Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Using LBG quantization for particle-based collision detection algorithm

SAENGHAENGTHAM Nida, KANONGCHAIYOS Pizzanu

(*Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand*)

E-mail: g47nsn@cp.eng.chula.ac.th; pizzanu@cp.eng.chula.ac.th

Received Apr. 11, 2006; revision accepted Apr. 24, 2006

Abstract: Most collision detection algorithms can be efficiently used only with solid and rigid objects, for instance, Hierarchical methods which must have their bounding representation recalculated every time deformation occurs. An alternative algorithm using particle-based method is then proposed which can detect the collision among non-rigid deformable polygonal models. However, the original particle-based collision detection algorithm might not be sufficient enough in some situations due to the improper particle dispersion. Therefore, this research presents an improved algorithm which provides a particle to detect in each separated area so that particles always covered all over the object. The surface partitioning can be efficiently performed by using LBG quantization since it can classify object vertices into several groups base on a number of factors as required. A particle is then assigned to move between vertices in a group by the attractive forces received from other particles on neighbouring objects. Collision is detected when the distance between a pair of corresponding particles becomes very small. Lastly, the proposed algorithm has been implemented to show that collision detection can be conducted in real-time.

Key words: Collision detection, Deformable object, Particle, LBG, Vector quantization
doi:10.1631/jzus.2006.A1225 **Document code:** A **CLC number:** TP39

INTRODUCTION

Collision detection is an important task in many fields such as robotics, computer games, computational geometry (Gottschalk *et al*., 1996), computer simulation, virtual reality, etc. Most collision detection algorithms work efficiently only with solid and rigid objects, so collision detection between deformable objects is a challenge. A number of researches on non-rigid objects are proposed such as cloth simulation (Bridson *et al*., 2003; Teschner *et al*., 2004) and biological structures (Raghupathi *et al*., 2003). As the object shape is not static, most algorithms based on bounding structure cannot detect collisions effectively. Bounding volume hierarchical method (Hubbard, 1996; Palmer and Grimsdale, 1995; Quinlan, 1994; Cohen *et al*., 1995; Held *et al*., 1995) has to recalculate its bounding representation every time surface deformation occurs and its cost is quite expensive.

An approach using particle determination (Senin *et al*., 2003) is then proposed which relies on the idea of interacting particles distributed on a surface. It can be applied to deformable object since recalculation is not required. However, the efficiency of this particle-based algorithm might decrease when there are several objects which can cause improper particle dispersion. Therefore, this research presents an improved particle-based collision detection algorithm for deformable surface. In order to enhance the efficiency of particle dispersion, each particle is assigned to control in its separated area. In the proposed algorithm, we apply the vector quantization (Abut *et al*., 1992; Linde *et al*., 1980; Gersho and Gray, 1992) with LBG algorithm (Linde *et al*., 1980) to partition the surface area. LBG quantization is a technique to classify vector data into several groups. The advantage of this technique is its ability to consider many factors by setting the value of each determinant as dimension of a vector.

It is shown that the proposed algorithm can efficiently detect the collision of several objects in real-time. The remainder of this paper is divided into six sections. In the next Section, we discuss some previous collision detection methods. After that, a new particle-based collision detection algorithm is presented. In Section 4, the analysis of proposed algorithm is described, followed by the result in Section 5 and conclusion in the final section.

COLLISION DETECTION

One of the most interesting problems in computer animation is the widely researched collision detection. There are many techniques developed to increase the detecting efficiency and, at the same time, shorten the computing time such as Spheres (Hubbard, 1996; Palmer and Grimsdale, 1995; Quinlan, 1994), Axis Aligned Bounding Boxes (AABBs) (Cohen *et al*., 1995; Held *et al*., 1995), Oriented Bounding Boxes (OBBs) (Gottschalk *et al*., 1996), K Dops (Klosowski *et al*., 1998; Zachmann, 1998), Quantized Orientation Slabs with Primary Orientations (QuOSPOs) (He, 1999) and Spherical shells (Krishnan *et al*., 1998). These techniques seem to be effective for solid and non-deformable objects. However, the usage of these techniques with deformable objects has some drawbacks due to their restructuring process which is not efficient in real-time. The entire recalculation for bounding representation is required to be processed every time surface deformation occurs. It is not the case for a method based on particle determination (Zachmann, 1998; Parent, 2001) using interactive forces between particles. This method offers significant benefit since the interaction forces between particles can be calculated every time objects deformation occurs without any recalculation of shape bounding structure.

The particle-based collision detection algorithm (Senin *et al*., 2003) uses the interactions forces between particles that spread over the objects surface as the main principle as shown in Fig.1. Particles are randomly spread over the surface and charged with same-charged ions, for particles on the same objects, and different-charged ions, for the particles in distinct objects. There are 2 types of forces acting on each particle which can be calculated as follows:

(1) The attraction between particles that are differently charged, in order to pull the particles into each other when the distinct objects move closer.

(2) The repulsion between particles that have the same charge to prevent the particles aggregation.

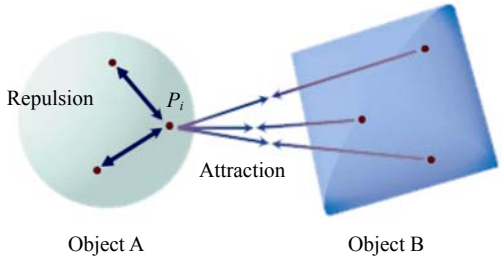

Fig.1 Particle-based collision detection

The position of each particle can be estimated from the summation of attractive and repulsive forces. The collision can be detected by measuring the distance between particles. When the particles come closer enough, there is high probability of collision leading to the investigation of the collision between those corresponding areas.

However, the particle dispersion might not be efficient when there are more than 2 objects in the system. This is because when the other object is coming close, it can attract all particles agglomerated at one side. If there is another object crashing on another side, it may not have enough attractive forces to pull the particles back, so collision detection fails.

Moreover, this method might have less efficiency when the objects have extreme expansion. As the number of particles is initially fixed, it may not be sufficient enough to cope with the broadened surface area.

Besides, the random placement of particles can lead to improper particle dispersion and thus decreases the collision detection efficiency. For example, as shown in Fig.2, three particles are randomly dispersed at the right hand side of object B and so that collision cannot be detected when object A crashes on the other side.

ALGORITHM

An adaptive collision detection algorithm is described in this section. First, in order to eliminate the inappropriate dispersion problem, an object surface is

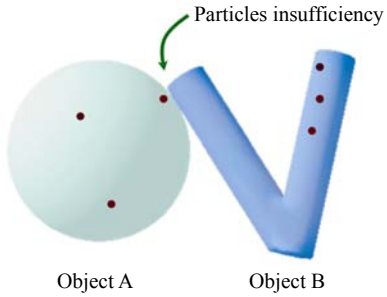

Fig.2 Undetected collision caused by improper particle dispersion at the initialization

divided into several small areas with each area filled with a particle moving only within its corresponding boundary as shown in Fig.3. This can assure that the particles always cover the entire surface. Therefore, this can avoid the insufficient particles case when more than one object crashes on another object at the same time. After that, attractive forces received from particle on different objects are applied. Consequently, when objects come closer, the particles at the corresponding surface will also move into each other. The distance between this pair of particles will be observed and determined. If it is shorter than some tolerable distance, the collision is then examined precisely between these two corresponding areas.

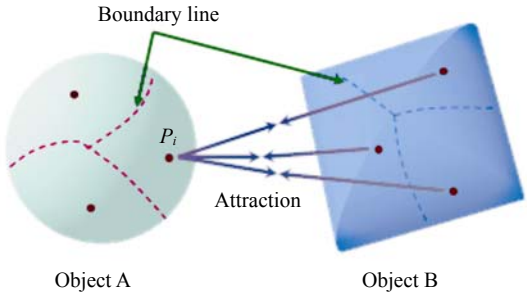

Fig.3 Interaction force on particle P_i based on proposed algorithm

Partitioning process

To partition surface area into several groups, we have to know the number of partitions which is equal to the number of particles needed. Hence, the number of particles needed for each object is first estimated, followed by the surface partitioning.

(1) The number of particles

A proper number of particles for each object can be calculated as the maximum number of objects that can collide with the object. This can ensure that there are enough particles to detect the collision of several objects at the same time. In this algorithm, we apply the kissing number (Conway and Slone, 1993a; 1993b) to estimate the maximum number of objects that can attach to an object at the same time.

Kissing number is the number of equivalent spheres which can touch an equivalent sphere without any intersections. The kissing number in three dimensions has been proved to be 12, as shown in Fig.4.

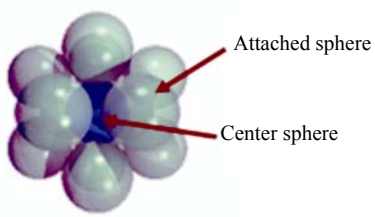

Fig.4 Twelve equivalent spheres which can touch an equivalent sphere without any intersections

Therefore, 12 can be used as the number of particles when there are all the equivalent spheres in the scene. If there are less than 12 objects, the number of objects can be used instead.

However, this number of particles might not be sufficient when there are several sizes of spheres. The number of particles needed for an object can be estimated as follows. The maximum number of the smallest sphere in the scene that can touch the object can be computed as follows: let X be the considered object and Y be the smallest sphere in the scene. Deposit the whole arrangement into an imaginary sphere, as shown in Fig.5. Imagine a lamp at the centre of the object X that casts shadows of surrounding spheres onto the inside surface of the imaginary sphere. Each circular shadow has an area of B and cannot overlap.

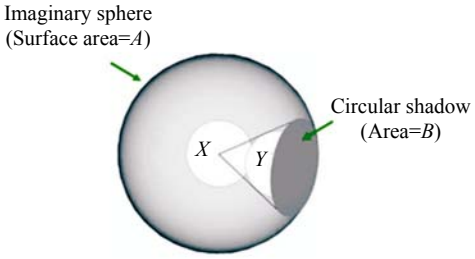

Fig.5 Circular shadow on the imaginary sphere covering a spherical object X and the smallest spherical object Y

Then the number of particles needed for X is not more than A/B when A is the surface area of the imaginary sphere.

We can also apply this technique to an object that has other shapes apart from sphere by completing a modification. First, each object is approximated as a collection of spheres by analyzing model geometry as shown in Fig.6. To locate a collection of spheres, all vertices have to be checked for critical points whose normal vector changes much. The suitable number of particles can then be achieved for each part of the object.

Fig.6 Approximated spheres for objects

(2) Surface partitioning

In order to partition object surface into a number of areas which can be calculated from the previous subsection "The number of particles", the object vertices must be grouped. Each group is filled with a particle that can only move between the vertices in the same group.

In this algorithm, LBG quantization (Linde *et al*., 1980) is applied to partition surface area. LBG quantization is a grouping technique which classifies vector data into several groups. The value of each group can be represented by its average value called code vector as shown in Fig.7. As the input data can be unlimited dimension vector, this technique can therefore efficiently partition object surface by considering several factors such as vertex position, normal vector, and color, etc. Each determinant is arranged into each vector dimension which, in this algorithm, has 2 factors to consider as will be described in the following paragraph.

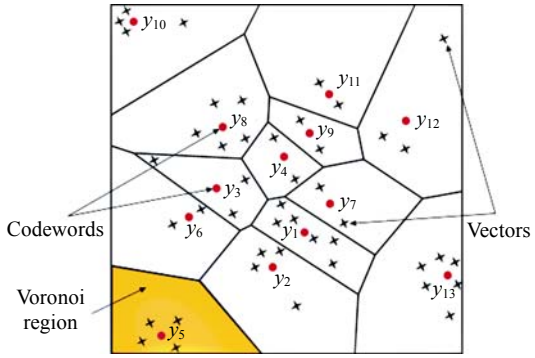

Fig.7 Vector quantization

Since some closed adjacent vertices should not be included in the same area due to their different planes, the input vectors for partitioning should not be dependent only on their positions, but should be dependent also on their normal vectors. Therefore, each input vector for the surface partitioning process is received as a 6-dimensional vector composed of vertex coordinate and its normal vector. The average value is then computed which can divide the data into 2 groups. Each part can be divided continuously until the required number of groups is met.

The algorithm for partitioning vertices is shown below:

1. Each vertex is converted into a 6-dimensional vector form

$$\text{Input vector: } \boldsymbol{X}_m=(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6),$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ represent vertex position and $\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6$ represent its normal vectors.

The set of all source vectors M is:

$$\text{Training sequence: } (\tau)=(\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_M).$$

2. The precision ε of the optimization process is set; $\varepsilon>0$ has a small value.

3. Dimension of data is specified, k=6.

4. The first initial code vector is estimated, by setting the number of required code vectors N=1. The average value of all source vectors can then be calculated as follows:

$$\boldsymbol{C}_1^* = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{X}_m. \tag{1}$$

After that, the squared error distortion value at point $\boldsymbol{C}_1^*$ is computed:

$$D_{\text{ave}}^* = \frac{1}{kM}\sum_{m=1}^{M}\| \boldsymbol{X}_m - \boldsymbol{C}_1^* \|^2. \tag{2}$$

5. Each reference code vector, $\boldsymbol{C}_i^*$, is used as a divider and 2 other code vectors for each part can be found. The procedure is repeated for all code vectors used as dividers.

For i=1, 2, ..., N

$$\boldsymbol{C}_i^{(0)} = (1+\varepsilon)\boldsymbol{C}_i^*, \tag{3}$$

$$\boldsymbol{C}_{N+i}^{(0)} = (1-\varepsilon)\boldsymbol{C}_i^*. \tag{4}$$

6. The N value is then doubled, N=2N.

7. Iteration process

7.1. Source vectors are classified into groups using the Euclidian distance between each input vector and its corresponding code vector.

Let iteration index i=0

For m=1, 2, ..., M find the lowest value of

$$d(\boldsymbol{C}_n^{(i)}, \boldsymbol{X}_m) = \sqrt{\sum_{j=1}^{k}(C_{nj} - X_{mj})^2} \tag{5}$$

For n=1, 2, ..., N

If $\boldsymbol{C}_{n^*}^{(i)}$ is a code vector that creates the lowest value, then put it into the group.

$$Q(\boldsymbol{X}_m) = \boldsymbol{C}_{n^*}^{(i)} \tag{6}$$

7.2. New code vectors are calculated from the average in each group when the grouping is finished.

For all value n=1, 2, ..., N

$$\boldsymbol{C}_n^{(i+1)} = \sum_{Q(V_m)=\boldsymbol{C}_n^{(i)}} X_m \Big/ \sum_{Q(V_m)=\boldsymbol{C}_n^{(i)}} 1. \tag{7}$$

7.3. Set i=i+1

7.4. Calculate

$$D_{\text{ave}}^{(i)} = \frac{1}{Mk}\sum_{m=1}^{M} \| \boldsymbol{X}_m - Q(\boldsymbol{X}_m) \|^2. \tag{8}$$

7.5. Stable state is checked by considering the following condition:

$$\frac{D_{\text{ave}}^{(i-1)} - D_{\text{ave}}^{(i)}}{D_{\text{ave}}^{(i-1)}} \le \varepsilon. \tag{9}$$

The difference between $\boldsymbol{C}_n^{(i)}$ and $\boldsymbol{C}_n^{(i-1)}$ are still too much if the condition is not as in Eq.(9). Thus this iteration process must be repeated until it is stable.

8. When the code vectors are already stable, they are set as reference code vectors ($\boldsymbol{C}_i^*$).

Let $D_{\text{ave}}^* = D_{\text{ave}}^i$,

For n=1, 2, ..., N

Set $\boldsymbol{C}_n^* = \boldsymbol{C}_n^{(i)}$.

9. We repeat Step 4 to Step 8 until the required number of the code vectors is reached.

10. Input vector ($\boldsymbol{X}_m$) is then classified into each group as in Step 7.1. All N groups of 6-dimensional vectors are then achieved.

Each 6-dimensional vector is converted into a vertex element by extracting the first 3 elements of the vector. Lastly, N groups of vertices, N areas, can be achieved.

11. A particle is assigned to each area.

Collision detection process

The procedure of this process is divided into 3 parts. First part is state the calculation of interaction forces between particles. Then, the movement of particles is discussed in the second part. Finally, the collision checking can be processed by considering the collision distance.

(1) Interaction forces

The interaction forces applied to each particle are calculated by summarizing the attractive force received from other particles on the neighboring objects. This attractive force corresponds to the distance between particles which can be computed as follows:

$$f(p_i,p_j)=0 \quad \text{when } r \ge R_{\text{eff}}, \tag{10}$$

$$f(p_i,p_j)=1/r^2 \quad \text{when } r < R_{\text{eff}}, \tag{11}$$

where $f(p_i,p_j)$=Force between particle i and j; r=Distance between particles; R_{eff}=The effective radius of attraction.

According to these equations, the attractive force becomes stronger when the distance becomes shorter. In contrast, if the distance becomes longer than the effective radius of attraction, then the attractive force becomes zero. This value can be approximately equal to the longest distance between the initial particles of each object.

Finally, the forces acting on each particle, p_i, is then obtained from the summation of attractive forces acting on the particle.

$$F_{p_i} = \sum_{j}^{N_p} f(p_i, p_j). \tag{12}$$

(2) Movement of particles

In the simplest case, particles are assigned to be located only at vertices within its specific area. Each particle is moved along the vertices to the neighboring vertex which received maximum force. To choose particle destination, we have to know the particle position (P_i) and its neighboring coordinates ($Q_{P_i,j}$) as shown in Fig.8. The forces acting on these vertices are then calculated using Eq.(12) as shown in the previous subsection "Interaction forces". After that, each particle is moved to the vertex which received maximum force. Finally, the process is checked if it changes to the relaxed state which can be defined as follows.

The relaxed state is a state when either the number of particles moved is less than $R_{\min}$ (when $R_{\min}$ should be about 10 times smaller than number of particles, N), or else, the number of iterations becomes equal to $R_{\max}$ (when $R_{\max}$ should be approximately equal to the maximum number of vertices along the shortest routes between any two particles on the object).

The movements of N particles are calculated and then wait for the particle system to reach the relaxed state.

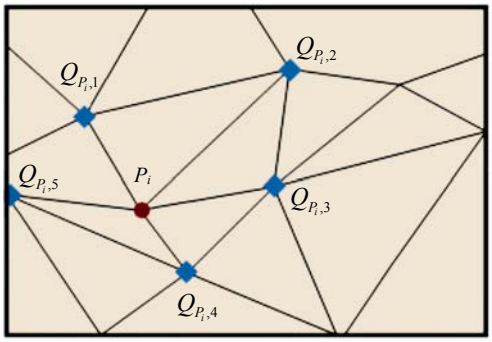

Fig.8 Particle position and its neighboring coordinate

(3) Collision distance

After the relaxed state is reached, the distance between a pair of the closest particles is then measured. We compare this value to the collision distance (μ) which can be calculated as the Euclidian distance between vertices shown in Eq.(13).

$$\mu=avg[d(\boldsymbol{v}_i,\boldsymbol{v}_j)]|\boldsymbol{v}_i,\ \boldsymbol{v}_j\in V, \tag{13}$$

where V is the set of vertex coordinates and $d(\boldsymbol{v}_i,\boldsymbol{v}_j)$ is the Euclidian distance between vertex $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ which can be calculated from the following equation:

$$d(\boldsymbol{A},\boldsymbol{B})=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}, \tag{14}$$

where $d(\boldsymbol{A},\boldsymbol{B})$ is the Euclidian distance between 2 points $\boldsymbol{A}=(a_1,a_1,\ldots,a_n)$ and $\boldsymbol{B}=(b_1,b_1,\ldots,b_n)$.

When the distance is shorter than μ, it can be concluded that there is high possibility of collision. Then, the collision at the corresponding area is precisely checked.

Repartition checking

The repartitioning does not have to be processed every time the object deforms since most objects used in computer graphics do not deform noticeably. However, when there is great deformation, some area might become too large so that only one particle is not enough to detect the entire area. When the deformed area is larger than an acceptable area (A^*), repartitioning process is required. The surface of the objects should be repartitioned when the deformed area is larger than the acceptable area (A^*).

The area of each region can be computed by estimating an approximated rectangular area. First, the distance along the vertices from the particles to the edge around the area is measured in four directions, $+X$, $-X$, $+Y$ and $-Y$. A rectangular area A can be calculated as shown in Fig.9. This approximated area is then compared to the acceptable area (A^*) which, in this algorithm, is set to the largest approximated rectangular area among of all initial areas. If the

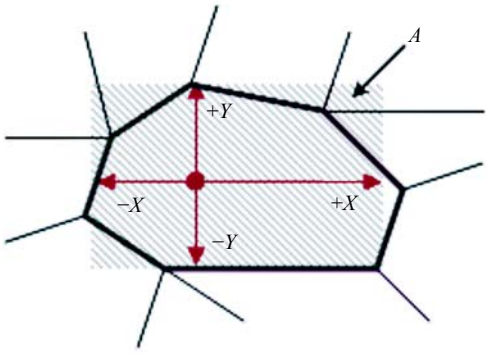

Fig.9 Four directions used to find size of areas

computed area is larger than the acceptable value, the repartitioning for the entire object is then processed.

ALGORITHM ANALYSIS

The complexity of the proposed algorithm corresponds to the computational time of 3 processes which are the surface partitioning process, the collision detection process and the repartition checking process.

The number of particles needed for each object can be estimated by checking every vertex for critical points, so that the computation takes time $O(M)$ where M is number of vertices. The surface partitioning algorithm for each object has complexity of $O(kMN)$ where k, M, N represent the number of iterations, number of vertices and number of particles, respectively.

However, the partitioning process does not have to be computed every time an object deforms as it is in the preprocessing procedure. Therefore, the complexity of the proposed algorithm can be estimated from the other two processes.

The complexity of the algorithm for detecting a collision is $O(N^2)$ while the complexity of the algorithm for repartition checking is $O(N)$ where N is the number of particles.

RESULTS

The proposed algorithm has been simulated on an Intel Pentium IV 2.8 GHz processor. We tested our algorithm on different polygonal models. Three frames from our implementation using two polygonal models of teapot defined by 5784 polygons and sphere defined by 1260 polygons are shown in Fig.10. Experiments showed that this algorithm can detect collision with accuracy of more than 90% using short computational time which can show the average result by 29 fps. From the results, it can be concluded that the presented algorithm can accurately detect collisions in real-time.

CONCLUSION

This paper presents an adaptive particle-based collision detection algorithm which can improve the particle dispersion of the original algorithm by adding the surface partitioning process. LBG quantization theory is applied in order to efficiently partition the surface area. The proposed algorithm provides a particle separately to each corresponding area in order to achieve equal and fair dispersion. Our algorithm can detect also the collision of several deformable objects at the same time while the previous particle-based method (Senin *et al*., 2003) might fail since the given particles are not sufficient and the particle dispersion is not appropriate. After the deformation of objects, the acceptance area is monitored so that the proper number of particles can be altered by completing the repartitioning process.

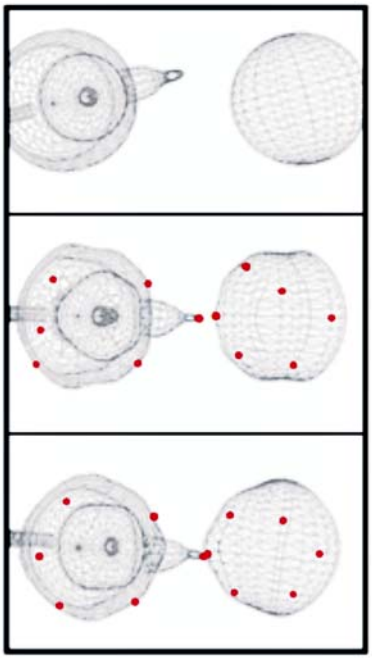

Fig.10 Collision detection between teapot and sphere

References

Abut, H., Gray, R.M., Rebolledo, G., 1992. Vector Quantization of Speech and Speech-like Waveforms. IEEE Transactions on Acoustics Speech and Signal Processing, ASSP-30, p.423-435.

Bridson, R., Marino, S., Fedxew, R., 2003. Simulation of Clothing with Folds and Wrinkles. Proceedings of ACM/Eurographics Symposium on Computer Animation, p.28-36.

Conway, J.H., Slone, N.J.A., 1993a. Bounds on Kissing Numbers. *In*: Sphere Packings, Lattices, and Groups, 2nd Ed., Springer-Verlag, New York.

Conway, J.H., Slone, N.J.A., 1993b. The Kissing Number Problem. *In*: Sphere Packings, Lattices, and Groups, 2nd Ed., Springer-Verlag, New York.

Cohen, J.D., Lin, M.C., Manocha, D., Ponamgi, M., 1995. I-COLLIDE: An Interactive and Exaxt Collision Detection System for Large-scale Environments. Proceedings of the Symposium on Interactive 3D Graphics, p.189-196.

Gersho, A., Gray, R.M., 1992. Vector Quantization and Signal Compression. Kluwer International Series in Engineering and Computer Science, 159. Kluwer Academic Publishers.

Gottschalk, S., Lin, M.C., Manocha, D., 1996. OOB Tree: A Hierachical Structure for Rapid Interference Detection. ACM Computer Graphics (Proc. SIGGRAPH'96), p.171-180.

He, T.S., 1999. Fast Collision Detection Using QuOSPO Trees. Proceedings of the Symposium on Interactive 3D Graphics, p.55-62.

Held, M., Klosowski, J.T., Mitchell, J.S.B., 1995. Evaluation of Collision Detection Methods for Virtual Reality Fly-troughs. Proceedings Seventh Canadian Conference on Computational Geormetry, p.205-210.

Hubbard, P., 1996. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, **15**(3):179-210. [doi:10.1145/231731.231732]

Klosowski, J.T., Held, M., Mitchell, J.S., Sowrizal, H., Zikan, K., 1998. Efficient collision detection using bounding volume hierarchies K-DOPs. IEEE Transactions on Visualization and Computer Graphics, p.21-36.

Krishnan, S., Pattekar, A., Lin, M., Manocha, D., 1998. Spherical Shell: A Higher Order Bounding Volume for Fast Proximity Queries. Proceedings of WAFR'98, p.287-296.

Linde, Y., Buzo, A., Gray, R.M., 1980. An algorithm for vector quantizer design. *IEEE Transaction on Communications*, **28**(1):84-95. [doi:10.1109/TCOM.1980.1094577]

Palmer, I., Grimsdale, R., 1995. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, **14**(2):105-116. [doi:10.1111/1467-8659.1420105]

Parent, R., 2001. Computer Animation: Algorithms and Techniques.

Quinlan, S., 1994. Efficient Distance Computation between Non-convex Objects. Proceedings of IEEE International Conference on Robotics and Automation, p.3324-3329.

Raghupathi, L., Cantin, V., Faure, F., Cani, M.P., 2003. Vision, Modeling and Visualization (VMV) Real-time Simulation of Self Collisions for Virtual Intestinal Surgery. Surg. Sim. & Soft Tis. Model, p.15-26.

Senin, M., Kojekine, N., Savchenko, V., Hagiwara, I., 2003. Particle-based Collision Detection. EUROGRAHICS.

Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.P., Faure, F., Magnenat-Thaimann, N., Strasser, W., Volino, P., 2004. Collision Detection for Deformable Objects. EUROGRAPHICS.

Zachmann, G., 1998. Rapid Collision Detection by Dynamically Aligned DOP-trees. Proceedings of IEEE Virtual Reallity Annual International Symposium, p.90-97.

Welcome visiting our journal website: ***http://www.zju.edu.cn/jzus***
Welcome contributions & subscription from all over the world
The editor would welcome your view or comments on any item in the journal, or related matters
Please write to: Helen Zhang, Managing Editor of *JZUS*
E-mail: **jzus@zju.edu.cn** Tel/Fax: 86-571-87952276/87952331