Collision Detection: Applications and Techniques

Norhaida Mohd Suaib
Dept. of Computer Graphics & Multimedia,
FSKSM, UTM Skudai.
haida@utm.my

Abdullah Bade
Dept. of Computer Graphics & Multimedia,
FSKSM, UTM Skudai.
abade@utm.my

Siti Mariyam Hj. Shamsuddin
Dept. of Computer Graphics & Multimedia,
FSKSM, UTM Skudai.
mariyam@utm.my

ABSTRACT
Collision Detection or interference detection has been studied extensively in robotics, computer graphics, physically-based modeling, molecular modeling, visualization and virtual environment. Predominantly, the input to a collision detection algorithm comprises large and complex geometric object either as static or dynamic object within the environment. The term collision detection refers to a task of determining whether two or more objects intersect. In order to represent the geometric interaction in a dynamical system with the utmost efficiency or to simulate realistic physical behaviour of entities in the virtual environment, it is vital to perform the collision detection event with fast and minimal computation, yet with high precision. In this paper we will present an overview of various collision detection applications and also describe several methods or techniques that had been used by various authors.

Categories and Subject Descriptors
I.3.7 [Three-Dimensional Graphics and Realism] – Animation, Color, shading, shadowing, and texture, Hidden line/surface removal, Raytracing, Virtual reality, Visible line/surface algorithms.

General Terms
Algorithms, Performance, Theory.

Keywords
Collision detection, bounding volumes, virtual reality, computer graphics.

1. INTRODUCTION
Collision detection (CD) has gained much attention of research since more than two decades ago. In the early days, most of the researches originate from the area of robotics and computational geometry. These later developed into a need in a variety of fields such as computer graphics, physically-based modeling, molecular modeling and virtual environment [1-8]. Nowadays, any interactive applications depend on real-time high-speed and efficient collision detection method.

The main objective of collision detection is to generate or produce a report on any intersection that exists between two or more objects [4, 5, 9]. The actual problem and the degree of accuracy of collision detection algorithm usually depend on the type of application that incorporates it. However, most of the proposed algorithms and techniques cannot be considered as universal or optimal solution to this problem. This poses an interesting challenge to researchers in this field - to develop a very efficient and robust algorithm. This is important in order to achieve real-time collision detection in virtual environment, irrespective of the number of objects present in the setting [5].

This paper first discusses the applications of CD in different fields (section 2) to highlight the importance of CD algorithms. This is followed by standard approach taken to perform CD in section 3. Section 4 outlines the different CD algorithms as discussed in the literature and section 5 concludes the discussion.

2. APPLICATION OF COLLISION DETECTION
The field of collision detection receives more attention and has been used in many applications. For instance, collision detection is becoming a significant component in Virtual Reality (VR) applications. This is due to the fact that without collision detection, high degree of realism for haptic sensation in the VR application is almost impossible to achive. In the field of robotics, notably in the vehicle industry using robot manipulators, the aim is to produce an efficient path planning procedure with the intention of avoiding any potential collision from occurring. It should be noted that in most motion planning, about 90% of the time is spent in computing the distance between the robot manipulators and the obstacles around it [10]. The complexity behind the path-planner theory, execution speed, environment domain and the non-generality of path-planning programs, pose great challenge in producing an optimal collision avoidance solutions for robotic applications [11].

In spite of using general collision detection approach in robotics, collision avoidance techniques are also useful in steering away the robot arm from colliding with surrounding obstacles. OxSim [11] is considered to be one of the earliest path planning program that is based upon the separation of geometry and planning. The prototype for collision detection and path planning schemes involves the use of large and complex geometry.

Conference’04, Month 1–2, 2004, City, State, Country.

Another area where collision detection plays an important role is in the field of computer animation. One of the most crucial considerations in achieving realistic objects movement is the ability to control their interactions or their contacts between objects. Animating a single rigid body is a straightforward exercise but as the objects become increasingly complex their interactions become more difficult to handle. Cyrus-Beck Clipping algorithm [12] is considered as one of the earliest example of collision detection algorithm, intended for flexible surface models (using triangle-triangle intersection) and convex polyhedra.

Computer vision is another exciting research field that poses many elegant and challenging problems, especially for real-time collision detection. Direct applications of the researches can be found in robotics, HCI (Human Computer Interaction), computer graphics and in search engines for images. Basically, the computer vision program uses the extracted information from the collision detection subprogram to prevent the appropriate action to be taken in order to prevent collision with the surrounding obstacles. The extracted information is usually the scene which is captured from a camera or similar device [13].

Virtual reality is one of the fastest growing fields of computer technology where collision detection is heavily put to use. Virtual reality applications become more appealing particularly in the area of medicine, architecture, engineering, science, education and entertainment. In the virtual environment, most application are in the virtual prototyping mode. Objects created in the artificial world are in the form of geometric models [7]. The problems related to the geometric computation involved in designing and developing the applications pose some challenge. This is especially true in a real-time simulation of realistic objects within a massive environment. There are basically three important characteristics which must be considered whilst developing VR applications for large environment. Firstly there is the fact that all object models consists of complex polygons. Secondly, the created objects do not have any topological information (maybe cracks in object models and the presence of non-manifold geometry), and lastly, the objects motion cannot be predicted or anticipated beforehand. One of the examples is cutting and tearing of soft tissues in medical simulator [14]. Nevertheless, in virtual reality applications human participation typically involves the scene modification aspect or initiating the movement of objects within the environment [7].

In order to preserve the high degree of realism in virtual reality applications, real-time interactive intersection procedures or collision detection must be applied continuously. This is obviously important as real-time interactive intersection and detection produces realistic object motions in the synthetic world. At this stage, there are several approaches to collision detection which have been implemented in virtual reality application, such as ray shooting using meshes (triangulation) of low stabbing and bounding volume hierarchies. Ray shooting uses meshes (triangulation) to perform the intersection tests by way of tracking the motion of flying objects within the tetrahedral mesh. In the bounding volume hierarchies approach, the collision test is performed using different types of bounding objects (sphere, AABB and OBB) to achieve the outcome.

3. STANDARD APPROACH

Collision detection is a computationally intensive process. Therefore, it is crucial to minimize the collision test. In general, collision detection algorithms follow two standard phases in order to filter the pairs that do not possibly collide in tandem to overcome the three main issues inherent in virtual environment application, namely real-time response, efficient computation and high-speed execution [4, 5]. These two phases are known as the broad-phase level and narrow-phase level [6, 11, 15, 16]. These two-phases can also be combined to form a hybrid-collision detection algorithm [16], where an approximation to all possible interference in the environment is done in the first stage, followed by a subsequent more accurate calculation test at a later stage.

3.1 Broad-phase Level

In the broad-phase level, the algorithm customarily identifies all pairs of objects which can possibly collide, and irrelevant pairs will be ignored. Generally, the broad phase collision detection method is very conservative and most of the calculations are not very complex.

One of the distinctive behaviors of the broad phase level is the filtering aspect. This aspect made the broad-phase stage works independently of other stages in determining collision between objects [17]. Generally the type of algorithms performed in broad phase level are bounding volumes, spatial subdivision and time-coherence [5, 15]. Bounding volume algorithms encompasses techniques such as, Axis-aligned bounding box (AABB), bounding sphere, oriented bounding box (OBB) and discrete orientation polytopes (k-DOPs) (Figure 1) [1]. Spatial subdivision uses techniques such as octree and BSP-Tree to determine the likelyhood of a collision occurring, whilst the time-coherence algorithm uses techniques such as four-dimensional space-time bounding [5, 15].

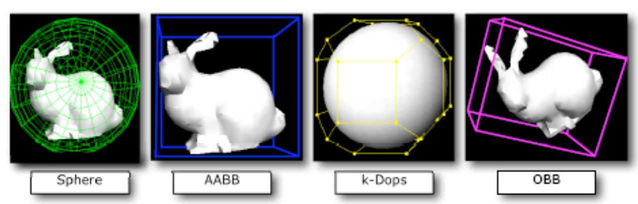

Figure 1. Different bounding volume types [1].

3.2 Narrow-phase Level

The narrow-phase level algorithm performs a relatively more accurate and exact collision detection method in pruning the identified object pairs during broad phase level. Results varies; objects that do not collide will be discarded while those that engaged in collision will produce information required to determine time of impact (TOI), appropriate collision response as well as contact determination [6]. A collection of algorithms will be discussed in section 4.

3.3 Discrete versus Continuous Collision Detection

Based on how CD algorithm deals with the movement of objects, they can either fall into a discrete CD or continuous CD [18]. Earlier CD research mostly focused on discrete CD, where collisions among objects are detected at discrete time steps. In some cases, CD algorithm failed to detect collision that happens between different intervals.

In order to overcome this problem, researchers resort to continuous CD (CCD) that takes continuous motion into account. Therefore, it reports first time of contact (TOC) and object penetration can be avoided. However, there is a drawback to CCD compared to its counterpart; speed – CCD is much slower compared to discrete CD.

4. COLLISION DETECTION TECHNIQUES

There are a number of ways to group these CD techniques. In this paper, we will briefly review some of popular techniques used for broad phase and narrow phase according to the classifications used by [6].

4.1 Broad Phase

Broad phase CD algorithm can be broken down into three categories; all-pair test, sweep and prune and hierarchical hash table. In all-pair test (also known as the 'exhaustive test'), each BVs that made up an objects will be tested against other BVs (of other objects). Sweep and prune test basically use BV projection onto the axes – any pair of colliding objects will show overlapped region in all major axes. Hierarchical hash table basically divides the whole scene into grids; objects that share the same region indicate collision.

4.2 Narrow Phase

There are five categories as described by in narrow phase CD as illustrated in figure 2.

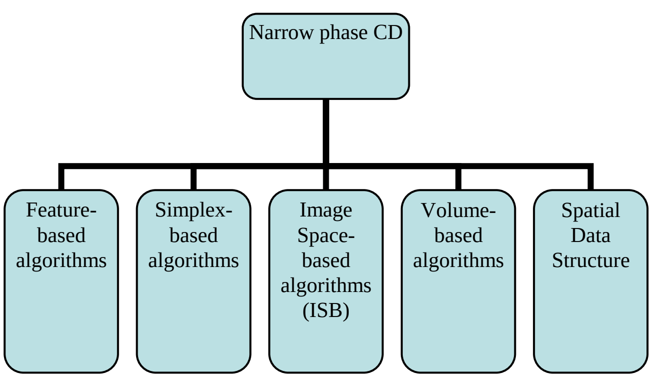

Figure 2. Five categories of narrow phase CD algorithms.

Feature-based CD algorithms are applied directly on objects' geometry. One of the popular algorithm in this category is the Lin-Canny Closest Features Algorithm [4]. Other algorithms are Voronoi-Clip (V-Clip), polygonal intersection and SWIFT.

The Gilbert, Johnson and Keerthy (GJK) algorithm is the ancestor to the simplex-based algorithms [19]. Using the Minkowski difference of the two sets of vertices of the objects, it determines whether or not these objects are in collision [20].

Image Space-Based algorithms (ISB) basically manipulate image-space occlusion queries to detect collision. Due to their nature, they are suitable to be implemented on the GPU as well as the CPU. Although GPU implementation usually yield faster result, two of the drawbacks are identified as they are not as fast as hierarchical approaches and error might occure during rasterization since image space is discrete. Two popular examples of ISB implementation are CINDER[21] and CULLIDE[22].

Volume-based algorithms generally share the similar design of ISB in terms of comparing 'depth' of the objects in the object space and GPU implementable. However, they use different computation the 'depth'.

Bounding volume hierarchies can further be broken down to spatial sub-division and bounding volume hierarchies (BVH). As the name implies, spatial sub-division (also known as spatial decomposition) divides object space while BVH divides the object. Spatial decomposition techniques are represented by algorithms such as Octrees, k-d trees, BSP-trees, Brep-indices, tetrahedral meshes and grids. On the other hand, some of the example of BVH algorithms are AABB, OBB, BOXTREE and Backface-culling [1, 4].

5. CONCLUSION

Collision detection poses an important role in fields such as robotics, computer graphics, physical based modeling, molecular modeling and virtual environment. This requirement is due to the fact that all of them need collision detection in order to perform precise operations and to preserve the physical behaviour of objects interaction in the real world. Although research in CD has been going on for more than 20 years, it still offers many research opportunities. This is manifasted in new algorithms and refinements to the existing methods and techniques being introduced in this field.

6. ACKNOWLEDGMENTS

We would like to express our gratitude to MOSTI (IRPA 74019 & 74224 and eScience 79237 grants), without which our research path may be winding. Our thanks also go to the people directly and indirectly involved in producing this paper.

7. REFERENCES

1. Bade, A., et al., *Oriented convex polyhedra for collision detection in 3D computer animation*, in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. 2006, ACM: Kuala Lumpur, Malaysia.

2. Suaib, N.M., et al. *Collision Detection Strategy for Interactive Virtual Environment*. in *The IASTED International Conference on Applied Simulation and Modelling 2003 (ASM 2003)* 2003. Marbella, Spain: ACTA Press.
3. Sunil, H., et al., *Collision detection and proximity queries*, in *ACM SIGGRAPH 2004 Course Notes*. 2004, ACM: Los Angeles, CA.
4. Klosowski, J.T., *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environment*. 1998, State University of New York at Stony Brook.
5. He, T., *Fast collision detection using QuOSPO trees*, in *In Proc. of ACM Symp. on Interactive 3D Graphics*. 1999.
6. Kockara, S., et al. *Collision detection: A survey*. in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. 2007.
7. Ming Chieh, L., *Efficient Collision Detection for Animation and Robotics*. 1993, University of California, Berkeley. p. 159.
8. Bade, A., et al. *An Overview of Collision Detection in Virtual Environment*. in *The 18Th International Conference on CAD/CAM, Robotics and Factories of The Future (CARS&FOF2002)*. 2002. Porto, Portugal.
9. Gottschalk, S., M.C. Lin, and D. Manocha, *OBBTree: A hierarchical structure for rapid interference detection*, in *Computer Graphics, 30 (Annual Conference Series*. 1996.
10. Tsai-Yen, L. and C. Jin-Shin, *Incremental 3D collision detection with hierarchical data structures*, in *Proceedings of the ACM symposium on Virtual reality software and technology*. 1998, ACM: Taipei, Taiwan.
11. Cameron, S. *Motion planning and collision avoidance with complex geometry*. in *Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE*. 1998.
12. Matthew, M. and W. Jane, *Collision detection and response for computer animationr3.* SIGGRAPH Comput. Graph., 1988. **22**(4): p. 289-298.
13. Pamela, R., et al., *Plan-N-Scan: A Robotic System for Collision-Free Autonomous Exploration and Workspace Mapping.* J. Intell. Robotics Syst., 1999. **24**(3): p. 207-234.
14. Klein, J.B., Dirk; Friman, Ola; Hadwiger, Markus; Preim, Bernhard; Ritter, Felix; Vilanova, Anna; Zachman, Gabriel;, *Advanced Algorithms in Medical Computer Graphics*, in *Eurographics 2008*. 2008, Eurographics: Crete, Greece.
15. Hubbard, P.M., *Collision detection for interactive graphics applications.* Visualization and Computer Graphics, IEEE Transactions on, 1995. **1**(3): p. 218-230.
16. Watt, A., *3D Computer Graphics*. 3rd. ed. 2000: Addison-Wesley.
17. Mirtich, B., *Efficient algorithms for two-phase collision detection*. 1997, Mitsubishi Electrical Research Laboratory.
18. Zhang, X., M. Lee, and Y.J. Kim, *Interactive continuous collision detection for non-convex polyhedra.* The Visual Computer, 2006. **22**(9): p. 749-760.
19. van den Bergen, G., *Collision Detection in Interactive 3D Environment*. 2004, San Francisco: Morgan Kaufmann Publishers Inc. 300.
20. Ericson, C., *Real-time collision detection*. Morgan Kaufmann series in interactive 3D technology. 2005, Amsterdam ; Boston: Elsevier. xxxviii, 593 p.
21. Knott, D. and D. Pai. *CInDeR: Collision and Interference Detection in Real-time using Graphics Hardware*. in *Graphics Interface '03*. 2003.
22. Govindaraju, N.K., M.C. Lin, and D. Manocha, *Fast and reliable collision culling using graphics hardware.* Visualization and Computer Graphics, IEEE Transactions on, 2006. **12**(2): p. 143-154.