Hierarchical Volumetric Approximation for 3D Object Similarity Database Searching

[*Efficient Geometry-based Similarity Search of 3D Spatial Databases*, Daniel A Keim, ACM SIGMOD '99]

Overview

Goal:

- Given a voxel-based query object, find ‘similar’ objects in the ‘large’ database.

Issues:

- *Registration* - Assume this is done!
- *Speed* - Comparisons are expensive, numerous
- *Storage* – Compact search structure

Solution:

- Hierarchical sets of volumetric approximations used as search keys into a balanced tree structure.

Similarity?

$$\delta_{VD}(v_1, v_2) \equiv 1 - \frac{\|v_1 \cap v_2\|}{\|v_1 \cup v_2\|}$$

Types:

- Congruent: $\delta_{VD}(v_s, v') = 0$
- ε-Similar: $\delta_{VD}(v_s, v') \leq \varepsilon$
- NN-Similar: $v' \mid \forall\, v \in$ DB: $\delta_{VD}(v_s, v') \leq \delta_{VD}(v_s, v)$

Valid metric: Has identity-preservation, commutativity, and triangle inequality properties.

Two Volumetric Approximations

Maximum Included Volume (*MIV*):

- MIV(v) $\subseteq v$, max. for a given type of approximation

Minimum Surounding Volume (*MSV*):

- $v \subseteq$ MSV(v), min. for a given type of approximation

a. MIV Approximation

b. MSV Approximation

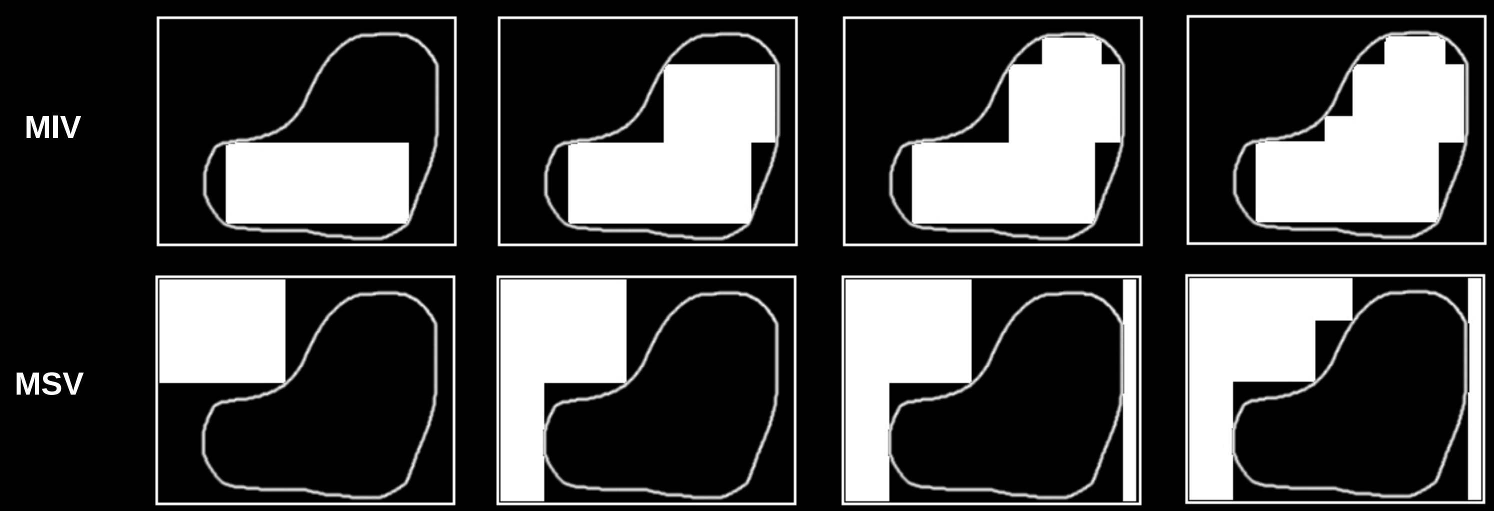
Hierarchy Defined

Accuracy parameter, *i*:

- $\forall i = 1\ldots k$: $MIV_i(v) \subseteq MIV_{i+1}(v)$
- $\forall j = 1\ldots k$: $MSV_{j+1}(v) \subseteq MSV_j(v)$

Example:

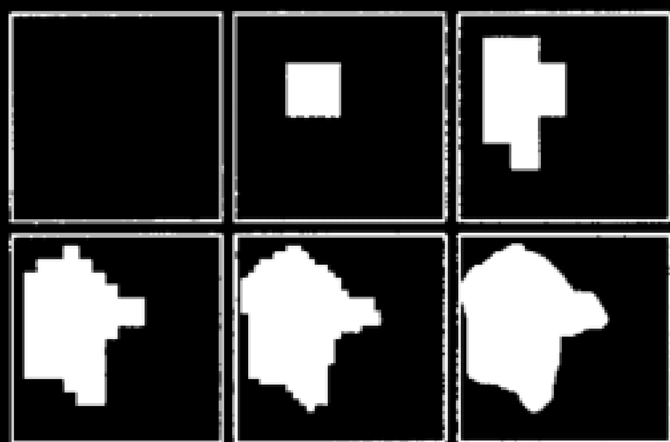
- Cuboids:



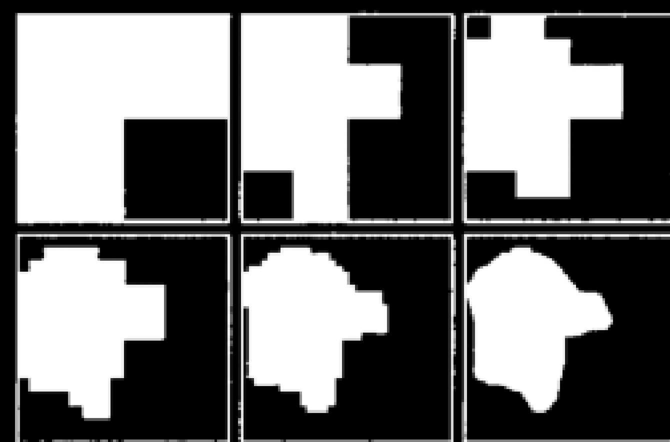
Hierarchy Defined (con't)

Example:

- Octrees: Faster comparisons



MIV



MSV

Differences:

- Octrees: Faster comparisons
- Cuboids: Better spatial resolution

Hierarchy Properties

Monotonicity:

- $|MIV_i(v) \cap/\cup v_s| \leq |MIV_{i+1}(v) \cap/\cup v_s|$
- $|MSV_j(v) \cap/\cup v_s| \geq |MSV_{j+1}(v) \cap/\cup v_s|$

Monotonicity of Union and Intersection:

- $\bigcap^{n}_{k=1} MIV_i(v_k) \subseteq \bigcap^{n}_{k=1} MIV_{i+1}(v_k)$
- $\bigcup^{n}_{k=1} MSV_{j+1}(v_k) \subseteq \bigcup^{n}_{k=1} MSV_j(v_k)$

Database Structure

Geometry-based Similarity Search Tree (GSST):

- Cluster ‘similar’ objects together in leaf nodes
- Store leaf objects as MIV, MSV approx.
- Store progressively coarse approximations for intersections/unions of child nodes’ MIVs/MSVs in the interior directory nodes.
- Use as minimally accurate approx. as possible.
- Keep the tree balanced (B-Tree)

Database Structure Defined

Leaf Nodes (LN):

- LN = (e_1, e_2,…, e_n)
- e = ($MIV_{i'}$, $MSV_{j'}$, *object.ptr*)
- $i' = min\{ i \mid \forall\ e_1,e_2 \in LN: e_1.MIV_i \neq e_2.MIV_i\}$
- $j' = min\{ j \mid \forall\ e_1,e_2 \in LN: e_1.MSV_j \neq e_2.MSV_j\}$

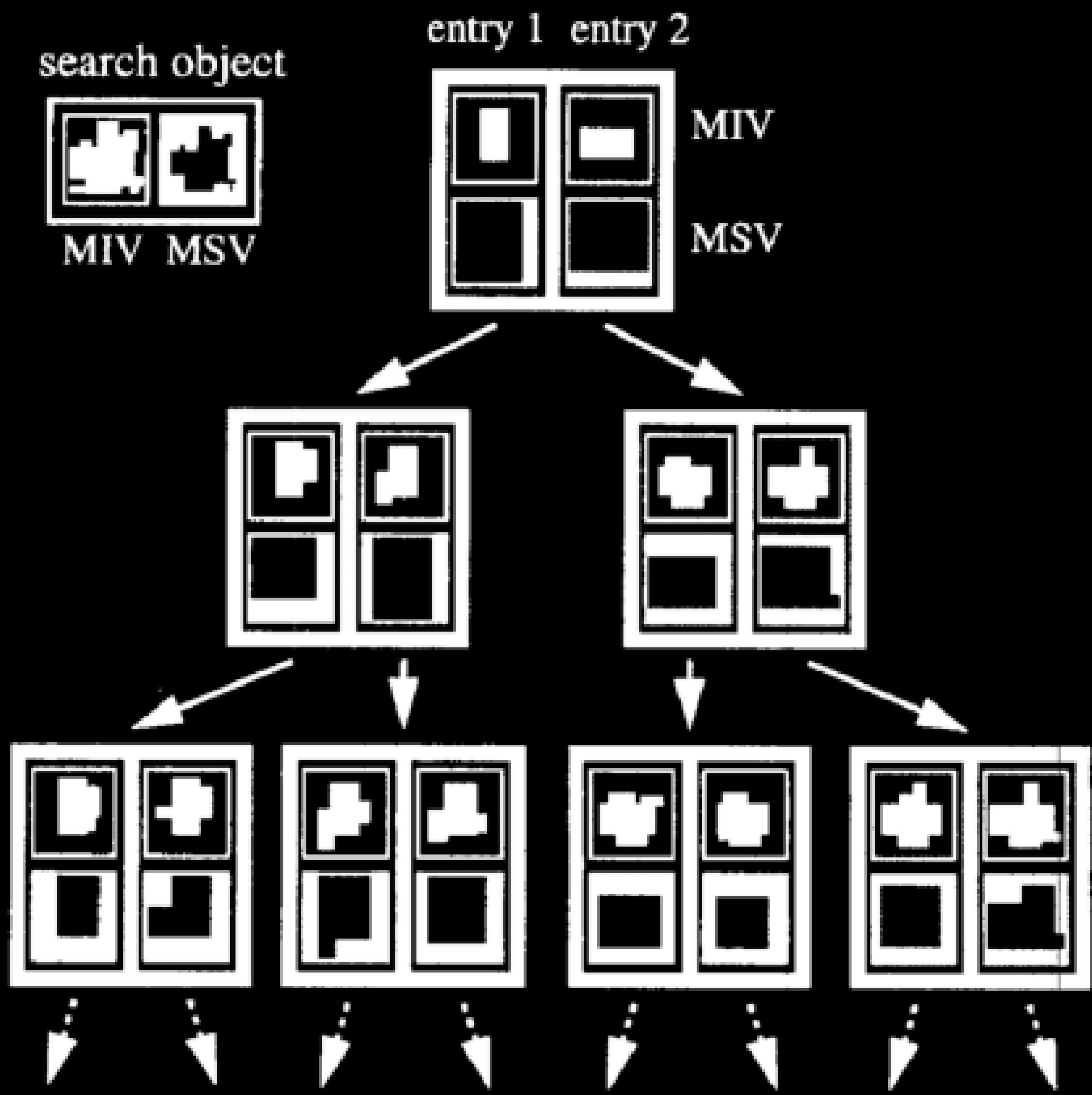
Database Structure Defined (cont'd)

Directory Nodes (DN):

- DN = $(e_1, e_2, \ldots, e_n)$
- $e = (MIV^{l}_{i'}, MSV^{l}_{j'}, \textit{child ptr})$
- $e.MIV^{l}_{i'} = \bigcap_{e' \in e.child} e'.MIV^{l+1}_{i'}$
- $e.MSV^{l}_{j'} = \bigcup_{e' \in e.child} e'.MSV^{l+1}_{i'}$
- $i' = min\{ i \mid i \leq i^*, \forall\ e_1, e_2 \in DN: e_1.MIV^{l}_{i} \neq e_2.MIV^{l}_{i}\}$
- $j' = min\{ j \mid j \leq j^*, \forall\ e_1, e_2 \in DN: e_1.MSV^{l}_{j} \neq e_2.MSV^{l}_{j}\}$
- *i*, j** are min *i, j* amongst DN's child node entries.

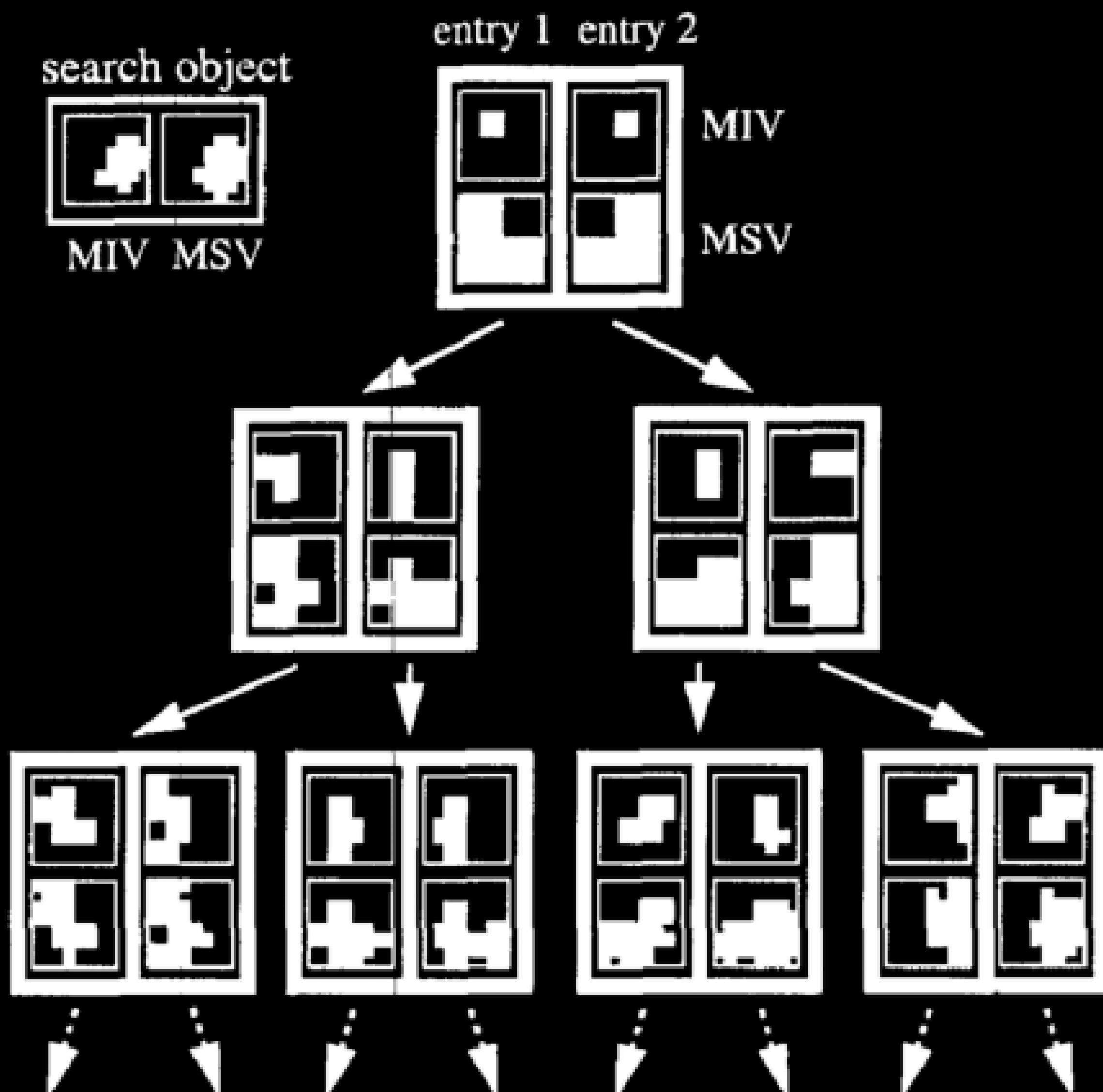
Database Structure Defined (cont'd)

Cuboid:



Database Structure Defined (cont'd)

Octree:



Similarity Metric

$$\delta_{VD}^{\min}(v_s, e) \equiv 1 - \frac{\|v_s \cap e.MSV\|}{\|v_s \cup e.MIV\|}$$

$$\delta_{VD}^{\max}(v_s, e) \equiv 1 - \frac{\|v_s \cap e.MIV\|}{\|v_s \cup e.MSV\|}$$

Properties:

- $\forall e' \in e.child : \delta^{min}(v_s,e) \leq \delta^{min}(v_s,e')$
- $\forall e' \in e.child : \delta^{max}(v_s,e) \geq \delta^{max}(v_s,e')$

- $\forall v \in subtree(e) : \delta^{min}(v_s,e) \leq \delta(v_s,v) \leq \delta^{max}(v_s,e)$

Follows from the properties of hierarchical volumes and the definition of the search tree…

Search Algorithm

ε-Similarity:

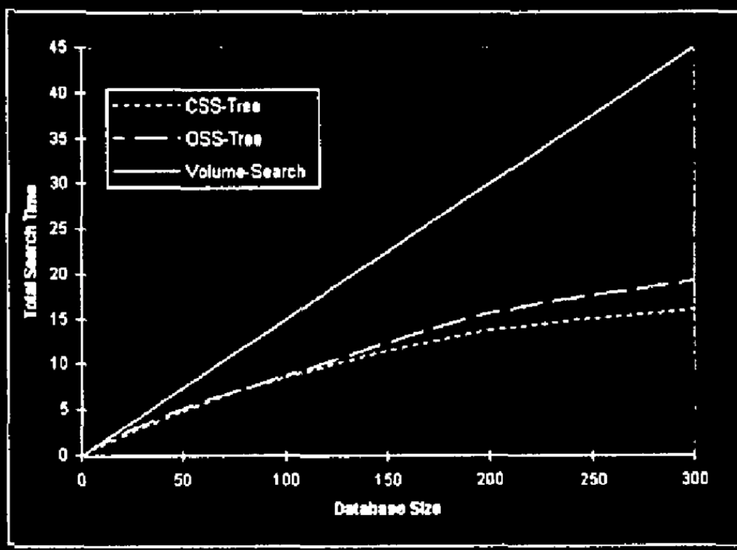
- Start at the root level
- Compute $\delta^{min}, \delta^{max}$ for every e $\in$ DN
- Prune subtrees with $\delta^{min} > \varepsilon$
- Add to result list leaves in subtrees with $\delta^{max} \leq \varepsilon$
- Move to next level and process remaining subtrees

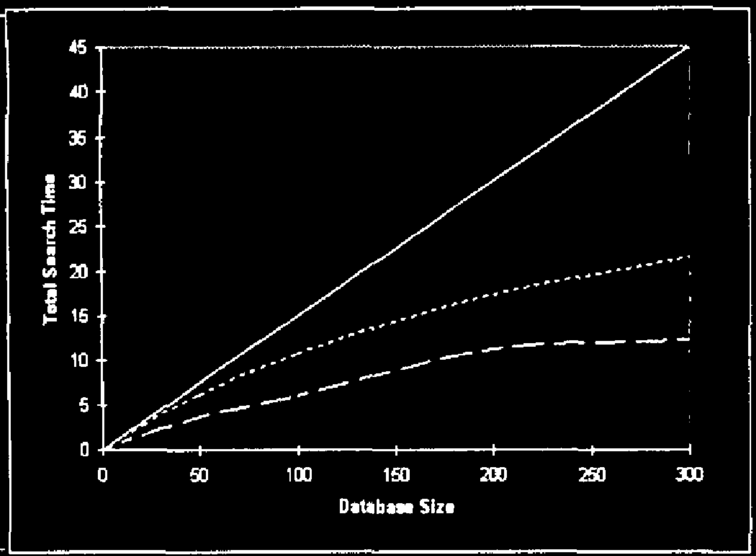
NN-Similarity:

- Start at Start at the root level
- Maintain smallest δ^{max} or $\delta(v, v_s)$ encountered, δ^*
- Prune subtrees/leaves with $\delta^{min}/\delta > \delta^*$
- Determine 'most promising candidate'
- Repeat until one leaf remains.

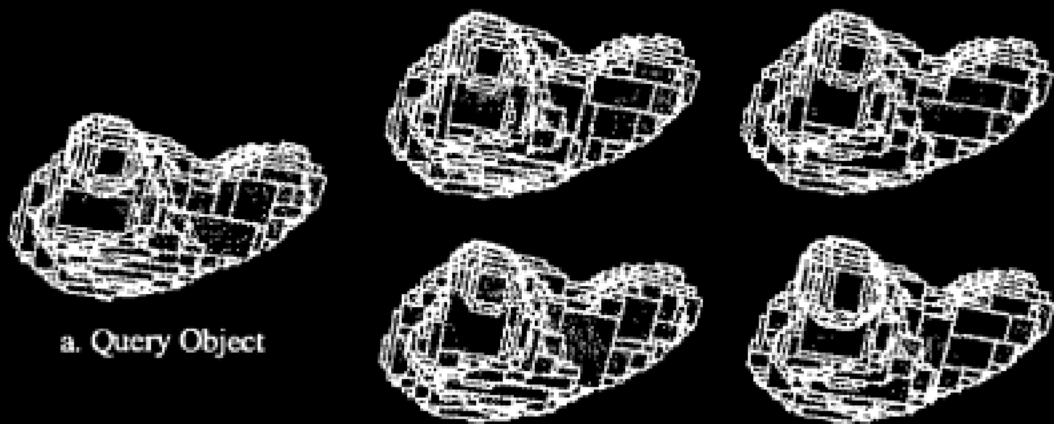
Results

- Search times sublinear in database size
- Can save space by saving incremental volume differences between node levels.
- Octrees incur lower CPU times – easier intersections, unions
- Cuboids incur fewer data accesses – more accurate spatially – faster pruning



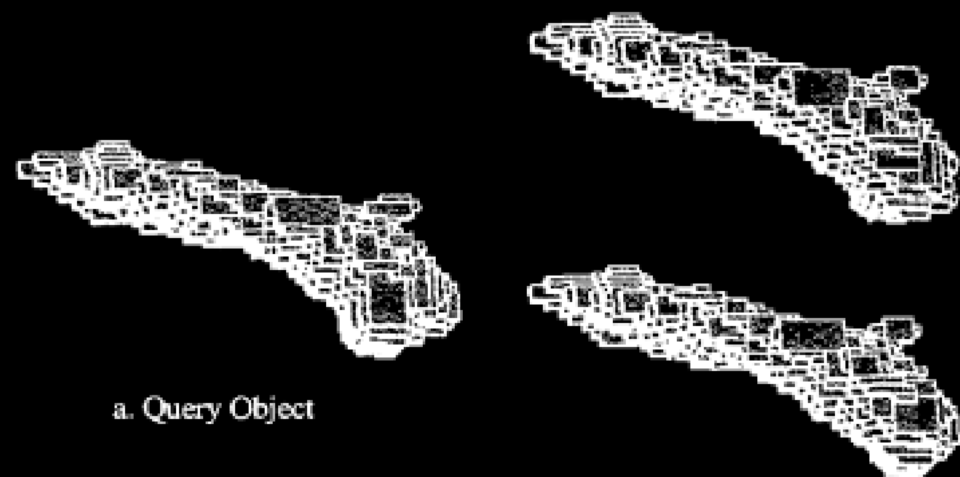


Results



a. Query Object

b. Query Results



a. Query Object

b. Query Results

[*Efficient Geometry-based Similarity Search of 3D Spatial Databases*,
Daniel A Keim, Proc. ACM SIGMOD Int. Conf. On Management of Data, 1999]