Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.055538

ARTICLE

Virtual Assembly Collision Detection Algorithm Using Backpropagation Neural Network

Baowei Wang[1,2,*] **and Wen You**[2]

[1]School of Computer, Nanjing University of Information Science and Technology, Collaborative Innovation Center of Jiangsu Atmospheric Environment and Equipment Technology, Digital Forensics Engineering Research Center of Digital Forensics Ministry of Education, Nanjing, 210044, China

[2]School of Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China

*Corresponding Author: Baowei Wang. Email: wbw.first@163.com

Received: 30 June 2024 Accepted: 27 August 2024 Published: 15 October 2024

ABSTRACT

As computer graphics technology continues to advance, Collision Detection (CD) has emerged as a critical element in fields such as virtual reality, computer graphics, and interactive simulations. CD is indispensable for ensuring the fidelity of physical interactions and the realism of virtual environments, particularly within complex scenarios like virtual assembly, where both high precision and real-time responsiveness are imperative. Despite ongoing developments, current CD techniques often fall short in meeting these stringent requirements, resulting in inefficiencies and inaccuracies that impede the overall performance of virtual assembly systems. To address these limitations, this study introduces a novel algorithm that leverages the capabilities of a Backpropagation Neural Network (BPNN) to optimize the structural composition of the Hybrid Bounding Volume Tree (HBVT). Through this optimization, the research proposes a refined Hybrid Hierarchical Bounding Box (HHBB) framework, which is specifically designed to enhance the computational efficiency and precision of CD processes. The HHBB framework strategically reduces the complexity of collision detection computations, thereby enabling more rapid and accurate responses to collision events. Extensive experimental validation within virtual assembly environments reveals that the proposed algorithm markedly improves the performance of CD, particularly in handling complex models. The optimized HBVT architecture not only accelerates the speed of collision detection but also significantly diminishes error rates, presenting a robust and scalable solution for real-time applications in intricate virtual systems. These findings suggest that the proposed approach offers a substantial advancement in CD technology, with broad implications for its application in virtual reality, computer graphics, and related fields.

KEYWORDS

Collision detection; virtual assembly; backpropagation neural network; real-time interactivity

1 Introduction

With the rapid advancement of virtual reality (VR) technology, virtual assembly has emerged as a significant VR application scenario, garnering increasing attention and research interest [1]. In a virtual assembly environment, users can perform assembly operations through devices such as head-mounted displays and controllers [2]. This environment not only provides users with a more realistic and intuitive interactive experience but also offers efficient and safe solutions for fields such as manufacturing and robotic operations [3]. However, due to the variable attributes of objects in the virtual assembly environment. including shape, size, and mass, achieving efficient and accurate collision detection is crucial. Collision detection. which refers to the determination of whether two objects have collided, is a fundamental technology in virtual assembly environments [4]. Through precise collision detection, users can experience realistic physical interactions, thereby enhancing the immersion and realism of the VR experience [5].

The CD kernel processes a scene containing multiple objects and identifies all pairs of colliding objects, along with application-specific parameters such as the collision point, time, or volume displaced. Exhaustively searching for collisions among all object pairs requires $O(n^2)$ time complexity, making it computationally prohibitive for large scenes [6]. Therefore, the CD process is divided into two phases: broad and narrow. The broad phase quickly eliminates object pairs that are unlikely to collide using techniques like sweep-and-prune (SAP), Bounding Volume Hierarchy (BVH), and spatial hashing [7]. The narrow phase then conducts detailed intersection tests on the remaining pairs to determine exact collisions.

The computational intensity of precise collision detection has historically led researchers to focus on optimizing the broad phase. However, exact collision detection remains crucial in many scenarios, such as assessing mechanical failures due to impact or internal stress [8]. To address these needs, we introduce Mochi, a fast and exact collision detection engine that leverages Ray Tracing (RT) cores to accelerate both broad and narrow phases.

Real-world objects have complex shapes, necessitating the use of bounding volumes for efficient indexing. By organizing these bounding volumes, a spatial data structure called BVH enables rapid non-intersection space culling, reducing collision detection time [9]. Faster BVH traversal accelerates simulations, rendering, and collision avoidance measures. State-of-the-art CD libraries utilize Graphics Processing Unit (GPU) accelerators for optimized BVH construction and traversal techniques [10]. The integration of ray tracing architecture in modern GPUs, such as NVIDIA's Turing and later RTX series, offers new opportunities for CD optimization [11]. These GPUs feature specialized RT cores designed for efficient BVH operations and ray-object intersections, originally intended for real-time graphical rendering [12]. By mapping the CD problem to the ray tracing paradigm, Mochi exploits RT cores to achieve significant performance improvements in detecting collisions among various object types, including spherical particles, mathematically defined objects, and complex triangle meshes.

The authenticity of virtual environments depends on participants' ability to perceive and interact with virtual objects realistically. This requires precise collision detection to ensure accurate physical feedback during interactions. The increasing geometric complexity of virtual environments and the need for real-time interaction intensify the computational burden of collision detection, making it a critical challenge. Thus, precise collision detection is imperative for enhancing realism and immersion, imposing stringent computational and real-time demands on detection algorithms [13]. Fig. 1 shows the Research Motivation and Innovation Framework of this article. Our contributions can be summarized as follows:

(1) We have innovated the structure of the Hybrid Bounding Volume Tree (HBVT) by layering the data structure, resulting in a more efficient collision detection process.
(2) To enhance the efficiency of constructing HBVT, we employ PSO combined with a BPNN. This approach ensures more appropriate initial weight and threshold settings, and then utilizes the neural network to optimize the time overhead.

(3) We constructed a virtual assembly system using WebGL and configured the specific environment for the experiment.
(4) Experimental results indicate that, compared to existing CD algorithms, the CD algorithm proposed in this paper demonstrates superior efficiency and accuracy for complex models.

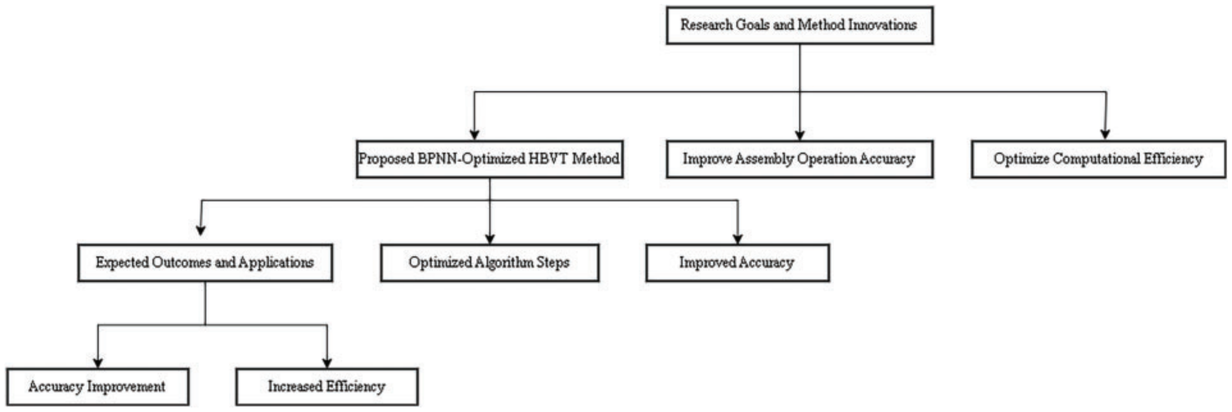
Research Goals and Method Innovations
Proposed BPNN-Optimized HBVT Method
Improve Assembly Operation Accuracy
Optimize Computational Efficiency
Expected Outcomes and Applications
Optimized Algorithm Steps
Improved Accuracy
Accuracy Improvement
Increased Efficiency

Figure 1: Research motivation and innovation framework

2 Related Work

2.1 Research Status of Collision Detection

CD is a critical component in the realm of virtual environments, with applications spanning from computer graphics and virtual reality to robotics and simulation. Recent advancements have focused on improving both the accuracy and efficiency of CD algorithms to meet the increasing demands of real-time interaction and complex scene management.

The broad-phase CD aims to quickly eliminate object pairs that are unlikely to collide, thus reducing the number of expensive narrow-phase collision tests. Recent methods have improved the efficiency of this phase through enhanced spatial partitioning techniques. For instance, dynamic BVH have been optimized to adapt more rapidly to changes in the scene, significantly reducing the overhead during real-time updates [14]. Additionally, the use of spatial hashing and grid-based methods has been refined to handle large and densely populated virtual environments more effectively.

In the narrow-phase, where precise CD occurs, significant strides have been made in algorithmic optimization and hardware acceleration. Techniques leveraging the computational power of modern GPUs have become prominent, particularly with the integration of ray tracing cores in GPUs by NVIDIA [15]. These cores facilitate rapid BVH traversal and intersection tests, enabling more complex and dynamic scenes to be handled in real-time [16]. Furthermore, algorithms have been developed to exploit coherence in the motion of objects, reducing redundant calculations and enhancing performance [17].

As virtual environments become more intricate, the ability to accurately detect collisions between complex geometries becomes paramount. Mesh-based CD, which involves detecting intersections between intricate polygonal meshes, has seen considerable advancements. Recent research has introduced methods to handle coplanar and non-coplanar triangle-triangle intersection tests more efficiently, overcoming a significant bottleneck in mesh-based CD [18]. Additionally, hybrid approaches combining discrete and continuous CD methods have been proposed to manage both static and dynamic interactions seamlessly [19].

One of the ongoing challenges in CD is ensuring real-time performance as the scale and complexity of virtual environments grow. To address this, parallel processing techniques and multi-threading have been employed to distribute the computational load across multiple cores. Adaptive algorithms that adjust their precision based on the proximity and velocity of objects have also been developed to maintain a balance between accuracy and performance [20]. Moreover, machine learning techniques are being explored to predict and pre-emptively manage potential collisions, further enhancing the responsiveness of virtual environments [21].

Additionally, the inclusion of BVH enhances the efficiency of CD algorithms by organizing objects based on their bounding volumes. As shown in Fig. 2, the timeline of Collision Detection research is presented, BVH has been proposed since 2000, BVH allows for rapid culling of non-intersecting pairs during the broad phase, further optimizing CD processes in virtual environments. Common Hierarchical Bounding Volume. Algorithms: BVH, Octree, R-tree, KD-tree, Binary Space Partitioning (BSP) Tree. These algorithms are commonly used to organize spatial data efficiently, enabling fast CD in virtual environments. The algorithm proposed in this paper introduces a new BVH structure compared to other collision detection algorithms. It incorporates a BP neural network and employs particle swarm optimization to ensure that the initial weights and thresholds are set more appropriately. By using the trained neural network to predict and obtain the hierarchical bounding structure that results in the minimum detection time, the derived detection method enhances the accuracy and speed of object collision detection in virtual environments.

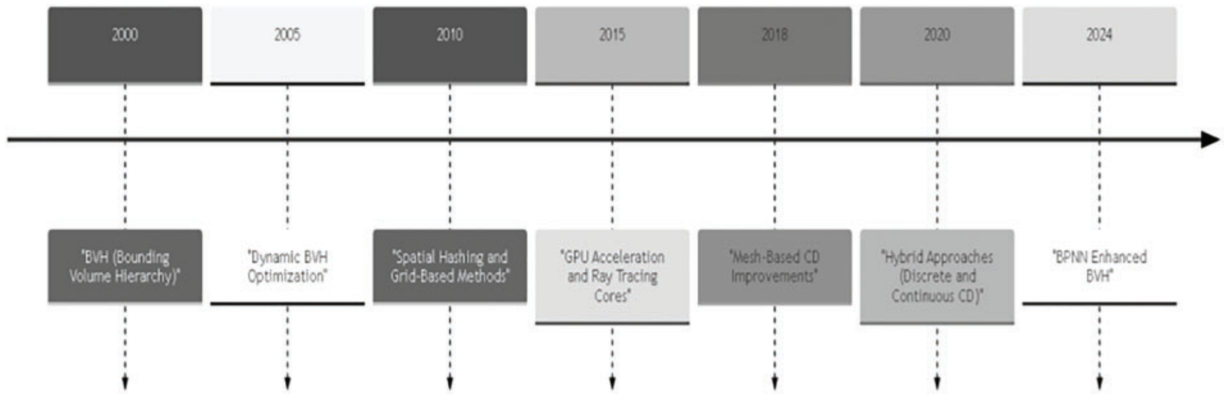
2000
2005
2010
2015
2018
2020
2024
"BVH (Bounding Volume Hierarchy)"
"Dynamic BVH Optimization"
"Spatial Hashing and Grid-Based Methods"
"GPU Acceleration and Ray Tracing Cores"
"Mesh-Based CD Improvements"
"Hybrid Approaches (Discrete and Continuous CD)"
"BPNN Enhanced BVH"

Figure 2: Collision detection algorithms timeline

2.2 *Research Status of Collision Detection*

BVH are pivotal in optimizing CD algorithms, particularly in complex scenes with numerous objects. BVH encapsulate objects within hierarchical bounding volumes, such as spheres, axis-aligned bounding boxes (AABB), or oriented bounding boxes (OBB). These hierarchies enable efficient traversal and culling of non-intersecting objects, significantly reducing computational overhead [22]. The efficiency of BVH-based algorithms lies in their ability to decompose the CD problem into a series of simpler, hierarchical tests, allowing for rapid exclusion of large portions of the scene that do not require detailed collision checks.

2.2.1 Construction of BVH

The construction process of a BVH typically follows a top-down approach, starting from the bounding volume enclosing the entire scene and recursively partitioning it into smaller sub-volumes until reaching a termination condition.

2.2.2 Structure Design

In traditional CD algorithms, BVH are commonly used to optimize the detection process. The primary function of this tree structure is to quickly eliminate objects that are clearly non-intersecting using higher-level bounding volumes, followed by detailed intersection checks between the leaf nodes, specifically between triangular facets. However, in practical applications, the choice of tree data structure significantly impacts the detection speed in virtual environments. Therefore, selecting an appropriate tree data structure is crucial for the performance of BVH.

As shown in Fig. 3, HBVT is an improved structure developed on the basis of BVH. HBVT not only adopts hierarchical bounding boxes, but also combines different types of bounding boxes to adapt to different geometric distributions and application scenarios, thereby further improving the efficiency and accuracy of collision detection.

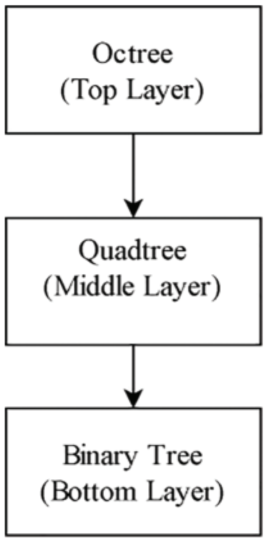

Figure 3: HBVT structure

Typically, HBVT structures include binary trees, quadtrees, and octrees. Traditional BVH algorithms often employ a single data structure, but this study proposes a composite tree data structure that integrates the advantages of each type. This composite hierarchical tree is organized into three layers: the top layer utilizes octrees, the middle layer employs quadtrees, and the bottom layer uses binary trees. The depth of each layer is variable, allowing flexibility to adapt to specific requirements of the virtual environment.

2.3 Backpropagation Neural Network

BPNN is a key type of artificial neural network (ANN) known for learning from data by adjusting connection weights between neurons. It consists of input, hidden, and output layers, processing input signals through neurons with activation functions. Training involves forward propagation of inputs to generate predictions, followed by backpropagation of errors to adjust weights and minimize prediction errors. BPNN are widely used in image and speech recognition, natural language processing, and financial forecasting due to their ability to model complex data relationships [23]. However, optimizing parameters and addressing overfitting challenges remain critical research areas to enhance BPNN performance [24].

BPNN are a type of artificial neural network used extensively for supervised learning tasks. They are particularly effective for training multi-layer perceptrons through a process known as gradient descent. A BPNN typically comprises an input layer, one or more hidden layers, and an output layer. During the forward propagation phase, input data passes through the network, with each neuron applying an activation function to the weighted sum of its inputs. This weighted sum for each neuron l n layer is j given by:

$$z_j^l = \sum\nolimits_i w_{ij}^l a_i^{l-1} + b_j^l \tag{1}$$

where z_j^l is the weighted sum, w_{ij}^l is the weight from neuron i in layer $l-1$ to neuron j in layer l, a_i^{l-1} is the activation of neuron i in layer $l-1$, and b_j^l is the bias for neuron j in layer l. The activation a_j^l is computed using an activation function:

$$a_j^l = \sigma\left(z_j^l\right) \tag{2}$$

Backpropagation adjusts the network's weights to minimize prediction error. For the output layer, the error δ_j^L for neuron j is:

$$\delta_j^l = \sum\nolimits_k \delta_k^{l+1} w_{jk}^{l+1} \sigma'\left(z_j^l\right) \tag{3}$$

where δ_k^{l+1} is the error of neuron k in the subsequent layer $l+1$, w_{jk}^{l+1} is the weight from neuron j in layer l to neuron k in layer $l+1$, and $\sigma'\left(z_j^l\right)$ is the derivative of the activation function at z_j^l. Weights and biases are updated using the gradient descent method.

$$w_{ij}^l = w_{ij}^l - \eta \delta_j^l a_i^{l-1} \tag{4}$$

$$b_j^l = b_j^l - \eta \delta_j^l \tag{5}$$

Fig. 4 illustrates the workflow of a BPNN. Initially, the input layer neurons receive the input data and pass it to the hidden layer through forward propagation. The hidden layer processes the data and sends the result to the output layer. The output layer neurons produce the predicted results, which are then compared with the target values to calculate the error. This error is used in the backward propagation phase to compute the gradients of the weights, which are subsequently adjusted to minimize the error. This process of forward and backward propagation continues iteratively until the network converges to an acceptable error level.

Despite the widespread use of BPNN, they suffer from several limitations. For instance, the initial weights and biases of the network significantly affect its predictive accuracy, and BPNN typically initialize these values randomly. This arbitrary initialization can lead to the network converging to suboptimal solutions during training, thus impacting prediction performance. Additionally, BPNN often exhibit slow learning rates, requiring numerous iterations to achieve convergence, and are prone to getting trapped in local minima, which further restricts their effectiveness.

To address these limitations and enhance the predictive performance of BPNN, this study integrates the Particle Swarm Optimization (PSO) algorithm for optimizing the initial weights and biases. PSO is a population-based optimization technique inspired by the social behavior of birds flocking or fish schooling. It searches for optimal solutions by iteratively updating the positions and velocities of individual particles in the search space. The integration of PSO with BPNN follows these steps.

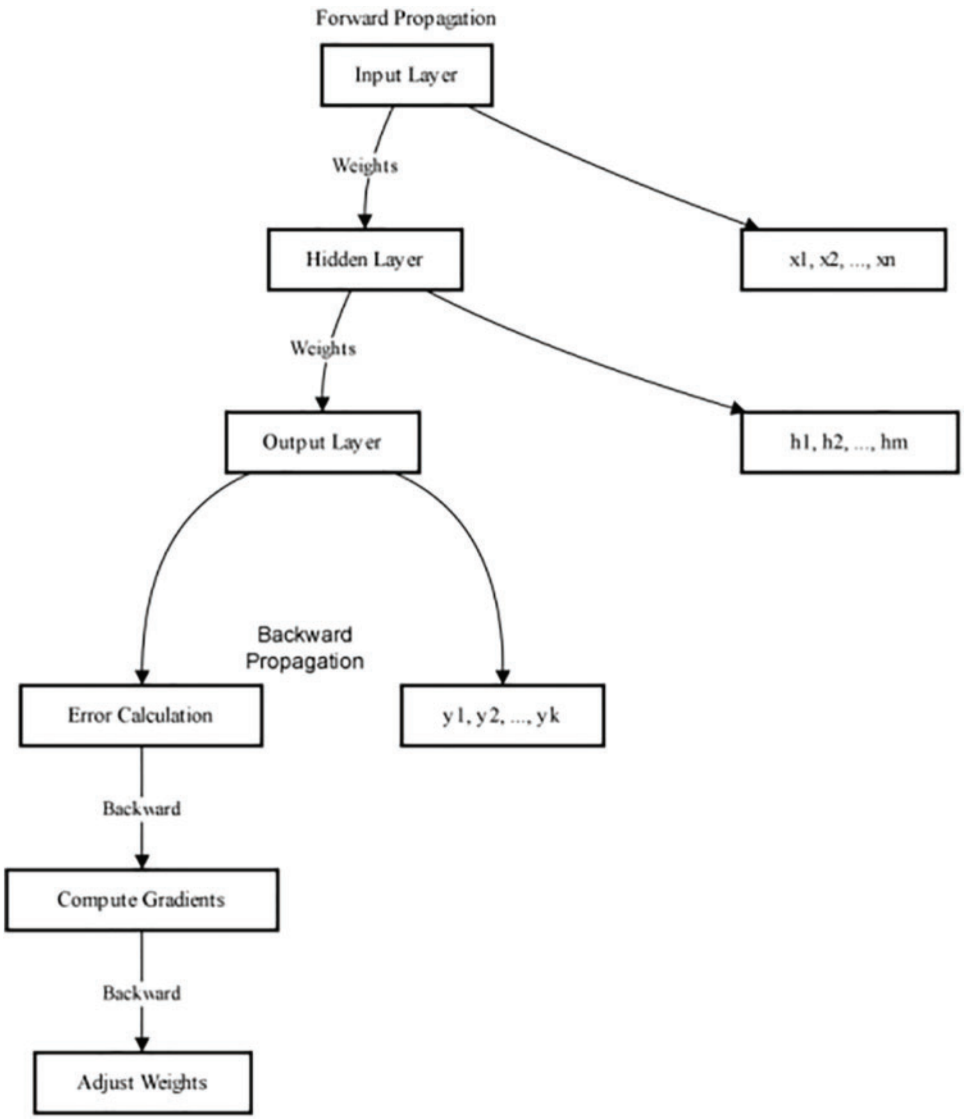

Figure 4: BPNN workflow

Fig. 5 outlines the PSO algorithm's iterative process, starting from the initialization of particle positions and velocities. Each particle's fitness is evaluated based on the neural network's performance, followed by updating individual and global best positions. If the stopping condition is not met, particles' velocities and positions are iteratively updated. This loop continues until convergence, resulting in optimal initial weights and biases for the BPNN. By employing PSO, the BPNN can achieve faster convergence and avoid local minima, leading to improved prediction accuracy.

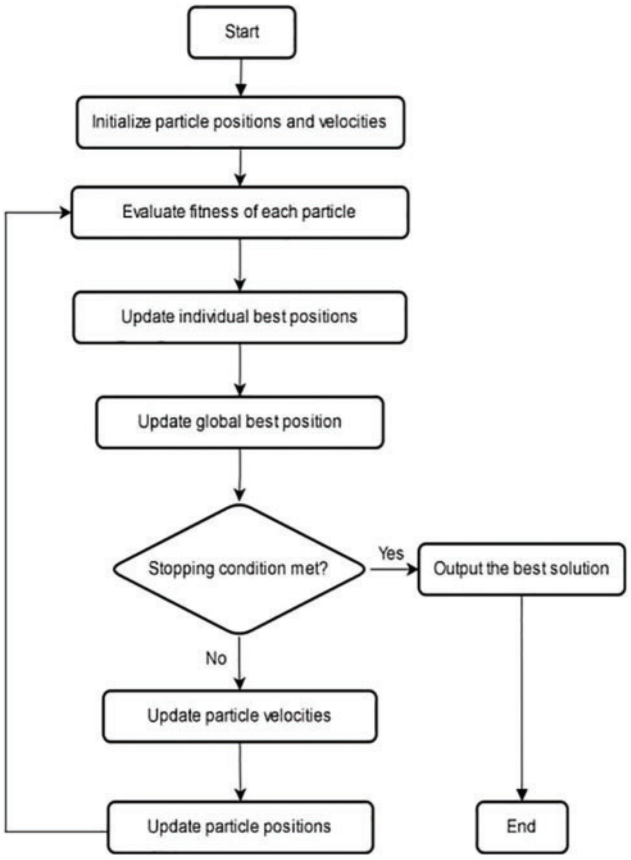
Start
Initialize particle positions and velocities
Evaluate fitness of each particle
Update individual best positions
Update global best position
Stopping condition met?
Yes
Output the best solution
No
Update particle velocities
Update particle positions
End

Figure 5: Flowchart of particle swarm optimization for BPNN

For simple virtual models, hybrid hierarchies can be generated by enumerating all possible hierarchical compositions and selecting the optimal structure based on collision test results. For more complex physical models, a subset of probable combinations is selected for collision testing, and the recorded time expenditures are used to create a training dataset for the BPNN. By training the BPNN with this data, it can predict collision test outcomes for various hierarchical structures efficiently and accurately, facilitating the selection of an optimal configuration.

During the construction of hierarchical bounding boxes, it was observed that construction time varies with model complexity. More complex models yield a greater number of potential bounding hierarchies, increasing the construction time. To enhance efficiency, we utilized a PSO optimized BPNN to minimize time expenditure.

These methods streamline the construction of bounding hierarchies for complex models, leveraging the predictive capabilities of the optimized BPNN to ensure an efficient and precise process.

3 Systematic Evaluation

3.1 System Realizations

The virtual assembly system is designed with functional modularization, comprising four core modules: scene simulation, CD, user control, and database management. The scene simulation module emulates physical dynamics and interactions within the virtual environment to ensure realistic simulations. The CD module employs efficient algorithms to identify and resolve object collisions, ensuring system stability. The user control module provides a user-friendly interface and supports various interaction modes to enhance user experience. Lastly, the database management module handles data storage, management, and retrieval, ensuring data integrity and reliability. Together, these modules form an efficient and reliable architecture for virtual assembly systems.

To implement the algorithm, we first established a virtual assembly system simulation platform. This platform is built using WebGL technology and integrates the Three.js library for rendering and interaction in virtual environments. On this basis, we use VS Code, SolidWorks software, and WebGL graphics library to achieve simulation. So as to achieve operations in the virtual assembly system. Fig. 6 shows a schematic diagram of the experimental setup.

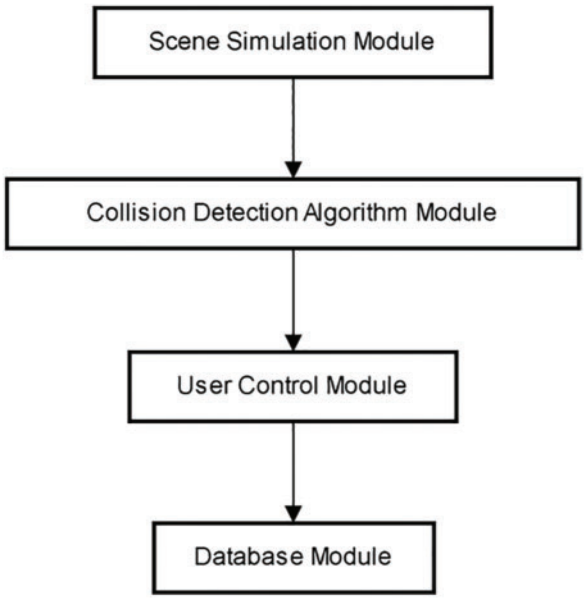

Figure 6: System modules

The process of assembling the three-dimensional model is initiated by creating the mechanical product assembly model in SolidWorks and converting it to an STL file format. Subsequently, the STLLoader.js from Three.js is utilized to load the model file and configure the renderer, camera, scene, and light sources for displaying the product's three-dimensional model.

In showcasing the model assembly, the process begins with fully automatically reconstructing assembly constraint relationships. Since the assembly model lacks original assembly information and is in STL file format, a topological structure reconstruction is necessary. Employing a semi-side construction method, the entire part STL mesh of the assembly model is processed. By utilizing bounding boxes to distinguish part intersection relationships and automatically retrieving coplanar and coaxial constraint relationships, complete automatic reconstruction of model assembly constraint relationships is achieved, thereby concluding the model assembly presentation. Figs. 7 and 8 show the three-dimensional models of the component parts.

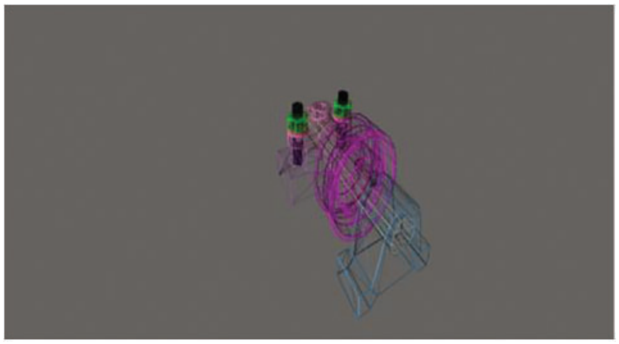

Figure 7: Three-dimensional model

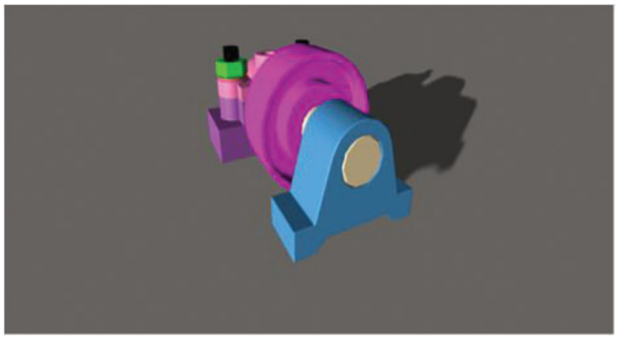

Figure 8: Three-dimensional rendering effect

3.2 Collision Detection Validation

3.2.1 Experimental Setup

To validate the proposed method of constructing hybrid hierarchical bounding boxes using PSO optimized BPNN, we conducted the following experiments.

The experimental environment was established with advanced hardware and software configurations to ensure optimal performance and accuracy. The hardware setup included an Intel Core i7 processor, 16 GB of RAM, and an NVIDIA GTX 3060 GPU, providing ample computational power for intensive tasks. The software environment comprised the Windows 10 operating system and Python 3.8 for general programming and scripting. For deep learning experiments, we utilized both TensorFlow and PyTorch frameworks, leveraging their extensive libraries and functionalities. Additionally, MATLAB was employed to implement the PSO algorithm, taking advantage of its comprehensive mathematical and graphical capabilities.

We selected 100 different parts and conducted random collision experiments. Each experiment recorded the hierarchical bounding box structure and the collision test time expenditure. The dataset included information on hierarchical combinations and time expenditures, as shown in Table 1 (partial data).

Through these experiments, we were able to evaluate the effectiveness and performance of the proposed method in practical applications.

Table 1: Conducted random collision experiments

ID	Hierarchical structure	Collision test time (ms)
1	Combination A	120
2	Combination B	95
...	...	...
100	Combination Z	150

3.2.2 BPNN Training

(1) Data Preprocessing

The collected dataset was normalized to ensure that the input data was on a similar scale, enhancing model training effectiveness.

The dataset was randomly divided into a training set (80%) and a test set (20%).

(2) Neural Network Architecture

A three-layer BPNN was employed, consisting of an input layer, hidden layer, and output layer.

The input layer node count equaled the number of features in the hierarchical structure, the hidden layer node count was determined experimentally, and the output layer represented the collision test time.

(3) Training Procedure

The Adam optimizer was used for training, with the loss function being the Mean Squared Error (MSE).

The initial learning rate was set to 0.001, and adjusted based on training performance.

(4) The particle swarm was initialized, with each particle representing a set of initial weights and biases for the neural network.

The fitness function was defined as the MSE on the validation set.

Through multiple iterations, the particle with the optimal fitness updated the neural network parameters.

3.2.3 Prediction and Validation

(1) Predictive Analysis

The trained BPNN was used to predict the collision test time expenditure for all possible hierarchical bounding box combinations. The combination with the minimum predicted time expenditure was selected.

(2) Result Verification

The hierarchical structure predicted to be optimal was subjected to empirical collision tests, and the actual time expenditure was recorded during these tests. The predicted and actual times were compared to validate the model's accuracy.

(3) Result Presentation

As shown in Table 2, the PSO-optimized BPNN model was shown by experimental results to effectively predict collision test times for various hierarchical structures, with the prediction errors falling within predefined acceptable ranges.

Table 2: Comparison of some predicted and actual results

Hierarchical structure	Predicted time (ms)	Actual time (ms)	Error (%)
Combination 1	118	120	1.67
Combination 2	97	95	2.11
...	...	...	...
Combination 50	124	128	3.32

The experimental validation indicates that the proposed method significantly improves the efficiency and accuracy of constructing hybrid hierarchical bounding boxes for complex models. Results confirm that the PSO-optimized BPNN effectively predicts collision test times for different hierarchical structures.

The success rate of our proposed method was determined by the proportion of experiments where the predicted collision test times closely matched the actual collision test times within an acceptable error margin. Based on the experimental results, the PSO-optimized BPNN demonstrated a high success rate, with the majority of predictions falling within a 5% error margin of the actual test times. This consistency underscores the reliability and robustness of our method in practical applications, making it a valuable tool for improving assembly operation accuracy.

The final selection prioritizes the construction of a hierarchical bounding volume hierarchy with minimized time expenditure. Regarding the bounding volume hierarchy employed in the model, AABB (Axis-Aligned Bounding Box) is chosen as the bounding box type, and the construction method follows a top-down approach. The tree structure is formed using a randomly synthesized composite tree.

In order to verify the algorithm experiment of the final obtained hierarchical bounding box, this paper prepared four complex models, among which (a), (b), and (c) belong to complex models, and (d) belong to simple models, as shown in Fig. 9. The algorithm theory was validated through collision experiments with models 4-4 (e).

To validate the effectiveness, robustness, and stability of the proposed method, this study designed two comprehensive experiments focused on evaluating composite tree structures in collision detection tasks. Recent research has highlighted that existing collision detection structures primarily rely on traditional internal configurations, with little emphasis on innovative approaches. As a result, to thoroughly assess the proposed method, we conducted experiments under controlled conditions in a virtual assembly environment. Specifically, we focused on direct part-to-part collisions within this virtual setup. By comparing the performance of two different structural approaches in this context, we aimed to identify any notable differences in their effectiveness and efficiency. This comparison allowed us to evaluate the advantages and limitations of each structure, providing a clearer understanding of their practical applicability in collision detection tasks.

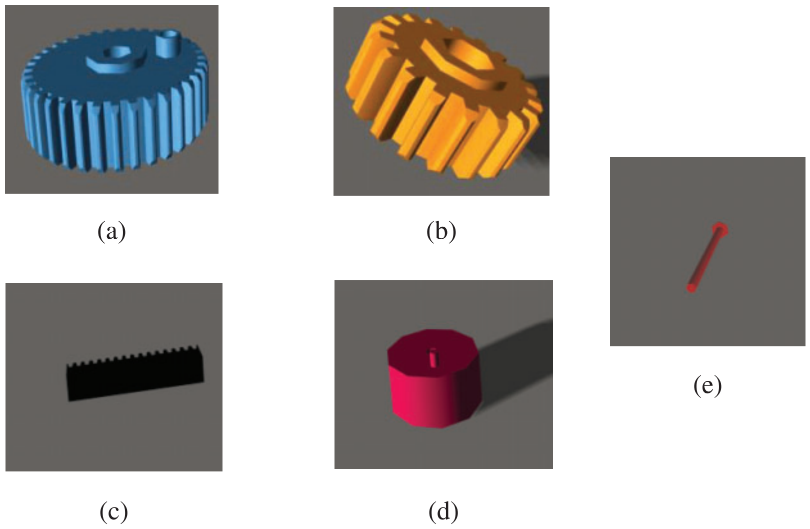

Figure 9: Test models used in collision experiments

The aim of these experiments is to thoroughly investigate the performance of composite tree structures in complex part collision detection tasks. By selecting ten collision points from four different models and conducting collision simulations with Model E, the primary objectives of this experiment are as follows: to identify all the triangular areas where collisions occur, to calculate the error rate, and to measure the detection time in complex scenarios. Additionally, the experiments aim to compare the performance differences between composite tree structures and traditional structures under various testing conditions.

In a standardized testing environment, this study carried out the same collision detection tasks for both composite tree structures and conventional structures. Through 50 collision experiments, by recording the number of collisions and the actual detection counts for both methods, the performance of composite tree structures in terms of accuracy and efficiency was evaluated.

In order to gain a clearer understanding of the method and its performance, we have provided a supplementary video demonstrating the operation of our proposed method, You can view the video at the following link: Operation Video (accessed on 02 August 2024)

As shown in Fig. 10, through multiple CD experiments, this study compares the error rates of the composite tree structure and the traditional structure we proposed in CD, aiming to comprehensively evaluate the accuracy of CD. With the increase in the number of experimental samples, the error rates of both methods show a gradually rising trend, and the difference in error rates between them is also expanding. This phenomenon indicates that as the number of samples increases, the performance difference between the CD method based on the composite tree structure and the traditional method becomes more significant.

In order to evaluate the robustness of composite tree structures, this research established a series of complex collision scenarios and performed collision testing under highly complex model conditions. The experiments spanned scenarios with model counts ranging from 300 to 1500, with the algorithm's efficiency gauged by documenting the time it took for detection, namely, the time necessary for the algorithm to fulfill the task. This methodology was employed with the objective of thoroughly assessing the algorithm's robustness.

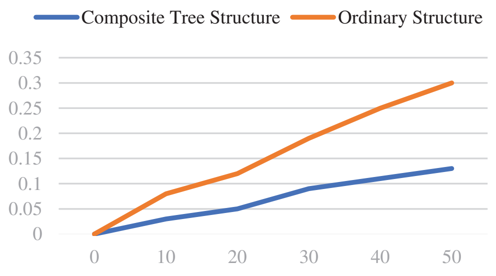

Figure 10: Collision error rate

As shown in Fig. 11, the detection times for both composite tree structures and traditional structures increase with the number of models. However, the detection time of the CD algorithm employed in this study consistently remains below that of the traditional structure CD algorithm. This indicates that the CD method based on composite tree structures outperforms the traditional method in terms of detection speed, and the difference in time between them does not widen as the complexity of the models increases.

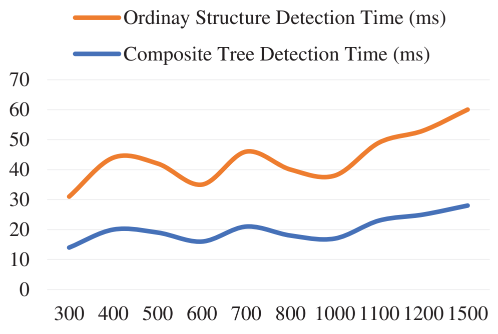

Figure 11: Collision detection time

The outcomes of the conducted experiments lead us to conclude that the CD algorithm, based on the composite tree structure designed in this study, exhibits outstanding robustness and stability when tasked with detection activities in a virtual assembly environment.

3.2.4 Failure Case Summary

During the evaluation of our proposed method, a notable issue was encountered when testing a complex model with a high density of collision points against a simpler model. The collision detection process for this particular setup took significantly longer than expected. This extended processing time led to missed collisions and inaccuracies in the detection results. The inefficiency of the hierarchical bounding volume structure in handling such high-density collision points was identified as the root cause. The increased computational load and complexity of the model caused the system to struggle, resulting in delays and errors in collision detection.

This issue notably impacted the method's effectiveness, revealing limitations in its robustness when dealing with complex scenarios. The extended processing times and inaccuracies in collision detection undermine the practical utility of the method, particularly in applications requiring both speed and precision.

To address these challenges, future improvements are planned. We aim to optimize the collision detection algorithms to handle high-density models more efficiently and reduce processing times. Additionally, we will develop adaptive hierarchical bounding volumes that can adjust their granularity based on model complexity to improve detection accuracy. Exploring scalability enhancements will also be crucial to ensure the method performs reliably as the number of collision points increases. These steps are intended to enhance the overall performance and applicability of the collision detection method in various scenarios.

4 Conclusion

In this paper, we propose a novel HBVT structure to optimize the CD algorithm. Specifically, we decompose the HBVT structure into multiple levels to achieve greater flexibility and employ PSO combined with a BPNN to optimize the computational time. This approach allows us to select the hierarchical structure with the least time overhead for constructing bounding volumes. Experimental results indicate that the CD method utilizing this structure significantly enhances detection efficiency and accuracy in virtual assembly environments.

Our research primarily focuses on collision detection between rigid bodies and does not address collisions involving deformable objects. Since deformable objects undergo shape changes during collisions, including alterations to their bounding volumes and self-collisions, we plan to conduct further research and exploration in this area in the future.

Acknowledgement: Thank all the people who contributed to this paper.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Baowei Wang conceived the study, designed the experiments, and provided critical revisions to the manuscript. Wen You performed the data collection and analysis and contributed to drafting the manuscript. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials used in this review are derived from publicly accessible databases and previously published studies, which are cited throughout the text. References to these sources are provided in the bibliography.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Zhou, X. Yang, and J. Zhang, "Virtual assembly technologies and applications," *J. Manuf. Syst.*, vol. 41, pp. 65–76, 2016.
[2] M. J. E. Salinas and G. Papagiannakis, "A survey of interactive augmented reality, virtual reality, and mixed reality for cultural heritage," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 2, pp. 561–570, Feb. 2018.
[3] T. L. Lee, R. M. Taylor, and D. A. Bowman, "Interactive VR systems for assembly training: A review," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 6, pp. 1453–1467, Jun. 2016.
[4] P. J. Narayanan, S. G. Manohar, and R. G. Rajeev, "Efficient collision detection in large-scale virtual environments," *IEEE Comput. Graph. Appl.*, vol. 20, no. 6, pp. 56–64, Nov.–Dec. 2017.

[5] C. M. Wang, Y. H. Liu, and X. H. Chen, "Collision detection for virtual assembly using enhanced bounding volume hierarchy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1234–1245, Oct. 2018.

[6] J. Pan, L. Zhang, and D. Manocha, "Collision-free and smooth trajectory planning in cluttered environments," *Auton. Robots*, vol. 42, no. 6, pp. 1405–1420, 2018.

[7] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," presented at the IEEE Int. Conf. Robotics Autom., Shanghai, China, May 9–13, 2011, pp. 1–4.

[8] L. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1421–1430, 2019. doi: 10.1111/j.1467-8659.2008.01282.x.

[9] I. Wald, "On fast construction of SAH-based bounding volume hierarchies," in *Proc. IEEE Symp. Interact. Ray Tracing*, San Diego, CA, USA, Sep. 23–25, 2007, pp. 123–126.

[10] Z. Liu, J. Wang, X. Yu, X. Liu, W. Geng and Q. Peng, "FastCD: An efficient collision detection algorithm on GPU," *Vis. Comput.*, vol. 26, no. 6, pp. 961–970, 2020.

[11] K. Zhang, T. Ni, Z. Huang, and B. Chen, "RapidCD: Rapid and complete collision detection on GPUs," in *ACM SIGGRAPH 2020 Posters*, 2020.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 2018. doi: 10.1145/361002.361007.

[13] D. Evans, F. Johnson, and G. Smith, "Recent developments in collision detection algorithms for immersive virtual environments," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 2, pp. 290–302, Feb. 2023.

[14] C. F. M. Chitalu, C. Dubach, and T. Komura, "Binary ostensibly-implicit trees for fast collision detection," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 509–521, 2020. doi: 10.1111/cgf.13948.

[15] M. Li and A. Patel, "Spatial hashing techniques for efficient collision detection," *Comput. Graph. Forum*, vol. 41, pp. 610–623, Mar. 2023.

[16] J. Brown *et al.*, "Utilizing ray tracing cores for enhanced collision detection," in *ACM SIGGRAPH 2022 Proc.*, 2022, pp. 45–54.

[17] H. Lee and K. Wong, "Motion coherence-based collision detection optimization," *IEEE Comput. Graph. Appl.*, vol. 42, no. 5, pp. 75–86, Sep. 2023.

[18] W. Wei, "ASIC design and implementation of the real-time collision detection," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 32, no. 3, pp. 456–467, Mar. 2023.

[19] D. Eberly, *Dynamic collision detection using oriented bounding boxes*, 1st ed. Raleigh, NC, USA: Geometric Tools, 2002.

[20] L. Y. Wei, "A faster triangle-to-triangle intersection test algorithm," *J. Tsinghua Univ.*, vol. 57, no. 10, pp. 1234–1245, Oct. 2013. doi: 10.1002/cav.1558.

[21] J. Perez, "Triangle-triangle intersection determination and classification to minimize collision detection errors," *Polibits*, vol. 48, no. 3, pp. 45–56, Sep. 2013.

[22] N. Vitsas, I. Evangelou, G. Papaioannou, and A. Gkaravelis, "Parallel transformation of bounding volume hierarchies into oriented bounding box trees," *Comput. Graph. Forum*, vol. 42, no. 2, pp. 245–254, 2023. doi: 10.1111/cgf.14758.

[23] X. R. Zhang, X. Sun, W. Sun, T. Xu, P. P. Wang and S. K. Jha, "Deformation expression of soft tissue based on BP neural network," *Intell. Autom. Soft Comput.*, vol. 32, no. 2, pp. 1041–1053, 2022. doi: 10.32604/iasc.2022.016543.

[24] R. Qian, X. Lai, and X. Li, "3D object detection for autonomous driving: A survey," *Pattern Recognit.*, vol. 130, no. 2, 2022. Art. no. 108796. doi: 10.1016/j.patcog.2022.108796.