PROCEEDINGS OF SCIENCE

OBB Intersect Test Algorithm Based on Effective Constraint

Lifang Bai
Information Engineering University, Zhengzhou, 450000, China
Email: mompidan@hotmail.com

Chaowen Chang, Yutong Wang, Zhao Ni, Ruiyun Wang
Information Engineering University, Zhengzhou, 450000, China
Email: wry0068@126.com

In light of collision detection problems in virtual environments, this paper makes a further study on the OBB(Oriented Bounding Box) intersection test.We describe the convex polyhedron by linear inequalities, achieve a matrix dimension reduction by simplifying the original coefficient matrix utilizing the corresponding row of the hyperplane which corresponds to the effective constraint inequality, and give a mathematical theoretical condition of the matrix dimension reduction, i.e, the effective constraint theorem. The algorithm only needs to operate the first column of matrix elements through the elementary transformation, and if necessary, related rows and columns will be rejected, then we can determine whether the test OBB intersects .Comparing the test speed and accuracy in the static case and dynamic case, experimental results show that speed of the new algorithm is 2-3times faster than that of the traditional OBB intersection algorithm, and the accuracy of the proposed algorithm is also not inferior to that of the SAT-based algorithm.

ISCC2015
18-19,December,2015
Guangzhou,China

PoS(ISCC2015)010

1. Introduction

The collision detection in the virtual environment is a common problem in the field of virtual reality, path planning for robots, computer game using 3D technology, physical simulation and engineering simulation. Over the past three decades, conventional algorithms for the collision detection always consist of two processes---the rough detection and precise detection[1]. Rough detection could reduce a quantity of objects that wouldn't make the collision pass fewer objects to the next precise detection[1-2] by using Bound Volume(BV), which is of large volume but simple geometric characteristics to describe complex virtual objects. Some of the typical and basic BV include Sphere[3], Axis Aligned Bounding Box (AABB)[4-5], Oriented Bounding Box (OBB)[6-7], k-Discrete Orientation Polytopes(k-DOP) [8-9], etc. All of them have their own advantages and disadvantages as well. An advantage common to Sphere and AABB is that they could be quickly constructed, simply expressed and available to make an intersection test. They have, however, a low fitting degree for tested objects, leaving many objects that should have been rejected. Especially, compared with Sphere and OBB, AABB needs a new alignment when those tested objects have a rotation, which increases the cost of calculation. k-DOP and OBB, however, have an excellent fitting degree. Furthermore, with the increasing value of k, k-DOP fits with the convex body of tested objects better than OBB. But OBB is better when testing geometric objects which can't reach a good alignment with the k-DOP axis and make an approximative approaching calculation[10]. Dueing the intersection test, k-ODP performs better than OBB in the test speed. It, however, has a fatal weakness----even there is rare collisions among objects, k-ODP still runs the update and alignment operation which results in a high cost [8][10].

High general performance makes OBB widely used in many collision detection fields of the rigid body. As we metioned before, there exists a high complexity in its intersection test. In order to solve this problem, we put forward the OBB intersection test algorithm based on effective constraint and present its reasoning process. The main idea of the algorithm is that we change the problem of OBB intersection test into a problem of simplification on the matrix, which results from whether the space constrainted by several lineal inequalities is null.

1.1 Relevant Research

In this section, we mainly describe relevant researches about the OBB intersection test algorithm and give analysis of them. The basic and most classical algorithm is proposed by Gottschalk S[6] which is based on Separating Axis Theorem (SAT). SAT results from Separating Theorem on Hyperplane in the field of Convex Analysis[11]----if convex bodies $A \cap B = \Phi$ $A, B \in R^{3\times 3}$, then, there must be a separating hyperplane P that can separate A and B. Sum up the two tested OBB's projection radiuses, and if the result is less than the projection distance between these two central points, then we call it OBB non-intersection. For example, as we can see in Fig. 1, when $|t \bullet l| > r_A + r_B$, A and B are disjointed where t is the displacement vector from A to B, and l is the unit directional vector of Axis L.

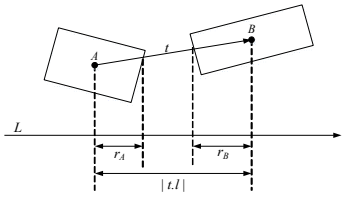


Figure 1: Separation Axis Theory

In a threee-dimensional virtual environment, we can determine whether there is an axis by searching for the potential separating axis. This algorithm needs to test 15 (3×2 are parallel to each plane's normal vector, 3×3 are cross products of vectors parallel to A and B's edges.) separating axes at most to examine whether OBB intersects. If there is no intersection on these 15 axes, we can conclude that it shows an OBB non-intersection; when meeting an intersection

on some axes, quit the test and conclude that it shows an OBB intersection. We should pay more attention to the order of these 15 axes when testing, sometimes, we can get the conclusion and quit the test after taking 6 of 15 axes.

Analysis of the algorithm: advantages of this algorithm are that it is based on a mature theory and provides a relatively precise test result by comprehensively checking every potential separating axis.In addition, its disadvantages are obvious----high cost of calculation because the algorithm involves a large number of operations such as addition, subtraction, point multiplication, cross multiplication and solving absolute value [12]. The robustness problem caused by that cross multiplication will reult in 0 vector when two vectors are parallel or approximately parallel, which can falsely report for the coincidence of separation. It can be adjusted by bringing in a small value ε to serve as a main controller, which can cause extra cost [10].

As to the weakness of the above algorithm, Bergen G[4] put forward a rough and simple algorithm SAT, which only needs to test the prior 6 axes. Bergen G pointed out that, mostly, we can get results after testing 6 of 15 axes. Considering the purpose of the intersection test (rejecting objects fast which can't have collisions), there is no need to achieve harsh precision. So it's worth to improve the performance of the OBB intersection test at the cost of accuracy. Furthermore, contribution from the last 9 axes to the eventual result is only 15%, and even if we ignore them, it can merely bring out a 6% error rate.

Analysis of algorithm: compared with the one in paper [10]. Obviously, the cost of this algorithm is small, and the robustness problem doesn't exist anymore. Its disadvantage is less accuracy, namely, there exists the error report for non-intersection instead of the intersection.

On the basis of papers [4][6], Chang J W,et al. held that some axes can bring earlier quit of the intersection test than others when two OBB are tightly close to each other. Based on this concept and a geometry intuition, and combined with the knowledge of Minkowski summation, they raised a selection method for the potential separation axis, conducted a series of tests using the greedy idea and reached a conclusion by selecting merely 5 of 15 axes to test.

Analysis of algorithm: this algorithm has made improvements in the detection speed compared with the prior two algorithms [12], but its frame of the intuition and greedy theory leaves a doubt in the aspect of accuracy and it also has the robustness problem owing to the same reason as what the algorithm raised in the paper [6].

All algorithms above are on the foundation of SAT, and differences between them consist in their method for selecting axes and the number of axes [7][14-15]. Similar to this thought, we will put forward a new intersection test algorithm based on the matrix operation to improve the performance of the OBB intersection test.

2. New Algorithm Proposed

2.1 Theoretical Basis

A closed convex polyhedron can be seen as an intersection of several finite half-spaces [20]. A half-space can be described by some inequalities, so the issue whether some convex polyhedrons intersect can be transferred into the issue whether the space enclosed by some linear inequalities is empty[21]. We give a description of a closed convex polyhedron as follows,

$$\begin{aligned} &a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n\le b_1\\ &a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n\le b_2\\ &\ldots\\ &a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n\le b_m \end{aligned} \quad ; \tag{2.1}$$

where m is the number of half-spaces, n is the dimension of space, so (2.1) can be written as (2.), called a matrix inequality.

$$A_0X\le b \quad ; \tag{2.2}$$

where A_0 is a $m*n$ matrix, X is a $n*1$ matrix, b is a $m*1$ matrix, so the issue whether two convex polyhedrons intersect, can be transferred into the following issue that

whether the space enclosed by several linear inequalities is empty in a three-dimensional space. For convenience, we assume several linear inequalities as follows:

$$\begin{aligned} &a_{11}x_1+a_{12}x_2+a_{13}x_3\le b_1\\ &a_{21}x_1+a_{22}x_2+a_{23}x_3\le b_2\\ &\cdots\\ &a_{m1}x_1+a_{m2}x_2+a_{m3}x_3\le b_m \end{aligned} \tag{2.3}$$

Let (2.3) be expressed as the form (2.2), then A_0 is a $m*3$ matrix, X is a $3*1$ matrix, and b is a $m*1$ matrix. Suppose $\alpha_i=(a_{i1},a_{i2},a_{i3})$, set that $S=\{X/\alpha_i X\le b_i(i=1,2,\ldots,m)\}$ is the solution set of (2.3). Then the definition of effective constraint in a three-dimensional space is given in the following.

Definition 1 (Effective Constraint): if the solution set S of the inequality system $\alpha_i X\le b_i(i=1,2,\ldots,m)$ meets $S=\{X/\alpha_i X\le b_i(i=1,2,\ldots i_0-1,i_0+1,\ldots,m)\}$, then we call the inequality $\alpha_{i0}X\le b_{i0}(i=1,2,\ldots,m)$ Ineffective Constraint, otherwise, we call it Effective Constraint($1\le i_0\le m$).

Definition 2 (Hyperplane[11]**): a**ssume α is a nonzero vector of an n-dimensional Euclid Space R and b is a real number, the set X whose every element in R meets $\alpha X=b$ is called Hyperplane in R.

We assume:

$Pos=\{i\mid a_{i1}>0\}$; $Neg=\{i\mid a_{i1}<0\}$; $Z=\{i\mid a_{i1}=0\}$; $S_1=\{X/\alpha_i X\le b_i, i\in Pos\}$; $S_2=\{X/\alpha_i X\le b_i, i\in Z\}$; $S_3=\{X/\alpha_i X\le b_i, i\in Neg\cup Z\}$.

Theorem 1 (Effective Constraint Theorem): if the convex space enclosed by some linear inequalities is not empty and elements of the first column in the coefficient matrix have both positive and negative numbers, then there exists $i\in Pos$, $j\in Neg$ which make $\alpha_i X\le b_i$ and $\alpha_j X\le b_j$ both effective constraints.

Proof: we prove there exists $i\in Pos$, which can make $\alpha_i X\le b_i$ the effective constraint (the proof $j\in Neg$ is similar to this). Suppose $\forall i\in Pos$, $\alpha_i X\le b_i$ are all ineffective constraints, then $S_3=S=S_3\cap S_2$, $S_3\subset S_1$. If $Z=\Phi$, for $\forall s_0\in R^3$, let $t_0=Max\{(b_k-\alpha_k s_0)/\alpha_{k1}, k\in Neg\}$, $s_1=(1,0,0)$, then $\forall t>t_0$, and we have $\alpha_{k1}t+\alpha_k s_0=\alpha_k(s_0+ts_1)<b_k(k\in Z\cup Neg)$, namely, $s_0+ts_1\in S_3$. For $S_3\in S_1$, $\forall t>t_0$, we get $s_0+ts_1\in S_1$, i.e., $\forall i\in Pos$,we have $\alpha_i(s_0+ts_1)<b_i(i\in Pos)$ (*). Let $t_1>Max\{(b_i-\alpha_i s_0)/\alpha_{i1}, i\in Pos\}$, and consider $t>Max\{t_0,t_1\}$. since $a_{i1}=\alpha_i s_1>0$, we have $\alpha_{i1}t+\alpha_i s_0=\alpha_i(s_0+ts_1)>b_i(i\in Pos)$, which contradicts the inequality (*). So the assumption is false. It is proved up.

2.2 the OBB Intersection Test Algorithm

OBB is a cube with directions, which can be seen as a special convex polyhedron. According to 3.1, its linear inequality systems can be described as $\begin{cases} PX\le U_1\\ -PX\le U_2 \end{cases}$, where P is a rotation matrix, U_1,U_2 are both $3*1$ matrixes. We assume linear inequality systems of the two tested OBB (A and B) are presented as follows:

$$\begin{cases} P_A X\le U_1\\ -P_A X\le U_2 \end{cases};\ \begin{cases} P_B X\le V_1\\ -P_B X\le V_2 \end{cases} \tag{2.4}$$

For the sake of less operation, the coordinate system of B can be transferred into that of A, so (2.4) can be written as:

$$\begin{cases} X\le P_A^{-1}U_1\\ -X\le P_A^{-1}U_2 \end{cases};\ \begin{cases} P_A^{-1}P_B X\le P_A^{-1}V_1\\ -P_A^{-1}P_B X\le P_A^{-1}V_2 \end{cases}$$

Let $P_0 = P_A^{-1}P_B$, then we can suppose : $A=\begin{bmatrix} E & P_A^{-1}U_1 \\ -E & P_A^{-1}U_2 \\ P_0 & P_A^{-1}V_1 \\ -P_0 & P_A^{-1}V_2 \end{bmatrix}$,

where E is a $3*3$ unit matrix. In a three-dimensional space, we call $a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 = b_i$ the equation determined by one corresponding to $a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 \le b_i$, and the equation determines a hyperplane. According to the effective constraint theorem, if two OBBs intersect, there must exist $i \in Pos$, $j \in Neg$ which make the hyperplane $\{X / \alpha_i X = b_i\} \cap S \neq \Phi$, $\{X / \alpha_j X = b_j\} \cap S \neq \Phi$. So we consider to use this hyperplane,which corresponds to the effective constraint inequality to simplify inequality systems. In other words, we just need to consider how to reduce the dimension of the matrix A. So reducing the matrix dimension is equal to finding the effective constraint inequality.

a: at first, there must exist both positive and negative elements in the first column of the matrix A, and the number of positive elements are equal to that of negative ones. Since $1 \in Pos$, we can use the first row to simplify other rows to make their first element equal to 0. Since the first row shows in the form of $(1\ 0\ 0\ *)$, we just need to consider the fourth column for simplification, and if the inequality corresponding to the first row is an ineffective constraint, we just need to use the next Row $i_0 (i_0 \in Pos)$ of the original matrix to do the simplification again.When i runs out, we can conclude $\alpha_i X \le b_i$ are all ineffective constraints, namely, $S = \Phi$. Otherwise, delete the row i_0 and the first column and continue the simplification. The process is described as follows:

$Gauss1(A)$ {

Step 1: find all rows of the matrix A whose first element (denoted by a_{i1}, $i \neq 1$) isn't equal to 0. Then use their own last element a_{i4} to minus the result after multiplying the last element a_{14} of the first row by a_{i1} ;

Step 2: during the operation, if there exists the row t whose newer last element $a_{t4} < 0$ and the rest of elements are all equal to 0, then quit this step and turn to Step 3; else, turn to Step 4.

Step 3: if there is still an element in the positive set never being used (of the original matrix A), get new $i_0 \in Pos$ and use Row i_0 to do the corresponding operation on the rest of rows to make their first element equal to 0 (namely, $a_{i1} = 0 (i \neq i_0)$), then turn to Step 2. If all elements of the positive set have already been used once, return NULL.

Step 4: reject the Row i_0 and Column 1 of the matrix A to get a new matrix A_1, return A_1 }

b: the concept about the following simplification of the matrix is same as $Gauss1(A)$, but here we are facing a new situation where the number of positive and negative elements of the first column in the new matrix A_1 may not be same. We denote the numbe of the set S as $|S|$. We will choose the smaller one from $|Pos|$ and $|Neg|$ and use the row, which the element means, in the chosen set to simplify our new matrix.

$Gauss2(A)$ {

Step1 when we have $|Pos| \le |Neg|$, select the Pos set; when having $|Pos| \ge |Neg|$, select the Neg set, then turn to Step 2.

Step 2: if there is still an element in the chosen set never being used (of the original matrix A), get new i_0 , use Row i_0 to do the corresponding operation on the rest of rows to make their first element equal to 0 (namely, $a_{i1} = 0 (i \neq i_0)$) and turn to Step 3, else, return NULL.

Step 3: during the **operation**, if there exists some row t whose newer last element $a_{t4} < 0$ and the rest elements are all equal to 0, then quit this step and turn to Step 2; else, turn to Step 4.

Step 4; reject the Row i_0 and Column 1 of the matrix A to get a new matrix A_1, return A_1 .}.

c: after $Gauss2(A)$, if the new matrix is not null, it only has two columns. So we can not have a rejection or return a newer matrix to improve the algorithm for executing efficiency. Return the BOOL value directly.

$BOOL\ Gauss3(A)$ {

Step 1: if $|Pos| \le |Neg|$, select the Pos set; else, select the Neg set, then turn to Step 2.

Step 2: if there is an element in the chosen set never being used, get new i_0 and use the row i_0 to do the corresponding operation on the rest of rows to make their first element equal to 0 (namely, $a_{i1}=0(i \neq i_0)$), then turn to Step 3, else, return FALSE.

Step 3: during the operation, if there exists some row t whose newer last element $a_{t4} < 0$ and the rest of elements are all equal to 0, then quit this step and turn to Step 2; else, return TRUE.}.

d: enter the function of the OBB intersection test :

$BOOL\ OBBtest(P_A, P_B, U_1, U_2, V_1, V_2)$ {

Step 1; $A_0 = P_A^{-1} P_B$, $A = \begin{bmatrix} E & P_A^{-1}U_1 \\ -E & P_A^{-1}U_2 \\ P_0 & P_A^{-1}V_1 \\ -P_0 & P_A^{-1}V_2 \end{bmatrix}$;

Step 2: $A_1 = Gauss1(A)$;

Step 3: if $A_1 = NULL$, return FALSE; otherwise, $A_2 = Gauss2(A_1)$;

Step 4. if $A_2 = NULL$, return FALSE; otherwise, $A_3 = Gauss3(A_2)$.}

Obviously, there is no need to consider other situations, because the new matrix must have both positive and negative elements after each rejection, when $BOOL\ Gauss3(A)$ returns TRUE, we can conclude that intersection happens.

3. Experiment Results and Analysis

3.1 Experimental Contents and Environment

a. Comparison Objects

The accuracy of the algorithm proposed by Chang JW remains to be verified, and taking into account the application breadth and the commercial acceptance of the algorithm, our experiment mainly compares our algorithm with the most classic testing algorithms (the exact algorithm SAT) based on the separating axis theorem proposed by Gottschalk S and the SAT lite algorithm proposed by Bergen G (the Fast SAT algorithm).

b. Comparison Contents

In this paper, we mainly consider the test speed, accuracy, memory consumption of the algorithm. Theoretically, the relative error of the new algorithm is 0 (see the accuracy analysis below), but in the actual operation, the calculation errors' accumulation may cause false positive rate which is the cause of using floating-point type storage, and the error is an absolute and inevitable error. Thus the proposed algorithm theoretically has more predictable advantage over those traditional SAT algorithms in accuracy. In storage consumption, they are about the same, since both use the expression that contains one rotation matrix and two vectors which have a particular meaning when storing the OBB. So this experiment aims to compare the test speed

and accuracy of intersection test algorithms. In summary, the general idea of the experiment is shown in Table 1.

Experimental contents	Comparison objects	Comparison contents
Static Comparison	Exact SAT algorithms	Test speed
Dynamic comparison (Rotation,translation)	Fast SAT algorithm	Accuracy

Table 1: Shows The General Idea of the Experiment

c. Experimental Case Description

In order to ensure the algorithm is feasible in engineering scenarios such as superiority in the field of the practical application, this paper discusses the static and dynamic test by comparing two kinds of cases. The static test case aims to test whether the relatively static OBB objects intersect in the scene as shown in Fig. 1. (a) and (b) are the case of relatively static OBB objects to be tested for intersection and non-intersection respectively.

(a) Static Intersection (b) Static Non-intersection

Figure2: Static Case

Naturally, the dynamic test case aims to test whether the relatively motional OBB objects intersect. As the movement of objects can be abstracted as rotation and translation, when the objects rotate or translate, the OBB can have the same action. So in this paper, the dynamic case includes rotation and translation as shown in Fig. 3 and 4 respectively. We only consider the initial state for the non-inersection, while in the intersection state it shows a similar result.

(a) Dynamic Rotation Intersection (b) Dynamic Rotation Non-intersection

Figure 3: Dynamic Rotation Case

(a) Dynamic Translation Intersection (b) Dynamic Translation Non-intersection

Figure 4: Dynamic Translation Case

d. Experimental Environment

The experimental environment is shown in Table .2 and 3 respectively. The number of static and dynamic cases is 10000 and 1000 respectively. In the actual application scenario, and in most cases, only a few objects make relative movement, while most of the OBB for detection is relatively static, so the number of case set is reasonable.

Case number.	System	CPU	RAM	Execution Environment
10000(Static) 1000(Dynamic)	Win7(32)	Intel(R) Core(TM) I3-2330M CPU@2.20GHz	2.92G	Matlab (R2010b)

Table 2 :Experimental Environment

3.2 Experimental Results and Analysis

First, this paper compares the proposed algorithm with the exact SAT algorithm in static and dynamic cases. Details are presented as follows.

a. Static Case

We make a comparison between the new algorithm and exact SAT algorithm in the static intersection case and non-intersecttion case, and test results of the test speed are shown in Fig. 5 and Fig. 6 respectively. Table 3 is a data record with a case number of 1000(in which 1 means the new algorithm and 2 means the exact SAT algorithm). The execution time (time execution, ET) unit is milliseconds (ms). We consider the standard deviation (10^{-3}) of the execution time to estimate the stability of the algorithm and measure the accuracy of the algorithm with false positives. It can be seen that, from the experimental results and data, the new algorithm performs better than the exact SAT algorithm in the case of the intersection or non-intersection of the OBB object, which is about 3 times of the SAT algorithm.

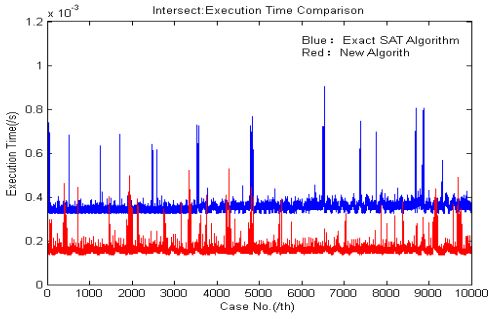


Figure5: Static Intersection Comparison

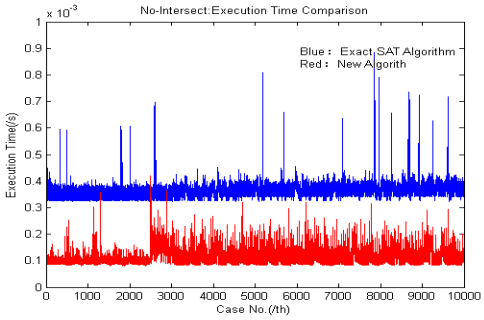


Figure 6 : Static Non-intersection Comparison

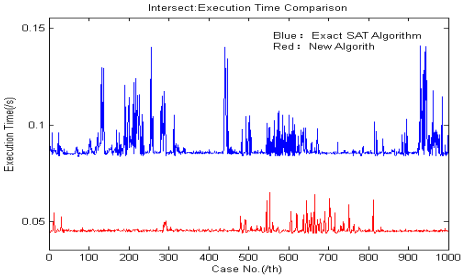
	Max ET		Min ET		Ave ET		ET SD		False	
	1	2	1	2	1	2	1	2	1	2
Inter.	0.86	1.2	0.13	0.32	0.15	0.36	0.03	0.04	2	7
Non Inter.	0.42	0.88	0.08	0.33	0.11	0.36	0.03	0.03	0	0

Table 3: Static Execution time (ET) Comparison

b. Dynamic Rotating Case

Now we consider the dynamic rotation of OBB objects (the initial state for the non-intersection, which is same for the below)---having a rotation of 360 degree. There are two situations: the occurrence of intersection or always non-intersection. If the intersection occurs, we compare the new algorithm with the exact SAT algorithm to examine which can detect the intersection earlier. So the experiment is executed from the beginning of the OBB rotation to the detection of intersection. If there is always non-intersection, the case of the dynamic rotation is same as that in the static case, and the test time both consume is determined by the number of the non-intersection test, without a contrast experiment which can provide direct reference to static case Similarly, so is for the dynamic translation case.

Comparison results between the proposed algorithm and the precise SAT algorithm under the dynamic rotation case are shown in Fig .7 and Table 4. From experimental results and data, the algorithm can detect the intersection of the SAT algorithm in the process of the rotation of the OBB object, which is 2 times faster than the SAT algorithm.



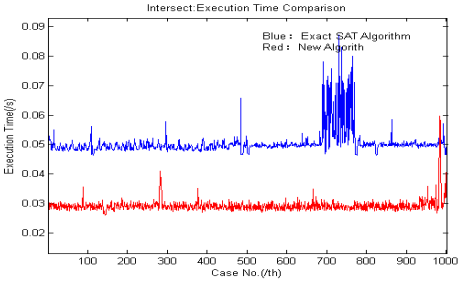


Figure 7: Dynamic Rotation Inter.Comparison **Figure 8:**Dynamic Translation Inter.Comparison

Max ET		Min ET		Ave ET		ET SD		False	
1	2	1	2	1	2	1	2	1	2
65.2	139.7	44.1	83.8	45.7	88.6	2.0	7.6	0	0

Table 4: Dynamic Rotation ET Comparison

c. Dynamic Translation Case

Similarly, comparison results between the proposed algorithm and the precise SAT algorithm under the dynamic translation case are shown in Fig. 8 and Table 5, where we can see that the new algorithm can detect the intersection of OBB objects in the translation process, and the speed is nearly 2 times faster than the SAT algorithm.

Max ET		Min ET		Ave ET		ET SD		False	
1	2	1	2	1	2	1	2	1	2
58.9	90.9	26.0	46.3	29.0	50.5	2.2	4.5	0	0

Table 5: Dynamic Translation ET Comparison

Then we compares the new algorithm with fast SAT algorithm in this paper. Specific steps mentioned above are presented as well.The experimental data are shown in Table 6.

From the table, we can see that whether it is in the static or dynamic OBB intersect test, the new algorithm is superior to the traditional SAT in the test of speed. It is important to note that the number of false positives is zero in the dynamic case. The reason is that the OBB always intersect in a certain period of time, and after the intersection occurs, OBB moves away. During this peiod, the new algorithm and the traditional one can detect the intersection, but they are just different in when the detection occurs.

	Max ET		Min ET		Ave ET		ET SD		False	
	1	2	1	2	1	2	1	2	1	2
Static	0.88	0.92	0.12	0.27	0.14	0.29	0.02	0.15	4	13
	0.36	0.09	0.07	0.28	0.12	0.26	0.03	0.11	0	0
Rotation	54.1	112.4	44.3	74.2	40.6	76.7	0.33	1.82	0	0
Translation	40.6	84.3	29.2	46.4	25.5	44.8	0.45	1.65	0	0

Table 6: Fast SAT Algorithm/New Algorithm Comparison

d. Expanding Experiment

Test will not exit until the traditional algorithm detects intersection. In other words, it will conduct all the 15 potential separating axes tests so that the result of non-intersecton is obtained. However, it is opposite for the new algorithm, the test will exit as long as the non-intersection is detected. If intersection occurs, it performs the matrix reduction operation three times at most. Of course, each matrix reduction operation has numbers of the primary row transformation. In theory, the traditional algorithm in the OBB intersection test is faster than the non-intersection, yet the new algorithm shows just the opposite side. In the virtual environment, only a few objects are close to each other and possible to collide in most cases and most OBBs maintain in a non-intersection state at the moment. In other words, the proposed algorithm in this paper will exit earlier in most cases and all tests are executed only when OBB intersects, which indicates the overall speed will increase further and it will hold more superiority than the traditional algorithm.

In order to verify the above argument, this paper validates the result shown in Fig. 8. Experiment data are shown in Table 7 where 1 means intersection, and2 means non-intersection.

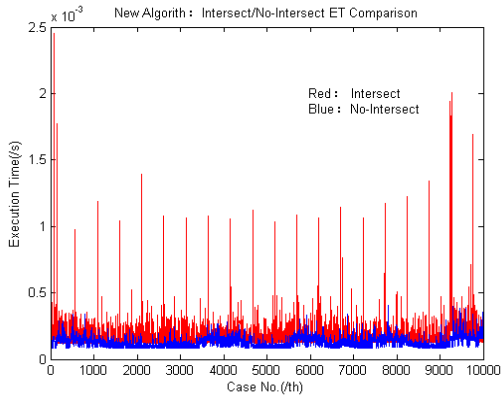


Figure 9: New Algorithm Inter./Non-inter.Comparison

Max ET		Min ET		Ave ET		ET SD	
1	2	1	2	1	2	1	2
2.4	0.48	0.12	0.08	0.14	0.10	0.06	0.03

Table 7: New Algorithm Inter./Non-inter.ET Comparison

3.3 Accuracy Analysis

Based on the algorithm description in 2.2, every prerequisite for selected rows of the new algorithm in the matrix of each dimension reduction is that the inequality is an effective constraint, i.e., if the solution set is not empty, the corresponding hyperplane intersection of the reconciliation set is not empty, that is to say, if the solution set is not empty, there must be a solution concentration points on the hyperplane. So every dimension reduction is a matrix of accuracy, and it is not at the expense of the precision until determining it is in the intersection or not. In theory, the relative error is 0.

The traditional precise SAT algorithm is equivalent to iterating through all the possible separation axes. The relative error is close to zero (cross-product results close to 0 may lead to false positives in related researches as mentioned above). Fast speed of the SAT algorithm is at the expense of a certain accuracy to improve the speed of the test.

As a result, in terms of accuracy of the new algorithm, it is better than the traditional SAT algorithm. The previous experimental data can also confirm this conclusion.

4. Conclusion

Based on the proposed mathematical theory conditions of the matrix dimensionality reduction, namely, the proposed effective constraint theorem, this paper utilizes a certain row to simplify the original matrix. The result shows that if there is no valid constraints during the simplifying process, it means non-intersection, otherwise, reject related ranks to achieve dimensionality reduction, then continue the simplification; if it eventually returns TRUE, it means intersection, otherwise, it means non- intersection.

Experiments under static and dynamic cases (rotation and translation) show that the test speed of the proposed is 2 to 3 times faster than the traditional OBB intersection test, and the accuracy is also guaranteed. Eventually, accuracy of the new algorithm and the traditional one is analyzed. The expanding experiment also shows that when the new algorithm is applied to the specific virtual environment, its advantages will become more apparent.

References

[1] Weller R.*A Brief Overview of Collision Detection*[M]//New Geometric Data Structures for Collision Detection and Haptics.Springer International Publishing,2013: 9-46.

[2] Wang Y,Hu Y,Fan J,et al.*Collision Detection Based on Bounding Box for NC Machining Simulati -on*[J].Physics Procedia,2012,24: 247-252.

[3] Hubbard P M.*Approximating polyhedra with spheres for time-critical collision detection*[J].ACM Transactions on Graphics (TOG),1996,15(3): 179-210.

[4] Bergen G.*Efficient collision detection of complex deformable models using AABB trees*[J].Journal of Graphics Tools,1998,2(4): 1-13.

[5] Cai P, Indhumathi C, Cai Y, et al. *Collision detection using axis aligned bounding boxes*[M]// Simulations,Serious Games and Their Applications.Springer Singapore,2014: 1-14.

[6] Gottschalk S,Lin M C,Manocha D.*OBBTree: A hierarchical structure for rapid interference detection*[C]//Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.ACM,1996: 171-180.

[7] Eberly D.*Dynamic collision detection using oriented bounding boxes*[J].Geometric Tools,Inc,2002.

[8] Klosowski J T,Held M,Mitchell J,et al.*Efficient collision detection using bounding volume hierarchies of k-DOPs*[J].Visualization and Computer Graphics,IEEE,1998,4(1): 21-36.

[9] Tao N,Li-juan W,Bo L I.*A Novel Method Based on K_DOPs and Hybrid Bounding Box to Optimize Collision Detection*[J].Journal of Convergence Information Technology,2012,7(12).

[10] Ericson C.*Real-time collision detection*[M].CRC Press,2004.

[11] Boyd S,*Vandenberghe L.Convex optimization*[M].Cambridge university press,2004.

[12] MA Deng-wu,YE Wen,LI Ying.*Survey of Box-based System Simulation*,2006,18(4): 1058-1064.

[13] Chang J W,Kim M S.*Efficient triangle–triangle intersection test for OBB-based collision detection*[J].Computers & Graphics,2009,33(3): 235-240.

[14] Ding S,et al.*Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces*[J].Computer-Aided Design,2004,36(13): 1281-1294.

[15] Tang T D,et al.*A new collision avoidance strategy and its integration with collision detection for five-axis NC machining*[J].The International Journal of Advanced Manufacturing Technology,2015: 1-12.

[16] Gan Jianhong,Peng Qiang,Dai Peidong,et al.*Improved Collision Detection Algorithm Based on Oriented Bounding Box*[J].Journal of System Simulation,2011,23(10)：2169-2173(in Chinese).

[17] Liu Xiaoping,Zhang Yingkai,Xie Wenjun,et al.*Collision Detection Algorithm Based on Sphere-OBB Bounding Box for Character Animation.*Journal of System Simulation,2014,7: 022(in Chinese).

[18] Zhu Yuanfeng,Meng Jun,et al.*Research on Real-time Collision DetectionBased on Hybrid Hierarchical Bounding Volume.*Journal of System Simulation,2008,20(2): 372-377 (in Chinese).

[19] Chen Wei,Ma Ruijin,Zheng Wenting,et al.*Efficeient processing of meshiness geometrical data based on OBB tree structure*[J].Chinese Journal of Computers,2007,30(2): 330-336.

[20] Rockafellar R T.*Convex analysis*[M].Princeton university press,1-3, 2015.

[21] Ren Shijun,Hong Bingrong.*A Fast Algorithm to Determine Whether Convex Regions Bounded byMultiple Linear Constraints Are Empty*[J].Chinese Journal of Computers,1998,21(10): 896-901.