



Real-Time Collision Detection for Dynamic Virtual Environments

Gabriel Zachmann, Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger,
Laks Raghupathi, Arnulph Fuhrmann

► To cite this version:

Gabriel Zachmann, Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Laks Raghupathi, et al.. Real-Time Collision Detection for Dynamic Virtual Environments. IEEE VR Tutorials, Mar 2005, Bonn, Germany. IEEE Computer Society, pp.310, 2005, 10.1109/VR.2005.1492815 . inria-00537446

HAL Id: inria-00537446

<https://inria.hal.science/inria-00537446v1>

Submitted on 18 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Tutorial:
**Real-Time Collision Detection
for Dynamic Virtual
Environments**

**Bounding Volume
Hierarchies**

Stefan Kimmerle
WSI/GRIS
University of Tübingen



- **Introduction**
- **Bounding Volume Types**
- **Hierarchy**
 - Hierarchy Construction
 - Hierarchy Update
 - Hierarchy Traversal
- **Comparison Rigid-Deformable Objects**
- **Examples and Conclusion**



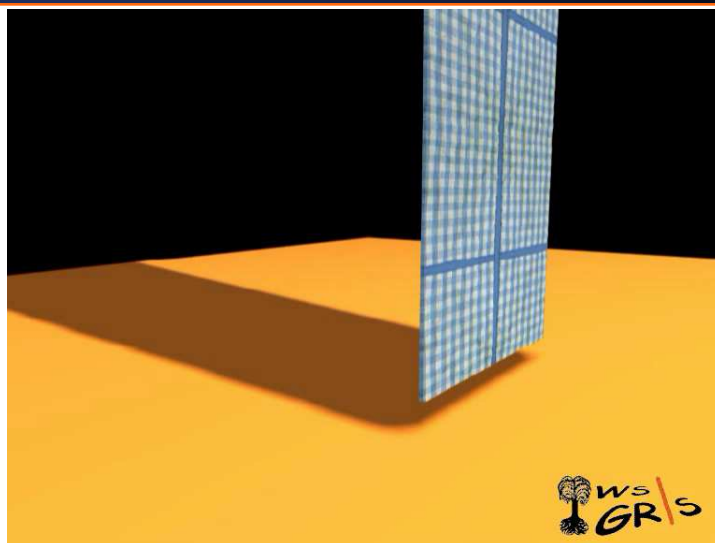
Problem of Collision Detection:

Object representations in simulation environments
do not consider impenetrability.

Collision Detection: Detection of interpenetrating objects.

The problem is encountered in

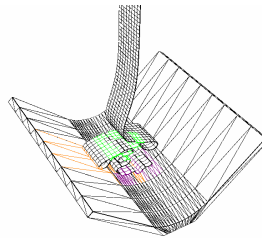
- computer-aided design and machining (CAD/CAM),
- robotics,
- automation, manufacturing,
- computer graphics,
- animation and computer simulated environments.





Definition of Bounding Volume Hierarchy (BVH):

Each node of a tree is associated with a subset of primitives of the objects together with a bounding volume (BV) that encloses this subset with the smallest instance of some specified class of shape.

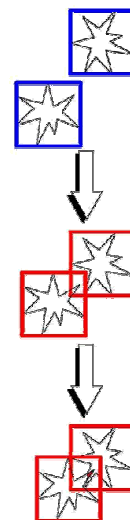


Use these BVs as simplified surface representation for fast approximate collision detection test:

Examples of BVs:

- Spheres
- Discrete oriented polytopes (k-DOPs)
 - Axis-aligned bounding boxes (AABB)
- Object-oriented bounding boxes (OBB)

- Check bounding volumes to get the information whether bounded objects **could** interfere.
- Avoid checking all object primitives against each other.
- Assumption that collisions between objects are rare.

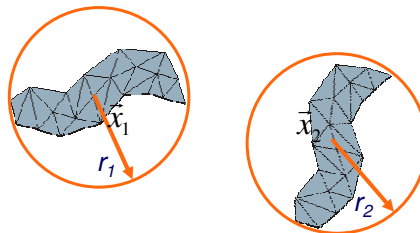




- Introduction
- **Bounding Volume Types**
- Hierarchy
 - Hierarchy Construction
 - Hierarchy Update
 - Hierarchy Traversal
- Comparison Rigid-Deformable Objects
- Examples and Conclusion



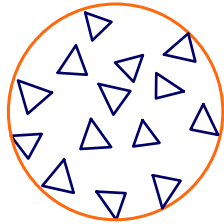
Spheres are represented by center \vec{x} and radius r .



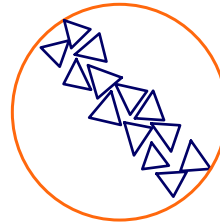
Two spheres do not overlap if $(\vec{x}_1 - \vec{x}_2) \cdot (\vec{x}_1 - \vec{x}_2) > (r_1 + r_2)^2$



Sphere as bounding volume:



good choice



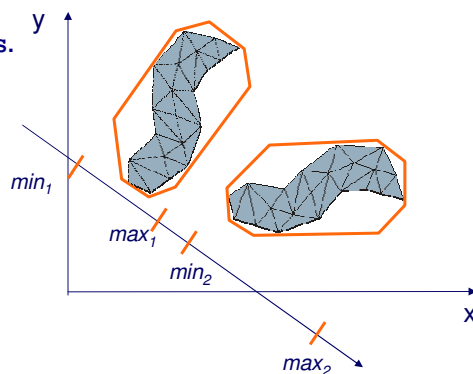
bad choice



Discrete oriented polytopes (k -DOP) are a generalization of axis aligned bounding boxes (AABB) defined by k hyperplanes with normals in **discrete** directions ($n_k: n_{k,j} \in \{0, \pm 1\}$).

k -DOP is defined by $k/2$ pairs of min, max values in k directions.

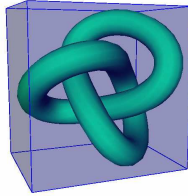
Two k -DOPs do not overlap, if the intervals in one direction do not overlap.



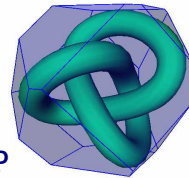


Different *k*-DOPs:

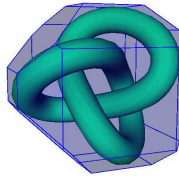
6-DOP
(AABB)



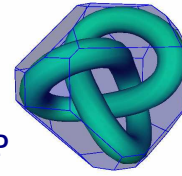
14-DOP



18-DOP



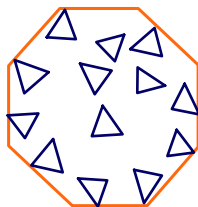
26-DOP



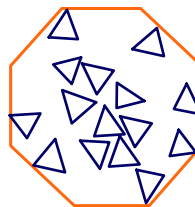
DOP



14-DOP as bounding volume:



optimal choice



also good choice



DOP



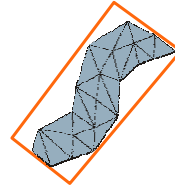
Object oriented bounding boxes (OBB) can be represented by the principal axes of a set of vertices. These axes have **no discrete orientation**. They move together with the object.

The axes are given by the Eigenvectors of the covariance matrix:

Centre of vertices \bar{x}_i :
$$\bar{\mu} = \frac{1}{n} \sum_{i=1}^n \bar{x}_i$$

Covariance matrix:
$$C_{jk} = \frac{1}{n} \sum_{i=1}^n \bar{x}_{ij} \bar{x}_{ik} \quad j, k = 1..3$$

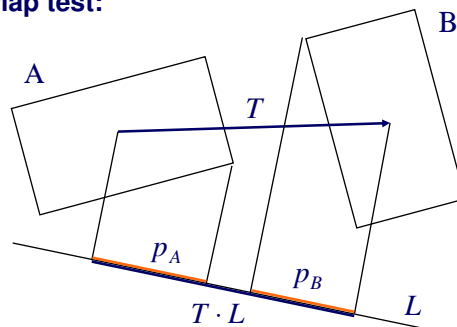
$$\bar{x}_i = \bar{x}_i - \bar{\mu}$$



OBB



OBB overlap test:



A and B do not overlap if:
$$\exists L: |T \cdot L| > p_A + p_B$$

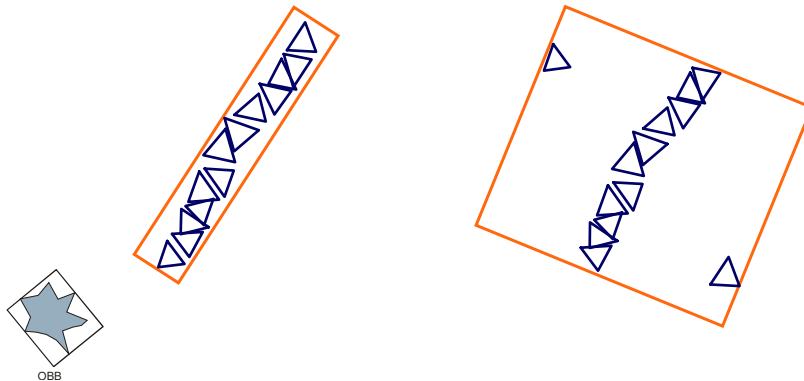
Problem: Find direction of L



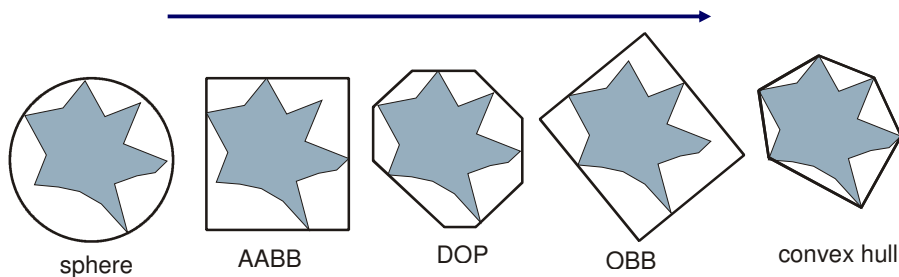
OBB



- Principal axes of an object are not always a good choice for the main axes of an OBB!
- Inhomogeneous vertex distribution can cause bad OBBs.



Better approximation,
higher build and update costs



Smaller computational costs
for overlap test

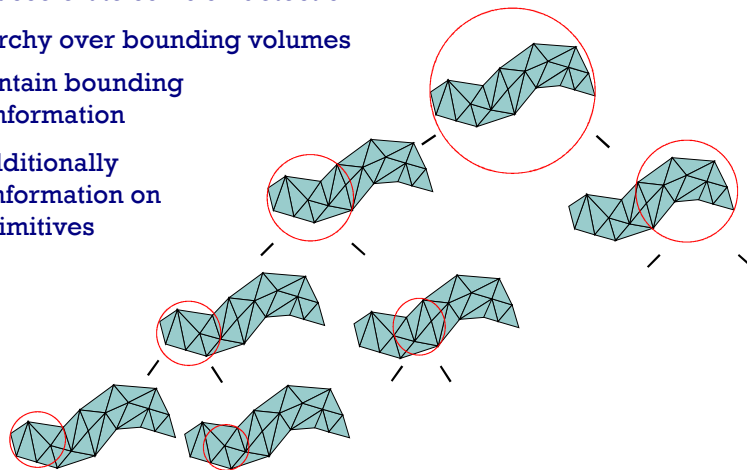


- Introduction
- Bounding Volume Types
- **Hierarchy**
 - Hierarchy Construction
 - Hierarchy Update
 - Hierarchy Traversal
- Comparison Rigid-Deformable Objects
- Examples and Conclusion



To further accelerate collision detection:

- use hierarchy over bounding volumes
- nodes contain bounding volume information
- leaves additionally contain information on object primitives



**Parameters**

- Bounding volume
- Type of tree (binary, 4-ary, k-d-tree, ...)
- Bottom-up/top-down
- Heuristic to subdivide/group object primitives or bounding volumes
- How many primitives in each leaf of the BV tree

Goals

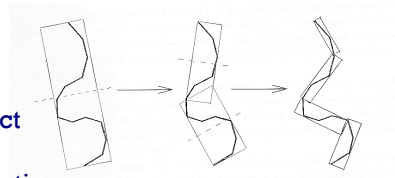
- Balanced tree
- Tight-fitting bounding volumes
- Minimal redundancy (primitives in more than one BV per level)

**Bottom-Up**

- Start with object-representing primitives
- Fit a bounding volume to given number of primitives
- Group primitives and bounding volumes recursively
- Stop in case of a single bounding volume at a hierarchy level

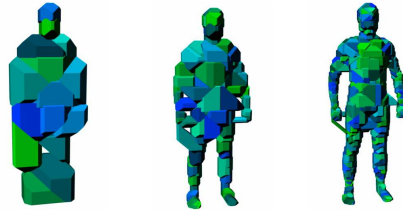
Top-Down

- Start with object
- Fit a bounding volume to the object
- Split object and bounding volume recursively according to heuristic
- Stop, if all bounding volumes in a level contain less than n primitives



**Top-Down Node-split:**

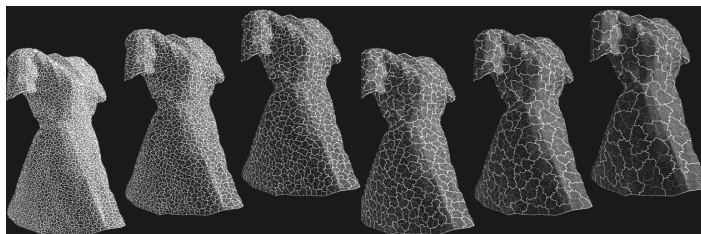
- Split k -DOP using heuristic:
 - Try to minimize volume of children (Zachmann VRST02).
 - Split along the longest side of the k-DOP (Mezger et al. WSCG03).



- The splitting continues until n single elements remain per leaf.

**Bottom-Up Node-grouping:**

- Group nodes using heuristic:
 - Try to get round-shaped patches by improving a shape factor for the area (Volino et al. CGF94).



- Group until all elements are grouped and the root node of the hierarchy is reached.

**Updating is necessary in each time step due to movement/deformation of simulated object.**

Difference between rigid and deformable objects:

- For rigid objects: transformations can be applied to complete object.
- For deformable objects: all BVs need to be updated separately.
 - Update is possible top-down or bottom-up.
 - To avoid a complete update of all nodes in each step, different update strategies have been proposed.



Some object transformations can be simply applied to all elements of the bounding-volume tree:

Spheres

- Translation, rotation

**Discrete Orientation Polytopes**

- Translation, no rotation
(discrete orientations of k hyperplanes for all objects)

**Object-Oriented Bounding Boxes**

- Translation, rotation
(box orientations are not fixed)





Larsson and Akenine-Möller (EG 2001):

- If many deep nodes are reached, bottom-up update is faster.
- For only some deep nodes reached, top-down update is faster.

-> Update top half of hierarchy bottom-up

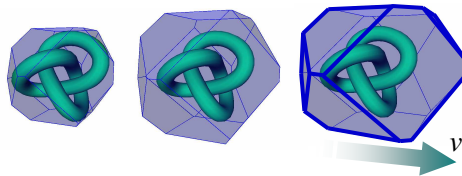
-> only if non-updated nodes are reached update them top-down.

- Reduction of unnecessarily updated nodes!
- Leaf information of vertices/faces has to be stored also in internal nodes -> higher memory requirements.



Mezger et al. (WSCG 2003):

- Inflate bounding volumes by a certain distance depending on velocity.



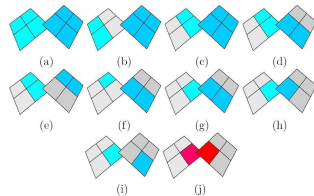
Update is only necessary if enclosed objects moved farther than that distance.

-> Fewer updates necessary.

-> More false positive collisions of BVs.

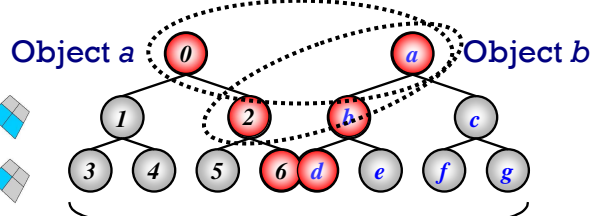


Binary trees:

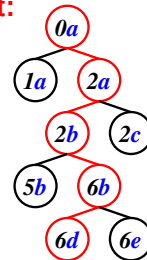


Minimize probability of intersection as fast as possible:

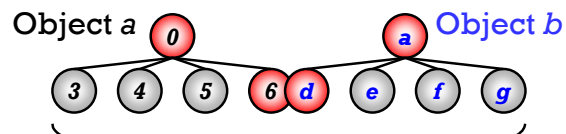
- Test node with smaller volume against the children of the node with larger volume.



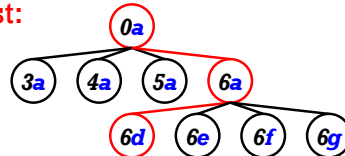
Collision test:



4-ary Trees:



Collision test:



Higher order trees:

- Fewer nodes
- Total update costs are lower
- Recursion depth during overlap tests is lower, therefore lower memory requirements on stack



- Introduction
- Bounding Volume Types
- Hierarchy
 - Hierarchy Construction
 - Hierarchy Update
 - Hierarchy Traversal
- **Comparison Rigid-Deformable Objects**
- Examples and Conclusion



Rigid Objects:

- use OBBs as they are usually tighter fitting and can be updated by applying translations and rotations.
- update complete BVH by applying transformations
- usually small number of collisions occur

Deformable Object:

- use DOPs as update costs are lower than for OBBs
- update by refitting or rebuilding each BV separately (top-down, bottom-up)
- high number of collisions may occur
- Self-collisions need to be detected
- use higher order trees (4-ary, 8-ary)



- Introduction
- Bounding Volume Types
- Hierarchy
 - Hierarchy Construction
 - Hierarchy Update
 - Hierarchy Traversal
- Comparison Rigid-Deformable Objects
- Examples and Conclusion



Interactive Cutting and Sewing



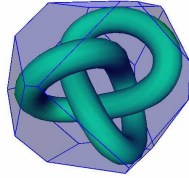
Conclusions



- BVHs are well-suited for animations or interactive applications, since updating can be done very efficiently.
- BVHs can be used to detect self-collisions of deformable objects while applying additional heuristics to accelerate this process.
- BVHs work with triangles or tetrahedrons which allow for a more sophisticated collision response compared to a pure vertex-based response.
- Optimal BVH and BV dependent on application (collision or proximity detection) and type of objects (rigid / deformable object)



Thank you!



Thanks to Matthias Teschner (University of Freiburg) and Johannes Mezger (University of Tübingen) for contributions to the slides!