New collision detection method for simulating virtual plant populations

Weilong Ding[1*], Zangxin Wan[1], Yan Xu[1], Nelson Max[2]

(1. *College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310026, China*;
2. *Department of Computer Science, University of California, Davis, 95616, USA*)

Abstract: Detecting and resolving the collision of organs between different plants or the collision of different organs of a single plant are key issues in the realistic construction of a virtual plant population. A suitable collision detection scheme is necessary to prevent a reduction in realism caused by organ penetration. A mixed bounding volume tree construction scheme based on the growth characteristics of tomato plants is proposed in this paper, and the construction mode of the bounding at all levels is simplified by using a digital tomato model. Using a parallel GPU approach, we designed a tomato plant population collision detection program with CUDA acceleration. The proposed method reduces the total collision detection time by 92%-96%.

Keywords: plant simulation, collision detection, bounding volume, GPU processing
DOI: 10.25165/j.ijabe.20191206.4888

Citation: Ding W L, Wan Z X, Xu Y, Max N. New collision detection method for simulating virtual plant populations. Int J Agric & Biol Eng, 2019; 12(6): 156–151.

1 Introduction

Plant populations are common in nature and one of the basic elements of virtual scenes. Realistic simulation of plant population scenes has a great practical value in many areas, such as landscape planning, crop cultivation, community design, and 3D games. An accurate virtual plant population model can provide a platform for research on many agricultural science problems[1], such as optimization of crop spacing and density and cultivation of an ideal plant type[2]. Modeling, simulation, and visualization of plant populations have become a popular research subject in computer science, applied mathematics, botany, biology, and ecology. Establishing a realistic 3D model for this type of scene is difficult because of the complex morphological structure of plant populations.

Typical organs considered in 3D models are stems, leaves, flowers, and fruits. In a plant population, the geometric organs between adjacent plants often exhibit contact or collision due to the direction of spatial growth and the distance between them. Contact or collision can occur among leaflets or between leaflets and petioles even within one plant. In the simulation of a plant population, if collision detection is not implemented, the established model will involve organ penetration, which seriously affects realism. Therefore, detecting and resolving the collision of organs between different plants or the collision of different organs of a single plant are key issues in the realistic construction of a virtual plant population. Furthermore, the efficiency of a collision detection algorithm is a serious issue due to a large number of organs in a plant population.

Collision detection has been widely used in many fields, including robotics, object modeling, animation, and video games[1]. Inexpensive methods must be used to replace the intersection test among facets that make up an object to determine the location, angle, and collision time among the objects in a complex and diverse scene. Optimization of collision detection that is efficient in different stages of different plant species was performed under different scenarios. Several aspects of the current studies must be improved. First, several traditional collision detection optimization algorithms are not optimized for plant scenes, and they cannot be directly applied to plant population collision detection. Second, many collision detection algorithms for large-scale scenes focused on the collisions among different plants, but few related with the internal organs in a plant. Third, several current plant collision algorithms focus on uniform leaf morphology, such as long and sword leaves, and may not be suitable for other types of leaves and plant organs.

The contribution of this article is an optimization algorithm for the bounding volume tree of tomato plants, based on the morphological characteristics of tomato plants and using parallel computing technology. A hybrid bounding volume structure based on the differences between the structure of tomato plants and that of other plants was constructed to suit the characteristics of tomato plants. An optimized collision detection process was designed according to the morphological characteristics of tomato for different hierarchical bounding volumes. A plant structure tree was also built and applied to internal organ collision culling of tomato by using the structure of tomato plant models. The efficiency of the algorithm was improved via CUDA parallel acceleration.

2 Related works

At present, commonly used collision detection algorithms include geometric, spatial subdivision, and hierarchical bounding volume approaches. As a type of scene-management algorithm, the spatial subdivision can limit the number of pair-wise primitive collision detection tests to several small ranges by dividing a scene into multiple sub-grids. Common spatial subdivision data structures include uniform grids, quad-trees, octrees[2], k-d trees[3],

Received date: 2018-12-27 **Accepted date:** 2019-10-19
Biographies: Zangxin Wan, Master student, research interest: virtual plant modeling, Email: 593735243@qq.com; **Yan Xu,** MS, research interest: virtual plant modeling, Email: 402625942@qq.com; **Nelson Max,** PhD, Professor, research interest: computer graphics, Email: max@cs.ucdavis.edu.
***Corresponding author: Weilong Ding,** PhD, Professor, research interest: virtual plant modeling. No.288, Liuhe Road, Liuxia Town, Hangzhou 310026, China. Tel: +86-571-85290527, Email: wlding@zjut.edu.cn.

and BSP trees. The objects in different sub-grids can usually be regarded as not colliding in a single time step. Moreover, the spatial subdivision method is easy to use with geometric or bounding volume methods. Therefore, many people have applied this method to N-body collision detection. For example, Zhou et al.[4] proposed a k-d tree algorithm based on GPU parallelism, constructed the k–d tree by using the breadth-first order, and applied it to the ray tracing algorithm. Avril et al.[5] designed a real-time collision detection algorithm for dynamic object groups for a multi-core GPU. The algorithm improves detection efficiency by reducing global memory access and simplifying the workload of a single thread. The collision detection algorithm in this work does not include the intersection test of triangles and polygons and can only run on a single computer. Wong et al.[6] proposed a GPU-based algorithm for spatial segmentation and collision detection. This method is based on the fusion of an octree and a grid and reduces the frequency of collision detection per frame by a layering test. It also provides a triangle-level collision detection scheme. Du et al.[7] divided a scene into uniform grids and implemented a distributed simulation of N-body collision detection on a super computer cluster. However, this method does not support collision detection between deformable objects.

The bounding volume method is one of the most common methods of collision detection, and it simplifies the intersection test calculation by bounding one or more objects with simple geometry. The most commonly used bounding volumes include balls, boxes, ellipsoids, and K-DOPs[1]. Objects with a complex morphological structure usually employ a bounding volume hierarchy (BVH) for collision detection. The selection of bounding volumes and the construction scheme of the BVH must be closely related to actual application scenarios to balance tightness, efficiency, storage consumption, and other factors, and extensive research has been conducted on this selection process. For example, Eloeet al.[8] optimized the bottom-up construction of a BVH algorithm and adopted the dual graph segmentation method to improve the quality of the constructed BVH. However, the accuracy and parallelism of the approach still require improvement. Kimet al.[9] proposed a hybrid, parallel, continuous collision detection (HPCCD) algorithm, in which the CPU performs BVH traversal and elimination, and the GPU performs simplified object collision detection. The scheme improves the collision detection efficiency for complex deformed objects. Sagardia et al.[10] proposed a collision detection and resolution algorithm based on voxels and a point cloud surface. The algorithm was implemented using the open-source physics engine Bullet and achieved better results than Bullet when dealing with the same scene. Qian et al.[11] proposed an adaptive spherical collision detection algorithm for deformable objects and utilized object-based collision processing using position-based dynamics. Ganestam et al.[12] proposed a top-down method based on an improved-sweep surface area heuristic (SAH) algorithm for quickly building a plant model bounding volume tree and applying it to ray tracing. Schwesinger et al.[13] applied bounding box-based collision detection technology to motion planning and proposed a fast collision detection method. Kim et al.[14] detected the intersection of tree branches with the line swept spheres (LSS) bounding volume tree and applied an interactive tree-modeling method by dealing with the collisions. However, this method only supports limited tree topologies. Owens et al.[15] used the axis-aligned bounding box (AABB) to voxelize flower patches in flower modeling and perform flower collision detection and deformation. However, the collision detection and deformation simulation process caused problems when a collision occurred in the initial stage of flower modeling.

3 Design of a tomato plant collision detection algorithm

3.1 Analysis

3.1.1 Tomato plant characteristics

Compared with the leaf morphology of wheat, rice, corn, and other crops, the leaf morphology of tomato plants has unique characteristics, which bring difficulties to the design of a collision detection scheme. For instance, rice leaves are slender, so abounding volume tree can be directly constructed with leaves, and collision detection can be performed[18]. However, tomato plants have compound leaves (see Figure 1), which mean both sides of the petiole and tip grow with pinnate leaflets. Thus, constructing a tight bounding volume for an entire leaf is difficult. Moreover, the growth of the pinnate leaflets of tomato plants is free, so simplifying the construction and cross-testing the bounding volume based on the leaves' growth pattern is challenging.

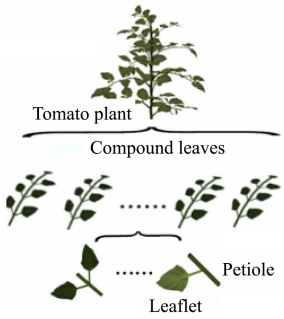


Figure 1 Schematic diagram of the structure tree of a tomato plant

Nevertheless, several physiological characteristics of tomato plants contribute to the design of our collision detection scheme. For example, the relatively fixed phyllotaxy and helical growth pattern of tomato compound leaves make the numbering of compound leaves and the design of a space geometry collision culling schemes easy. In the case of suitable plant spacing, the most common type of leaf petiole crossover is the crossover of the tip. Therefore, in a compound leaf structure, the farther an organ is from the main petiole of the plant, the more likely it is to collide with other plants. On the contrary, the closer an organ is to the center of the plant, the more likely it is to collide with another leaf petiole of the same plant.

3.1.2 Model division and related assumptions

Before collision detection, the tomato plant model was divided into three scales according to its morphological structure. The top is the plant level. After receiving the tomato model set, the system divides it into objects in which tomato plants are used as units and determines if the objects intersect. The detection of tomato internal organ collision is restricted inside a tomato plant. The middle scale is the main petiole and the compound leaf structure, which includes branches. The leaflets grow in the branches and tip. When plants collide, the collision culling scheme is used to identify the compound leaf pairs that have collided. In this scale, the compound leaf subscripts are saved in the order of growth from the bottom to the top of the main stem. The lowest scale is the leaflets and branch segments of the plants. The collision position is culled to a specific leaf or petiole. In this scale, the subscripts of leaflets and branch segments are saved in the order of growth from the inner to the outer part in a compound

leaf structure.

3.2 Building the BVT of tomato plant populations

Our hybrid bounding volume hierarchy was determined based on an existing digital model of tomato plant geometry. Then a collision detection optimization scheme was developed according to the characteristics of the bounding volume hierarchy population.

3.2.1 Pretreatment of the tomato plant model

Each tomato plant contains several compound leaf structures, see Figure 1. A mixed bounding volume tree (MBVT) is constructed for each compound leaf. An organ bounding volume needs to be constructed in turn for a single compound leaf structure with leaflets, other organs, and the petiole segment.

3.2.2 Construction scheme of each scale of bounding volume

On the basis of the analysis of the morphological characteristics of the tomato compound leaf structure, a bottom-up approach was selected to construct the MBVT for the compound leaves of the tomato plant.

1) Simplified OBB construction of the leaflet/leaflet petiole

For each leaflet petiole and leaflet, a simplified oriented bounding box (OBB) was used for collision detection.

An OBB is defined as the smallest rectangular solid with an unrestricted orientation that contains the target object. Thus, the most important task in the construction of an OBB is the acquisition of the directional axes for the orientation. Traditionally, direction axes were obtained by calculating the covariance matrix of the vertex data of the target object, which is computationally intensive and time consuming. The method to construct tomato plants in our system involved transforming the pre-defined organ model through a series of parameters. All leaf petiole segments and leaf organs were constructed based on the original coordinates, and they were assembled on tomato plants through translating, rotating, and scaling operations. Considering that OBB can also be regarded as a cuboid formed with the maximum and minimum projection coordinates, we can obtain the vertex sequence of each underlying organ object before the transformation and the transformation matrix that is assembled on the plant. With the transformation matrix, the translated local coordinate system and coordinate axes can be calculated and regarded as simplified OBB direction axes. Then, the projection range of the projection coordinate of the object in the original coordinate system can be calculated according to the initial vertex sequence of the target object. A simplified OBB for the target object is produced by combining the two.

2) BVT construction for the compound leaf structure in the middle scale

For the compound leaf structure in the middle scale, the traditional bounding volume construction scheme uses an axis-aligned bounding box (AABB), which is a rectangular solid oriented to fixed axes. This structure is relatively simple. During collision detection, three standard axes are tested as separate axes, which is convenient. However, if the target object is transformed, then the AABB needs to be completely reconstructed, which means updating is expensive. The leaf collision detection (LCD) bounding volume (Song, 2013) used in this work can be considered a simplified OBB. The LCD uses the vertical axis as a fixed direction, and the main local coordinate system orientation calculation and collision detection separation axis test are concentrated on the horizontal plane. The growth direction of the compound leaf petiole can be regarded as another of the orientation axes. Then the last orientation axis can be calculated from the first two orientation axes. OBB needs 15 separate axis tests while LCD needs only 5 during the collision test as one of the axes is fixed (Song, 2013). Similar to an OBB, the LCD has an advantage of low update operation costs. LCD performs better than AABB in terms of tightness of the elongated blade or tomato compound leaf structure.

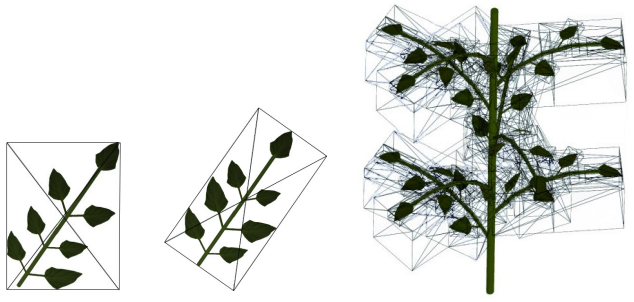
We constructed three groups of bounding volumes, namely, AABB, OBB, and LCD, for the compound leaves of the tomato plant. Table1 shows a comparison of the average time consumption and the volume ratio of bounding boxes. In Table 1, the volume ratio means (volume in AABB or LCD)/(volume in OBB). From it, we can see that AABB and LCD performed better in construction, and LCD and OBB performed better in bounding volume update operations by transferring the compound leaf model. In the intersection test, the time consumed by LCD was less than that by OBB. Given that a tomato compound leaf structure extends horizontally outward, the tightness gap between LCD and OBB is not too large.

Table 1 Comparison among three methods applied to tomato plants

Method	Average time-consumption of bounding volume construction/μs	Time-consumption of bounding volume intersection test/μs	Time-consumption of bounding volume update/μs	Volume ratio based on OBB
AABB	1924	50	2400	1.08-1.72
LCD	2078	200	7.9	1.08
OBB	2988	550	7.5	1.0

After obtaining the orientation axes of the LCD bounding volume for a single compound leaf structure, the model was transformed into a local coordinate system. A bottom-up plan was used to construct the LCD of the parent node for the underlying organ model. The parent bounding volume node of the organ model was constructed along the leaf petiole according to the growth characteristics of the leaflets in the compound leaf structure. The LCD-OBB bounding volume tree was then constructed. The maximum length of each branch in the entire tomato plant can be regarded as a basis for collision detection. If the distance between two tomato plants is less than the sum of their maximum outward extension distance, then the two plants may collide.

The constructed tomato plant MBVT is shown in Figure 2. First, the OBB is constructed on the leaf segments and blades of all the compound leaf structures, and then the LCD bounding volume tree is constructed on the upper scales of these OBB envelops to simplify the collision culling process.



a. AABB b. LCD c. MBVT

Figure 2 Schematic diagram of the MBVT of a tomato plant

3.3 Collision detection of plant population

The model information for the tomato plants is stored in a 1D array and sorted by position, and all potential plant collisions are preserved in a table. A plant can collide with up to eight plants around it because collision mostly occurs between neighbor plants. On this basis, we traversed the plants in the scene by subscript order and added to a table of potential plant collisions whenever the current plant may collide with a neighboring plant with a higher subscript.

All potential plant collisions were identified after completing the traversal. For each potential collision in the table, we traversed the compound leaf structures of both plants and performed collision detection for their MBVTs in pairs. In view of branch collision among the plants, collision is more likely to occur in the tip-end structure than in a branch of the segment. Therefore, collision detection of the tip-bounding volume should be performed with higher priority. The full method is as follows:

Step 1: Enter a tomato plant's bounding volume dataset and traverse it to obtain 3-8 plants (denoted as *Around* (*i*)) around tomato *i*.

Step 2: Perform a collision test for each pair of plant *i* and plant from *Around*(*i*). Add a pair of indexes to the table of plant collisions if it passes the test.

Step 3: Repeat steps 1 and 2 until the entire bounding volume dataset is traversed.

Step 4: Make another table of compound leaf intersection pairs. For two plants whose numbers of compound leaves are *m* and *n*, $m \times n$ tests are needed.

Step 5: Apply leaf by leaf OBB collision tests for all leaves and petioles in each pair and save the collision test results.

Step 6: Repeat steps 4 and 5 until collision detection for all plant pairs in the table is completed.

3.4 Internal organ collision detection for a tomato plant

In plant population modeling, in addition to the collision between plants, the internal organs of a single plant, such as branches and leaflets, cross other branches for many reasons, which affect the realism of the plant model. Thus, a method for internal organ collision detection in a single plant is also needed.

A plant structure tree is a tree structure that describes the construction of a digital tomato model and is used for the collision culling of internal organs of a single tomato plant. By regarding all of the lower organs as leaf nodes, we constructed the structure tree of a single plant, as shown in Figure1. First, the petioles and the leaflets were grown on the petioles were combined to form a segment-based leaflet group. By default, no collision detection was performed inside the group. Second, all of the petioles of each compound leaf structure were combined into a leaf. Plant internal collision detection was carried out according to the leaf group. Finally, the tomato structure tree was created with all the leaves.

Step 1: The tomato plant bounding volume dataset was entered and traversed. For each plant, collision detection was performed between each group of compound leaves and the bounding volume tree of other compound leaves according to the leaf number of the compound leaves. To prevent duplicate tests, each group of compound leaf structures was only collision-tested with a larger-numbered compound leaf structure. Therefore, for a plant with *m* compound leaves, a total of $m\times(m-1)/2$ tests were needed.

Step 2: The collision test of each pair of bounding volume trees was a synchronous recursive downward intersection test from the inner sub-bounding volume of the compound leaf structure to the underlying OBB bounding volume. The returning collision detection results were stored.

Step 3: Within each group of compound leaf structures, all of the underlying OBB bounding volumes were traversed by the order of subscripts for the collision test between this bounding volume and the other underlying bounding volume.

Step 4: For each bounding body in step 3, we obtained the node position of the contained object in the plant structure tree and removed the parent node of the current node. Then, we performed collision-testing on the OBB bounding volume of the other objects. The result was saved.

Step 5: Steps 1 to 4 are repeated until all plant internal organ collision detection results are returned.

4 Acceleration of the collision detection algorithm for tomato plant population

The efficiency of collision detection of tomato plant population was improved with the proposed collision detection optimization scheme. However, the time consumed for collision detection in large-scale scenes is too large, which seriously affects the efficiency of realistic rendering and dynamic simulation of tomato plants. Therefore, improving the collision detection efficiency for large-scale plant populations through a parallelization scheme is of practical significance. This work used a GPU-based parallelization scheme implemented by CUDA to accelerate the collision detection process.

4.1 Introduction to CUDA

CUDA is a general-purpose computing architecture from NVIDIA that allows developers to write programs that work on the CPU and GPU in an extended C language. The CUDA architecture abstracts the GPU into a three-tier structure of thread grids, thread blocks, and threads. The kernel functions written by developers run highly parallel on these threads. The CUDA programming model includes the CPU side, namely, the host, and the GPU side, namely, the device. The host side calls the traditional C/C++ API, and the device side calls one of CUDA's parallel programs, which is called a kernel, and reads the data in the video memory by calling the thread according to the index of block and thread.

4.2 Analysis of algorithms

The following results were obtained from the analysis of the plant population collision detection algorithm in Section 3.3: (1) Collision detection between each pair of compound leaf structures is independent and can be mapped to different threads for parallelized processing; (2) In the internal collision detection of the compound leaf structure, the detection of child nodes of MBVT depends on the collision detection result of the parent node. Thus, dividing and parallelizing are difficult; (3) The process of obtaining a potential collision pair dependent on collision detection for compound the leaf structures inside a plant. The complexity of collision detection for compound leaf structures is much higher than that of collision culling between plants. Therefore, the process of obtaining potential plant collision pairs should be implemented by the CPU.

To verify the correctness of the analysis of collision detection parallelism, we conducted a set of experiments to measure the time consumption at various stages of tomato plant population collision detection. The experiments were performed on a Lenovo Y50 laptop (Intel Core i7-4710HQ CPU at 2.50 GHz, 16 G of RAM, NVIDIA GeForce GTX860M graphics), and the results are shown in Table 2.

Table 2 Time-consumption under different population size of BVH construction and collision detection

Plant count in scene	200	400	600	800	1000
Total triangle counts	1.7M	3.5M	5.0M	6.5M	8.1M
Total time-consumption/ms	323	613	1001	1357	2159
Time-consumption of BVH construction/ms	43	54	94	125	184
Time-consumption percentage of BVH construction	13%	9%	9%	9 %	8%
Time-consumption of obtaining potential collision pair/ms	1	1	1	1.5	1.5
Time-consumption percentage of obtaining potential collision pair	1%	1%	1%	1%	1%
Time-consumption of BVH intersection testing/ms	279	557	907	1230	1973
Time-consumption percentage of BVH intersection testing	86%	90%	89%	90%	91%

Analysis of the data in the table revealed the following points. (1) As the scale of the plant population increases, the total duration of collision detection also increases. (2) The ratio of construction time of the bounding volumes to total time decreases with the increase in the scene scale. Given that the construction of the bounding volume is only executed once during the initialization of the scene, serial execution can be considered. (3) In the bounding volume intersection test, the time for obtaining the potential collision pair is always about 1 ms in different plant counts, which means parallelization is not needed. (4) Most of the collision detection time is used for the MBVT intersection test of the compound leaf structure, which is the focus of the parallel scheme.

4.3 Algorithm implementation

According to the analysis in Section 4.2, the following parallel program design was proposed using the design principles of a CUDA parallel program. (1) In the task allocation stage, collision detection of a single potential plant collision pair is regarded as the basic unit and performed by the GPU. The process of constructing the bounding volume and obtaining the potential collision pairs is implemented by the CPU. (2) In the thread mapping stage, the collision detection process is independent for each pair of tomato plants, so this process is mapped into a single thread block. Each thread in the thread block is responsible for processing the intersection test of a pair of compound leaf structures in these two plants. (3) In the GPU memory management stage, the entire MBVT data of a plant are stored in the global memory because they are too large and read by multiple thread blocks. Collision test results data, which are returned from all the threads, need to be transmitted back to the CPU for processing and stored in the global memory.

5 Experiments results and analysis

Interface and model rendering were implemented withUnity3D, and parallel collision detection was implemented with CUDAC++. Our computer hardware is Intel (R) Core i5 4590 with 2.66 GHz CPU, 8 GB memory, and NVIDIA GTX 970 graphics card.

The tomato digital model used in this experiment was from the L-system-based tomato plant modeling software developed by our laboratory[19]. To reduce the time consumption of scene rendering in large-scale plant populations, a simplified tomato plant architecture model was adopted. A parametric L-system was used to describe the topological structure of individual tomato plants. Vertex shader and fragment shader were respectively utilized to do the computing of graphical interpretation of generated strings and the calculation of texture, light, and color of different organs. The level of detail is employed to describe different simulation levels according to the distance between the plant position and the viewpoint. The rendering result of a large field of tomato plants is shown in Figure 3, made up of regularly spaced differently rotated version of the same tomato plant.

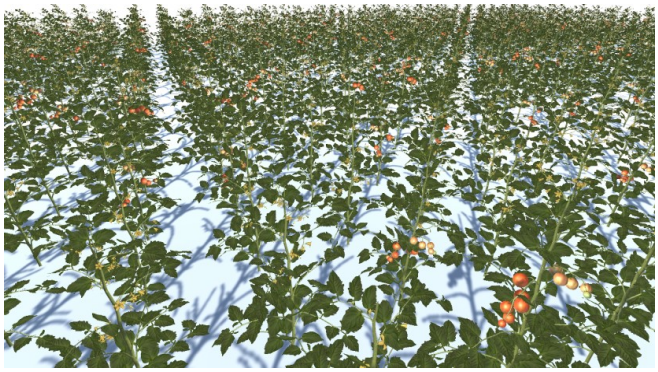


Figure 3 Rendering of tomato plant population

To verify the efficiency of our method, we compared the MBVT and AABB methods in sparse cases of 100 tomato plants. Neither of the two methods tested the collision between elements. The results are shown in Table 3.

Table 3 Comparison between our method and AABB Tree

Method	Average time-consumption of BVH construction/ms	Time-consumption of BVH intersection test/ms	Time-consumption of BVH update /ms
AABB	644.9	8.5	458.9
MBVT	28.5	45.5	8.2

As shown in Table 3, the MBVT method consumed much less time than the AABB tree in the construction phase because the copying and updating operations used when multiple rotated replicas of the same model appeared in the underlying construction of the hybrid bounding volume greatly reduced the computational cost.

In the intersection test stage, the collision detection of the OBB bounding volume in MBVT consumed more time than that of the AABB tree. However, after the scene update performed (such as movement, rotation and deformation of the plant models), the update cost of the MBVT is much smaller than the AABB tree. Therefore, this method is advantageous in large-scale motion scenarios.

To verify the efficiency of CUDA parallel acceleration, we performed an experiment by setting the tomato plant row spacing to 14 with 2000, 4000, 6000, 8000, and 10 000 plants. The results are shown in Table 4. The collision detection of 10 000 tomato plants on the CPU consumed more than 50 seconds. In this case, the parallel acceleration collision detection program based on CUDA consumed about 2 seconds. The GPU-based solutions saved 92%-96% of collision detection time at different scales, which shows that the GPU-based parallel acceleration scheme effectively enhances collision detection efficiency in large-scale scenarios.

The collision detection in a simulation of a plant population has been studied from various aspects. Summarizing, a set of MBVT construction schemes was proposed, and a GPU was utilized to accelerate the calculation. The scheme is based on bounding volumes, which are commonly used in collision detection. A tomato plant was divided into different scales according to its structure, and then bounding volumes were built for each scale.

Several experiments were performed to choose bounding volume for each scale and test performance. For a concrete scale, a variety of distinct bounding volumes were tested. Factors such as calculation cost, update cost, and tightness were considered. In performance tests, different sizes of plant populations were assembled under the same conditions to construct the MBVT and calculate collisions to obtain the efficiency of the method. Test results proved that the proposed method is more efficient than the AABB method. Based on the above experimental results, we suggest that the proposed method can accelerate the collision detection of plant populations.

Table 4 Comparison between CPU and GPU

Plants number	Time-consumption of CPU/s	Time-consumption of GPU/s
2000	5.14	0.41
4000	13.35	0.77
6000	23.34	1.15
8000	36.47	1.55
10000	52.06	1.93

However, there are still some shortcomings in this research due to the limitations of experimental conditions. First, the tomato model is not authentic enough. In this paper, a simplified tomato plant model was utilized. Although the compound leaf structure was constructed, other physiological characteristics were not considered yet, which made the results less convincing. Second, this article only tested the method on tomato, and there was no experiment on other plants, which need further study. Also, this paper only studied the collision among one kind of plant but did not consider factors such as growth, environment, and mixed plants. Finally, this paper only uses the bounding volume and GPU acceleration for completing collision detection but did not consider scene management algorithms that are often used simultaneously with the bounding volume such as a spatial subdivision.

6 Conclusions

A set of optimization schemes for collision detection in a tomato plant population was proposed. First, a suitable LCD-OBB hybrid tree structure was selected based on the characteristics of the tomato model. Second, by using model transformation data, the structure of the bounding body at all levels was simplified. Third, the possibility of parallelism in plant collision detection was analyzed, and CUDA was used to further improve efficiency.

Thus far, the simplification scheme of the bounding volume and collision detection schemes are only applicable to plant models from parametric modeling, such as from an L-system. For other types of digital models, sufficient morphological features need to be obtained, or the model should be processed to obtain corresponding hierarchical models. Therefore, the next step to be carried out is to further extend this method to many types of models. Plant morphology becomes increasingly diversified in non-ideal environments. Therefore, further studies are needed to deal with plant collisions in the case of twisting, bending, and breaking of leaf petioles or leaflets.

Acknowledgements

This work was supported by the National Natural Science Foundations of China (61571400, 31471416) and Natural Science Foundations of Zhejiang Province (LY18C130012). The authors are grateful to the anonymous reviewers whose comments helped to improve this paper.

[References]

[1] Dinas S, Bañón J M. A literature review of bounding volumes hierarchy focused on collision detection. Ingeniería y Competitividad, 2015; 17(1): 49–62.

[2] Zhang W, Xie Q, Zhong J, Liu J, Hao Q, Guo G. Acceleration algorithm in ray tracing by the octree neighbor finding. Journal of Graphics, 2015; 36(3): 339–344.

[3] Hapala M, Havran V. Review Kd‐tree Traversal Algorithms for Ray Tracing. Computer Graphics Forum, 2015; 30(1): 199–213.

[4] Zhou K, Hou Q, Wang R, Guo B. Real-time KD-tree construction on graphics hardware. ACM Transactions on Graphics, 2008; 27(5): 126.

[5] Avril Q, Gouranton V, Arnaldi B. Fast collision culling in large-scale environments using GPU mapping function. Eurographics Symposium on Parallel Graphics and Visualization, Euro-graphics Association, 2012; pp.71–80.

[6] Wong T H, Leach G, Zambetta F. An adaptive octree grid for GPU-based collision detection of deformable objects. The Visual Computer, 2014; 30(6): 729–738.

[7] Du P, Liu E S, Suzumura T. Parallel continuous collision detection for high-performance GPU cluster. ACM SIGGRAPH Symposium on Interactive 3d Graphics and Games. ACM, 2017.

[8] Eloe N W, Steurer J A, Leopold J L, Sabharwal C L. Dual graph partitioning for Bottom-Up BVH construction. Journal of Visual Languages & Computing, 2014; 25(6): 764–771.

[9] Kim D, Heo J P, Huh J, Kim J, Yoon S E. HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs. Computer Graphics Forum, 2010; 28(7): 1791–1800.

[10] Sagardia M, Stouraitis T, e Silva J L. A new fast and robust collision detection and force computation algorithm applied to the physics engine bullet: method, integration, and evaluation. Conference & Exhibition of the European Association of Virtual & Augmented Reality, 2014.

[11] Qian K, Yang X, Zhang J, Wang M. An adaptive spherical collision detection and resolution method for deformable object simulation. International Conference on Computer-Aided Design and Computer. Graphics IEEE, 2015; pp.8–17.

[12] Ganestam P. Bonsai: rapid bounding volume hierarchy generation using mini trees. Journal of Computer Graphics Techniques, 2015; 4(3): 23–42.

[13] Schwesinger U, Siegwart R, Furgale P. Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners. IEEE International Conference on Robotics and Automation, 2015; pp.63–68.

[14] Kim Y J, Woo J H, Kim M S, Elber G. Interactive tree modeling and deformation with collision detection and avoidance. Computer Animation & Virtual Worlds, 2015; 26(3-4): 423–432.

[15] Owens A, Cieslak M, Hart J, Classen-Bockhoff R, Prusinkiewicz P. Modeling dense inflorescences. ACM Transactions on Graphics, 2016; 35(4): 136.

[16] Song W G. Visualization technology of wheat growth. MS thesis. Nanjing Agricultural University, 2013.

[17] Cheng J, Grossman M, Mckercher T. Professional CUDA C Programming. Wrox Press Ltd, 2014.

[18] Zhang X B, Hu B, Tang L, Wu Y L, Jiang H Y. Fast collision detection for rice leaf population based on improved bounded box tree and GPU. Transactions of the CSAE, 2018; 34(1): 171–177. (in Chinese)

[19] Ding W L, Jin H J, Cheng Z J, Chen Q. A Visualization System for Tomato Plant Modeling. Proceedings of 8th International Conference Computer Graphics, Imaging and Visualization, 2011; pp.160–165.