* Introduction Introduce yourself and the title of your presentation
* 10 pages or more

− Overview − Give an idea, but not too detailed

− Motivation − Illustrate the principle and / or applications

− Explain the goal of your presentation − The audience should be eager to listen your presentation

− Main part

− Should consist of distinguished sections

− Separate different sections of the presentation explicitly

− Each section should be introduced and summarized

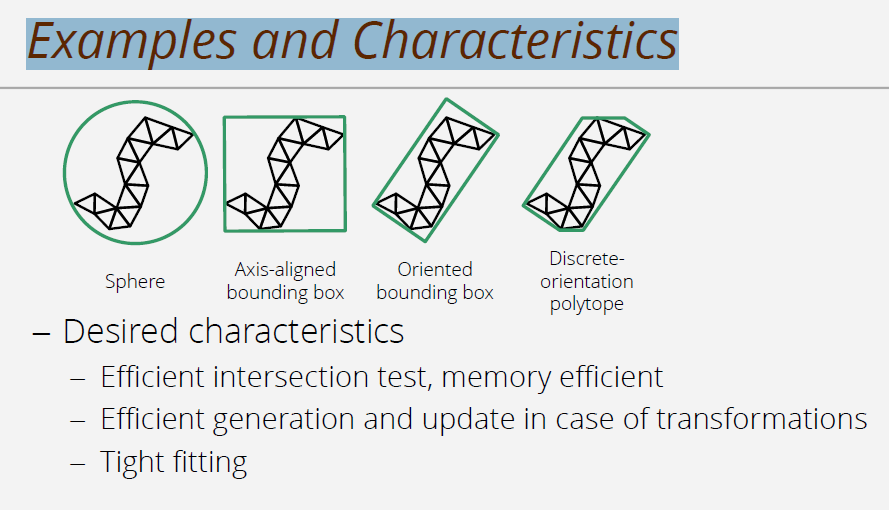
− Summary − Tell the audience what you have told them − Ask for questions

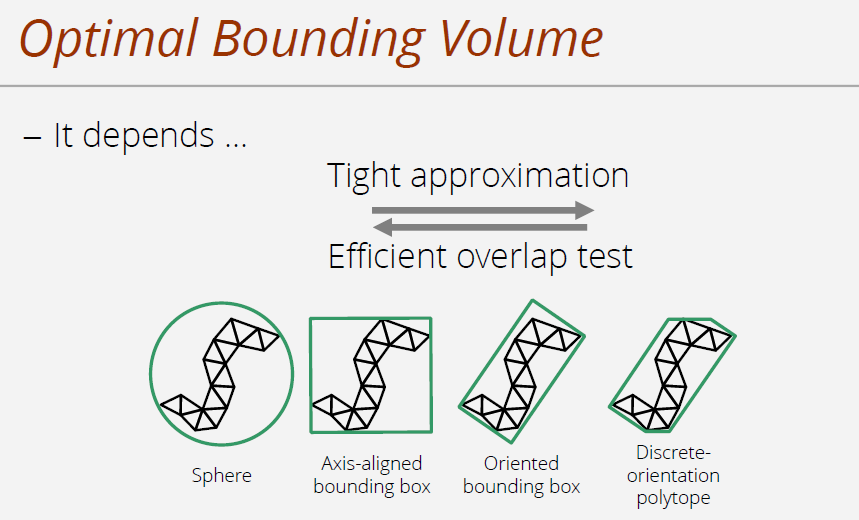
**− Data Structures**

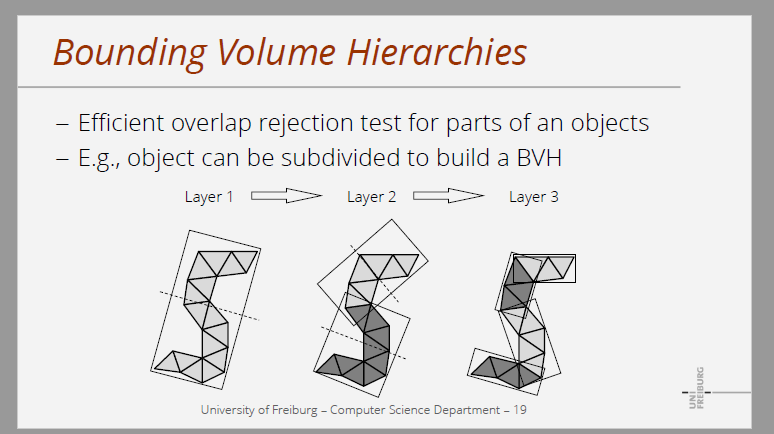
1. **INTRODUCTION** The first 1 or 2 slides should introduce your subject to the audience. Very briefly (you only have about 20-25 minutes total) give a concise background. Explicitly state the question addressed in the paper. Start with the “big picture” and then immediately drive to how your study fits in the big picture (one or two sentences.) One key difference of the talk versus the paper is that you should state your major conclusion(s) up front. That is, in a few sentences, tell the audience where you will lead them in this presentation. (e.g. “Although previous studies have found that inferotemporal neuronal receptive fields are very large, in this talk I will show that, under certain conditions, IT receptive fields are remarkably small.”)
2. **METHODS** There should be 1 or 2 methods slides that allow the audience to understand how the experiment was conducted. You might include a flow chart describing the “recipe” of the experiment. Do not put in details that might be appropriate in a paper (people can ask about them at the end if they are interested). For example, “ extracellular, single-unit recording was used to measure …” NOT: “The electrode impedance was 1 megaohm when measured at 1 kilohertz with a xx instrument…”
3. **RESULTS** The next slides should show the major results. If appropriate, it is nice to start with a slide showing the basic phenomenon (in neurophysiology, this might be a “a single cell” example). It reminds your audience of the variables that were manipulated and introduces your audience to the basic unit of measure. Next, show figures that clearly illustrate the main results. Do not show charts of raw data. All figures should be clearly labeled. When showing figures, be sure to explain the figure axes before you talk about the data (e.g., “the X axis shows time. The Yaxis shows level of neural activity”).
4. **DISCUSSION** (Conclusions) List the conclusions in clear, easy to understand language. You can read them to the audience. Also give one or two sentences about what this likely means (your interpretation) in the big picture (i.e. come full circle back to your introduction) and perhaps some future directions. 5. CRITIQUE Please end your presentation with at least two or three major things that should be discussed. These should consist of things like: things that might be improved in the study, additional experiments that you think might be appropriate (better?), and general issues about object recognition (i.e. put the study in the “big picture” of the course). Discussion from the audience should be especially encouraged at this point, but you should be prepared to foster this by raising these issues (e.g. one slide with a list of issues).

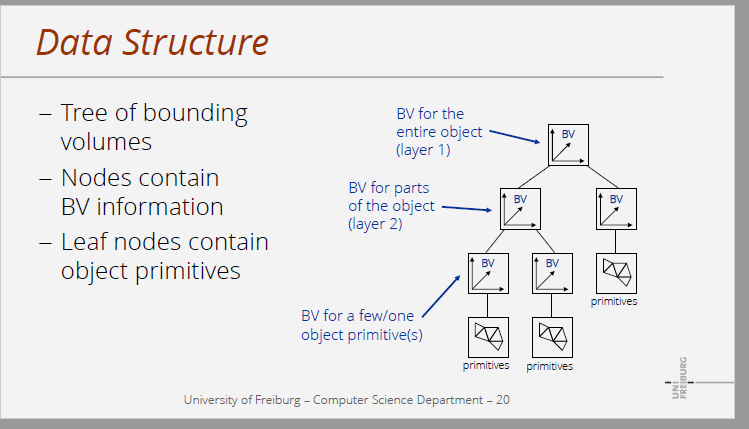
**Bounding Volume Hierarchies (**BVH**)**

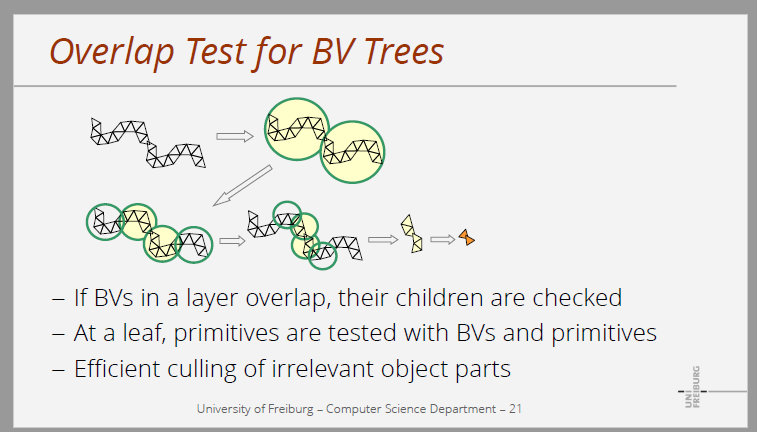
1. **INTRODUCTION**

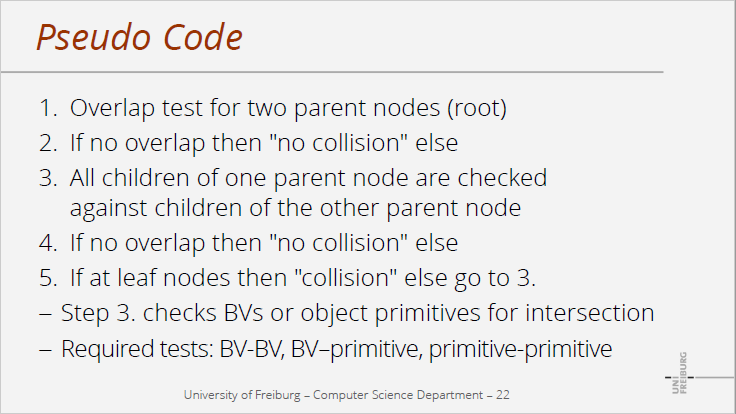
* Spatial data structures are important for many queries in animation. While space subdivision concepts, e.g. uniform grids, are popular in the neighbor search of SPH fluids, bounding volume hierarchies BVH are generally popular in all kinds of collision queries, in particular when dynamic rigid bodies are involved. The idea of a BVH is to enclose a complex geometry with a simple bounding shape and then to recursively subdivide the geometry into smaller parts which are enclosed by smaller bounding shapes. In a collision test, the resulting hierarchy is queried which is significantly more efficient than testing all primitives of the geometries.
* Collision detection is an essential part of physically realistic dynamic simulations
* In each time step  
  - Detect collisions  
  - Resolve collisions   
  - Compute dynamics
* Collision detection for polygonal models is in O(n^2) without the BV methods
* Simple bounding volumes –encapsulating geometrically complex objects –can accelerate the detection of collisions
* 

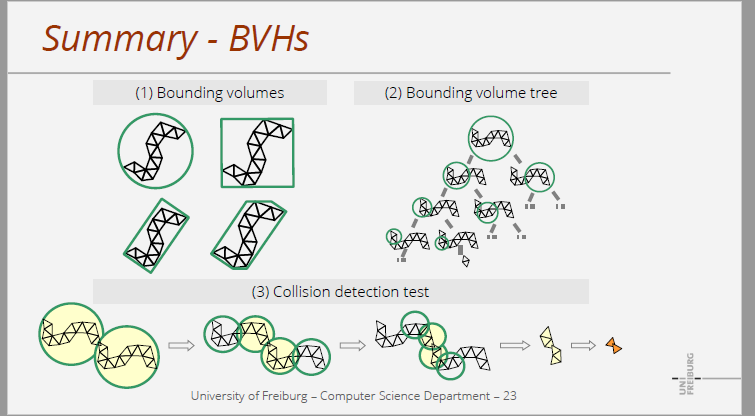


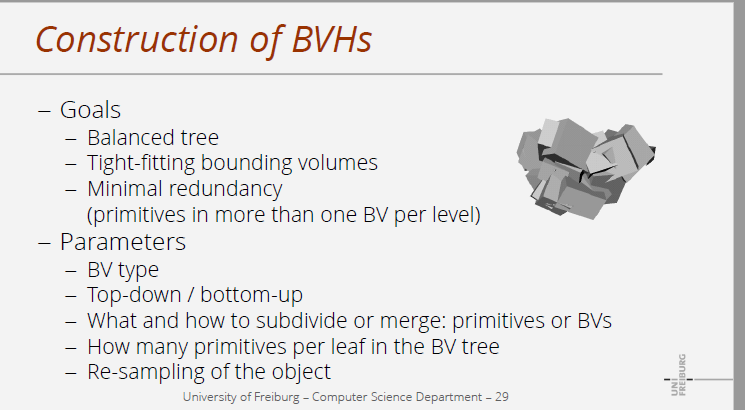












**Efficient Collision Detection Using: this is the most relevant paper to read form**

− why bvh is Alternative to space subdivision? − Useful for collision queries

**• Introduction**

**• Bounding Volume Types**

**• Hierarchy**

**• Hierarchy Construction**

**• Hierarchy Update**

**• Hierarchy Traversal**

**• Comparison Rigid-Deformable Objects**

**• Examples and Conclusion**

**Introduction:**

*What is BVH?*

A Bounding Volume Hierarchy (BVH) is a tree structure on a set of geometric objects like a [3D configurator](https://viscircle.de/?page_id=18591&lang=en). All geometric objects are wrapped in bounding volumes that form the leaf nodes of the tree. These [nodes](https://viscircle.de/why-skinning-is-so-important-for-any-rigging-project/?lang=en) are then grouped as small sets and enclosed in larger sets. These in turn are also grouped recursively and included in other larger boundary volumes, resulting in a tree structure with a single boundary volume at the top of the tree. Volume boundary hierarchies are used to efficiently support multiple operations on sets of geometric objects, such as collision detection and [raytracing](https://viscircle.de/hybrid-rendering-for-real-time-lighting-raytracing-vs-screening/?lang=en). **https://viscircle.de/guide-for-beginners-what-is-a-bounding-volume-hierarchy/?lang=en**

Bounding volume hierarchies (BVH) have become a widely used

alternative to kD-trees as the acceleration structure of choice in

modern ray tracing systems. However, BVHs adapt poorly to nonuniformly

tessellated scenes, which leads to increased ray shooting

costs. This paper presents a novel and practical BVH construction

algorithm, which addresses the issue by utilizing spatial

splitting similar to kD-trees. In contrast to previous preprocessing

approaches, our method uses the surface area heuristic to control

primitive splitting during tree construction. We show that our algorithm

produces significantly more efficient hierarchies than other

techniques. In addition, user parameters that directly influence

splitting are eliminated, making the algorithm easily controllable.

*Challenge:*

The choice of bound volume is determined by a compromise between two targets. On the one hand, we want to use boundary bands that have a very simple shape. So we only need a few bytes to store them. In addition, cut tests and distance calculations are simple and fast. On the other hand, we would like to have bounding volumes that fit very closely to the corresponding data objects. One of the most commonly used bounding volumes is an axially parallel minimum bounding frame. The axis-oriented minimum boundary box for a given set of data objects is easy to calculate, requires only a few bytes of memory, robust cut tests are easy to implement, and extremely fast. **https://viscircle.de/guide-for-beginners-what-is-a-bounding-volume-hierarchy/?lang=en**

*What makes BVH good?*

There are several desired properties for a BVH that should be considered when designing for a particular application.

* The nodes contained in a particular subtree should be close together. The further down the tree, the closer the nodes should be to each other.
* Each node in the BVH should have a minimum volume.
* The sum of all limiting volumes should be minimal.
* More attention should be paid to the node near the root of the BVH.
* If you prune a node near the root of the tree, more objects will be removed from further consideration.
* The overlap volume of the sibling nodes should be minimal.
* The BVH should be balanced both in its node structure and in its contents. When balancing, as much of the BVH as possible can be cut if a branch is not crossed.

With regard to the structure of the BVH, it must be decided to what extent (number of children) and in what size the tree representing the BVH should be used. A tree of small degree will be of larger height. This increases the traversal time from root to leaf. On the other hand, each node visited requires less work to check its children for overlaps. The opposite is true for a high-grade tree: although the tree has a lower height, more work is required at each node. In practice, binary trees (grade = 2) are by far the most common. One of the main reasons is that binary trees are easier to create. **https://viscircle.de/guide-for-beginners-what-is-a-bounding-volume-hierarchy/?lang=en**

*Trees structure types:*

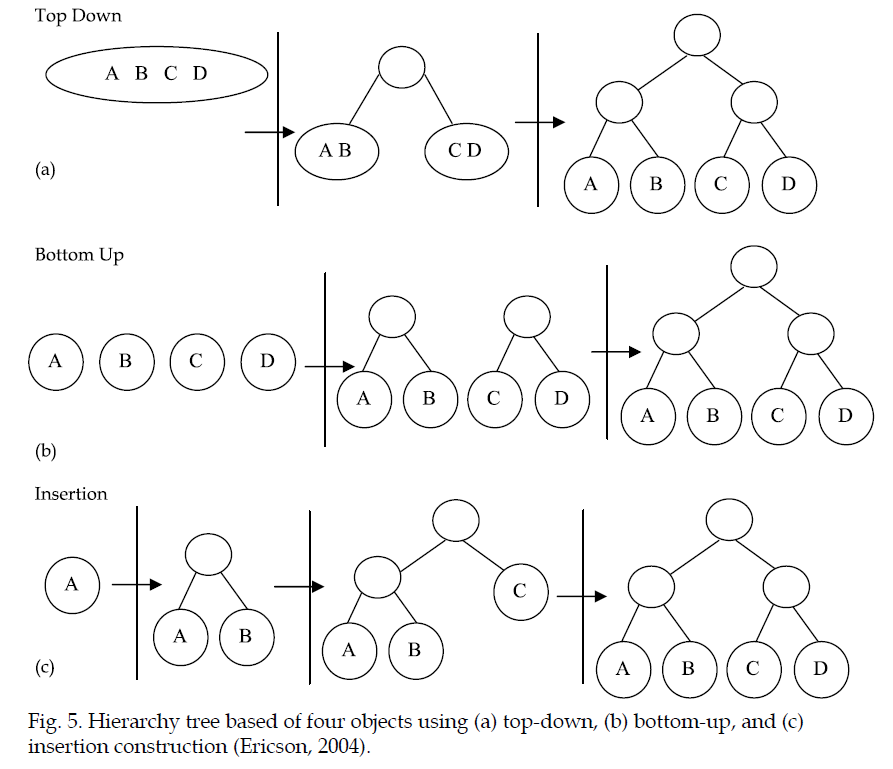
There are three main categories of tree building methods:

1. top-down
2. bottom-up
3. insert methods.

Top-down methods go so far as to partition the input set into two (or more) subsets, limit them in the selected boundary volume, and then partition them recursively until each subset consists of only a single primitive (leaf nodes are reached). Top-down methods are easy to implement, quick to construct, and by far the most popular, but do not lead to the best possible trees in general.

Bottom-up methods start with the input record as leaves of the tree and then group two of them into a new (internal) node, proceeding in the same way until everything is grouped under a single node. Bottom-up methods are more difficult to implement, but are generally likely to produce better trees. Some recent studies suggest that the construction speed in low-dimensional space can be significantly improved (equal to or superior to the top-down approaches) by sorting objects by the space fill curve and approximately bundling them based on this sequential order.

Both top-down and bottom-up methods are considered offline because they both require all primitives to be available before construction begins. Insert methods build the tree by inserting one object after another, starting with an empty tree. The insertion position should be chosen so that the tree grows as little as possible after a cost key figure. Insertion methods are considered to be online methods because they do not require all primitives to be available before construction begins and therefore updates can be performed at runtime. **https://viscircle.de/guide-for-beginners-what-is-a-bounding-volume-hierarchy/?lang=en**



**Ref image: Bounding Volume Hierarchies for Collision Detection**

**2.4 Partitioning strategies and splitting algorithm**

The simplest way to partition an object is to split the object equally with respect to local

coordination axes of the object. The split can be done according to the median-cut algorithm

style or several other strategies as follows(Ericson, 2004):-

 *Minimize the sum of the volumes (or surface areas) of the child volumes*. Minimizing the sum

of value can effectively minimize the probability of intersection - tightness bounding

volume must be used.

 *Minimize the maximum volume (surface area) of the child volumes.* Divides the child volume

into equal size by making larger volume as small as possible.

 *Minimize the volume (surface area) of the intersection of the child volumes.* Complex to be

implemented as it depends on the tightness of bounding volume.

 *Minimize the separation of the child volumes.* Where each child will be separated, it helps to

decrease the probability of both children being traversed at the same time.

 *Divide primitives equally between the child volumes*. Divide the object into two equal parts

as mentioned earlier.

 *Hybrid combination of multiple bounding-volumes* **Bounding Volume Hierarchies for Collision Detection**

**2.4.1 Choosing the splitting point**

**

Fig. 7. An example of (a) Object median splitting (b) Object mean splitting (c) Spatial median

splitting based on (Ericson, 2004)

Computer Graphics

48

 *Median of the centroid coordinates* (object median) – split at the object median (distributed

parts) resulting a well-balanced tree.

 *Mean of centroid coordinates* (object mean) – (Klosowski et al., 1998) stated that better

performance is obtained using object mean compared to object median. It was claimed

that splitting at the object mean gives smaller volume trees.

 *Median of the bounding-* **Bounding Volume Hierarchies for Collision Detection**

**3. Bounding volume hierarchies**

BVH is simply a tree structure that represents geometric models with specific bounding

volumes. It works like a tree that has a root (upper division), a group of leafs (middle

division) and a leaf (last division). Each node has it bounding-volumes that cover the

children nodes. The main idea of BVH is to build a tree that has a primary and secondary

root where each of the secondary nodes is stored as leaf. BVH allows intersection to occur

without searching for non-colliding pairs from the hierarchy tree. For example, given two

objects with their BVH, when root of the hierarchies do not intersect, the calculation will not

be done for both objects. However, when roots of both hierarchies intersect, it will check for

intersection between roots of one of the hierarchy’s tree with the children of the other

hierarchy’s tree. In this case, it recursively checks again whether there is intersection

between both objects at middle level until the correct intersection is found. Figure 1

previously shows how object is partitioned into several parts. While Figure 11 depicts the

basic algorithms for detecting collision between two hierarchies (Kamat and Martinez, 2007,

Nguyen, 2006)

Beginning at the root nodes of two given trees

1. Check for intersection between two parent nodes

2. If there is no intersection between two parents

3. Then stop and report “no collision”

4. Else check all children of one node against all

Children of the other node

5. If there is intersection between any children

6. Then If at leaf nodes

7. Then report “possible collision”

8. Else go to Step 4

9. Else skip and report “no collision”

Fig. 11. BVH traversal algorithm proposed by (Nguyen, 2006) for collision detection between

two BVH *-* **Bounding Volume Hierarchies for Collision Detection**

**3.1 BVH construction**

The construction of BVH is started by selecting type of the tree. There are multiple types of

BVH construction available. One of the most popular tree constructions is binary tree, which

has following nodes definition:-

a. **Nodes** – A node may contain specific value or a condition that represents a separate

data structure or a tree of its own. In BVH, it contains bounding-volume with their tree

ID. Each node may become parent, child or leaf nodes (in computer science tree grow

downward where the root is at the top of the tree). The node that produces child nodes

is called parent node, superior node or ancestor node. The length starting from root

node down to the leaf node determines the height of the tree. The depth of the tree is

the length of the path to its root.

b. **Root Nodes** – The topmost node in binary tree is called root node. Actually, root node

does not have any parents. It is the starting node where all the operations of the tree

will begin normally. Root Node is connected to each child nodes downward by using

branch or subtree connector. Branch of subtree connector is one of the important

elements that are connecting each node with their parent nodes.

c. **Leaf Nodes** – The bottommost node in binary tree is called the leaf nodes. It does not

have any child nodes and mostly contain the last value or conditions. In BVH, it may

contain few triangles or maybe one triangle.

d. **Internal Nodes** – The functional node where it has both parent nodes and child nodes.

It is not leaf nodes as it still has child nodes and linked by parent node.

e. **Subtree** – A portion of a tree data structure that can be viewed as a complete tree in

itself. It represents some other parts of the tree that can be manipulated by user in

programming. *-* **Bounding Volume Hierarchies for Collision Detection**

**3.2 BVH cost function**

This section will describe the overview of hierarchical method that is used in the proposed

urban simulation. BVH is proven to be the most efficient method for collision detection

(Sulaiman et al., 2009, Chang et al., 2008, Bergen, 2004, Bergen, 1999). Thus, in this research,

the hierarchical cost function that has been used by previous researchers will be used. Basic

cost function was first formulated by (Weghorst et al., 1984) for hierarchical method in ray

tracing and later was applied by (Gottschalk et al., 1996) and enhanced by (Klosowski et al.,

1998). The calculation of execution time is formulated as follows:

T = Nv X Cv + Np X Cp + Nu X Cu + Co (1)

Where

*T*: total execution time for detecting interference

*Nv*: number of BV pairs overlap tests

*Cv*: time require for testing a pair of bounding-volumes

*Np*: numbers of primitive pairs overlap tests

*Cp*: time require for testing a pair of primitives for interference

*Nu*: numbers of nodes that need to be updated

*Cu*: cost of updating each node

*Co*: Cost of one-time processing

**Bounding Volume Hierarchies for Collision Detection**

**Memory-optimized bounding-volume hierarchies:**

Bounding-volume hierarchies are the most common collision detection (CD) structures when

dealing with non-convex meshes or polygon soups [1]. They have been used in numerous

freely-available collision detection packages (RAPID [2], SOLID [4], QuickCD [3], PQP

[10], etc) to investigate the performance of various bounding volumes (BV). While using BVtrees

is a very efficient method when it comes to detecting collisions between arbitrary

geometries, it still has one major drawback : memory requirements.

A lot of papers have focused on the efficiency of collision queries : various ways of building

the trees have been explored [4], experiments have been made with hybrid hierarchies [14],

and almost the whole BV-spectrum (1) has been implemented in at least one available CD

library.

However, memory requirements have not received that much attention. BV-trees often require

a non-negligeable amount of bytes per triangle, which can ultimately lead to major problems

[8]. Memory-optimized bounding-volume hierarchies

A complete BV-tree is made of 2\*N-1 nodes, where N is the number of primitives (usually

triangles) in the input model. In complete trees each leaf node contains a single primitive,

which means there are N leaf nodes in them, and N-1 internal nodes. Memory-optimized bounding-volume hierarchies

**Early optimizations:**

1. The first way is to have more than one primitive in each leaf node
2. The second way is to reduce Nbi and Nbl.

**Aggressive optimizations**

1. Wiping the leaf nodes out
2. Quantized trees Memory-optimized bounding-volume hierarchies

BVHs are more efficient to build than kd-trees, which generally deliver slightly faster ray intersection tests than BVHs but take substantially longer to build. On the other hand, BVHs are generally more numerically robust and less prone to missed intersections due to round-off errors than kd-trees are. Implementation Bounding Volume Hierarchies https://www.pbr-book.org/3ed-2018/Primitives\_and\_Intersection\_Acceleration/Bounding\_Volume\_Hierarchies

*NOTE: look into `Implementation Bounding Volume Hierarchies` to understand the implementation.*

*WHERE WE USE IT `APPLICATIONS`?*

**Use.**

BVHs are often used in ray tracing to eliminate potential intersection candidates within a scene by omitting geometric objects in limited volumes that are not intersected by the current beam. Use these BVs as simplified surface representation for fast approximate collision detection test. **https://viscircle.de/guide-for-beginners-what-is-a-bounding-volume-hierarchy/?lang=en**

*Examples of BVs:*

• Spheres

• Discrete oriented polytopes (k-DOPs) Axis-aligned bounding boxes (AABB)

• Object-oriented bounding boxes (OBB)

WHY?

* ***Implementation pseudo code***:

*`Implementation Bounding Volume Hierarchies`*

**Ref: Bounding Volume Hierarchies for Collision Detection**

**Go in depth with these:**

* **Fitting an OBB “**collision queries using OBB**”**
* **AAC** Efficient BVH Construction via Approximate Agglomerative Clustering