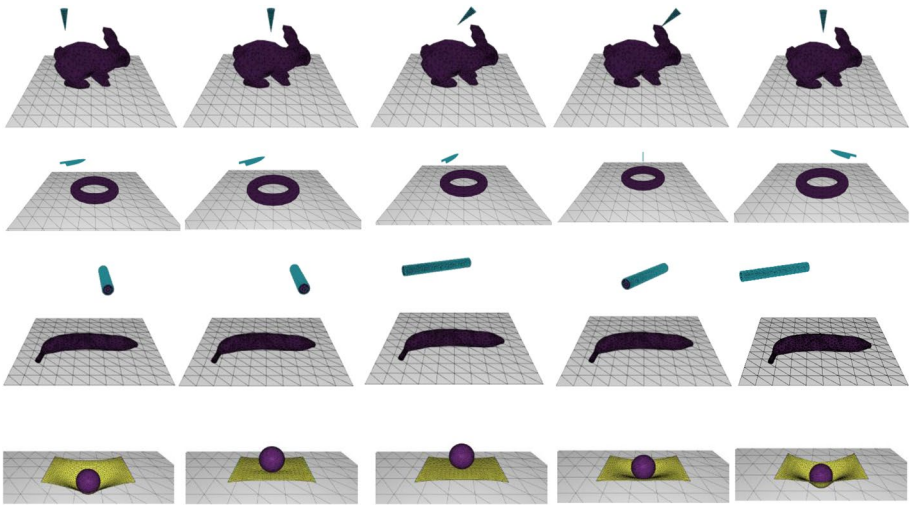

Fig. 5 Applying IPC [25] to generate the datasets used in this study. The IPC library can output accurate CDR simulation results

Table 1 Execution time of the proposed method and IPC

Scene	Vertex	Edge	Face	IPC (ms)	The proposed method (ms)	Speedup
Cone-bunny	3022	9054	6036	2070	75	27.60
Knife-torus	5926	17,772	11,848	6380	60	127.60
Sphere-mat	4439	13,305	8870	2440	135	18.07
Cylinder-banana	3607	10,809	7206	1200	90	13.33

type of accumulated noise. This means that when a rollout is generated by feeding the model with its own noisy, previous predictions as input, the fact that its inputs are outside the training distribution may lead to more substantial errors and thus rapidly accumulate further error. A simple approach to make the model more robust to noisy inputs by corrupting the input positions of the model with Gaussian noise is used; thus, the training distribution is closer to the distribution generated during rollouts.

Optimization procedures: The model parameters were optimized over this loss with the Adam optimizer [26], using a nominal mini-batch size of one. A maximum of 1×10^5 gradient update steps was performed with an exponential learning rate decay from 10^4 to 10^6. While models can be trained in fewer steps, this study avoided using aggressive learning rates to reduce variance across datasets and make comparisons across settings fairer.

Results

This section demonstrates that our model can reliably process collisions in the physical system and conduct several experiments comparing the baseline, quantitative evaluation, and qualitative evaluation in different simulation scenes: face-face collisions, edge-face collisions, vertex-face collisions, and ablation studies. The reader is referred to the supplemental video for the corresponding animations. The proposed model runs on a PC with a central processing unit Intel E5–2637, 128 GB RAM, and a GTX 1080 Ti graphics card.

Comparison

A sphere-mat scene is chosen to demonstrate the advantage of the proposed method in processing collisions compared with the baseline. The proposed method is compared with subphysics [1] and a baseline. The baseline comes from meshgraphnets [7] without remeshing because remeshing changes the topology of the data, which is not conducive to evaluating collisions. Figure 6 shows a comparison between the proposed method and the baseline. The top row is the ground truth, the second row is the subphysics simulation result, the third row is the baseline simulation result, and the bottom row is the proposed method simulation result. The results show that their method has a large interpenetration area, whereas the proposed method has none. Clearly, the proposed method detects collisions in the physical system.

Qualitative evaluation

To demonstrate the effectiveness of the proposed method in terms of quality, three different collision

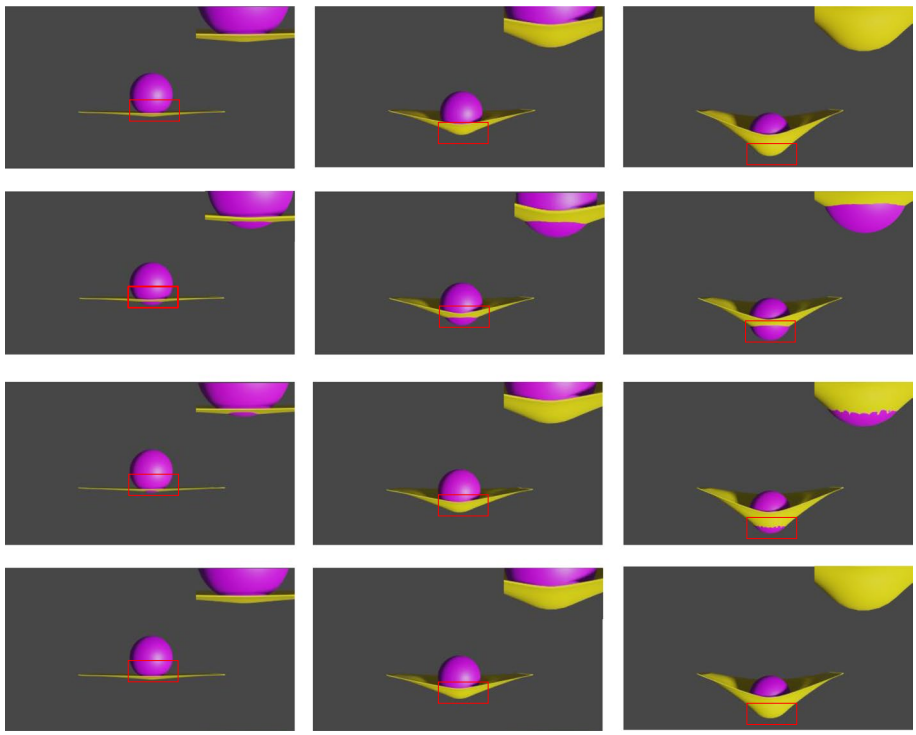

Fig. 6 Comparison. The top row is the ground truth, the second row is the subphysics simulation result, the third row is the baseline simulation result, and the bottom row is the proposed method's simulation result

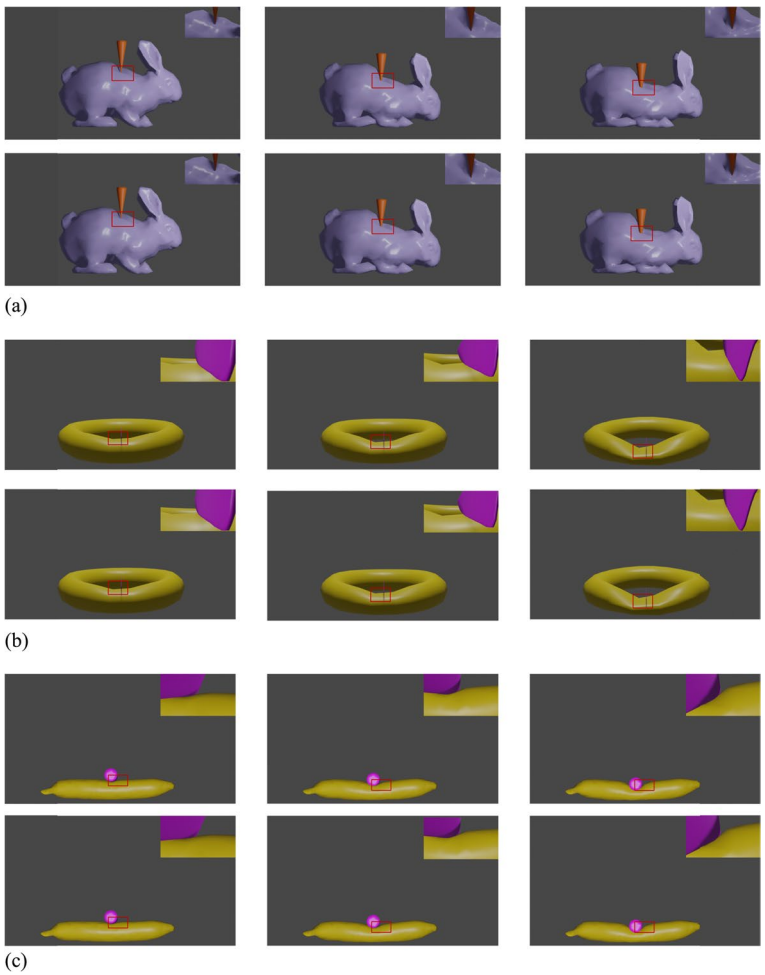

Fig. 7 Qualitative evaluation. The proposed method is evaluated in three different collision scenes: vertex-face, edge-face, and face-face collisions. For each scene, the first row is the simulation result of the proposed method, and the second row is the ground truth. **a**: Vertex-face collisions; **b**: Edge-face collisions; **c**: Face-face collisions

scenarios were defined: vertex-face collision, edge-face collision, and face-face collision. Figure 7(a) shows the vertex-face collision, Fig. 7(b) shows the edge-face collision, and Fig. 7(c) shows the face-face collision. In all three figures, the first row shows the simulation result of the proposed method, and the second row shows the ground truth. None of the three scenes had any interpenetration of the proposed method. Table 2 shows the quantitative evaluation of the collision elimination. The table shows that the proposed method can effectively eliminate collision errors. According to the results in the figures and table, the proposed method can process vertex-face, edge-face, and face-face collisions.

Table 2 Quantitative evaluation of collision elimination

Scene	N_{vf} (%)	$d_{vf,\,mean}$ (%)	$d_{vf,\,max}$ (%)	N_{ee} (%)	$d_{ee,\,mean}$ (%)	$d_{ee,\,max}$ (%)
Vertex-face collision	69.61	72.23	69.63	–	–	–
Edge-edge collision	62.50	53.67	49.84	–	–	–
Face-face collision	58.15	58.16	59.51	34.46	42.22	45.22

N_{vf}, $d_{vf,\,mean}$, $d_{vf,\,max}$, N_{ee}, $d_{ee,\,mean}$ and $d_{ee,\,max}$ represent mean vertex-vertex collision number, mean vertex-vertex collision interpenetration distance, max vertex-vertex collision interpenetration distance, mean edge-edge collision number, mean edge-edge collision interpenetration distance and max edge-edge collision interpenetration distance, respectively

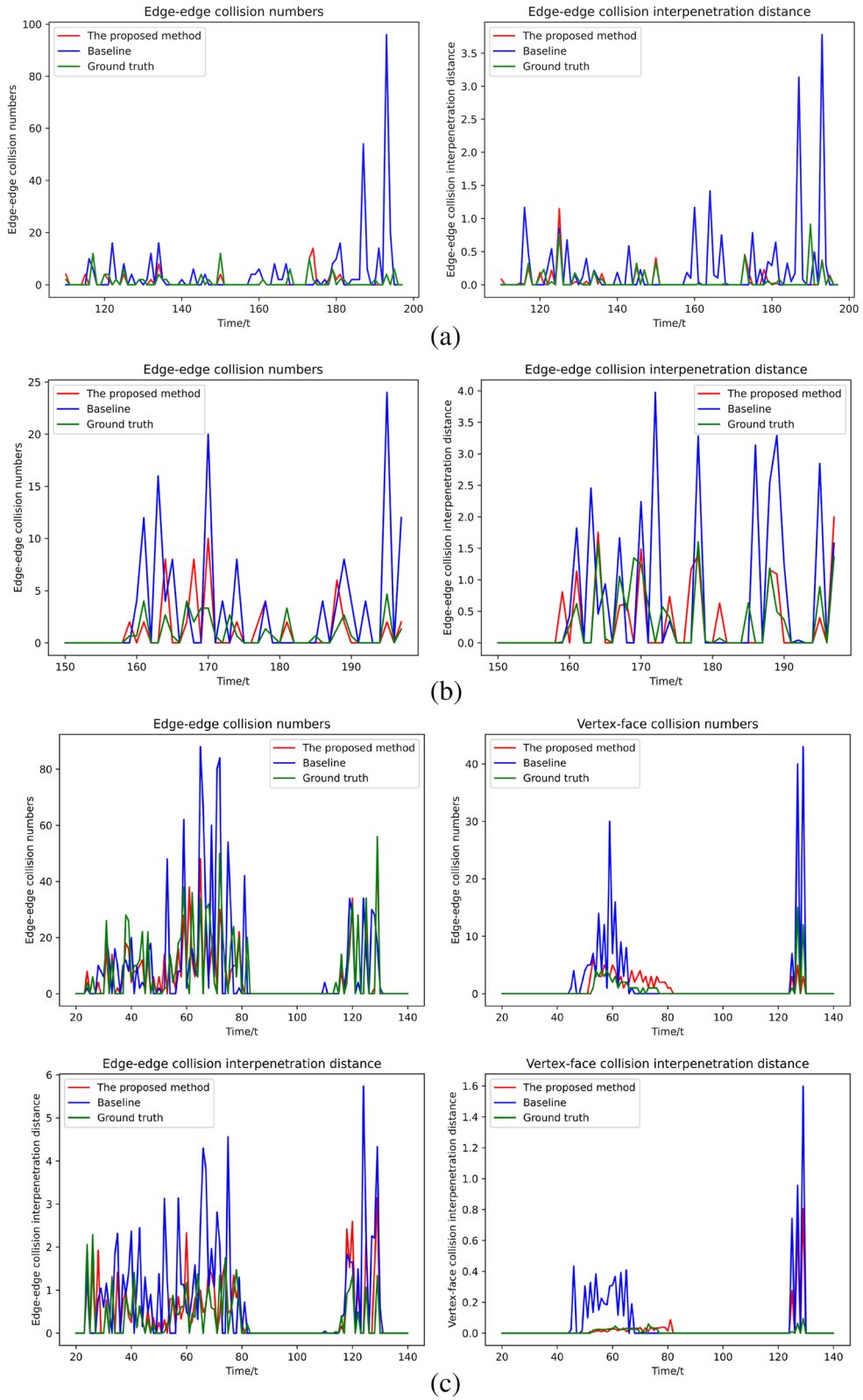

Fig. 8 Quantitative evaluation. The proposed method is evaluated using the numbers and interpenetration distance of vertex-face and edge-edge collisions. **a**: Vertex-face collision evaluation; **b**: Edge-face collision evaluation; **c**: Face-face collision evaluation

Quantitative evaluation

To demonstrate the effectiveness of the proposed method, three different scenes of collisions were defined: vertex-face, edge-face, and face-face collisions. Four collision quantitative evaluations were used: vertex-face collision numbers, vertex-face collision interpenetration distance, edge-edge collision numbers, and edge-edge collision interpenetration distance to judge the effectiveness of the proposed method's processing collision. Figure 8 shows four collision quantitative evaluation results for three collision scenes.

Because ref. [1] lacks a CCD module, it is excluded from the comparison. Clearly, for the four collision quantitative evaluations, the proposed method is fairly less accurate than the baseline [7] and is close to the ground truth. The results demonstrate that the proposed method can effectively reduce collision errors in physical systems.

Ablation study

The self-supervised term was removed from the proposed method to demonstrate the effectiveness of the random latent space in completing CDR. Figure 9 shows the comparison results. The first row shows the proposed method's simulation results, while the second row shows the ablation simulation results. The figure shows that there are interpenetrations if the self-supervised term is removed. The results show that using the self-supervised term to complete the collision response is crucial for the proposed method.

Performance

The proposed method is compared with the ground truth physical simulator IPC to evaluate its performance. Table 1 presents the results of this comparison. As the table shows, the proposed method leverages IPC by at least one order of magnitude.

Discussion

In this section, the advantages and challenges of the proposed method are discussed. The proposed method is compared with other state-of-the-art methods. Although the simulation results of the other methods have interpenetrations, the proposed method has none. Clearly, the proposed method detects collisions in the physical system better than the other methods. The proposed method was qualitatively and quantitatively evaluated in three collision scenarios: vertex-face, edge-face, and face-face collisions. The proposed method can visually produce no interpenetration results and effectively reduce the number of vertex-face collision and edge-edge collisions, resulting in visually excellent and physically accurate results. An ablation study was conducted to demonstrate the effectiveness of the random latent space for complete CDR. Some interpenetrations occur without a random latent space, demonstrating that our self-supervised term effectively reduces interpenetrations. Furthermore, compared to traditional CCD methods (IPC), the proposed method leverages at least one order of magnitude. In conclusion, to the best of our knowledge, the proposed deep learning-based framework can effectively address tool-object collisions and is a state-of-the-art method.

However, this study only focused on the interaction between a rigid tool and soft body. The penetration number of the vertex face and edge increases sharply as the model's complexity increases. The existing framework does not support large-scale interactive simulation computations owing to the limitations of the existing storage and computational power of the workstation. Future studies should introduce multiscale representations to achieve large-scale interactive simulations.

Conclusions

In this study, a deep interactive physical simulation framework that can effectively address tool-object collisions is presented. This was achieved using a GNN-based architecture and collision-aware recursive regression module to detect collisions. Additionally, a novel self-supervised collision term is introduced to provide a more compact collision response. The proposed method was

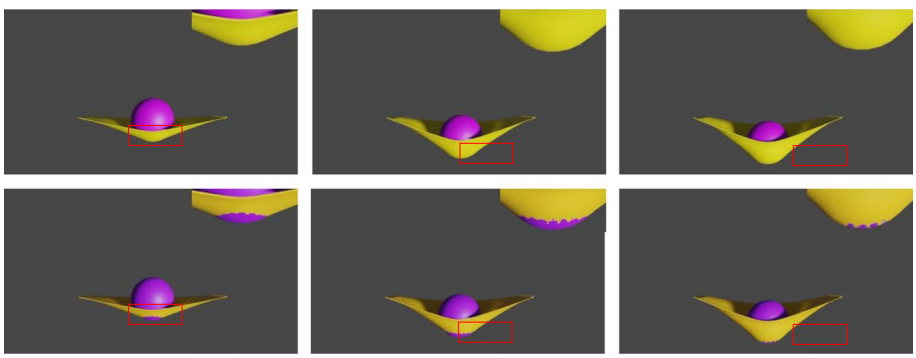

Fig. 9 Ablation study. The first row is the proposed method's simulation result, and the second row is the ablation simulation results

extensively evaluated and the results demonstrated that it can effectively reduce interpenetration artifacts while ensuring high simulation efficiency. However, the trained model could only be applied to simulations using the same tool object. Further research must be conducted to enhance the generalizability of this study's results. The existing framework does not support large-scale interactive simulation computations owing to the limitations of the existing storage and computational power of the workstation. Furthermore, future work must introduce multiscale representations to achieve large-scale interactive simulations.

Abbreviations

GNN: Graph neural network; CDR: Collision detection and response; CCD: Continuous collision detection; BV: Bounding volume; MLP: Multilayer perceptron; SDF: Signed distance field; IPC: Incremental potential contact.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s42492-022-00113-4.

Additional file 1.

Acknowledgements

The authors are grateful to the College of Computer Science, Sichuan University, and the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences.

Authors' contributions

XZ and YQ conceptualized the study; XZ implemented the machine learning model, conducted all the experiments, and produced the paper; YQ and PAH revised the manuscript; ZF and QW supplied significant information regarding the current research to the introduction and conclusion; The final manuscript has been read and approved by all authors; XZ and YQ contributed equally to this work.

Funding

This project was funded by Natural Science Foundation of Guangdong Province, No. 2020B010165004.

Availability of data and materials

The IPC library, available at https://github.com/ipc-sim/IPC, was used to generate simulation datasets.

Declarations

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this study.

Author details

[1]College of Computer Science, Sichuan University, Chengdu 610065, China. [2]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. [3]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.

Received: 14 February 2022 Accepted: 25 May 2022
Published online: 07 June 2022

References

1. Holden D, Duong BC, Datta S, Nowrouzezahrai D (2019) Subspace neural physics: Fast data-driven interactive simulation. Paper presented at the 18th annual ACM SIGGRAPH/Eurographics symposium on computer animation, ACM, Los Angeles, 26-28 July 2019. https://doi.org/10.1145/3309486.3340245
2. Santesteban I, Thuerey N, Otaduy MA, Casas D (2021) Self-supervised collision handling via generative 3D garment models for virtual try-on. Paper presented at the 2021 IEEE/CVF conference on computer vision and pattern recognition, IEEE, Nashville, 20-25 June 2021. https://doi.org/10.1109/CVPR46437.2021.01159
3. Luo R, Shao TJ, Wang HM, Xu WW, Chen X, Zhou K et al (2020) NNWarp: Neural network-based nonlinear deformation. IEEE Trans Vis Comput Graph 26(4):1745-1759.
4. Romero C, Casas D, Pérez J, Otaduy M (2021) Learning contact corrections for handle-based subspace dynamics. ACM Trans Graph 40(4): 131. https://doi.org/10.1145/3476576.3476703
5. Teng Y, Meyer M, DeRose T, Kim T (2015) Subspace condensation: Full space adaptivity for subspace deformations. ACM Trans Graph 34(4): 76. https://doi.org/10.1145/2766904
6. Tan QY, Pan ZR, Manocha D (2021) L Collision: Fast generation of collision-free human poses using learned non-penetration constraints. arXiv preprint arXiv:2011.03632.
7. Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW (2021) Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409.
8. Coming DS, Staadt OG (2008) Velocity-aligned discrete oriented polytopes for dynamic collision detection. IEEE Trans Vis Comput Graph 14(1):1-12. https://doi.org/10.1109/TVCG.2007.70405
9. Redon S, Kheddar A, Coquillart S (2002) Fast continuous collision detection between rigid bodies. Comput Graph Forum 21(3):279-287. https://doi.org/10.1111/1467-8659.t01-1-00587
10. Redon S, Lin MC (2006) A fast method for local penetration depth computation. J Graph Tools 11(2):37-50. https://doi.org/10.1080/2151237X.2006.10129216
11. Tang M, Manocha D, Otaduy MA, Tong RF (2012) Continuous penalty forces. ACM Trans Graph 31(4): 107. https://doi.org/10.1145/2185520.2185603
12. Choi YK, Wang WP, Mourrain B, Tu CH, Jia XH, Sun F (2014) Continuous collision detection for composite quadric models. Graph Models 76(5):566-579. https://doi.org/10.1016/j.gmod.2014.03.005
13. Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M et al (2018) Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261.
14. Kipf T, Fetaya E, Wang KC, Welling M, Zemel R (2018) Neural relational inference for interacting systems. Paper presented at the 35th international conference on machine learning, JMLR.org. Stockholm; 2018.
15. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. Paper presented at the 34th international conference on machine learning, JMLR.org. Sydney; 2017.
16. Seo S, Liu Y (2019) Differentiable physics-informed graph networks. arXiv preprint arXiv:1902.02950.
17. Chang MB, Ullman T, Torralba A, Tenenbaum JB (2017) A compositional object-based approach to learning physical dynamics. arXiv preprint arXiv:1612.00341.
18. Sanchez-Gonzalez A, Heess N, Springenberg JT, Merel J, Riedmiller M, Hadsell R. et al (2018) Graph networks as learnable physics engines for inference and control. Paper presented at the 35th international conference on machine learning, JMLR.org. Stockholm; 2018.
19. Li YZ, Wu JJ, Zhu JY, Tenenbaum JB, Torralba A, Tedrake R (2019) Propagation networks for model-based control under partial observation. Paper presented at the 2019 international conference on robotics and automation, IEEE, Montrl, 20-24 May 2019. https://doi.org/10.1109/ICRA.2019.8793509
20. Mrowca D, Zhuang CX, Wang E, Haber N, Fei-Fei L, Tenenbaum JB et al (2018) Flexible neural representation for physics prediction. arXiv preprint arXiv:1806.08047.
21. Sanchez-Gonzalez A, Bapst V, Cranmer K, Battaglia P (2019) Hamiltonian graph networks with ODE integrators. arXiv preprint arXiv:1909.12790.

22. Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia P (2020) Learning to simulate complex physics with graph networks. Paper presented at the 37th international conference on machine learning, PMLR, Vienna.
23. Tang M, Manocha D, Tong RF (2010) Fast continuous collision detection using deforming non-penetration filters. Paper presented at the 2010 ACM SIGGRAPH symposium on interactive 3D graphics and games, ACM, Washington. https://doi.org/10.1145/1730804.1730806
24. Kingma DP, Welling M (2014) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
25. Li MC, Ferguson Z, Schneider T, Langlois T, Zorin D, Panozzo D et al (2020) Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. ACM Trans Graph 39(4): 49. https://doi.org/10.1145/3386569.3392425
26. Kingma DB, Ba J (2017) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Publisher's Note