

Collision detection strategies for virtual construction simulation

Kuan-Chen Lai, Shih-Chung Kang*

Computer-Aided Engineering (CAE) Group, Department of Civil Engineering, National Taiwan University, Taipei, Taiwan

ARTICLE INFO

Article history:
Accepted 17 February 2009

Keyword:
Construction crane
Collision detection
Virtual reality
Construction simulation
Bounding volume box

ABSTRACT

This research aims at developing a simple and efficient collision detection method to support the rendering of a virtual construction scenario in real time. To expedite the collision detection algorithm, we approximated construction machineries and structural elements on a construction site by using spheres and cylinders. By modeling the objects using outer boundaries, the computational cost for collision detection can be significantly reduced. Using the outer boundary also provides the benefit of ensuring a conservative result (i.e. towards the safer side). VC-COLLIDE, the collision detection algorithm developed in this research, has also been presented. VC-COLLIDE has been implemented and tested by using three typical construction scenarios: small building scenarios (683 objects), large building scenarios (2143 objects) and tall building scenarios (2612 objects). The test results indicate that VC-COLLIDE can complete all the collision checks within 1/20th of a second, the upper bound of real-time refresh time in the three testing environments. We also compared VC-COLLIDE with the collision detection function in Open Dynamic Engine (ODE), a widely used physics engine for real-time visualization. Because VC-COLLIDE is designed specifically for using in a virtual construction scenario, its computational performance is significantly better than ODE.

Crown Copyright © 2009 Published by Elsevier B.V. All rights reserved.

1. Introduction

Virtual construction is the process of using computational methods to model, simulate, and visualize construction scenarios on computers [12]. Many successful applications have been recently presented worldwide, such as those by Wilkins and Barrett [13], Clayton et al. [3], and Kamat and Martinez [5]. Virtual construction allows stakeholders to envision the construction results, enables engineers to review construction problems early in the design phase, and allows constructors to manage the site more efficiently. Since virtual construction presents detailed construction processes visually, many construction problems, such as spatial conflicts and inefficient machine operations, can be identified and solved before the start of actual construction. Therefore, virtual construction is becoming an important management tool, especially in complicated or large-scaled projects [7].

Previous investigators working on related topics of virtual construction, such as Paulson et al. [10], Halpin and Riggs [4], and Martinez and Ioannou [9], pointed out that the computational cost of collision detection can be a great drawback in the simulation process. Similar to real construction sites, collisions between virtual construction machineries (e.g. cranes and excavators) and virtual structural elements (e.g. beams and columns) should be avoided. This means that the computer has to perform collision checks between all the moving parts of the machine and all the structural elements. We have to perform collision checks continuously to develop a real-time animated virtual

construction scenario. The required frame rates of real-time systems vary depending on the equipment. For a real-time system that is steering an oil tanker, a frame rate of 1 Hz may be sufficient. However, a rate of even 100 Hz may not be adequate for steering a guided missile in a computer game. The designer must choose a frame rate appropriate to the application's requirements. In this research, we tested scenarios on multiple computers and found that a frame rate of 20 Hz ensures that the quality of the animation rendering for virtual construction scenarios is sufficient for real-time viewing.

The challenge of performing collision checks is that the computational cost grows exponentially when the number of objects, including construction machinery and structural elements, increases in virtual construction. Simulation of simple construction scenarios may be solved and simulated in real-time. However, with the performance restriction of a computer, a complicated virtual construction scenario which contains a great number of structural elements may not be computed and visualized within 1/20th of a second using common collision detection packages. Therefore, it is necessary to develop an efficient collision detection method that can specifically support the real-time virtual construction system.

2. Existing collision detection methods

Collision detection methods have actually been in existence for years. They can be divided into two general groups. Methods in the first group can determine the collision status (i.e. whether objects have collided or not) quickly via multiple logic statements. The well-known axis-aligned boundary box (AABB) method [1] belongs to this group. The second group of collision detection methods can return

* Corresponding author. National Taiwan University, Department of Civil Engineering, Taipei (10617), Taiwan. Tel.: +886 2 3366 4346.
E-mail address: sckang@ntu.edu.tw (S.-C. Kang).

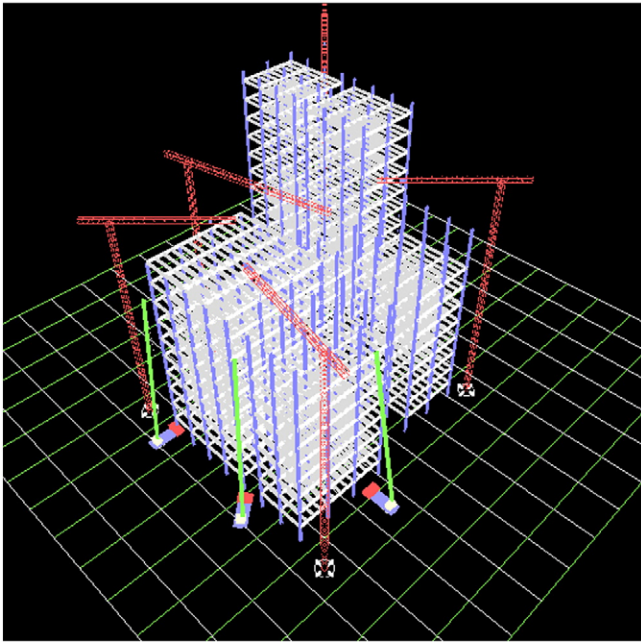


Fig. 1. Virtual construction scenario.

both the collision status and the distance to the nearest possible collision in the case of a collision-free status. Some broadly used collision detection libraries, such as the Open Dynamic Engine (ODE) [11], SOLID [2], and the algorithm developed by Lin and Canny [8] belong to this group.

Collision detection methods in the first group usually require less computation because they are not burdened by the need to compute the shortest distance between objects. However, they may not fulfill the needs of virtual construction. The free distance between the objects is critical for rendering virtual construction scenarios, especially when we need to simulate the detailed motions of virtual construction machines. Knowing the shortest possible collision distance can eliminate unnecessary collision checks within the collision-free region, significantly reducing the computational time required to search for collision-free paths.

Collision detection methods in the second group can always return the distance between objects. However, they require more computational resources and so may not be able to support real-time

computation for a virtual construction system. Since the efficiency of a collision detection computation greatly depends on the precision requirements, we propose the development of a collision detection method which supports the computation of collision detection in a real-time virtual construction system while at the same time providing distance calculations with sufficient accuracy.

3. Objectives

The objective of this research is to develop an algorithm which can be used to identify all undesirable conflicts that can occur among static and dynamic construction sources and allows the distance between objects in large dynamic 3D virtual construction scenarios to be determined. To achieve this objective, we would like to develop an algorithm to deal with the collision detection problems efficiently so that we can realize the real-time visualization for large construction scenarios. To support the automated simulation functions in virtual construction software such as iCrane [6], this collision detection algorithm needs to compute the collision-free distances. A collision detection algorithm which can return the free distance between objects can significantly improve the performance of motion planning methods. Because performance strategies differ between different construction scenarios, the collision detection algorithm needs to have the flexibility to adapt to different construction sites in order to obtain the best computational performance.

4. Approximations of objects in virtual construction

To efficiently identify the collision status of objects in virtual construction scenarios, objects should be suitably approximated. The following sections will present a typical virtual construction scenario and the approximation methods for the objects in the construction scenarios.

A typical construction scenario comprises of a number of structural elements and construction machineries. Structural elements, such as beams and columns, are construction materials which form a building. They are usually static, only moving one or two times during the entire project. Construction machineries are used to manipulate the structural elements, moving them from storage to the building site. They are basically dynamic. The position and configurations of construction machineries continuously change during the construction process.

Fig. 1 illustrates a typical virtual construction scenario. It includes a high-rise building on the ground formed by 2612 structural elements, including beams and columns. It also includes multiple construction

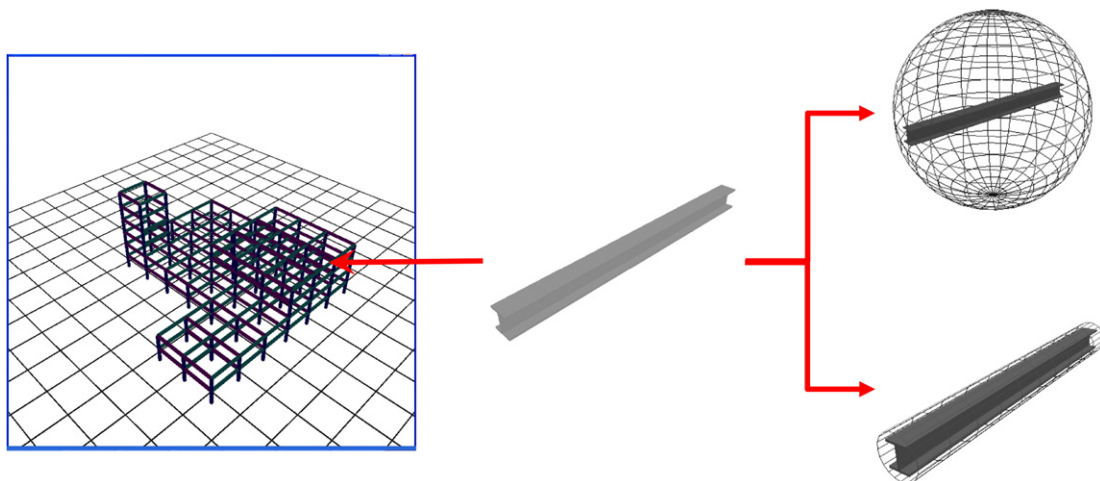


Fig. 2. Approximation of construction material.

machinery, including five tower cranes and three mobile cranes. Since the structural elements are assumed to be static, only the motion of construction machinery will cause collisions.

To develop high-performance algorithms for collision detection problems in such virtual scenarios, it is important to properly approximate the static and dynamic objects in the construction scenario. The following two sections will introduce the approximation methods developed in this research for structural elements and for construction machineries respectively.

5. Approximation of structural elements

Structural elements such as beams and columns on the construction scene can be approximated as different boundaries to accelerate the collision detection computations. Fig. 2 illustrates a small construction scenario which includes 338 structural elements. Most of the structural elements, such as beams and columns, in this construction scenario are usually long-shaped objects. Therefore, in this research, we propose the use of spheres and cylinders and define the methods to approximate these objects. Since a spherical boundary may result in a large error in some cases, we also propose two boundary levels, the rough boundary and the detailed boundary, to approximate the structural elements at different precision levels. The following sections will introduce (1) the rough spherical boundary, (2) the detailed spherical boundary and (3) the cylindrical boundary proposed in this research. A comparison between the methods is also presented.

5.1. Rough spherical boundary

The rough spherical boundary approximates a structural element using a spherical outer boundary. Fig. 3 presents the approximation boundary using perspective, front view, and side view projections. From the perspective view, we can see that an element is approxi-

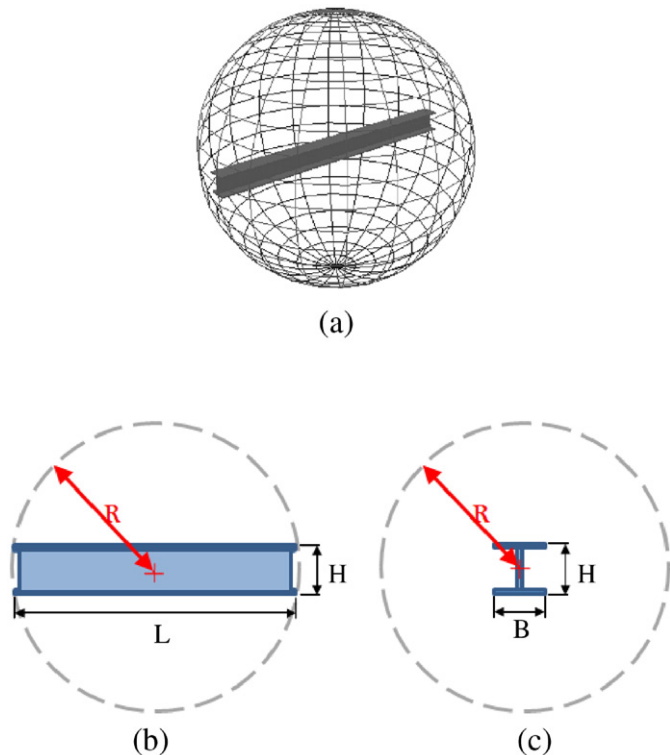


Fig. 3. Rough spherical boundary approximation (a) perspective view (b) side view (c) front view.

mated by an outer sphere. The center of the sphere is located at the centroid of the structural element. The radius R_{RS} is computed using Eq. (1).

$$R_{RS} = \sqrt{\left(\frac{L}{2}\right)^2 + \left(\frac{B}{2}\right)^2 + \left(\frac{H}{2}\right)^2} \quad (1)$$

where L is the length of the structural element; B is the width of the structural element; and H is the height of the structural element. Because this approximation method does not present the shape of the structural element precisely, some errors, which means differences between the real and computed distance, may be included when computing the distance between objects. The maximum error (worst case) may occur on the top center of the structural element or the side center of the structural element. The maximum error can be computed using Eq. (2).

$$\text{Error}_{\max-RS} = R_{RS} - \text{MIN}\left(\frac{B}{2}, \frac{H}{2}\right) \quad (2)$$

The advantage of using a spherical boundary is the low computational cost required to calculate the distance between spheres. If we approximate all the structural elements using spheres, in order to see the collision status between the objects, we can simply compute the distance between the centers of two spheres and subtract the radius of both spheres. If the result is greater or equal to zero, we can ensure that the two objects are collision free. Although some errors are included in the computation, the result is fortunately always conservative. In other words, the distance between objects is always underestimated. Since the computational load is very light, this method is especially suitable for the detection of collisions between a large numbers of elements which have a low possibility of colliding.

5.2. Detailed spherical boundary

The detailed spherical boundary approximates a structural element using multiple spherical outer boundaries. Fig. 4 shows this approximation boundary using perspective, front view, and side view projections. From the perspective view, we can see that a structural element is approximated using multiple spheres which cover the entire structural element. The radius R_{DS} is computed using Eq. (3):

$$R_{DS} = \sqrt{\left(\frac{B}{2}\right)^2 + \left(\frac{H}{2}\right)^2} \quad (3)$$

Although this detailed spherical boundary bounds the structural element tightly, there are also errors that occur when using this approximation method. Here, we assume that structural elements, including I-beams, are modeled as boxes. This assumption is reasonable when simulating the operation of the construction machineries. Few operators are willing to or are capable of moving two objects so closely. This assumption ensures that the system always yields a conservative result to avoid the undesired operations. The maximum error (worst case) can be observed in Fig. 4c and computed using Eq. (4).

$$\text{Error}_{\max-DS} = R_{DS} - \text{MIN}\left(\frac{B}{2}, \frac{H}{2}\right) \quad (4)$$

The number of the spheres required is determined by the ratio between the length and width or the ratio between the length and height. This can be calculated using Eq. (5).

$$N_{DS} = \text{MAX}\left(\frac{[2L]}{B}, \frac{[2L]}{H}\right) \quad (5)$$

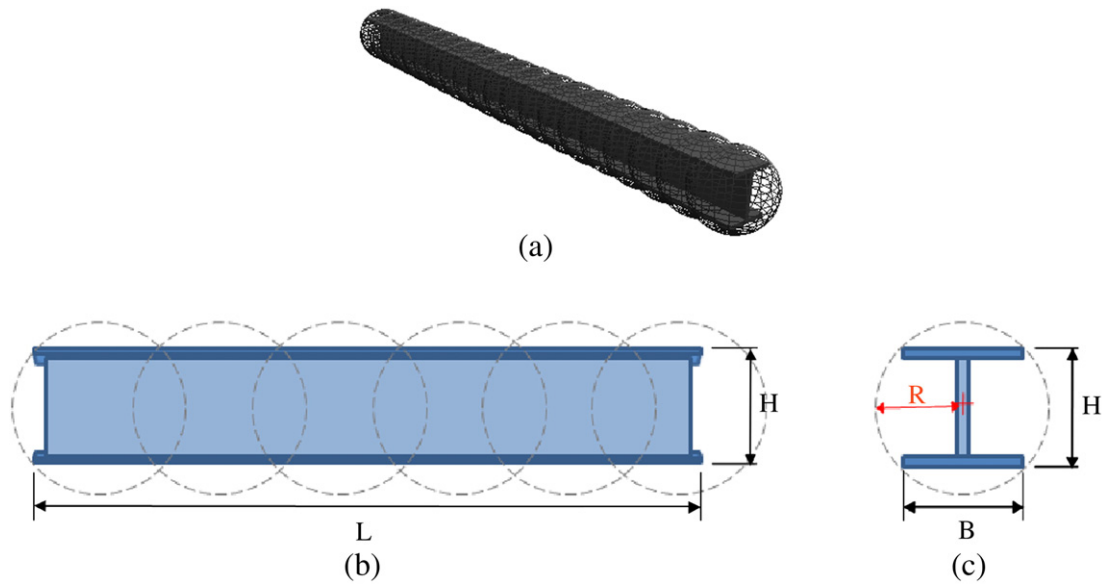


Fig. 4. Detailed spherical boundary approximation (a) perspective view (b) side view (c) front view.

To ensure the use of the minimum necessary number of spheres to approximate the structural element, we need to compute the location of the center of the spheres using a two-step method. The first step is to determine the center of the end spheres, which are located at the two ends of the structural element, i.e. the first and last spheres in the sphere list. The second step is to compute the center of the internal spheres by equally dividing the center line. This process ensures that the entire structural element can be covered using the minimum necessary number of spheres.

The detailed spherical approximation boundary provides better accuracy than bounding a structural element using a rough sphere. However, the computational cost is also higher because more spheres are involved in the computational processes. Therefore, we need to find a balance between the computational cost and accuracy to meet the needs for the application of virtual construction.

5.3. Cylindrical boundary

The cylindrical boundary approximates a structural element using a cylindrical outer boundary. Fig. 5 shows the approximation boundary using perspective, front view, and side view projections. The perspective view illustrates a cylinder which approximates the structural element. The center line of the cylinder is located at the centroid line of the structural element. The radius R can be computed using Eq. (3). Its maximum possible error (in a worst case scenario) can be calculated using Eq. (4).

The advantage of using a cylindrical boundary is that we obtain relatively precise results when computing the distance between structural elements. The cylindrical boundary approximation can obtain the same accuracy as the detailed spherical approximation. However, to compute the distance between the cylinders or between a

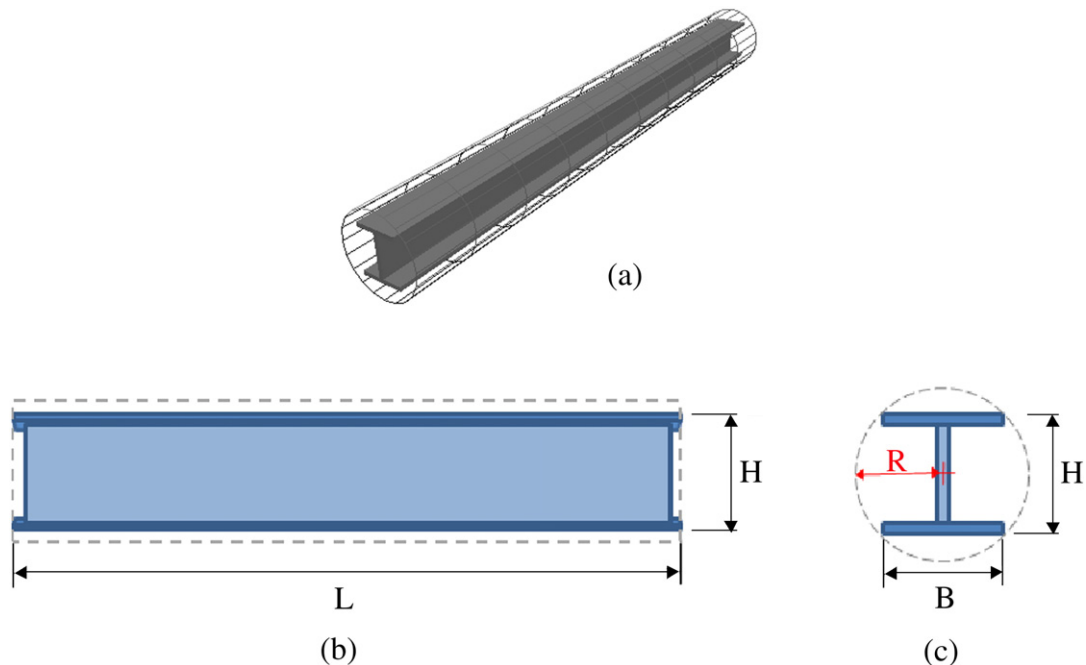


Fig. 5. Cylindrical boundary approximation (a) perspective view (b) side view (c) front view.

Table 1
Error and computational cost for different types of approximations.

Approximation type	Max error	Computational cost*
Rough sphere	$\sqrt{(\frac{L}{2})^2 + (\frac{B}{2})^2 + (\frac{H}{2})^2} - \text{Min}(\frac{B}{2}, \frac{H}{2})$	1
Detailed Sphere	$\sqrt{(\frac{B}{2})^2 + (\frac{H}{2})^2} - \text{Min}(\frac{B}{2}, \frac{H}{2})$	$\text{Max}(\frac{2L}{B}, \frac{2L}{H})$
Cylinder	$\sqrt{(\frac{B}{2})^2 + (\frac{H}{2})^2} - \text{Min}(\frac{B}{2}, \frac{H}{2})$	≈ 3

*The time for computing the distance between two spheres equals 1 unit.

sphere and cylinder usually requires multiple steps to compute the distance between end points and lines. The computational cost is higher than using the rough sphere approximation and in some cases is also higher than using the detailed sphere approximation.

In short, although the cylindrical boundary provides better accuracy than a rough sphere, the computational cost incurred when using a cylinder (line) is always much more than a sphere (point). However, in some situations, the cylindrical boundary is still a preferable choice, such as when dealing with a long element requiring higher accuracy.

5.4. Comparison of the three approximation methods

Table 1 lists the max errors and computational costs associated with collision detection using different approximations of structural elements. In the rough spherical approximation, we only use one sphere to represent one structural element. The computational cost of collision detection in this approximation is the cost of calculating the distance between two spheres, and the error, in general cases, is approximately half of the longest length (L). Since structural elements of a typical building range from 6 to 12 m, the errors of the rough spherical approximation will be 300 to 600 cm. In the detailed spherical approximation, the computational cost is related to the length ratio between L , B , and H . It is typically about 10–15 times higher than the computational cost using the rough spherical approximation, and the

error in most cases drops to within the range of 20 to 40 cm. The computational cost of detecting a collision between a cylinder and a sphere is approximately 3–5 times the cost of detecting a collision between two spheres depending on the implementation methods.

5.5. Integration strategies for approximation methods

In this research, we integrated the above-mentioned approximation boundaries and developed four strategies to represent structural elements: (1) detailed spherical boundary, (2) cylindrical boundary, (3) detailed spherical boundary with rough spherical boundary and (4) cylindrical boundary with rough spherical boundary (as shown in Fig. 6). The first two strategies employed either only spherical boundaries or cylindrical boundaries to approximate the structural elements. The other two strategies are two-stage approaches. In order to meet the accuracy requirements of some construction projects, it is impossible to adopt only the rough spherical boundary approximation for the whole procedure. The detailed spherical boundary approximation and the cylindrical boundary approximation are much more accurate.

Detailed approximations (spherical and cylindrical) require significant computing time when applied directly. Therefore, we developed a two-stage method. In the first stage, an external sphere was assigned to present the structural element. Although the rough boundary may add additional computational costs, it can reduce the number of collision checks between objects that have a low possibility of colliding. In the second stage, we use either multiple spheres or a cylinder to represent the element with better detail. The second stage will guarantee the accuracy of the results of collision checks.

6. Approximation of construction machinery

In this research, we focused on approximating two typical types of construction machinery. One is a mobile crane and the other a tower crane. Similar approximation methods can also be applied to different


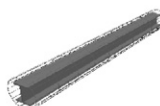
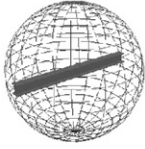

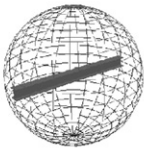
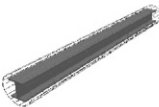
Strategies	Stages		Description
	Stage 1	Stage 2	
Sphere Only			Using detailed spherical approximation only
Cylinder Only			Using cylindrical approximation only
Sphere -> Sphere			Using rough spherical approximation as stage 1 and detailed spherical approximation as stage 2
Sphere -> Cylinder			Using rough spherical approximation as stage 1 and cylindrical approximation as stage 2

Fig. 6. Four strategies to approximate a structural element.

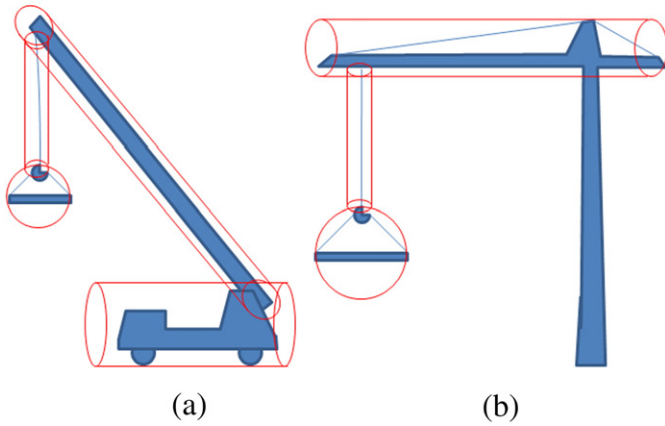


Fig. 7. Construction machinery: (a) mobile crane; (b) tower crane.

construction machinery, such as excavators or fold-lifters on other virtual construction sites.

A two-stage approach was also developed to approximate mobile cranes and tower cranes. An outer boundary for cranes eliminates structural elements that are a relatively long distance away from the cranes before each part of the crane starts to search for potential collisions. If structural elements are inside the outer boundary, all parts of the crane will detect a collision with these structural elements. On the other hand, if structural elements are not inside the outer boundary, the crane will ignore these structural elements when collision detection commences.

6.1. Approximation of parts in a mobile crane

Fig. 7a shows a simplified model of a typical mobile crane and its geometric boundary. We approximate the mobile crane with its four major parts. The first one is the main body (also called the car). The second part is an expandable jib. At the end of the jib is a cable with a hook. Under the hook, there may be a structural element hanging at the bottom of the cable. We approximate the four parts of the mobile crane using their basic geometric shapes. We employed cylinders to approximate the car, jib, and the cable. Because the structural element hanging under the hook may rotate in its available space (i.e. the working space of the element is a sphere), we defined a sphere to represent the structural element.

6.2. Approximation of parts in a tower crane

The tower crane is a modern form of the balance crane. Fixed to the ground, tower cranes often give the best combination of height and lifting capacity and are used in the construction of tall buildings. A horizontal jib is balanced asymmetrically across the top of the tower. Its short arm carries a counterweight of concrete blocks, and its long arm carries the lifting gear. A typical tower crane consists of three

major parts: the tower, jib, cable, and the hook. We developed an approximation model for the tower crane shown in Fig. 7b. We ignore the boundary of the tower because the tower always remains still during erection. Since erection planners usually arrange the erection activities away from the tower, there is little possibility of colliding with the structural elements. We used a cylinder to represent the crane jib and the cable. Here, we also used a sphere to represent the structural element hanging under the hook.

6.3. Rough approximation boundary of a crane

To reduce the computational cost between objects that have little possibility of colliding, we introduced two types of rough approximation boundaries outside the crane to eliminate the collision checks between objects which are far from each other. One is a spherical boundary and the other is a cylindrical boundary. Fig. 8 shows the rough spherical approximation of a crane. The radius R_{STC} in the tower crane case, is determined using Eq. (6).

$$R_{STC} = \sqrt{\left(\frac{H_{\text{partial}}}{2}\right)^2 + \left(\frac{L_{\text{partial-jib}}}{2}\right)^2} \quad (6)$$

where H_{partial} is the height of the tower measured from the ground to the bottom of the jib; $L_{\text{partial-jib}}$ is the partial length of the jib measured from the center of the tower to the tip of the jib, as shown in Fig. 8. The center of the rough spherical outer boundary is located at $\left(\frac{L_{\text{partial-jib}}}{2}, \frac{H_{\text{partial}}}{2}\right)$ while the base of the tower crane is considered to be the origin; in the mobile crane case, the radius R_{SMC} of the rough spherical boundary is computed using Eq. (7).

$$R_{SMC} = \frac{L_{\text{jib}}}{2} \quad (7)$$

where L_{jib} is the length of the jib of the mobile crane and the center of the rough spherical outer boundary is located at the center of the jib of the mobile crane. The rough spherical outer boundary is considered effective for eliminating structural elements in the first phase of the computation of collision detection. However, when the tower crane is tall enough, the radius of the rough spherical outer boundary will grow as large as the height of the tower crane. At that time, the rough outer boundary merely has the effect of increasing the computing time.

Fig. 9 presents the rough cylindrical approximation of a crane. In the case of a tower crane, the radius R_{CTC} of the rough outer cylindrical boundary is computed using Eq. (8).

$$R_{CTC} = \frac{L_{\text{jib}}}{2} \quad (8)$$

where L_{jib} is the length of the jib of the tower crane shown in Fig. 9. The center line of the rough cylindrical outer boundary is perpendicular to

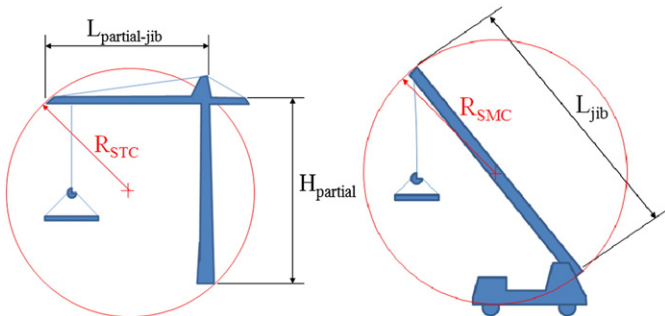


Fig. 8. Rough spherical approximation of a crane.

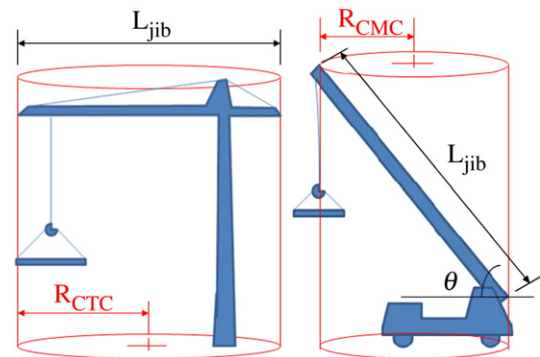


Fig. 9. Rough cylindrical approximation of a crane.

the ground and passes through the middle point of the jib; in the mobile crane case, the radius R_{CMC} of the rough outer cylindrical boundary is defined by Eq. (9):

$$R_{CMC} = \frac{L_{jib}}{2} \cos \theta \quad (9)$$

where L_{jib} is the length of the jib of the mobile crane and θ is the angle of elevation of the jib of the mobile crane. The center line of the rough cylindrical outer boundary is perpendicular to the ground and passes through the middle point of the jib. When dealing with a high crane unit, it is preferable to use the rough cylindrical outer boundary; it has the benefit that the radius of the cylinder does not change when the height of the crane grows. This type of rough outer boundary is strategically beneficial when dealing with high-rise buildings.

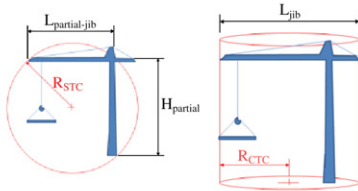

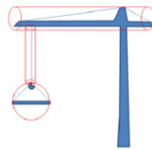
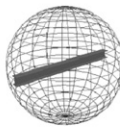
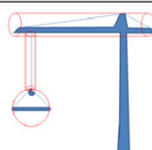
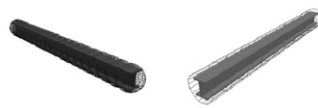
7. Collision detection algorithm for virtual construction

An algorithm, VC-COLLIDE, was developed to deal with the collision detection problems in a virtual construction environment. This algorithm uses the above-mentioned approximation methods for structural elements and construction machineries to reduce the computational cost of the entire virtual construction scenario. Approximations of a structural element focus on reducing the consumed time of each individual collision detection computation; approximations of construction machinery, especially the outer crane boundary, take advantage of the reduction in collision calculations. In this research, we integrated the above two principles in VC-COLLIDE to efficiently solve the problem of collision detection in virtual construction scenarios.

7.1. Working procedure of VC-COLLIDE

For performing collision detection in a virtual construction site, we separate all the objects into two groups, a dynamic group and a static group. The dynamic group includes the parts of construction machines while the static group includes the structural elements and other static objects on the site, such as the scaffolds. Objects in the static group do not have any possibility of colliding with other objects of this group. However, objects in the dynamic group may collide with objects in the static group or other objects in the dynamic group (collision between machineries). In VC-COLLIDE we generally define the objects in the static group as structural elements and the objects in dynamic group as machineries. They are the two inputs of the VC-COLLIDE algorithm.

The whole procedure can be separated into three phases: the rough-rough phase, detailed-rough phase, and the detailed-detailed phase, as shown in Fig. 10. The rough-rough phase (R-R phase) uses rough crane approximation to represent the machinery group and rough element approximation to represent the structural element group. Figs. 8 and 9 illustrate the two above-mentioned rough approximation methods, cylindrical and spherical approximations, for a tower crane and an approximation method, spherical approximation, for structural elements in the R-R phase. In this phase, VC-COLLIDE attempts to eliminate unnecessary collision checks between the machineries and structural elements as much as possible. This process is usually very time-consuming, especially when the number of structural elements and machineries on the construction site are large. However, this process can significantly reduce the computation time for later phases. After checking all pairs of rough approximation boundaries, a list of collided structural elements will be identified as the return result of the R-R phase.

Phase	Approximation of Construction Machinery*	Approximation of Structural Element
Rough-Rough (R-R)		
Detailed-Rough (D-R) (Optional)		
Detailed-Detailed (D-D)		

*using tower crane as an example

Fig. 10. Three phases of VC-COLLIDE.

Algorithm VC-COLLIDE(*constructionMachineryList*, *structuralElementList*): check the collision status between construction machineries and structural elements. Return a list of structural elements which collided with construction machineries.

```

1:   ML ← constructionMachineryList
2:   EL ← structuralElementList
3:   CEL ← collidedStructuralElementList
4:   CEL ← RoughRoughCheck(ML, EL)
5:   CEL ← DetailedRoughCheck(ML, CEL)
6:   CEL ← DetailedDetailedCheck(ML, CEL)
7:   RETURN CEL

```

Fig. 11. Algorithm: VC-COLLIDE.

The detailed-rough phase (D-R phase) is performed after the rough-rough phase. Only the structural elements identified as the collided elements of the R-R phase (i.e. within the rough crane approximation) are checked to see whether these suspicious elements actually collide with working cranes in this environment. In the D-R phase, construction machineries were modeled in more detail using multiple smaller cylinders and spheres. The structural elements were still approximated using external spheres. Similar to the process in the R-R phase, a list of elements whose approximation boundaries interfere with others is returned.

After the D-R phase, the list of structural elements classified as collided objects needs to be confirmed by an even more detailed approximation method. The detailed-detailed phase (D-D phase) is designed for this purpose. The tower crane remains approximated using multiple cylinders and spheres and the structural elements can be approximated by either external cylinders or a number of smaller spheres. Although the computational cost can be much higher for computing each pair of structural elements and construction machinery than the previous phases, higher accuracy (less than 10 cm in most cases) can be achieved.

7.2. The VC-COLLIDE algorithm

The following sections present the VC-COLLIDE algorithm and three major functions used in VC-COLLIDE, including *RoughRoughCheck*, *DetailedRoughCheck*, and *DetailDetailCheck*.

7.2.1. VC-COLLIDE algorithm

VC-COLLIDE is the algorithm developed in this research that is focused on solving collision detection problems in virtual construction

scenarios. Fig. 11 shows the process of VC-COLLIDE. VC-COLLIDE takes two variables as parameters. The first one is *constructionMachineryList* and the second one is *structuralElementList*. The parameter *constructionMachineryList* is a list of stored information on construction machineries in the virtual construction scenario to be solved. The parameter *structuralElementList* is a list of stored information on structural elements in the virtual construction scenario. After these two lists pass into VC-COLLIDE, VC-COLLIDE will perform the three phases of collision detection to check if any collision occurred in the virtual construction scenario. The three phases are shown above: the R-R phase, D-R phase, and the D-D phase. Each phase will filter out structural elements which have no possibility of colliding with construction machineries in the virtual construction scenario through specific approximation methods for each phase.

In the end, a list, CEL (standing for the list of collision elements), which stores information on collided structural elements, will be returned. Structural elements stored in CEL are structural elements which collided with construction machineries in the virtual construction scenario within the appropriate tolerance and accuracy restriction of a construction requirement.

7.2.2. RoughRoughCheck function

RoughRoughCheck is a function first used in VC-COLLIDE. Fig. 12 shows the process of the algorithm. The goal of this algorithm is to eliminate structural elements which are too far away from a construction machine to collide. Construction machineries and structural elements are approximated during the R-R phase. After completion of the distance calculations between construction machineries and structural elements which are approximated with the R-R phase style, structural elements which collided with construction

Algorithm RoughRoughCheck(*constructionMachineryList*, *structuralElementList*): check the collision status between rough approximated construction machineries and rough approximated structural elements. Return a list of structural elements which collided with construction machineries.

```

1:   ML constructionMachineryList
2:   EL structuralElementList
3:   CEL collidedStructuralElementList
4:   FOR EACH construction machinery M in ML
5:       M.roughApproximation()
6:       FOR EACH structural element E in EL
7:           E.roughApproximation()
8:           IF E collided with M THEN CEL.pushback(E)
9:   RETURN CEL

```

Fig. 12. Algorithm: RoughRoughCheck.

Algorithm *DetailedRoughCheck*(*constructionMachineryList*, *structuralElementList*): check the collision status between detailed approximated construction machineries and rough approximated structural elements. Return a list of structural elements which collided with construction machineries.

```

1:   ML constructionMachineryList
2:   EL structuralElementList
3:   CEL collidedStructuralElementList
4:   FOR EACH construction machinery M in ML
5:       M.detailedApproximation()
6:       FOR EACH structural element E in EL
7:           E.roughApproximation()
8:           IF E collided with M THEN CEL.pushback(E)
9:   RETURN CEL

```

Fig. 13. Algorithm: DetailedRoughCheck.

machinery are stored in a CEL, a list of collided elements. The list CEL will then be returned for the usage of the next phase. However, there are two types of rough approximations for construction machineries, spherical and cylindrical, and only spherical approximation for structural elements. Thus, the selection of rough approximation of construction machineries will result in two situations.

7.2.3. DetailedRoughCheck function

DetailedRoughCheck is a function in the second phase of VC-COLLIDE. Fig. 13 shows that construction machineries and structural elements filtered through RoughRoughCheck will be approximated with the D–R phase style. Thus, distance calculations between construction machinery and structural elements will be performed and structural elements that collided with construction machinery in this phase will be stored in a list called CEL which will be returned after the algorithm DetailedRoughCheck is completed. Construction machinery is approximated with the detailed mode (i.e. all parts of construction machineries are individual approximated boundary boxes) and structural elements are approximated as a sphere. Considering the selection of the collision detection strategy, this phase (or algorithm) is optional and can be skipped. In other words, it is possible that the D–D phase will tightly follow the R–R phase if the D–R phase is set to “skipped”.

7.2.4. DetailedDetailedCheck function

Fig. 14 shows the process of the third function used in VC-COLLIDE. As the last phase in VC-COLLIDE, construction machineries and

structural elements are approximated using the most detailed approximation. After calculations of the distance between construction machinery and structural elements are completed, the remaining structural elements which are considered to have collided with construction machinery under the most accurate condition will be stored back to CEL. CEL will be returned to VC-COLLIDE as the final result showing structural elements which collided with construction machinery in this virtual construction scenario. In the D–D phase, construction machineries are approximated in the detailed mode and structural elements are approximated with either a cylinder or multiple spheres. Thus, the selection of collision detection strategy will produce two results.

7.3. Strategies in the VC-COLLIDE algorithm

There are two selections in the R–R phase (i.e. construction machinery approximated as a rough sphere or rough cylinder), two selections in the D–R phase (i.e. whether or not this phase is skipped), and two selections in the D–D phase (i.e. structural elements approximated as a cylinder or multiple spheres). Thus, eight strategies resulted in total from this research.

By using this approach, the distances between construction machinery and structural elements is calculated efficiently following the principle of dealing with a large number of calculations by using the most efficient method. The largest number of structural elements occurs at the very start of the collision detection procedure. The rough

Algorithm *DetailedDetailedCheck*(*constructionMachineryList*, *structuralElementList*): check the collision status between detailed approximated construction machineries and detailed approximated structural elements. Return a list of structural elements which collided with construction machineries.

```

1:   ML constructionMachineryList
2:   EL structuralElementList
3:   CEL collidedStructuralElementList
4:   FOR EACH construction machinery M in ML
5:       M.detailedApproximation()
6:       FOR EACH structural element E in EL
7:           E.detailedApproximation()
8:           IF E collided with M THEN CEL.pushback(E)
9:   RETURN CEL

```

Fig. 14. Algorithm: DetailedDetailedCheck.

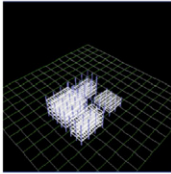
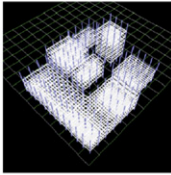
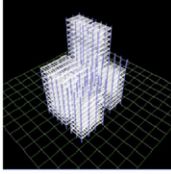
Scenario No	Type	Number of Structural Element	Number of Construction Machinery	Snapshot
1	Small	683	2 tower cranes 3 mobile cranes	
2	Large	2143	3 tower cranes 5 mobile cranes	
3	Tall	2612	5 tower cranes 3 mobile cranes	

Fig. 15. Three construction scenarios for testing.

approximation of construction machinery is a convenient method for eliminating structural elements which are too far away from specific construction machinery. When calculating the distance with the rough approximation of construction machinery, structural elements are simply approximated using the rough spherical boundary, which is most efficient in calculations. The multi-level hierarchical approximation for structural element and construction machinery also obeys the same principle.

8. Performance tests

To validate the performance of VC-COLLIDE in virtual construction scenarios, we conducted a series of performance tests in three test scenes. This section summarizes the processes of the tests and their results.

8.1. Test scenarios

We designed three typical virtual construction scenarios to test and compare the collision detection algorithm and approximation methods developed in this research. Fig. 15 presents detailed descriptions for the three scenarios.

The first scenario is a small building project, in which 683 structural elements with two tower cranes and three mobile cranes were involved. This is typical of a house or a low-rise office project. The project site is usually small so the machineries need to work in relatively narrow areas. Because the computational cost for collision detection is relatively low, this test scenario can be used as a baseline for the overall test.

The second scenario is a large construction project. It is a wide-area construction site like a shopping mall or a large apartment. For these types of projects, we usually need to consider using multiple working phases to expedite the construction progress and the reach of the machineries also needs to be taken into account. Therefore, we define 2143 structural elements, 3 tower cranes, and 5 mobile cranes to be involved in the project.

The third scenario is a tall building project which can be related to high-rise projects commonly seen in urban areas. The building in the

project contains 2612 structural elements. Because these high-rise projects rely more on tower cranes to support vertical transportation, we define five tower cranes and three mobile cranes in this scenario.

8.2. Implementation and test environment

Since the goal of this research is to develop a generic collision detection method to support various virtual construction scenarios for the general usage of construction purposes, we chose a typical personal computer to implement and test the VC-COLLIDE algorithm. The computer is equipped with a 1.73 GHz central processor and 1 GB memory; a mid-range computer in the year 2007 in terms of computational speed.

C#, an object-oriented programming (OOP) language, is used as the primary programming language in this research. In other words, the Microsoft.NET framework was implemented on the computer system. The Common Language Runtime (CLR) functionality of C# provides services such as security, memory management, and cross-language integration, which not only makes the computer software more robust and efficient but also potentially reduces the effort required to integrate the functions with other virtual construction software in the future.

8.3. Test results

We conducted four performance tests to evaluate the computational efficiency of VC-COLLIDE in three testing scenarios. Because VC-COLLIDE actually approximates all the construction machineries and structural elements using either cylinders or spheres or the combination of both, the first test focused on the most fundamental problem: the computational cost for distance-checking between spheres and cylinders. The second test was designed to test the benefit brought from the rough approximation for cranes (i.e. the benefit from the R-R phase). The third test was designed to quantify the computational costs of VC-COLLIDE using different approximation strategies and in different test scenarios. The fourth test was designed to compare the computational performance between the VC-COLLIDE algorithm and ODE, a physics engine widely used for checking for collisions in virtual environments.

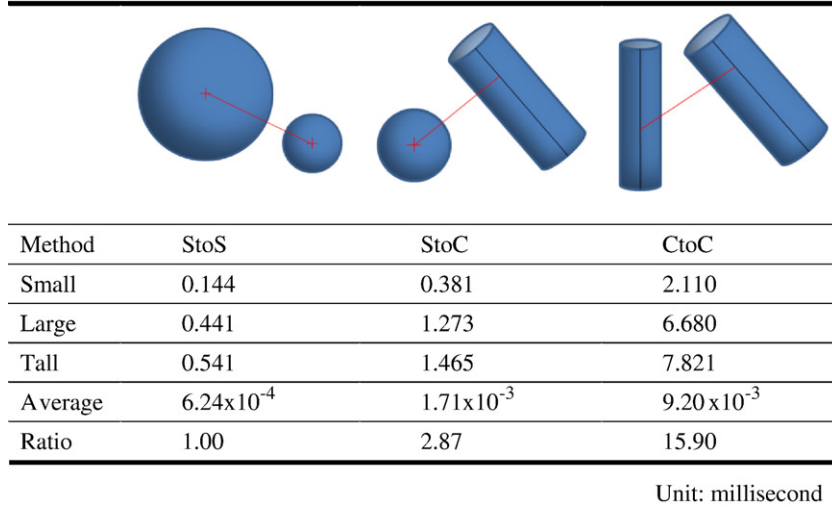


Fig. 16. Computational costs of different distance calculations.

8.3.1. Comparison between approximation methods

VC-COLLIDE actually decomposes a virtual scenario into a number of cylinders and spheres. The collision detection problems become a series of distance computations between two spheres, a sphere and a cylinder, and two cylinders. Therefore, we designed a test especially to evaluate the computational costs of each distance check in the three test environments. Take the case of the evaluation of the computational cost for computing the distance between spheres as an example. We placed a sphere in the center of the scenario and used spheres to approximate the structural elements individually and timed the computation duration for the whole process. Since the time duration is too small to be calculated on computers, we repeated the process 1000 times and took the average. The same procedure was also applied to evaluate the cost for computing the distance between a sphere and a cylinder and between two cylinders.

As seen in Fig. 16, the average costs for computing the distance between two spheres (StoS), a sphere and a cylinder (StoC), and two cylinders (CtoC) in three test scenarios are presented. We found the overall average costs (in milliseconds) for computing the distance between StoS, StoC, CtoC to be 6.24×10^{-4} : 1.71×10^{-3} : 9.20×10^{-3} . The ratio is 1.00: 2.87: 15.90. The correlation between the ratios in the three test environments is statistically high ($R^2 > 0.99$).

8.3.2. Computational costs for different rough approximation of cranes

During implementation of VC-COLLIDE, we noticed that the rough approximation of cranes can eliminate most of the unnecessary collision checks and rapidly increase the computing performance. Therefore, we designed a test to quantify the change in computational costs with and without rough approximation in three test scenarios. Fig. 17 shows the testing results. Without rough approximation, the computational cost for checking distances is proportional ($R^2 > 0.99$). With the spherical approximation, the computational cost of the large project and the tall project are significantly reduced to 47.88% and 32.53% respectively. With the cylindrical approximation, the computational cost of the large project and the tall project are also significantly reduced to 34.52% and 47.88% respectively. Here, we found that both spherical and cylindrical approximations can effectively reduce computational costs. Spherical approximation performs better in small and large construction projects while cylindrical approximation performs better in tall projects.

8.3.3. Computational performance in test scenarios

Table 2 shows the computational cost of VC-COLLIDE in three test scenarios. We conducted the tests eight times, each time to estimate one collision detection strategy, i.e. a specific combination of approxi-

mation methods and phases. As mentioned before, we used both spherical approximation and cylindrical approximation in the R–R phase. Since D–R is an optional phase, we further separated the tests into two groups: one with the D–R phase and one without it. In the D–D phase, we also separated the tests into the spherical D–D phase and the cylindrical D–D phase.

From the eight tests in three test scenarios, we found that the smallest computational cost (denoted by an asterisk '*' in Table 2) for the three scenarios occurred in different collision detection strategies. In both the small and large scenario, the smallest computational costs occurred (1) in the setup of the spherical R–R phase, (2) with the D–R phase, and (3) cylindrical D–D phase. In the tall scenario, the smallest computational cost occurred (1) in the setup of the cylindrical R–R phase, (2) with the D–R phase, and (3) in the spherical D–D phase. This shows that different scenarios (i.e. different combination of building structure and machineries) may need different collision detection strategies to obtain optimal results.

During the tests, we also found the choice of which rough-approximation method was optimal in the R–R phase greatly depended on the test scenario. From Table 2, we can see that in the small and large scenarios, selecting the spherical approximation boundary for cranes results in better computational efficiency (on average saving 16.76% of the computational time). On the contrary, in the tall scenario, selecting the cylindrical approximation in the R–R phase results in much better computational efficiency (38.11% on average) when performing the collision detection checks.

In addition, the importance of the D–R phase was also highlighted. Although the D–R phase is an intermediate phase requiring additional computational costs for computing the collisions between the detailed

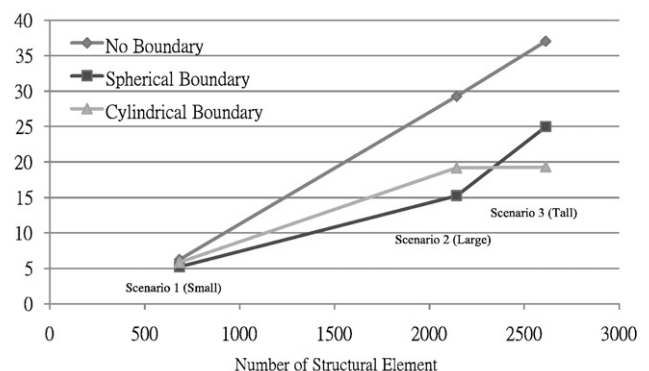


Fig. 17. Computational costs for different rough approximations of a crane.

Table 2

Computational cost for each strategy in three scenarios.

Testing Scenarios	Rough spherical approximation in R–R phase				Rough cylindrical approximation in R–R phase			
	With D–R phase		Without D–R phase		With D–R phase		Without D–R phase	
	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase
Small	5.216	4.556*	12.842	12.444	5.853	5.244	9.917	9.614
Large	15.257	14.909*	43.954	43.819	19.167	19.277	33.138	32.295
Tall	24.987	23.759	74.988	73.862	19.305*	19.334	33.505	32.684

Unit: millisecond.

*The most efficient strategies in specific virtual construction scenarios.

approximation of cranes and the rough approximation of structural elements, the overall computational cost shows reduced computational costs in all three scenarios. In these tests, an average of 38.11% of the computational cost can be saved due to the introduction of the D–R phase.

8.3.4. Performance comparison with ODE

To validate the performance of VC-COLLIDE, we compared the computational efficiency of VC-COLLIDE and ODE in the three test scenarios defined in this research. ODE is a tool for simulating rigid body dynamics, broadly used to deal with collision detection problems in 3D games, virtual reality, and scientific simulations [11]. The results are summarized in Table 3.

We ignored the R–R phase during the comparison test because it was designed only to reduce the number of collision checks. Therefore, twelve tests (four collision detection strategies in three scenarios) were conducted using both VC-COLLIDE and ODE. From the results, we found that VC-COLLIDE performed significantly better than ODE in all twelve tests. 77% to 96% of the computational time was saved by using VC-COLLIDE. If we consider 50 ms (20 fps) to be the limitation of real-time visualization, ODE is apparently unable to support the computation of collision detection during the time duration.

The major disadvantage of ODE in virtual construction scenarios is the extensive range of functions it provides. For example, the collision detection function in ODE calculates not only the distance between objects, but also the coordination of the contact position and even the depth to which the two bodies inter-penetrate each other. The normal to the contact surface is also calculated. Although these functions can be used for other purposes, they are redundant in dealing with collision detection in virtual construction and require extra computational costs.

9. Summary and conclusions

The major contribution of this research is the development of an efficient collision detection method that supports the real-time rendering of a construction scenario. To reduce computational efforts, proper approximation methods need to be developed to decrease the number of logic tests and distance calculations. We introduced two types of boundary boxes, cylindrical and spherical boundaries, to approximate construction machineries and structural elements commonly seen on construction sites. These methods simplify

collision detection between the construction machineries and structural elements into a series of collision checks between spheres and cylinders.

We developed the VC-COLLIDE algorithm to integrate the spherical and cylindrical boundary boxes. This algorithm actually transfers a list of construction machineries and structural elements in a construction scenario into a series of collision checks between spheres and cylinders. If there are any collisions between cylinders and spheres, VC-COLLIDE will return the collision status as a true value. If there are no collisions between the cylinders and spheres, then VC-COLLIDE will return a value which presents the conservative distance from collision. Four major strategies were included in VC-COLLIDE. Each one demonstrates different computational performances in different scenarios. VC-COLLIDE is straightforward and can be implemented in different visualization tools.

We also created three testing scenarios, construction sites for a small, large and tall building, as test examples and conducted performance tests by using VC-COLLIDE and the ODE software package. The results pointed out that for different virtual construction scenarios, the best collision detection strategy is not always the same one. Some situations are easy to predict (e.g. the spherical outer crane boundary cannot handle a high-rise building scenario well) but some are very difficult to be judged without actual tests.

In this research, we found that the compromised accuracy contributes majorly to the high computational performance in VC-COLLIDE. The construction machineries and structural elements are approximated by outer boxes, either spheres or cylinders. These approximation methods will introduce errors in the computational results. The distance between the list of cylinders and spheres which represent the machineries and structural elements is always less than or equal to the real distance between the construction objects. Because machinery movement in a construction site always requires additional safe distance, the conservative results computed by VC-COLLIDE are acceptable.

In short, VC-COLLIDE, the collision detection algorithm developed in this research, is an effective method for efficiently computing the collision status and approximating the free distance in various construction scenarios. This algorithm only includes the computed distance between spheres and cylinders. It is easy to implement and integrate with different visualization tools to render virtual construction scenarios in real-time.

Table 3

Performance comparison between VC-COLLIDE and ODE.

Testing Scenarios	VC-COLLIDE				Open dynamic engine			
	With D–R phase		Without D–R phase		With D–R phase		Without D–R phase	
	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase	Spherical D–D phase	Cylindrical D–D phase
Small	6.235	6.173	24.120	23.979	158.594	173.125	178.750	126.719
Large	29.272	29.122	132.536	134.236	669.271	853.646	917.708	638.542
Tall	37.036	35.682	169.744	166.344	797.396	1009.375	1055.729	741.667

Unit: millisecond.

References

- [1] G. van den Bergen, Efficient collision detection of complex deformable models using AABB Trees, *Journal of Graphics Tools* 2 (4) (1997) 1–13.
- [2] G van den Bergen, SOLID the software library for interference detection, 2004 Available at: (<http://www.win.tue.nl/~gino/solid/>).
- [3] M.J. Clayton, R.B. Warden, T.W. Parker, Virtual construction of architecture using 3D CAD and simulation, *Automation in Construction* 11 (2) (2002) 227–235.
- [4] D.W. Halpin, S. Riggs, *Design of construction and process operations*, Wiley, New York, N. Y., 1992, p. 598.
- [5] V.R. Kamat, J.C. Martinez, Automated generation of dynamic, operations level virtual construction scenarios, *Electronic Journal of Information Technology in Construction (ITcon)* 8 (2003) 65–84.
- [6] S.C. Kang, E. Miranda, Planning and visualization for automated robotic crane erection processes in construction, *Automation in Construction* 15 (4) (2006) 398–414.
- [7] B. Koo, M. Fischer, Feasibility study of 4D CAD in commercial construction, *Journal of Construction Engineering and Management* 126 (4) (2000) 251–260.
- [8] M.C. Lin, J.F. Canny, A fast algorithms for incremental distance computation, *Proceedings of IEEE Conference on Robotics and Automation*, Sacramento, CA, USA, vol. 2, 1991, pp. 1008–1014.
- [9] J.C. Martinez, P.G. Ioannou, General-purpose systems for effective construction simulation, *Journal of Construction Engineering and Management* 125 (4) (1999) 265–276.
- [10] B. Paulson, W. Chan, C.C. Koo, Construction operation simulation by microcomputers, *Journal of Construction Engineering and Management* 113 (2) (1987) 302–314.
- [11] R. Smith, Open Dynamic Engine, 2000 Available at: (<http://www.ode.org>).
- [12] A.F. Waly, W.Y. Thabet, A virtual construction environment for preconstruction planning, *Automation in Construction* 12 (2) (2003) 139–154.
- [13] B. Wilkins, J. Barrett, The virtual construction site: a web-based teaching/learning environment in construction technology, *Automation in Construction* 10 (1) (2000) 169–179.