Hindawi
Scientific Programming
Volume 2024, Article ID 7889278, 15 pages
https://doi.org/10.1155/2024/7889278

Research Article

Deep Neural Network-Based Cloth Collision Detection Algorithm

Yanxia Jin, Zhiru Shi, Jing Yang, Yabian Liu, Xingyu Qiao, and Ling Zhang

Data Science and Technology, North University of China, Taiyuan 030051, China

Correspondence should be addressed to Yanxia Jin; jyx@nuc.edu.cn

Received 9 August 2022; Revised 27 October 2023; Accepted 18 December 2023; Published 17 January 2024

Academic Editor: Roberto Natella

The quality of collision detection algorithm directly affects the performance of the whole simulation system. To address the low efficiency and low accuracy in detecting the collisions of flexible cloths in virtual environments, this paper proposes an oriented bounding box (OBB) algorithm with a simplified model, tree structure for a root-node double bounding box, and continuous collision detection algorithm incorporating an OpenNN-based neural network optimization. First, for objects interacting with the cloths with more complex modeling, the model is simplified with a surface simplification algorithm based on the quadric error metrics, and the simplified model is used to construct an OBB. Second, a bounding box technique commonly used for collision detection is improved, and a root-node double bounding box algorithm is proposed to reduce the construction time for the bounding box. Finally, neural networks are used to optimize the continuous collision detection algorithm, as neural networks can efficiently process large amounts of data and remove disjoint collision pairs. An experiment shows that the construction of an OBB using the simplified model is almost identical to that of the original model, but the taken to construct the OBB is reduced by a factor of approximately 2.7. For the same cloth, it takes 5.51%–11.32% less time to run the root-node double bounding box algorithm than the traditional-hybrid bounding box algorithm. With an average removal rate nearly identical to that of the traditional filtering method, the elapsed time is reduced by 7%–11% by using the continuous collision detection algorithm based on an OpenNN neural network optimization. The simulation results are realistic and in line with the requirements for real-time cloth simulations.

1. Introduction

Cloth simulation is an active area of research in the computer graphics, it is widely used in games, virtual fitting, animation, and other fields, and collision detection has a significant impact on the efficiency of cloth simulations. Cloths are typically flexible objects, and high-level performance is required to detect cloth collisions in real time. Thus, effectively improving the performance of cloth collision detection has become a key focus in the field of cloth simulation research. Hu and Qin [1] proposed a minimum-volume oriented bounding box (OBB) generation algorithm. The algorithm was based on a convex hull operation for quickly generating a well-fitting minimum OBB by enumerating all possible combinations of edges of the convex hull, allowing for selection of the optimal direction of the bounding box. Jin et al. [2] proposed a thickness-free circular bounding box for accommodating cloth self-collision detections, showing clear advantages in the scenarios where were a sufficient number of self-collisions. Wang et al. [3] proposed a new hierarchical OBB construction method for optimally solving solid mesh models to increase the speed of collision detection. Although, the use of an optimized and improved single bounding box algorithm is efficient for collision detection, each single bounding box has its own disadvantages; these be solved using a hybrid hierarchical bounding box algorithm. Using relatively simple bounding spheres, AABBs with relatively complex OBBs and k-DOPs can be used to build hybrid bounding boxes playing to the advantages of both the high-detection efficiency of the simple bounding box and high-removal rate of the complex bounding box [4–6]. However, there are issues concerning low accuracy and efficiency with the traversal process of hybrid hierarchical bounding. To address these issues, Li and Wang [7] improved the tree structure of the hierarchical bounding box.

Continuous collision detection algorithms are often used for the cloth simulations. These include algorithms based on motion interpolation [8], continuous collision detection algorithms based on algebraic methods [9], and distance-based collision detection algorithms [10]. Jin et al. [11] proposed a new algorithm for real-time and accurate collision detection for the deformable objects. This algorithm was based on the random collision detection, and used a differential evolution algorithm to improve the global search capability and speed of convergence of the particle swarm algorithm. Qu et al. [12] introduced an optimization operator for reinitialising a iterative ant colony algorithm to avoid premature convergence to the local optima. Du et al. [13] proposed a parallel continuous collision detection algorithm for multimodel scenarios to address the problem of performance gains being limited to the individual processors. Tayyub and Khan [14] proposed a successive approximation method for evaluating an optimal load ratio using heterogeneous processing based on a GPU–CPU platform, aiming to improve the efficiency of collision detection. Lu et al. [15] adopted an algebraic nonpenetration filtering method to optimize continuous collision detection by simplifying the solution process and improving the efficiency of the cloth simulation. Wan et al. [16] proposed a spatial linear projection filtering method for optimizing continuous collision detection, and used a combination of nonpenetration filters and spatial linear projection filters for collision removal, thereby improving the collision rejection rate.

Machine learning has the advantages of a high-classification accuracy, a high-learning capacity, and the ability to process large amounts of data quickly. Some studies have combined machine learning and cloth simulations. Holden et al. [17] combined subspace methods for simulation with machine learning, which when coupled, could enable very effective physical simulations of the supported subspaces. The use of machine learning combined with cloth simulations allows cloths to have greater details [18, 19] and enriches the pleated meshes of low-resolution cloths [20]. Jin et al. [2] used deep neural networks (DNN) to accelerate the detection speed for a bounding box and improve the efficiency of self-collision detection. Oh et al. [21] proposed a cloth simulation method by combining DNNs with traditional physical methods. Shi et al. [22] used machine learning to study the relationship between human movement and costume deformation in costume animation, so as to make the cloth more accurate. Jiang et al. [23] proposed a method to simulate cloth drape by using different cloth. By constructing BP neural network, the nonlinear relationship between cloth mechanics parameters and control parameters of 3D textile simulation system is obtained. Virtual clothing based on this method is more specific. Zhang [24] studied the relationship between the changes in the details of clothing folds in human body models and the human motion form, and used machine learning algorithm modeling to find the connection between the two, so as to quickly and accurately show the changes in the clothing folds. Runia et al. [25] established the deep network and took the real situation as a reference to compare whether the current simulation results were different. According to the comparison results, the parameters of the network model are adjusted. Correspondence is measured using the embedding feature, which maps physically similar examples to neighboring points. Gundogdu et al. [26] solved the problem of static cloth hanging on the human body, and learned the method based on physical simulation by using the network. Meanwhile, the required calculation time was reduced by two orders of magnitude, and the loss function was added to increase the cloth details. Ju and Choi [27] proposed a neural network learning method to estimate a set of clothing simulation parameters from the static drape of a given cloth. Montazeri et al. [28] collected the horizontal data generated by cloth and yarns in the simulation and used the regression neural network training model to represent the cloth shape in a very real way.

In this paper, the collision detection in cloth simulation is divided into rough collision detection and precise collision detection. Two improvements are proposed for the rough detection stage, as follows: (1) As an OBB has good compactness, a simplified model of the OBB algorithm is proposed to address the problem of the time-consuming construction of OBBs for complex modeled objects. The model is simplified by as much as possible while maintaining the characteristics of the original model, thereby greatly reducing the computational effort required for the OBB. (2) To address the problem of the time-consuming construction of the bounding box, a root-node double bounding box tree structure is proposed. For the precise detection stage, based on a continuous collision detection algorithm optimized by algebraic nonpenetration filtering and spatial linear projection filtering and neural network algorithms, this study proposes an OpenNN-based neural network training method for learning algebraic nonpenetration filtering and spatial linear projection filtering (for removing the collision pairs). Accordingly, it is possible to optimize the continuous collision detection algorithm.

2. Related Studies

2.1. Surface Simplification Algorithm Based on Quadric Error Metrics. In this study, we use a surface simplification algorithm based on the quadric error metrics [29] to quickly produce high-quality polygonal model approximations. The algorithm uses iterative shrinkage of a vertex pair to simplify the model, and uses a quadratic matrix to maintain the surface error approximations. The flow of the algorithm is shown in Figure 1.

The simplified algorithm for quadric error metrics is based on iterative shrinkage of the vertex pair, where the shrinkage pair $(v_1, v_2) \longrightarrow v$ denotes that the fixed points v_1 and v_2 are moved to a new position v. Any degraded edges or multiple faces are then removed, as shown in Figure 2.

Two individual parts of the model are joined at v, as shown in Figure 3.

We define a shrinkage cost to select the shrinkage pair suitable for performing the shrinkage. For this, we need to describe the error on each vertex. Here, we create a symmetric 4×4 matrix Q for each vertex, and define the error at the vertex as $\Delta(v) = v^{\mathrm{T}}\mathrm{Q}v$. Inspired by the heuristic algorithm by Ronfard and Rossignac [30], we can associate a set of planes with each vertex to form a set, and define the error of a vertex

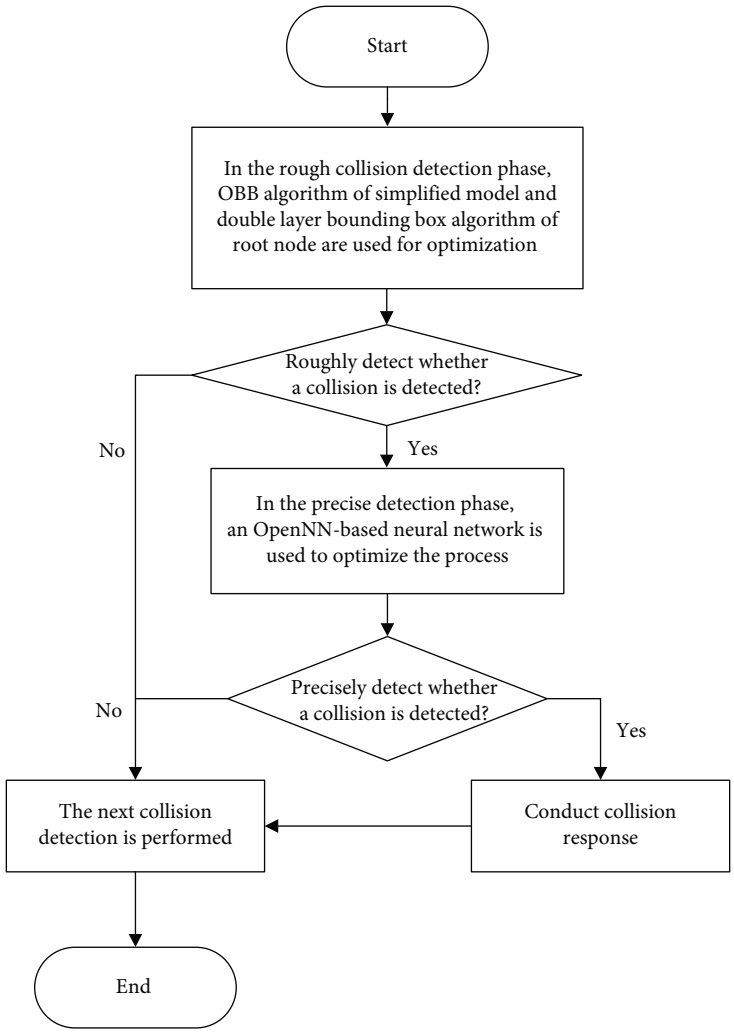

Figure 1: Flowchart of this paper.

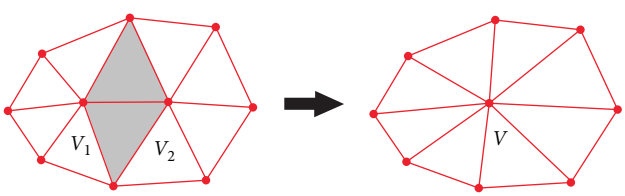

Figure 2: Edge contraction. The edge is shrunk to a point, shadow triangles degenerate and are removed during shrinkage.

with respect to this set as the sum of the squares of the distances from its plane, as follows:

$$\Delta(v) = \Delta\left(\left[v_x v_y v_z 1\right]^{\mathrm{T}}\right) = \sum_{p \in v} (p^{\mathrm{T}} v)^2, \tag{1}$$

$p = (\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d})^{\mathrm{T}}$ in Equation (1) is the plane defined in Equation $\mathrm{ax} + \mathrm{by} + \mathrm{cz} + d = 0$, where $\mathrm{a}^2 + \mathrm{b}^2 + \mathrm{c}^2 = 1$. Equation (1) can be rewritten as follows:

$$\begin{aligned} \Delta(v) &= \sum_{p \in v} (v^{\mathrm{T}} p)(p^{\mathrm{T}} v) \\ &= \sum_{p \in v} v^{\mathrm{T}} (p p^{\mathrm{T}}) v \\ &= v^{\mathrm{T}} \sum_{p \in v} K_p v, \end{aligned} \tag{2}$$

$$K_p = p p^{\mathrm{T}} = \begin{matrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{matrix}. \tag{3}$$

For a shrinkage pair $(v_1, v_2) \longrightarrow v$, we must provide the position of v after the shrinkage such that $\Delta(v)$ is minimal. As the error function is a quadratic function, finding the minimum value of the error function is a linear problem that can be solved to obtain v, as follows:

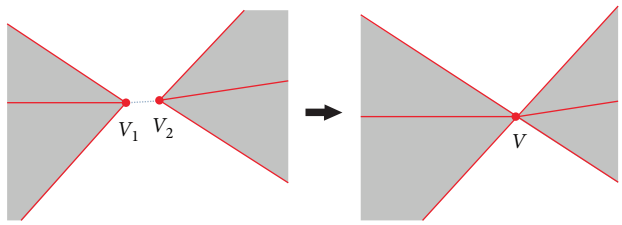

Figure 3: Nonedge shrinkage. When nonedge pairs are contracted, then unconnected parts of the model will connected.

$$\begin{matrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{matrix} \overline{v} = \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \end{matrix}. \tag{4}$$

If the matrix to the left of v is invertible, we get a calculation as follows:

$$\overline{v} = \begin{matrix} q_{11} & q_{12} & q_{13} & q_{14} & 0 \\ q_{12} & q_{22} & q_{23} & q_{24} & 0 \\ q_{13} & q_{23} & q_{33} & q_{34} & 0 \\ 0 & 0 & 0 & 1 & 1 \end{matrix}. \tag{5}$$

If the matrix to the left of v is not invertible, we find the optimal vertex along the line segment. If the optimal vertex is still not found, v is chosen from the endpoints or midpoint.

The surface simplification algorithm based on quadric error metrics can produce a high-quality simplified model with the characteristics of the original model in a fairly short period of time. A simplified model using this method is shown in Figure 4, and it can be seen that the characteristics of the 90%-simplified model are almost identical to those of the original model.

2.2. Continuous Collision Detection Algorithm Optimized by Algebraic Nonpenetration Filtering. According to the literature [15], we can quickly determine the presence of roots using the Descartes' rule of signs and Vincent's theorem on $D(\mathrm{t})$, as shown in Equation (7). If a root exists, the collision time t needs to be calculated exactly; otherwise, it indicates that no collision has occurred in the time interval.

The condition for a collision to occur is whether the four points are coplanar, which essentially requires solving a cubic equation. The collision time t is solved for using a four-point coplanarity feature. According to Brochu et al. [31], if edge–edge and point–plane collisions occur during a time interval, the four points of the collision lie on the same plane, and a calculation can be deduced as follows:

$$D(\mathrm{t}) = (x_{10} + tv_{10}) \times (x_{20} + tv_{20}) \cdot (x_{30} + tv_{30}) = 0. \tag{6}$$

Here, x_{10} is the positional difference between vertices x_0 and x_1 at time point t; v_{10} is the difference in velocity between vertices v_0 and v_1; the scalar triple product, i.e., $D(\mathrm{t})$, is 0, indicating that a collision must have occurred.

In response to the problem where multiple solutions may exist for $D(\mathrm{t}) = 0$ in the time interval, approximation testing is performed to avoid collisions occurring to the neglected errors. The above equation can be reduced to a standard cubic equation as follows:

$$D(\mathrm{t}) = a + bt + ct^2 + dt^3. \tag{7}$$

2.3. Continuous Collision Detection Algorithm Optimized by Spatial Linear Projection Filtering. To improve the removal rate in the precise collision detection stage and reduce the requirement for precise computation and correspondingly increased performance requirements, Wan et al. [16] proposed spatial linear filtering to remove collision pairs failing to satisfy the internal conditions. Their approach was based on the theorem that if two objects do not intersect in a given projective space during a certain time interval, the two objects will not intersect in the original space either. According to this method, after calculation and deduction, in the case of a point–plane collision pair, if Z_1, Z_2, Z_3, Z_4, Z_5, and Z_6 are of the same sign in Equation (8), it can be extrapolated that no collision has occurred during this time interval.

$$\begin{aligned} Z_1 &= (V_0 \cdot F_0)\mathrm{s} \\ Z_2 &= (V_1 \cdot F_1)s \\ Z_3 &= (V_0 \cdot G_0)s \\ Z_4 &= (V_1 \cdot G_1)s \\ Z_5 &= (V_0 \cdot H_0)s \\ Z_6 &= (V_1 \cdot H_1)s \end{aligned}. \tag{8}$$

In the above, V represents the point in the collision pair, F, G, and H are faces. During the time interval, V_0, F_0, G_0, and H_0 are the initial positions of the corresponding points, V_1, F_1, G_1, and H_1 are the end positions of the corresponding points, and s is the projection vector.

If Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, and Z_8 are of the same sign in Equation (9), in the case of edge–edge collision pairs, it can be inferred that no collision has occurred during this time interval.

$$\begin{aligned} Z_1 &= (A_0 \cdot C_0)\mathrm{s} \\ Z_2 &= (A_1 \cdot C_1)\mathrm{s} \\ Z_3 &= (A_0 \cdot D_0)\mathrm{s} \\ Z_4 &= (A_1 \cdot D_1)\mathrm{s} \\ Z_5 &= (B_0 \cdot C_0)\mathrm{s} \\ Z_6 &= (B_1 \cdot C_1)\mathrm{s} \\ Z_7 &= (B_0 \cdot D_0)\mathrm{s} \\ Z_8 &= (B_1 \cdot D_1)\mathrm{s} \end{aligned}. \tag{9}$$

Here, AB is one side and CD is the other side. During the time interval, A_0, B_0, C_0, and D_0 are the initial positions of the corresponding points, and A_1, B_1, C_1, and D_1 are the end

positions of the corresponding points. As it is considered that using too many projection vectors s will affect the computational efficiency, five projection vectors are finally selected for removal after the experiment, as follows:

$$s = \{(1,0,0), (\sqrt{2}/2, \sqrt{2}/2, 0), (\sqrt{2}/2, 0, \sqrt{2}/2), (0, \sqrt{2}/2, \sqrt{2}/2), (\sqrt{3}/3, \sqrt{3}/3, \sqrt{3}/3)\}. \tag{10}$$

3. Methods

3.1. Oriented Bounding Box (OBB) Simplified Modeling. In graphic rendering projects for large scenes, simplified models are often used for objects far away from the scene. As the objects are far away and do not need to be rendered overly accurately, they can be rendered using simplified models to improve the rendering efficiency of the entire scene. In this study, the simplified model approach is used for the construction update of the OBB. Thereby reducing the computational effort and improving the efficiency of the OBB.

3.1.1. OBB Construction with Simplified Modeling. The highly real-time nature of the OBB requires real-time updates, and the increasing refinement of today's models makes constructing this bounding box more time-consuming than other options. Accordingly, a simplified model is used for the OBB construction, and can effectively reduce the operational overhead of the program. In particular, in this study, we embed the simplified model into the original model to build the OBB, so as to improve the efficiency of the OBB algorithm.

The traditional OBB algorithm takes the average of the sum of all triangle vertices as its center position; each triangle block is unevenly sized, and will have a different orientation. This study uses a fast adaptive OBB algorithm [32] to reduce the influence of the triangular surfaces in the model. Correspondingly, the method has less overhead than the traditional OBB algorithm. The method is as follows.

Calculate the centroid O of the model, and denote the vertices of the t-th triangle of the model by a_t, b_t and c_t, respectively. O_t is its centroid, S_t is the surface area, and Z is the total area of the model. Calculate the centroid of the triangle as follows:

$$o_t = (a_t + b_t + c_t)/3. \tag{11}$$

The surface area of the triangle and the total area of the model are, respectively, expressed as follows:

$$S_t = |(a_t \cdot b_t) \times (a_t \cdot c_t)|/2, \tag{12}$$

$$Z = \sum_{t=1}^{n} S_t. \tag{13}$$

The center point of the bounding box is denoted by:

$$O = \sum_{t=1}^{n} (S_t \cdot o_t)/2. \tag{14}$$

Calculate the projection of the model on the x-axis. The maximum and minimum values of the projection area are $Q_{\max}$ and $Q_{\min}$, respectively. Subsequently, divide the model vertices according to the positions of the projection on the axes. Divide the projection area into blocks, k is the length, as follows:

$$k = \frac{Q_{\max} Q_{\min}}{s}. \tag{15}$$

Divide the set of model vertices S into s subsets according to the projected position of each vertex on the x-axis, and calculate its subset code as follows:

$$S_s = \{(x, y, z) | Q_{x\min} + s \times k \le x \quad Q_{x\min} < Q_{x\min} + (s+1) \times k\}, \tag{16}$$

where S_s is the s-th subset.

Extract the set of vertices away from the centroid of the model and construct the set of vertices S' of the bounding box. Similarly, find the coordinates corresponding to the Y-axis and Z-axis, and construct the maximum point S'. Construct the OBB on the extracted S'. Calculate the expected value of the vertex set of S' and its covariance matrix as follows:

$$u = \frac{1}{6n} \sum_{i=1}^{n} (p_i + q_i + r_i), \tag{17}$$

$$C_{jk} = \frac{1}{3m} \sum_{i=1}^{n} \overline{p_i^j}\,\overline{p_i^k} + \overline{q_i^j}\,\overline{q_i^k} + \overline{r_i^j}\,\overline{r_i^k} \quad \overline{p}_i = p_i - u \quad \overline{q}_i = q_i - u \quad \overline{r}_i = r_i - u. \tag{18}$$

In the above, n is the number of triangular faces. Calculate the eigenvectors based on the covariance matrix, and regard them as the three directions of the OBB. Finally, calculate the projection length of the vertex set S' in the three directions of the OBB to complete the bounding box calculation.

3.2. Root-Node Double Bounding Box Algorithm. During collision detection, the time needed for the construction and updating of the different bounding boxes and for intersection detection between the bounding boxes varies. To reduce the time consumed by the construction and updating of the bounding box while ensuring the removal rate, this study proposes a root-node double bounding box algorithm.

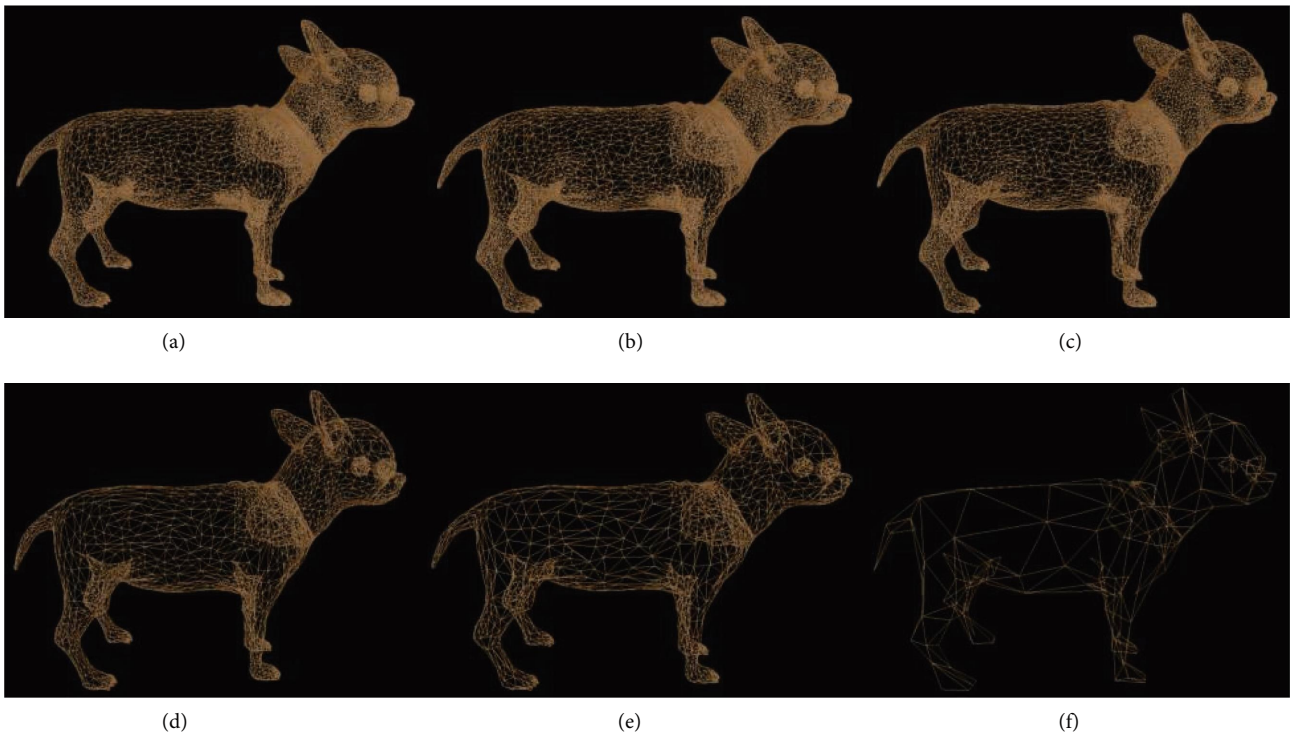

Figure 4: Simplify the rabbit model to varying degrees. (a) Original model, (b) 20%-simplified model, (c) 40%-simplified model, (d) 80%-simplified model, (e) 90%-simplified model, and (f) 99%-simplified model.

3.2.1. Construction and Collision Detection of the Root-Node Double Bounding Box. The bounding volume hierarchy (BVH) structure constructed by the AABB and bounding sphere is shown in Figure 5. In the BVH, bounding spheres are used for nonroot nodes, and double bounding boxes are used for root nodes. During the cloth simulation, a bounding sphere is first constructed for the root node, and when the bounding sphere collides, an AABB is constructed, at which point the root node has a double bounding box. When the collision occurs in the second AABB of the root node, the BVH structure is traversed down to the leaf node.

The exact process of collision detection is as follows.

(1) Starting from the root node, construct the BVH tree using a bounding sphere, and traverse the tree.
(2) When a collision is detected at the root node, construct an AABB for the root node; otherwise, perform Step 5.
(3) When a collision occurs in the AABB of the root node, a depth-first search principle is used to traverse the BVH tree up to the leaf node. If the collision does not occur, perform Step 5.
(4) When a collision is detected at a leaf node, collision detection is performed for the underlying fundamental primitives, followed by a collision response.
(5) Detect the collision occurrence again from the root node in the next time step.

3.2.2. Advantages of the Root-Node Double Bounding Box Algorithm. In this study, we use the level of compactness of the bounding box τ to better illustrate the advantages of the algorithm. τ As represented by Equation (19), τ_S, τ_A denotes the compactness of the bounding sphere and AABB, respectively. $V(O)$ denotes the volume of the cloth simulation, and $V(B)$ denotes the volume of the bounding box.

$$\tau = \frac{V(B)}{V(O)}, \tau_S = \frac{V_S(B)}{V(O)}, \tau_A = \frac{V_A(B)}{V(O)}, \tag{19}$$

$$\tau_S = \frac{\pi[(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2 + (z_{\max} - z_{\min})^2]^{\frac{3}{2}}}{6V(O)}, \tag{20}$$

$$\tau_A = \frac{(|x_{\max}| + |x_{\min}|)(|y_{\max}| + |y_{\min}|)(|z_{\max}| + |z_{\min}|)}{V(O)}. \tag{21}$$

From the above equation, it can be seen that the smaller the values of τ_S and τ_A, i.e., the larger $V(O)$ and smaller $V(B)$, the higher the level of compactness. The use of the root-node double bounding box algorithm combines the strengths of the bounding sphere and AABB. Thus, compared with the simple bounding box algorithm, the algorithm proposed herein has a higher level of compactness (as its low level of compactness is addressed by constructing the double bounding box).

3.3. Continuous Collision Detection Algorithm Based on OpenNN Optimization. The use of algebraic nonpenetration filtering and spatial linear projection filtering can effectively improve the removal rate of collision pairs. As a DNN is

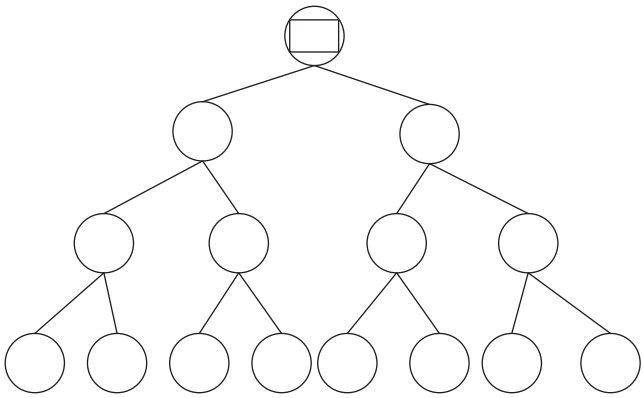

Figure 5: Double-layer bounding box BVH structure of root-node diagram.

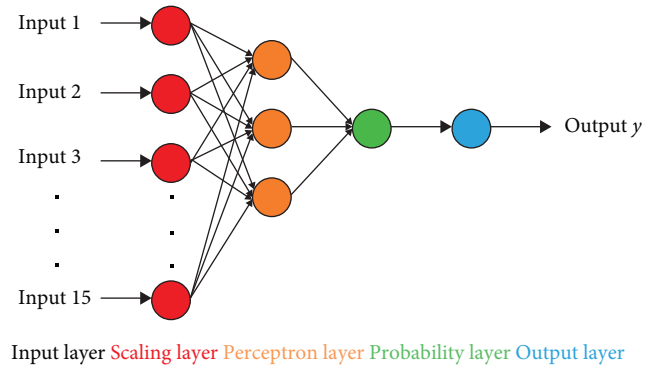

Figure 6: Initial structure diagram of neural network.

versatile owing to its powerful nonlinear fitting, strong feature extraction capabilities, and the ability to quickly process large amounts of data, this study proposes a removal method in which DNNs are used to learn algebraic nonpenetrating filtering and spatial linear projection filtering. This provides a way to quickly remove collision pairs within a time interval.

3.3.1. Neural Network Fusion-Based Collision Detection Algorithm. The neural network construction in this study was based on an open source neural network library, i.e., OpenNN. The neural network model was constructed through five main steps, as discussed more fully below.

(1) Datasets. The first step is dataset preparation, so as to provide the source of information for the problem solving. During the experiment, the collision pairs are removed using the filters mentioned in Sections 3.3.1 and 3.3.2. Those that are removed, i.e., where no collision occurred, are marked as 0, and the others are marked as 1. The resulting set of instances is divided into training, selection, and testing subsets accounting for 60%, 20%, and 20% of the original instances, respectively. To provide a suitable range for all inputs, the dataset is scaled using the min–max scaling as shown in Equation (22), where x denotes the data before scaling, X denotes the data after scaling, and *min* and *max* denote the minimum and maximum values in the dataset, respectively. The scaled dataset ranges from −1 to 1.

$$X = 2 \times \frac{x - \min}{\max - \min} - 1. \tag{22}$$

(2) Network Construction. The second step is to construct the correct neural network structure. This structure is mainly used for classification and to determine whether a collision has occurred. The neural network structure consists of the following parts: a scaling layer, two perceptron layers, and a probabilistic layer. The structure is shown in Figure 6, where there are 15 input variables in the input layer and 15 neurons in the corresponding scaling layer, three neurons in the first perceptron layer, one neuron in the second perceptron layer, and finally, one neuron in the probabilistic layer.

The "Neural Network" class constructs the neural network and uses the constructor to appropriately organize the neuron layers. Once the neural network is constructed, the input and output information can then be introduced into the layers.

The most important part of the perceptron layer is the perceptron neuron, as shown in Figure 7, where $x_1, x_2, \ldots, x_n$ represents the input information, $w_1, w_2, \ldots, w_n$ represents the weights, b represents the deviation, and c represents the value for combining the input values. act(x) represents the activation function, and y represents the final output. The output of the perceptron neuron can be represented as shown in Equation (24).

$$\mathrm{act}(x) = \frac{1}{1 + e^{-x}}, \tag{23}$$

$$y = \mathrm{act}\left(b + \sum_{i=1}^{n} w_i \cdot x_i\right). \tag{24}$$

The activation function used for the perceptron layer is the logical activation function as shown in Equation (23), and varies between 0 and 1.

The probabilistic layer provides a way of interpreting the output information as probabilities. In this study, the method used for the probabilistic layer is binary probability. Specifically, the binary probability method is used for binary classification, where the output y in Equation (25) can have the value 1 or 0, x is the input, and λ is the decision threshold.

$$y = \begin{cases} 0, x < \lambda \\ 1, x \geq \lambda \end{cases}. \tag{25}$$

(3) Loss Function. The loss function is the normalized squared error (NSE), and NSE is obtained by dividing the square of the difference between the output of the neural network out and target in the dataset by the normalization factor A. If the result is 1, the neural network will predict the data at the mean value; if the result is 0, it indicates a perfect prediction of the data. The NSE can be expressed as follows:

$$\mathrm{NSE} = \frac{\sum(\mathrm{out} - \mathrm{tar})^2}{A}. \tag{26}$$

(4) Optimization Algorithms. The optimization algorithm uses a quasi-Newton method to find the neural network parameters minimizing the loss function x^k as shown in Equation (27), where the learning rate η is adjusted by each neuron using a linear minimization method.

$$x^{(k+1)} = x^{(k)} - \boldsymbol{G_K} \cdot y_k \cdot \eta_k. \tag{27}$$

We derive G_K by expanding $f(x)$ at x^k using the second degree Taylor polynomial as follows:

$$f(x) = f\left(x^{(k)}\right) + g_k^{\mathrm{T}}\left(x - x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^{\mathrm{T}} \boldsymbol{H}\left(x^{(k)}\right)\left(x - x^{(k)}\right). \tag{28}$$

To derive x, we obtain the approximation function of $\nabla f(x)$in the domain $x = x_k$ as follows:

$$\nabla f(x) = g_k + \boldsymbol{H}_K\left(x - x^{(k)}\right). \tag{29}$$

Further, let $y_k = g_{k+1} - g_k, \delta_k = x^{(k+1)} - x^{(k)}$ then the quasi-Newton method is as follows:

$$g_{k+1} - g_k = \boldsymbol{H}_K\left(x^{(k+1)} - x^{(k)}\right), \tag{30}$$

$$\boldsymbol{H}_{\boldsymbol{K}}^{-1} y_k = \delta_k. \tag{31}$$

The iterative calculations of the quasi-Newton method using an n order matrix G_k to replace H_k^{-1} and G_k are as follows:

$$\boldsymbol{G_{k+1}} = \boldsymbol{G_k} + \Delta \boldsymbol{G_k}. \tag{32}$$

(5) Model Selection. In the fourth step, to find the network structure with the best generalization characteristics and minimize the error in the selected dataset instances, we need to select the number of neurons in the neural network model. During the experiment, an incremental sequence method is used to optimize the number of neurons in the network structure, starting with a small number of neurons. The incremental sequence method increases in complexity with each iteration, and Figure 8 demonstrates the neural network structure that best fits the proposed algorithm.

3.3.2. Advantages of the Neural Network Fusion-Based Algorithm. In addition to processing a large amount of particles at once, the proposed algorithm also improves efficiency by omitting edge–edge collision detections. In traditional detection, a collision detection among triangles is required after a leaf node collision. As shown in Figure 9, there are eight triangles in total, and one triangle at a time is taken for point-face intersection detection. This requires a total of 24 vertices to be detected, whereas the model has only nine vertices in reality.

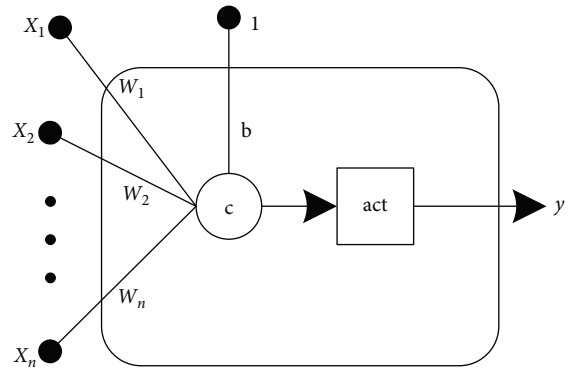

Figure 7: Perceptron structure.

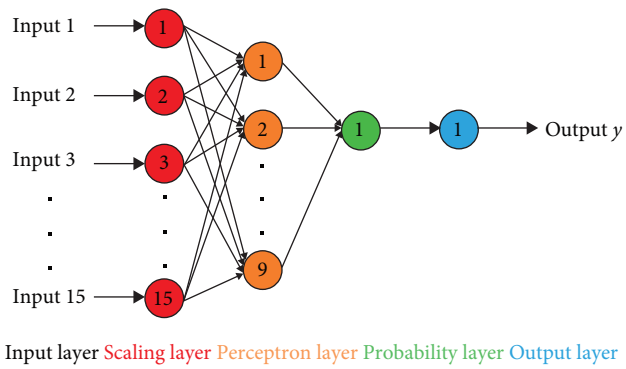

Figure 8: The most suitable neural network structure.

In practice, the model structure is far more complex than in Figure 9, and if vertices that have already been detected are not excluded, there will be a large number of repeatedly detected particles, increasing the computational effort of the algorithm and reducing the operating speed. To avoid this, the basic elements contained in the cloth used to construct the bounding boxes are particles, not triangles. In this way, many repetitive tests can be avoided, while allowing for a larger number of particles to be processed.

4. Experiments

To verify the effectiveness of the proposed algorithm, the simulation system was built under a Windows operating system using the development tool VS2017, using both C++ and OpenGL techniques. The models used in this article are downloaded from the network model library. The experimental results were obtained by comparing the proposed algorithm with existing algorithms.

4.1. OBB Algorithm with Simplified Model. In this study, we used an OBB algorithm with simplified model. As shown in Figure 10, the OBB constructed by the simplified model was approximately the same size as that constructed by the original model, but effectively reduced the amount of computation and accelerated the construction speed of the OBB. Moreover, the use of the fast adaptive OBB algorithm not only reduced the impact of the orientation bias created by the

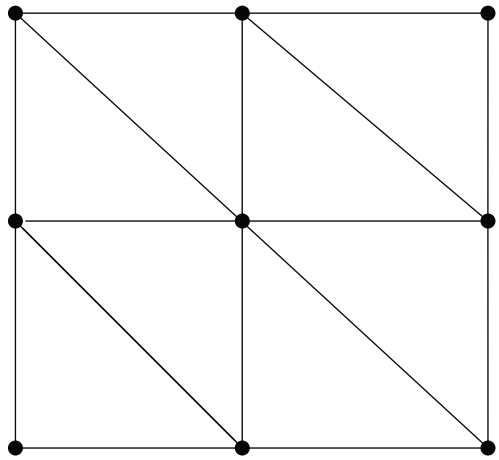

Figure 9: Simple fabric structure diagram.

Table 1: Time consuming comparison of OBB bounding box construction.

Method	Time consuming/s
OBB	2.817
Fast adaptive OBB algorithm [32]	1.554
OBB algorithm with simplified model	0.997

triangular faces in the model, but also sped up the construction of the bounding box. The detailed comparisons are shown in Table 1.

The proposed OBB algorithm with simplified model was not only effective in reducing the time consumption required for the construction of the bounding box, but also showed good performance in ensuring the bounding compactness of the model. Rabbit, cat, dinosaur, and human models were used in the experiments for comparison with a compactly constructed minimum-volume OBB generation algorithm [1]. The ratio of the constructed OBB to the model volume was used as the evaluation value, with a smaller value representing a higher level of compactness between the bounding box and model. As shown in Figure 11, the OBB algorithm with simplified model used herein was not as compact as the minimum-volume OBB generation algorithm, but the difference was negligible in the application scenario.

4.2. Optimization of Number of Neurons in the Neural Network. As mentioned above, to select the appropriate neural network structure, we used the incremental sequence method, starting with a small number of neurons and increasing the complexity of the method with each iteration. Figure 12 illustrates the track record of the selection and training errors for the different neuron selection processes, with the blue line indicating the training error and orange line indicating the selection error, respectively. The number of neurons for the minimum selection error was nine; therefore, a neural network with nine neurons in the first perceptron layer was selected.

4.3. Neural Network Performance Analysis. In this study, we evaluated the performance of the neural network by comparing the output of the neural network with the training instances. Herein, if instances collide, such instances are called positive classes. If no collision occurs, such an instance is called a negative class. The confusion matrix of the experimental results is shown in Table 2, and shows that all test instances were well-classified, with 435 correctly classified instances and 32 incorrectly classified instances. The classification accuracy is 93.75%, and the error rate is 6.25%.

The receiver operating characteristic curves of the experimental results are shown in Figure 13; they indicate the false positive class on the X-axis and true positive class on the Y-axis. The discriminatory ability is measured by calculating the area under the curve (AUC), and the closer the AUC is to 1, the better the classifier. In this study, AUC = 0.927, indicating that the neural network predicts well in most cases.

4.4. Comparison of Time Consumption of Different Algorithms for Collision Detection. We compared the removal efficiency when using algebraic nonpenetration filtering [15], spatial linear projection filtering [16], a combination of algebraic nonpenetration filtering and spatial linear projection filtering, and the DNN-based algorithm proposed in this study. As can be seen from Table 3, the use of the DNN-based algorithm appears to be less time-consuming. As can be seen from Table 4, although using both the nonpenetration filtering and spatial linear projection has the highest average removal rate, there is very little difference between this result and that using the DNN-based algorithm. Taken together, the DNN-based algorithm reduced the required time by an average of approximately 13% over the best removal method, i.e., the combination of algebraic nonpenetration filtering and the spatial linear projection filtering, while nearly equaling its average removal rate.

To compare the average time used by the different methods for collision detection, the classical bounding box algorithm, root-node double bounding box algorithm, fuzed DNN-based self-collision detection algorithm [2], and proposed algorithm were compared. Experiments were conducted with the same cloth model and the different collision object models, respectively, and the experimental results are shown in Table 5.

To more visually compare the methods in Table 5, we generated a histogram, as shown in Figure 14. As can be seen from Figure 14, it takes less time to use the traditional hybrid bounding box algorithm than the single bounding box algorithm. The root-node double bounding box is faster than the traditional hybrid bounding box algorithm in terms of running speed, with a time reduction of 5.51%–11.32%. The OpenNN-based collision detection algorithm outperforms all other algorithms in terms of speed, with a total time reduction of 11.70% compared to the root-node double bounding box, and a total time reduction of 6.62% compared to the fuzed DNN-based self-collision detection algorithm. The result indicates that the proposed algorithm has good detection efficiency in the simulation scenarios, along with high real-time performance.

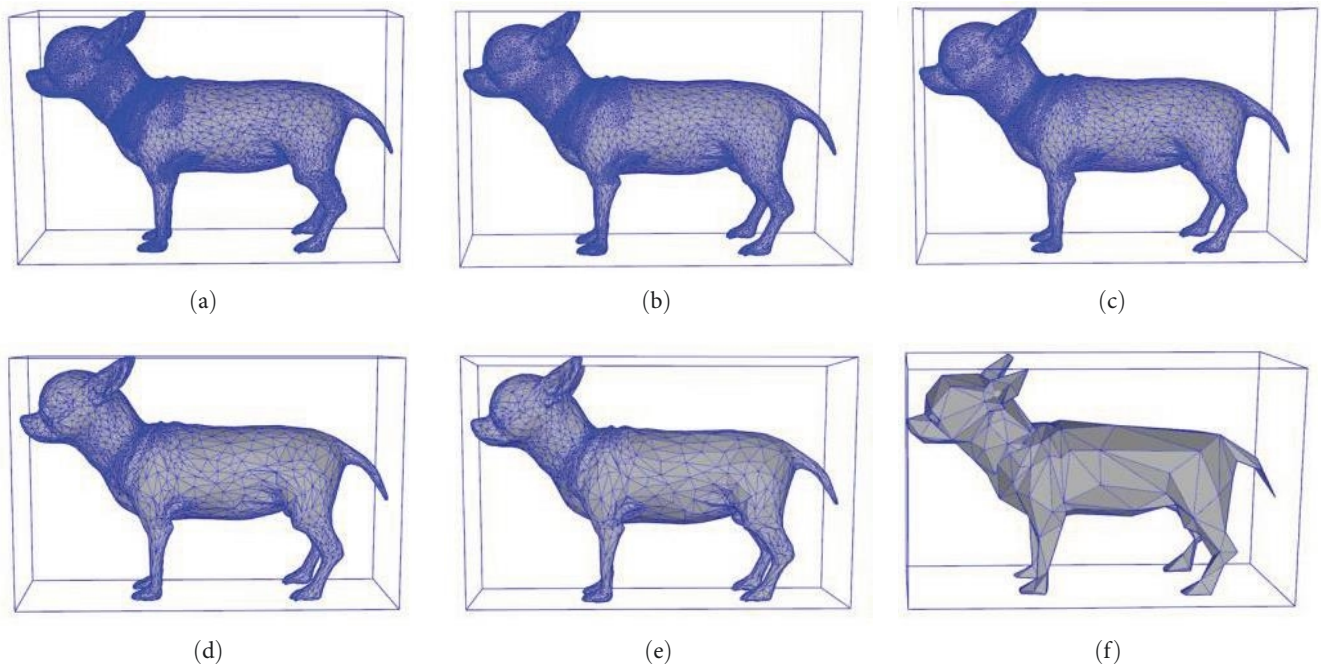

Figure 10: OBB bounding box construction by simplifying models of different degrees. (a) Original model, (b) 20%-simplified model, (c) 40%-simplified model, (d) 80%-simplified model, (e) 90%-simplified model, and (f) 99%-simplified model.

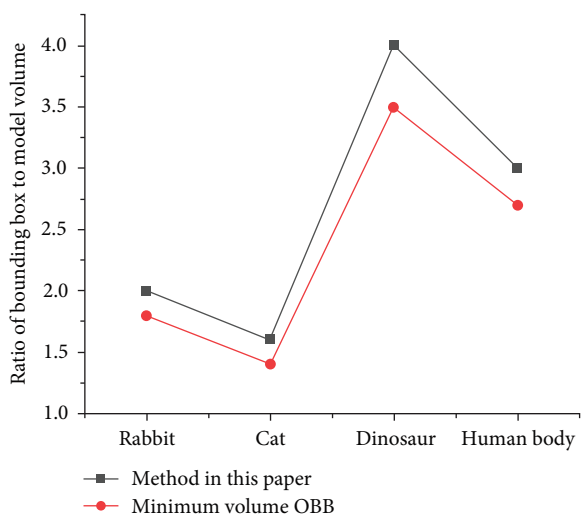

Figure 11: Tightness comparison with minimum volume OBB bounding box algorithm.

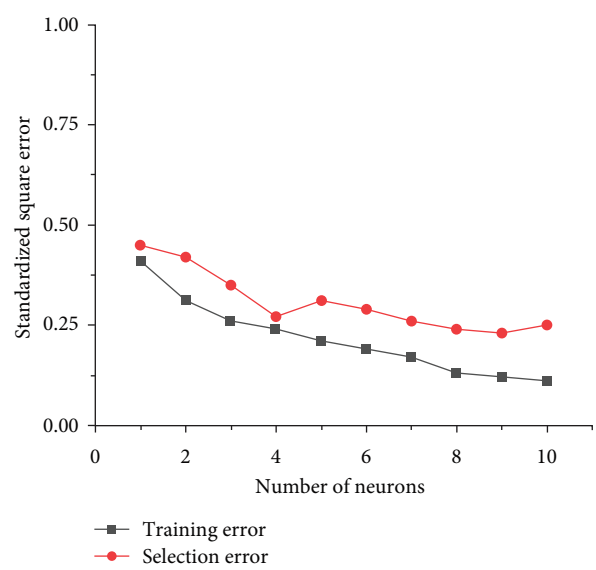

Figure 12: The influence of the number of neurons on the error.

Table 2: Confusion matrix.

	Prediction is positive class	Prediction is negative class
Result is positive class	291	12
Result is negative class	17	144

The comparison of different models in the experiments reveals that the more complex the model, the more evident the time advantage of the proposed algorithm. To further validate this conclusion, cloth materials A, B, and C with different levels of precision were subjected to collision simulations with a fish model. The detailed cloth model information is shown in Table 6, and the time consumption statistics are shown in Table 7.

To visually present the data in Table 7, a line graph was generated as shown in Figure 15. As can be seen from Figure 15, the collision detection time for each algorithm increases as the amount of cloth model data increases. When cloth model increased in precision by 84% from A to C, the two traditional algorithms and root-node double bounding box algorithm increased in time by 96% and the fuzed DNN-based self-collision detection algorithm increased

Table 3: Comparison of elimination time for different models interacting with cloth.

Model	Algebraic nonpenetration filtering [15]/s	Spatial linear projection filtering [16]/s	Combination of algebraic nonpenetration filtering and spatial linear projection filtering/s	Ours/s
Rabbit	19.08	19.29	19.56	17.34
Cat	19.12	19.34	19.87	16.84
Dinosaur	22.54	22.79	23.07	20.98
Human body	27.38	27.56	27.86	23.95

Table 4: Comparison of rejection rate between different models and cloth interactive collision.

Model	Algebraic nonpenetration filtering [15]	Spatial linear projection filtering [16]	Combination of algebraic nonpenetration filtering and spatial linear projection filtering	Ours
Rabbit	34.52%	51.89%	53.45%	52.28%
Cat	36.47%	53.23%	55.13%	55.87%
Dinosaur	35.23%	52.34%	54.76%	54.58%
Human body	38.84%	54.86%	57.69%	56.74%

Table 5: Collision detection time of different models.

Model	AABB/s	AABB-OBB/s	Root-node double bounding box algorithm/s	Fuzed DNN-based self-collision detection algorithm/s	Ours/s
Chicken	0.0607	0.0574	0.0509	0.0498	0.0461
Sphere	0.0682	0.0635	0.0600	0.0554	0.0521
Cube	0.0553	0.0536	0.0490	0.0458	0.0423
Fish	0.0677	0.0627	0.0588	0.0558	0.0526

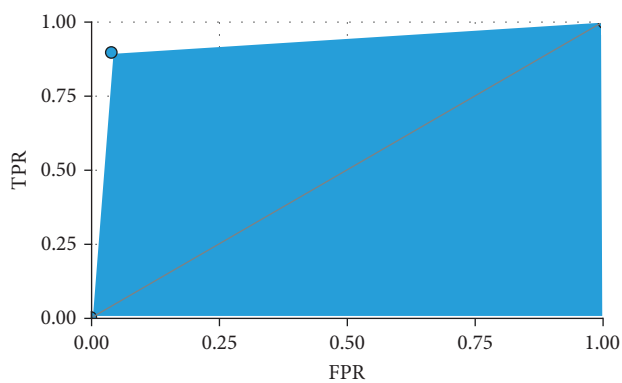

Figure 13: ROC curve.

Table 6: Comparison of different cloth models.

Cloth model	Vertex	Triangular face piece
Cloth A	400	722
Cloth B	900	1682
Cloth C	2,500	4,802

Table 7: Time for collision detection of different precision fabrics.

Cloth model	AABB/s	AABB-OBB/s	Root-node double bounding box algorithm/s	Fuzed DNN-based self-collision detection algorithm/s	Ours/s
Cloth A	0.0154	0.0151	0.0143	0.0295	0.0378
Cloth B	0.0677	0.0627	0.0588	0.0558	0.0526
Cloth C	0.3768	0.3724	0.3683	0.2983	0.1195

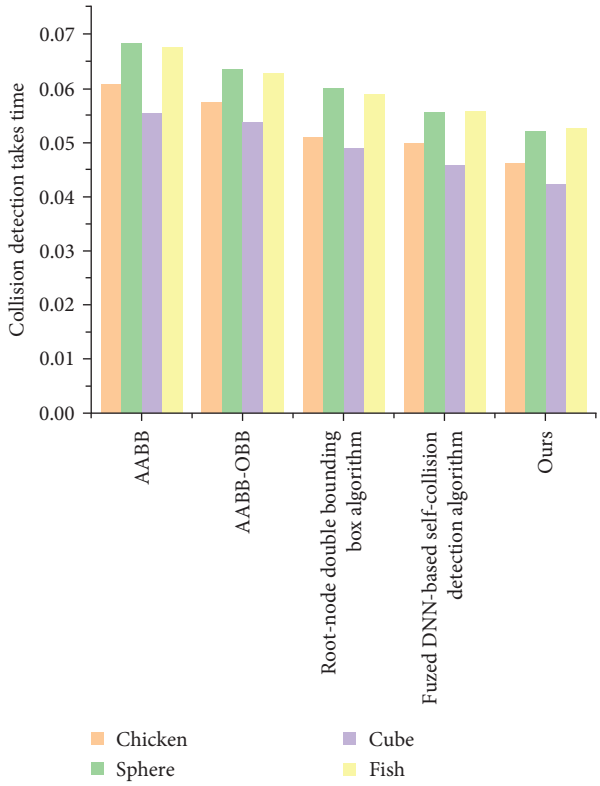

Figure 14: Time-consuming change of cloth collision detection with different models.

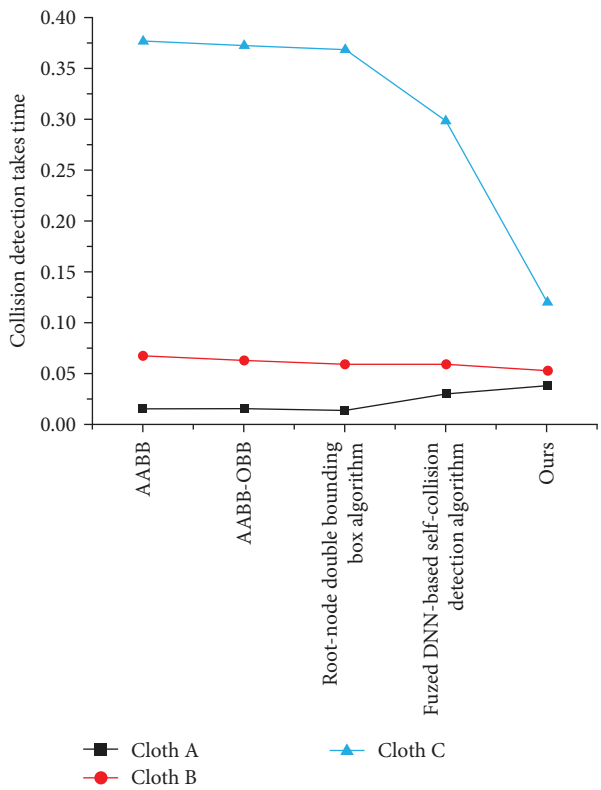

Figure 15: Time-consuming change of cloth collision detection with different precision.

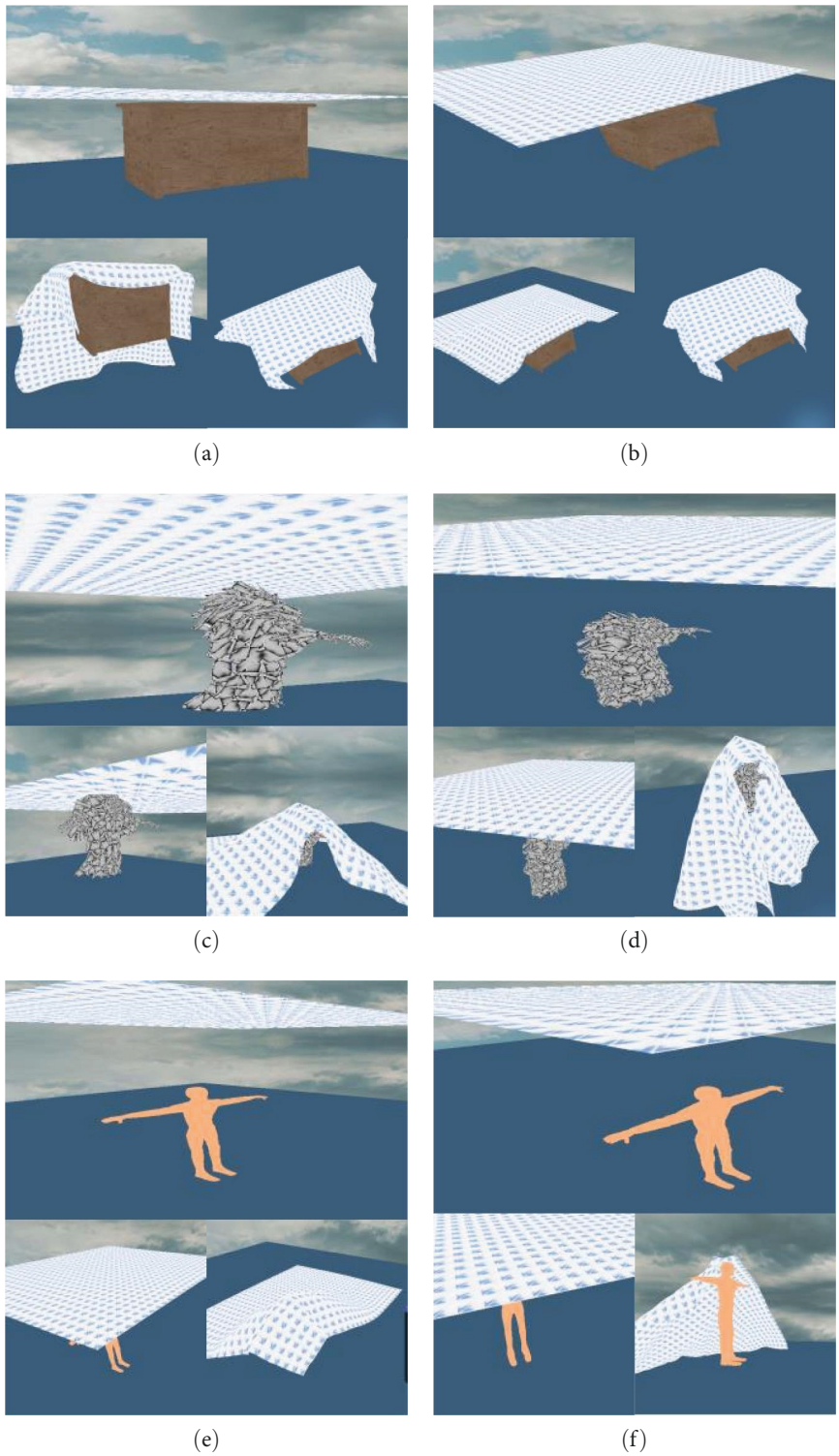

Figure 16: The effect of cloth collision with different models. (a, c, e) Using bounding box AABB simulation; (b, d, f) using neural network simulation.

in time by 90.11%, whereas the OpenNN-based algorithm increased by only 68.37%.

For cloth A (a simple model), the most time was spent when using the OpenNN-based algorithm, and among the traditional methods, the least time was spent when using the root-node double bounding box algorithm. This result indicates that the OpenNN-based algorithm is more suitable for dealing with complex and highly accurate models than simple models. The root-node double bounding box is suitable for small- and medium-sized models, similar to the

traditional bounding box algorithms. However, under the same conditions, the root-node double bounding box algorithm has an advantage in that it is less time-consuming than the traditional bounding box algorithms.

4.5. Validity Verification of the Algorithm Simulations. To verify that the proposed algorithm not only improves efficiency, but also ensures the validity of the simulations, cloth B was subjected to collision experiments with different models: (1) the cloth falls by gravity, collides with a wooden box and rests on the box; (2) the cloth collides with a game character with sharp edges and covers the game character; (3) a moving human body passes through the vertically falling cloth, and the cloth falls on the human body.

The simulation effectiveness values for different frames during the experiment are shown in Figure 16. The simulation effectiveness values using the traditional AABB algorithm are shown in Figures 16(a), 16(c), and 16(e). The simulation effectiveness values using the neural network are shown in Figures 16(b), 16(d), and 16(f). Through comparisons, the validity of the simulation effectiveness of the proposed algorithm is found to be approximately the same as that of the AABB algorithm. The simulation does not affect the visual perception, thereby ensuring the simulation effectiveness.

5. Conclusion

To address the problems, concerning the high-computational effort and low-running rate arising from the complex models when constructing OBBs, we propose an OBB algorithm with simplified model. By simplifying the model, the amount of computation required to construct the OBB is greatly reduced, and the shapes of the bounding box constructed before and after the simplified model are approximately the same. Therefore, the bounding box constructed by the simplified model can be treated as the bounding box of the model before the simplification. Second, we proposed a root-node double bounding box collision detection algorithm. A combination of the simplest AABB and bounding sphere is used to improve the removal rate of the bounding box and to enhance the efficiency of the collision detection algorithm. Third, we also propose an OpenNN-based collision detection algorithm for managing a large number of cloth particles in one time step, thereby reducing the construction time for the bounding box and accelerating the collision detection. In comparison with the traditional single bounding box algorithm, hybrid bounding box algorithm, and fuzed DNN-based self-collision detection algorithm, this proposed approach reduces the collision detection time while the accuracy is guaranteed and the performance is greatly improved.

The stability of the model as obtained by training the neural network in this experiment can be further improved. A larger and more comprehensive amount of samples should be obtained in the future to train the neural network to obtain a better model. To better integrate the collision detection algorithm with the neural networks, future research should attempt to use neural networks to predict the positions of cloth model particles after collisions, thereby reducing the time used for the collision response and improving the efficiency of the overall algorithm simulation.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was support in part by the Natural Science Foundation of China under grant 62071281, by the Nature Science Foundation of SHANXI Province under grant 202103021224218.

References

[1] Z. G. Hu and Q. F. Qin, "Minimum volume directed bounding box generation algorithm based on convex hull," *Journal of Hunan University (Natural Science Edition)*, vol. 46, no. 2, pp. 105–111, 2019.

[2] Y. X. Jin, Q. F. Cheng, J. R. Zhang, X. Qi, B. Ma, and Y. Jia, "Self-collision detection algorithm based on fused DNN and AABB-circular bounding box," *Journal of Image and Graphics*, vol. 25, no. 8, pp. 1674–1683, 2020.

[3] R. Wang, W. Hua, G. X. Xu, Y. Huo, and H. Bao, "Variational hierarchical directed bounding box construction for solid mesh models," 2203.

[4] J. R. Sun and X. M. Lu, "Optimized collision detection algorithm based on hybrid bounding box and intersection of triangles," *Computer Engineering and Applications*, vol. 54, no. 19, pp. 198–203, 2018.

[5] C. Liu, *Research and Implementation of Collision Detection Algorithm in Virtual Environment*, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2018.

[6] Y. C. Liu, H. X. Wang, and P. Yang, "A three-layer hybrid bounding volume hierarchy for collision detection," in *Proceedings of the 7th International Conference on Computer Engineering and Networks*, pp. 73–81, Proceedings of Science, Shanghai, China, 2017.

[7] Y. H. Li and Z. Y. Wang, "Improvement of collision algorithm based on hybrid hierarchical bounding box," *Journal of East China Jiaotong University*, vol. 36, no. 6, pp. 112–118, 2019.

[8] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "ICCD: interactive continuous collision detection between deform able models using connectivity-based culling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 544–557, 2009.

[9] X. Jia, Y.-K. Choi, B. Mourrain, and W. Wang, "An algebraic approach to continuous collision detection for ellipsoids," *Computer Aided Geometric Design*, vol. 28, no. 3, pp. 164–176, 2011.

[10] G. Van den Bergen, "A fast and robust GJK implementation for collision detection of convex objects," *Journal of Graphics Tools*, vol. 4, no. 2, pp. 7–25, 1999.

[11] Y. X. Jin, C. Ren, Z. Li, S. Y. Chen, H. Wang, and H. Y. Han, "Research on deformed body collision detection algorithm based on intelligent algorithm," *Computer Engineering and Applications*, vol. 53, no. 19, pp. 130–135, 2017.

[12] H.-Y. Qu, W. Zhao, and A.-H. Qin, "A fast collision detection algorithm based on optimization operator," *Journal of Jilin University*, vol. 47, no. 5, pp. 1598–1603, 2017.

[13] P. Du, E. S. Liu, and T. Suzumura, "Parallel continuous collision detection for high-performance GPU cluster," in *ACM Siggraph Symposium on Interactive 3d Graphics & Games*, pp. 1–7, Association for Computing Machinery, San Francisco, USA, 2017.

[14] M. Tayyub and G. N. Khan, "Heterogeneous CPU–GPU implementation of collision detection," *IADIS International Journal on Computer Science and Information Systems*, vol. 14, no. 2, pp. 25–40, 2019.

[15] M. Y. Lu, Y. C. Wan, W. Zhao, Y. Tang, and J. Zhao, "Fast cloth collision detection and effective contact friction algorithm," *Journal of Computer-Aided Design & Computer Graphics*, vol. 32, no. 3, pp. 392–400, 2020.

[16] Y. Wan, Y. L. Fu, and L. Yao, "Continuous collision detection algorithm based on spatial linear projection," *Intelligent Computer and Application*, vol. 12, no. 2, pp. 1–5, 2022.

[17] D. Holden, B. C. Duong, S. Datta, and D. Nowrouzezahrai, "Subspace neural physics: fast data-driven interactive simulation," in *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–12, ACM, Los Angeles, USA, 2019.

[18] H. Zhang, *Research on the Relationship between Posture and Folds of Virtual Clothing Network Display Based on Machine Learning*, Beijing Institute of Fashion Technology, Beijing, China, 2019.

[19] M. Shi, L. Yang, T. L. Mao, Y. W. Deng, and S. Q. Wang, "Study on the correlation between human motion and garment deformation in cloth animation," *Journal of Computer-Aided Design and Computer Graphics*, vol. 29, no. 10, pp. 1941–1951, 2017.

[20] L. Chen, J. Ye, L. Jiang, C. Ma, Z. Cheng, and X. Zhang, "Synthesizing cloth wrinkles by CNN-based geometry image superresolution," *Computer Animation and Virtual Worlds*, vol. 29, no. 3-4, Article ID e1810, 2018.

[21] Y. J. Oh, T. M. Lee, and I. K. Lee, "Hierarchical cloth simulation using deep neural networks," in *Proceedings of Computer Graphics International 2018*, pp. 139–146, Association for Computing Machinery, Bintan Island, Indonesia, 2018.

[22] M. Shi, Y. Liu, T. Y. Mao, Y. W. Deng, and S. Q. Wang, "Study on the correlation between human movement and costume deformation in costume animation," *Journal of Computer Aided Design and Graphics*, vol. 29, no. 10, pp. 1941–1951, 2017.

[23] Y. Jiang, R. Guo, F. Ma, and J. Shi, "Cloth simulation for Chinese traditional costumes," *Multimedia Tools and Applications*, vol. 78, no. 4, pp. 5025–5050, 2019.

[24] H. Zhang, *Study on the Relationship between Body Shape and Garment Fold in Virtual Online Display Based on Machine Learning*, Beijing Institute of Fashion Technology, Beijing, 2019.

[25] T. F. H. Runia, K. Gavrilyuk, C. G. M. Snoek, and A. W. M. Smeulders, "Cloth in the wind: a case study of physical measurement through simulation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10495–10504, IEEE, 2020.

[26] E. Gundogdu, V. Constantin, S. Parashar et al., "GarNet++: improving fast and accurate Static3D cloth draping by curvature loss," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2020.

[27] E. Ju and M. G. Choi, "Estimating cloth simulation parameters from a static drape using neural networks," *IEEE Access*, vol. 8, pp. 195113–195121, 2020.

[28] Z. Montazeri, C. Xiao, Y. R. Fei, C. Zheng, and S. Zhao, "Mechanics-aware modeling of cloth appearance," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 1, pp. 137–150, 2019.

[29] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on computer graphics and interactive techniques*, pp. 209–216, ACM Press, 1997.

[30] R. Ronfard and J. Rossignac, "Full-range approximation of triangulated polyhedra," *Computer Graphics Forum*, vol. 15, no. 3, pp. 67–76, 1996.

[31] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–7, 2012.

[32] C. Wang, Z. L. Zhang, Y. Long, and S. D. Wang, "Improved hybrid bounding box collision detection algorithm," *Journal of System Simulation*, vol. 30, no. 11, pp. 4236–4243, 2018.