Two Different Views On Collision Detection

Torsten Ullrich, Christoph Fünfzig, and Dieter W. Fellner

***Abstract*— The realistic movement of geometry governed by physics is necessary for many applications. Changes of movements occur at events, where geometries collide. For real-time applications the collision detection must be as fast as possible.**

While conceptually simple, the collision detection excels in its intrinsic complexity. Naive approaches to determine all collisions of n objects require a runtime of $O(n^2)$, which will rapidly be too much in practice.

Innumerable algorithms with reduced runtime complexity have been developed in the last decades. They can be classified according to different schemes regarding e.g. field of application or solution strategy. This article differentiates by the algorithms' background: computational geometry or signal processing, and presents a representative of each domain. The main characteristics of both algorithms are simplicity and efficiency.

***Index Terms*— collision detection, computational geometry, signal processing, distance fields, axis-aligned bounding boxes, wavelets**

I. Introduction

COLLISION detection is an algorithmic problem all areas of computer science related to the simulation of physical objects in motion have to deal with. The field of application ranges from robotics and computer-aided manufacturing to computer games and virtual reality.

Collision detection algorithms can be classified in various ways: according to the geometric object model used, to the characterization of the solving strategies or to theoretical concerns such as worst-case complexity. This article differentiates by the algorithms' background. Many collision detection algorithms have been developed in recent years, but all of them have a common origin in either computational geometry or signal processing.

The most important difference between computational geometry and signal processing is the discrete nature of computational geometry. Although all problems that are solved on digital computers must be formulated in a discrete form, some applications deal with discrete approximations to continuous phenomena.

In image processing, for example, filtering operators are written in a continuous setting and need to be discretized in order to find numerical solutions, whereas in geographic information systems (GIS) road networks have per se a discretized structure.

II. Collision Detection

Within physical simulations collision detection answers the question whether two or more given objects collide. To reduce the quadratic complexity of testing all objects with each other, proximity queries and space partitioning are used.

This broad phase extracts only potential collision pairs. The following narrow phase performs a precise collision detection for each potential collision pair with the objects' model representation.

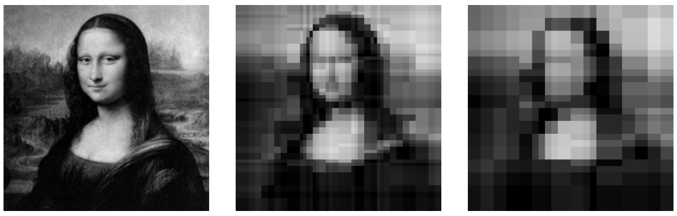
Fig. 1. Wavelet theory is an very effective tool in image processing based on multiresolution analysis. A wavelet compression algorithm represents an image as a set of real coefficients. Most of the wavelet coefficients of a typical image are nearly zero, so that the image is well-approximated with a small number of non-zero wavelet coefficients omitting the almost zero coefficients. The algorithm achieves a good compression ratio, while maintaining the image's quality.

In this article we present two algorithms for a precise collision detection between two potentially colliding objects. The first one uses axis-aligned bounding boxes (AABB) and is a typical representative of a computational geometry algorithm, whereas the second one uses spherical distance fields originating in image processing. Both approaches have to address the challenges of collision detection algorithms:

- **Just in Time**: Collision detection is a very time consuming task that can easily consume the majority of an application's total run time. A slow algorithm is a burden to every virtual reality system, whereas a rapid one may produce amazing experiences.
- **No resources**: Related to the time critical aspects of collision detection is the algorithmic time-space tradeoff that can be perceived perspicuously in large virtual environments: Complex scenes require a lot of memory themselves which forbids to use extensive, additional data structures.
- **All inclusive**: All collision detection algorithms use object representations that allow an early exclusion of far-flung objects. Bounding objects of simple shape are applied. These simple objects substitute the original objects in different resolutions. All simplified objects have in common that they enclose the original object (or parts of it) in order to not miss a potential collision.

The two algorithms are introduced in the following sections emphasizing their strengths and weaknesses.

III. Axis-Aligned Bounding Boxes

Many collision detection algorithms are based on bounding volumes. If the bounding volume is a box, whose axes are aligned with the axes of the coordinate system, it is called an axis-aligned bounding box (AABB). Bounding volume

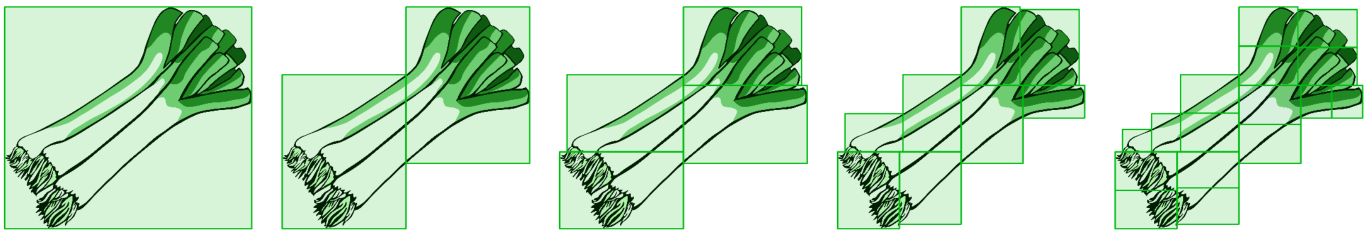

Fig. 2. During the preprocessing steps for the axis-aligned bounding box data structures a model is enclosed by its bounding box (outer left). Afterwards the bounding box is split at its longest edge into two bounding boxes that are refitted to the contained model parts. This subdivision process continues until the desired granularity is reached (outer right).

hierarchies based on AABB rank among the simplest collision detection algorithms. If for each object and object part the coordinate system is chosen newly based on the object's points, the resulting box is called an oriented bounding box (OBB).

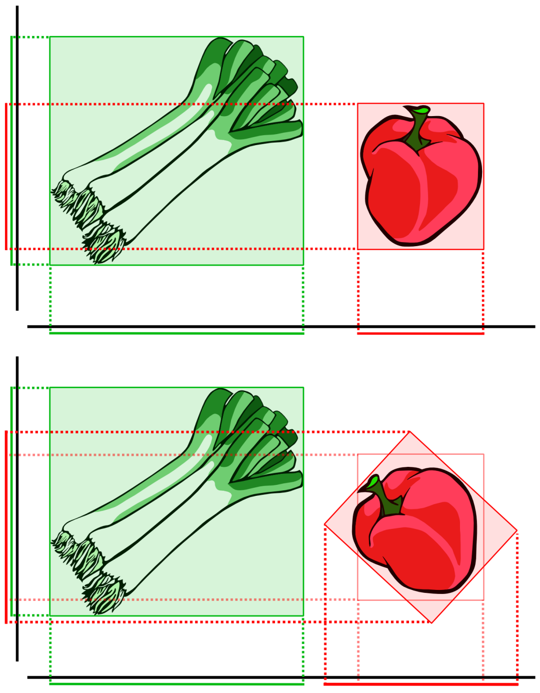

Fig. 3. The intersection test of two axis-aligned bounding boxes can be performed using a few interval comparisons. If and only if all coordinate intervals of both bounding boxes overlap, the bounding boxes intersect. In case of rotations, one's AABB has to be realigned to the other's coordinate axes.

A. Preprocessing

The preprocessing step determines a model's bounding box aligned to the coordinate system. In many commonly accepted model representations (polygonal descriptions, Bézier surfaces, etc.) this is an easy minimum/maximum search over all (control) vertices.

Afterwards, the bounding box is split at its longest edge into two bounding boxes that are refitted to the model. This subdivision process is called recursively by all boxes down to the desired granularity. This process is illustrated in Figure 2. The bounding boxes are stored in a tree structure (AABB tree).

B. Intersection Test

A single bounding box is typically stored using two points. The intersection test uses these two points in order to determine the boxes' projection onto the coordinate axes. The enclosing intervals on the coordinate axes can be read off easily.

Two bounding boxes overlap, if and only if all coordinate intervals of both bounding boxes intersect. A two-dimensional configuration is shown in Figure 3.

The algorithm uses the presented data structure to exclude irrelevant candidates as fast as possible. A collision check of two models starts by checking the root bounding boxes. If these boxes do not overlap, a collision is not possible. Otherwise, descending the AABB tree both bounding volumes are refined and checked again. The last refinement level has to work on the model's primitive representation.

C. Pros & Cons

The AABB based collision detection has many advantages.

- The algorithm is easy to understand and to implement. A simple AABB collision detection consists normally of a few lines of code.
- The implementation of this algorithm is numerically stable. Even the usage of floating-point values is extremely stable in combination with a bounding box offset within the magnitude of floating-point precision.

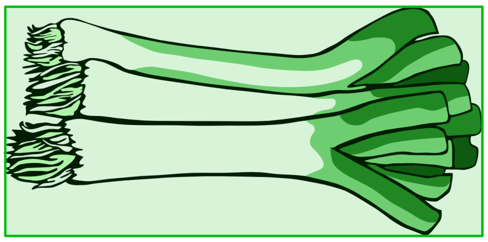

Fig. 4. Having a fixed orientation of a bounding volume (e.g. axis aligned) an optimal enclosure of an arbitrary object is not possible in general. This Figure shows the optimal object orientation in contrast to the orientation illustrated in Figure 2.

- The algorithm has no restrictions concerning the underlying model representation (polygonal models, free-form surfaces, B-Rep models, etc.).

Regrettably, the algorithm also has some inherent disadvantages.

- The alignment of all bounding boxes according to the coordinate system simplifies the intersection test, but it has suboptimal tightness in the general case. There can be a big difference between the optimal bounding box and the axis-aligned bounding box of an object.
- Another problem is the inability to reuse the AABB data structure during object rotations. If an object is rotated along an arbitrary axis, its bounding boxes have to be realigned (using the object's bounding box) as illustrated in Figure 3 or even recalculated (from the object's points).

IV. Wavelets/Distance fields

The second algorithm presented in this article has its origins in signal processing. It uses spherical distance fields to speed up collision checks.

A. Preprocessing

During the preprocessing step the algorithm takes an initial model and determines a model center. According to this center point the model is transformed into spherical coordinates and sampled into several intervals. For each sample the maximum radius is stored. Figure 5 illustrates this process. The 2D case is shown for clarity.

The basic concept used in the preprocessing step works similarly to a wavelet transform. It starts with the high resolution object and derives objects of lower resolution by storing the differences in detail coefficients, in order to be able to reverse this operation.

An efficient implementation of this process may store all coefficients within some fixed-sized arrays.

For correctness a low resolution model must bound all models of higher resolution. Therefore, the Haar basis functions, which are used among others in image processing, e.g. in Figure 1, and which describe an averaging and differencing process, are unsuitable. For our purpose, maximizing and minimizing functions have to be taken as presented in Figure 5. Therefore, each object will be transformed into an inscribed circle and a circumference in lowest resolution.

The Expected Running Time of Collision Detection with AABB Trees

In search of an algorithm suitable for a particular problem, its running time plays an important role – especially the expected running time of an algorithm is the measure of its quality. The worst case scenario plays a minor role.

In order to answer a collision query of two objects the AABB-based collision detection algorithm traverses both bounding volume hierarchies. The required time to answer the query consists of

- initial setup costs,
- the costs of a single bounding volume/bounding volume overlap test multiplied by the expected number of overlap tests, and
- the costs of a single primitive/primitive intersection test which also has to be multiplied by the expected number of intersection tests.

In an asymptotic analysis the number of overlap tests N defines the running time. While it is obvious that $N = n^2$ in worst case, this number is linear or even logarithmic for most practical cases (n denotes the number of bounding volumes each of the two query objects owns).

The conditional probabilities that a pair of bounding volumes overlap can be estimated using geometric reasoning. R. Weller, J. Klein, and G. Zachmann presented this reasoning in "A Model for the Expected Running Time of Collision Detection using AABB Trees" on the Eurographics Symposium on Virtual Environments 2006.

Assuming that the two root bounding volumes overlap the expected number of overlap tests only depends on scaling factors that relate the size of a bounding volume to the size of its children.

For the sake of clarity and simplicity these scaling factors α_x, α_y, and α_z along each axis shall be constant throughout the hierarchies of the query objects. Then the expected number of overlap tests can be estimated by

$$N(n) \leq \sum_{i=1}^{\lg n} 4^i \cdot \alpha_x^i \cdot \alpha_y^i \cdot \alpha_z^i \in O\left(n^{lg(4\alpha_x\alpha_y\alpha_z)}\right).$$

Plugging this estimation into a cost function $T(n)$ yields an overall time estimate that is illustrated in Table 1. The frequently used implementation, which divides a bounding volume in the middle of its longest edge into two parts (with a small overlap) has a scaling product of $\alpha_x \cdot \alpha_y \cdot \alpha_z \approx 1/2$, i.e. the expected running time is linear to the number of bounding volumes.

$\alpha_x \cdot \alpha_y \cdot \alpha_z$	$T(n)$
$< 1/4$	$O(1)$
$1/4$	$O(\lg n)$
$1/2$	$O(n)$
$3/4$	$O(n^{1.58})$
1	$O(n^2)$

Table 1. Effect of the scaling factor product on the running time of a AABB-based collision query.

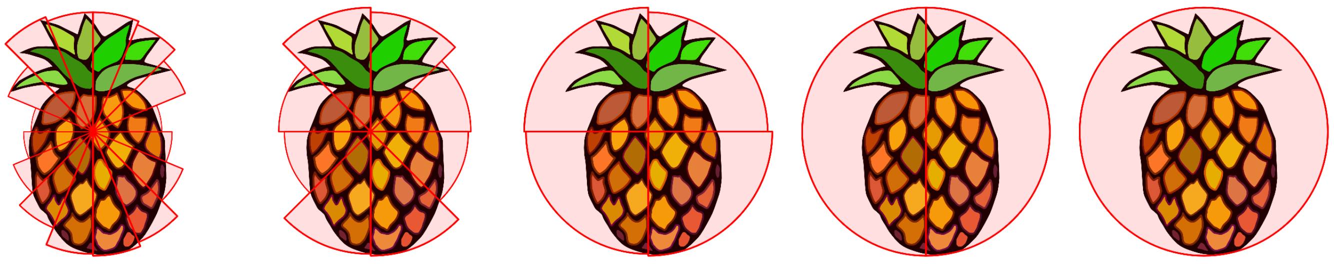

Fig. 5. The distance field based collision detection algorithm uses a spherical sampling of an object (outer left) and applies maximum filters to it. The results of these filters are shown in the figures. At the lowest level a bounding sphere encloses the whole object (outer right).

B. Intersection Test

Having transformed all objects this way, it is possible to perform a fast collision test, which will be demonstrated by a pineapple and a red pepper.

At runtime, the intersection test starts with the model representation at lowest resolution and tests whether they collide or not. If this test is positive, the level of detail will be increased. Thereby it is important for performance that only intersecting sectors are considered further on.

This principle is illustrated in Figure 7, which shows the various stages during a collision test.

As long as there are intersecting sectors of different objects, the algorithm refines the objects. If the algorithm reaches the highest resolution of an object, the collision test is performed on the object primitives.

In 2D the intersection test has to check whether two segments overlap. In 3D two capped cones have to be checked. In contrast to the previous algorithm where two boxes can be checked by comparing a few intervals, the intersection test for two capped cones is non-trivial.

Subject to the orientation of the cone axes, two cases are analyzed. The first case deals with nonparallel axes; the second one with parallel lines. A heuristically chosen (angle) threshold distinguishes between the nonparallel and the parallel case.

In the nonparallel case for both axes the shortest distance and the according perpendicular points P_1, P_2 are determined. The test itself considers cone section. One cone is intersected with a plane containing the first cone's center and the line through P_1 and P_2. This results in a well-known cone section. The first cone is reduced to a line which passes the first cone's center and the point which you get by translating P_1 within the considered plane towards the cone section's focus. The intersection test is then reduced to a 2D intersection test between a cone section and a line.

In the parallel case the algorithm analyzes the projection of one cone onto the other cone's axis and vice versa. For each projection two distance checks of points against their according radii are performed, which results in a total of four checks. The special case of parallel lines is handled separately for optimization, as in this case the whole test can be done by some interval checks, which are much faster.

Together with a quick rejection test (consider bounding cylinders instead of cones) and a quick acceptance test (consider spheres inside the cones) for the nonparallel case the algorithm is reasonably fast.

C. Pros & Cons

The distance field based algorithm has some advantages and disadvantages. The advantages involve among other things

- the algorithms ability to reuse an objects distance field while the object is translated and rotated. The initial preprocessing does not need any updates.
- The algorithm has no restrictions concerning the underlying model representation (polygonal models, free-form surfaces, B-Rep models, etc.).
- The algorithm is easy to understand, although it is not very easy to implement.

The only problem this algorithm is how to cope with non-star-shaped objects. This problem affects the algorithm's speed but not its correctness.

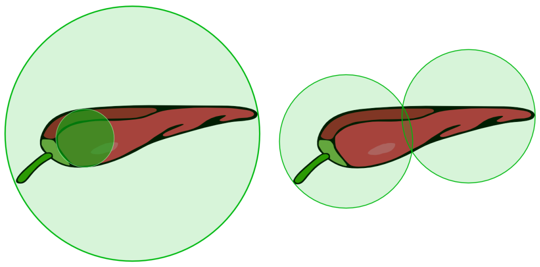

Fig. 6. For aspect ratios much larger (resp. much smaller) than one, the contained volume differs largely in sphere sections from one center point. Using several center points can improve extreme aspect ratio cases.

Concerning usage of a single center point, one has to pay attention to the object's aspect ratio (i.e. the volume ratio of the smallest bounding sphere to the largest contained sphere). For extreme ratios, it is reasonable to switch to a small number of center points instead of a single center point.

V. Summary

This article presented two collision detection algorithms, which are suitable for all model types, e.g., polygon soups, surfaces or volumetric models. They are simple to understand and their memory footprint is linear in the model complexity.

Both approaches are scalable in the information they give in collision determination. If they analyze a possible collision only up to a fixed refinement level, the collision time only depends on the granularity of the bounding volumes, but not on the primitives' count of the model. Due to this fact it

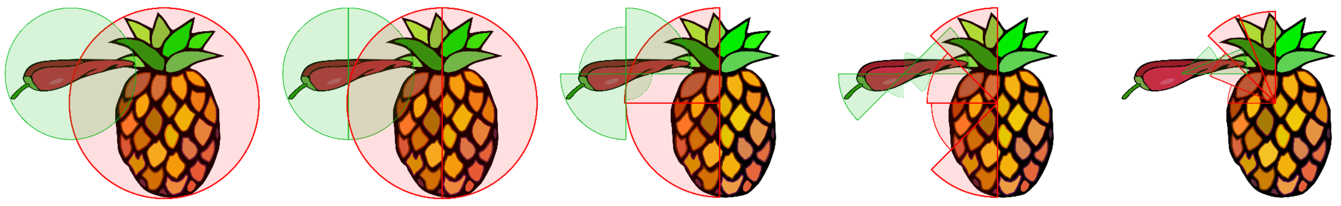

Fig. 7. At runtime the intersection test starts with the bounding volume at lowest resolution (outer left). As long as different bounding volumes intersect, the volumes are refined. Sectors which do not collide with other sectors are not considered further on. At highest resolution (outer right) only model part belonging to colliding sectors have to be checked for collision. Depending on the model representation appropriate algorithms have to be used.

is possible to estimate the time bounds for the collision test tightly.

Although both algorithms have many similarities and attributes in common, they have different origins: The AABB-based algorithm uses discrete structures such as partition trees to achive the necessary refinement, whereas the distance field-based algorithm uses a refinement strategy known from signal processing. It reads the model details out of an array as they are necessary. It therefore deals with discrete approximations to continuous phenomena.

VI. Further Readings

Collision detection based on AABB is a well-known approach. It is a standard algorithm described in introductory courses on computer graphics. A good overview is given, e.g., in the book "Real-Time Rendering" by Tomas Akenine-Möller and Eric Haines (A.K. Peters Limited) and further details can be found in the book "Realtime Collision Detection" by Christer Ericson (Series in Interactive 3D Technology, Morgan Kaufmann Publishers).

The second algorithm has been developed by C. Fünfzig, T. Ullrich and D. W. Fellner. It is described in the article "Hierarchical Spherical Distance Fields for Collision Detection" published in IEEE Computer Graphics & Applications (January/February 2006).