Bounding Volume Hierarchies

Object Partitioning

Verwendung:

- Ray Casting
- View-Frustum culling / occlusion culling
  (Szenegraph liefert BV-Hierarchy)
- Point location, nearest neighbor
- Datenbanken ("spatial joins", range queries, point queries, ...)
- Kollisionserkennung

Wird oft als Gegensatz zum space partitioning betrachtet (sehe ich weniger).

Def. BVH:

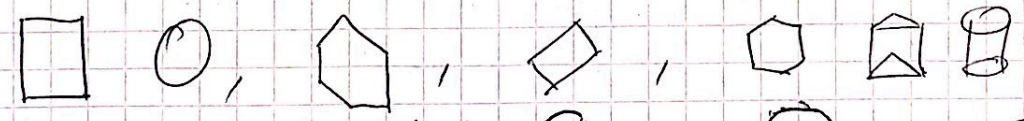
Gegeben Menge O von Objekten.

Falls $|O| \leq k$: $BVH(O) :=$ Blattknoten b, speichert O.

Sonst ($|O| > k$): $BVH(O) :=$ Knoten v mit $n(v)$ Kindern $v_1, \dots, v_n$, wobei $v_i := BVH(O_i)$, $O_i \subseteq O$, $\dot\bigcup O_i = O$.

Jeder Knoten v speichert außerdem ein "bounding volume" $BV(v) \in \mathcal{B}$ = Menge aller zur Verfügung stehenden BVs, so daß $\forall o \in O(v): o \subseteq BV(v)$.

Typ. BVs sind AABBs, Kugeln, DOPs, OBBs, konvexe H., Zylinder, Kombinationen.



(convex hull?) (implizite Flächen?) $\bigcirc \cap \square$ [SR-tree]

Bem.: Die Menge $\mathcal{B}$ muß nicht notw. homogen sein! D.h. sie kann auch Kugeln und Boxes und Zylinder enthalten.

Layered BVH: $\forall$ children v_i: $\forall i: BV(v_i) \subseteq BV(v)$

Wrapped BVH: $\forall$ leaves v_i underneath v: $O(v_i) \subseteq BV(v)$

Strategien:

- bottom - up
- top - down
- einfügen

Kriterien an BVs:

1) Tightness
2) Speicheraufwand
3) Schneller Test gegen Query - Objekt (Punkt, Strahl, anderes BV, ...)

Definition "Tightness": [OBB-Paper]

Sei B ein BV, G Geometrie, $G \subseteq B$.

Gerichtete Hausdorff-Distanz $h(B,G) = \max_{b \in B} \min_{g \in G} d(b,g)$

(d irgend eine Metrik, normalerweise L_2);

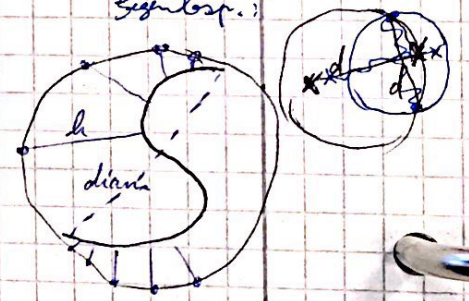
Durchmesser $\text{diam}(G) := \max_{g,f \in G} d(g,f)$.

Tightness $\tau := \frac{h(B,G)}{\text{diam}(G)}$, kleiner ist besser

(Achtung: Hausdorff ist sehr anfällig gegenüber Ausreißer!)
Gibt es bei BV's nicht

↑ Mit Kugeln als BV gilt: Damit ist $0 < \tau < \frac{1}{2}$

FRAGE: ist diam = Durchmesser der minimalen Bounding Sphere? ⇒ nein Gegenbsp.:



Meine Definition:

$$\tau := \frac{vol(BV(v))}{\sum_{o \in O(v)} vol(BV(o))},$$

mit $O(v)$ = Menge der Blätter unter v (die die Objekte speichern). Objekte in v

Bei dieser Def. kann $\tau < 1$ werden, weil sich überlappende Volumen mehrfach gezählt werden! I.A. ist hier $\tau > 1$.

Oder

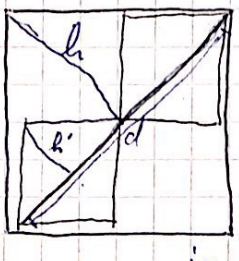
$$\tau := \frac{vol(BV(v))}{\sum_{c \in C(v)} vol(BV(c))},$$

mit $C(v)$ = Kinder

kleiner ist besser.
(Wir argumentieren: dieses Tightness-Maß macht mehr Sinn, weil es z.B. aus unserem Paper ableitet, daß bei der Splitting-Heuristik das Volumen der Kinder minimiert werden muß.)

Bem.:

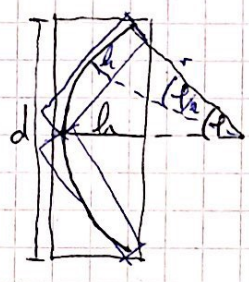
1. Tightness von AABBs in AABB-Hierarchien hängt hauptsächlich von "netter" Orientierung der Geom. ab (für Flächen niedriger Krümmung), und sie ist im wesentlichen konstant über die ganze Hierarchie!



$$\tau := \frac{h}{d}$$

$$\tau' = \frac{h'}{d/2} = \frac{h/2}{d/2} = \tau$$

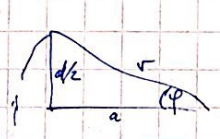
2. Tightness von OBBs in OBB-Hierarchien hängt nicht ab von Ori der Fläche, sondern eher von oberen Krümmung, und sie nimmt ungefähr linear mit der Tiefe in der Hierarchie ab (wird also besser):



$$h = r(1-\cos\varphi), \quad d = 2r\sin\varphi$$

$$\tau \cong \frac{1-\cos\varphi}{2\sin\varphi} \to 0 \quad \text{für } \varphi \to 0$$

verhält sich ungefähr $\frac{\varphi}{4}$. (s.u.)



$$\cos\varphi = \frac{a}{r} \qquad h = r - a = r - r\cos\varphi$$

$$\text{diam}(\Phi) = \sqrt{d^2+h^2} = \sqrt{r^2\left(4\sin^2\varphi + \underbrace{(1-\cos\varphi)^2}_{1-2\cos\varphi+\cos^2\varphi}\right)}$$

$$= r\sqrt{4\sin^2\varphi + \cos^2\varphi - 2\cos\varphi + 1}$$

$$3\sin^2 + \cos^2 + \sin^2 - 2\cos + 1 = 3\sin^2 + 2 - 2\cos$$

braucht man nicht

Taylor-Reihen einsetzen

$$\frac{1-\cos\varphi}{2\sin\varphi} \overset{!}{=} \frac{1-\left(1-\frac{1}{2!}\varphi^2+\frac{1}{4!}\varphi^4-\dots\right)}{2\left(\varphi-\frac{1}{3!}\varphi^3+\dots\right)}$$

$$\approx \frac{\frac{1}{2}\varphi^2}{2\varphi} = \frac{1}{4}\varphi = O(\varphi)$$

↑ entspricht "Tiefe" im BVH

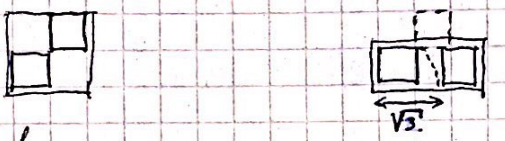
Aufbau-Strategien:

1. Bottom-up:

[Roussopoulos & Leifker: "Direct Spatial ... Packed R-trees" nur angedeutet; ca '84]

gegeben Menge S von BVs auf dem obersten Level der Hierarchie, soweit sie bis jetzt berechnet wurde;
für alle $b \in S$: bestimme (b_n, d_n) = nearest neighbor, dessen Abstand;
sortiere S bzgl. d_n;
fasse die ersten k zu einem Vaterknoten zus., die nächsten k, ...

Idee: diese Strategie produziert nicht mehr. BVs mit wenig "dead space":



(in 3D)

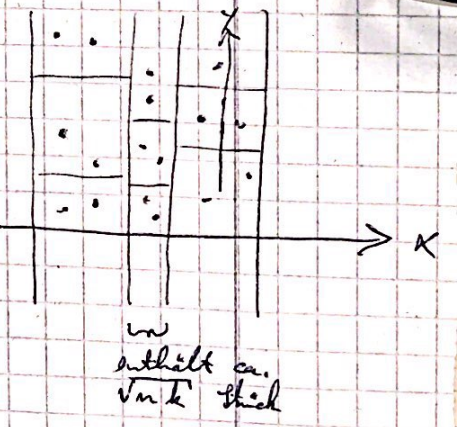
dead-space = $2^3 - 2 \cdot 1^3 = 6$

dead-space = $1^2 \cdot (\sqrt{3}+1) - 2 \cdot 1^3$
$= \sqrt{3} - 1 \approx 0.73$!

2. Bottom-up: (zuerst für die Ebene)

[Leutenegger, Edgington, Lopez: "STR ..." ~1995]

Start mit S, $|S| = n$.
Bestimme Mittelpkt c^i für alle $b^i \in S$;
sortiere S bzgl. c^i_x;
teile S in $\sqrt{\frac{n}{k}}$ vertikale "Slices" bzgl. c^i_x;
sortiere jeden Slice bzgl. c^i_y;
teile jeden Slice bzgl. c^i_y in $\sqrt{\frac{n}{k}}$ "Tiles";
fasse jeden Tile zu einem Vaterknoten zus.,
berechne BVs für jeden V.knoten,
bilde daraus die Menge S';
wiederhole ~~bis~~ mit S' bis $|S'| \leq k$.



Analog im $\mathbb{R}^d$: $\sqrt[d]{n/k}$ Slices, jeden Slice wieder in $\sqrt[d]{n/k}$ viele Unterslices unterteilen, so oft wie Dim.

3. Insertion:

[kommt für streaming geom. vielleicht wieder; s. Bither 2015]

Start mit leerem Baum.
Oder $\mathcal{B}$ = set of BV's of all O's
while $|S| \geq 1$

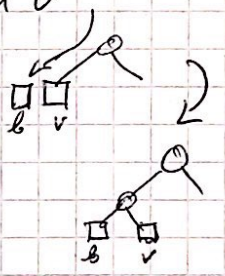
(*) wähle nächstes $b \in S$ ($\mathcal{B}$);

$v :=$ root

while $v \neq$ Blatt

(#) $v :=$ Kind w von v, so daß Einfügen von b in Teilbaum w den Gesamtbaum minimal verschlechtert

bilde neuen inneren Knoten mit Kindern b und v



Varianten betreffen nur Schritt (*) u. (#).

Anm. hier: $k=2$ (binärer Baum).

Für $k>2$ bekommt man einen B-tree,

und braucht zusätzliche Operation Split, wenn Knoten überläuft (s. R-tree und Co.)

Variante A: Surface area Heuristic [Macdonald '89/90] [R-trees & Co.] [Goldsmith & Salmon]

(#): wähle denjenigen Kind w, so daß area(bbox($w \oplus b$)) = min (Adding b to BVH(w))

(area = Betrag der Oberfläche)

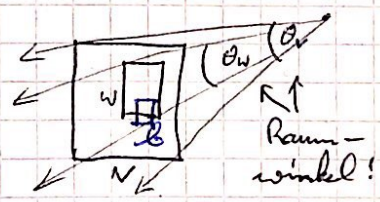
(*): "model order" oder zufällige Perm. davon (sortieren entlang einer Achse war ganz schlecht)

Begründung für area:

DV-Hierarchie wird für Strahltests verwendet;

Anm.: Strahl s kommt von etwas weiter weg;

$Pr[s \text{ trifft } w \oplus b \mid s \text{ trifft } v] = \frac{\theta_{w \oplus b}}{\theta_v} \approx \frac{\text{area}(w \oplus b)}{\text{area}(v)}$



Unentschieden auflösen durch Pgonarea

(passiert oft, wenn beide Kinder + b area = 0 liefern).

Statt Area Vol nehmen, falls Point-Loc.-Query

LSS-tree

Variante B für Ruzahn:

wähle Kind, dessen Mittelpkt am nächsten am M.pkt von B.

Problem bei Insertion: Reihenfolge —

wenn das erste BV schlecht gewählt ist, dann wird der ganze Baum zwangsläufig schlecht.

4. Top-down: (S aufteilen geschickt und Rekursion)

Generelle Idee: finde "gute" Kostenfkt für einen Split.

Variante A (für Raytracing):

[angelehnt an Gordon Müller '99]

$S =: L \dot\cup R$ (linke/rechte Teilmenge)

Kosten $C(L,R) := \frac{\text{area}(\text{bbox}(L))}{\text{area}(\text{bbox}(S))} \cdot C(L) + \frac{\text{area}(\text{bbox}(R))}{\text{area}(\text{bbox}(S))} \cdot C(R)$

$$C(S) := \min_{L \in \mathcal{P}(S)} C(L, S \setminus L)$$

Näherungsalgo:

[Gordon Müller '99; R* tree; Subramanian & Fussell: k-d trees]

forall $\alpha \in \{x,y,z\}$:

sort S entlang α bzgl. Mittelpkte der BVs / Objs;

bestimme $k^\alpha := \underset{j=0..n}{\text{argmin}} \left\{ \frac{\text{area}(b_1 .. b_j)}{\text{area}(S)} \cdot j + \frac{\text{area}(b_{j+1} .. b_n)}{\text{area}(S)} \cdot (n-j) \right\}$ (find min)

↑ nur ges. $L \subseteq S$ werden betrachtet

↑ Schätzung für C(L)

repeat along y, z

wähle davon bestes k^α.

Bem.: Sortieren muß man nur 1x machen, da sie erhalten bleibt beim Split.

Komplexität: $T(n) = \Theta(n) + T(\lceil \alpha n \rceil) + T'(\lfloor (1-\alpha) n \rfloor)$

$\Rightarrow T(n) \in \Theta(n) + \Theta(n \log n)$ (für Sortieren)

↑ ist mit jeder Reflexion ein anderes α!

Variante B (min. auch Vol):

wie vorher; zusätzlich:

bestimme auf jeder Achse α die m besten $k^\alpha \rightarrow L = \{k_i^\alpha\}$;

forall α:

bestimme $o^\alpha := \min \{ vol(bbox(b_1 .. b_{k_i^\alpha})) \cap vol(bbox(b_{k_i^\alpha+1} .. b_n)) \mid k_i^\alpha \in L, i = 1..m \}$

wähle das beste o^α

und den zugehörigen Split

[Meine Idee, basierend auf R* tree, 1990.]

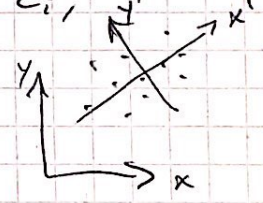
Variante C:

betrachte Mittelpkte c_i der BVs/Pgone $b_i \in S$;

Hauptachsentransformation liefert neue Pkte c_i';

lege Splitebene senkrecht zur H. achse mit längster Ausdehnung durch Schwerpunkt aller c_i'

(alternativ: Median)



[OBB-Paper]

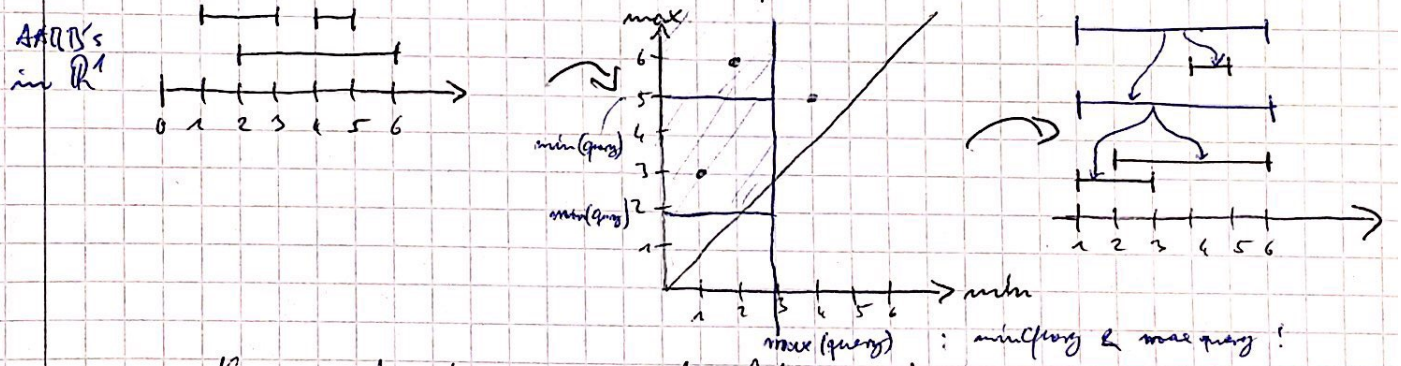
Configuration-space Box-tree:

[Vgl. vgl. de Berg, et al: "Box-trees and R-trees with near-optimal query time". 2001]

Betrachte Input-Boxes (AABB's) $b_i \in S$ im $\mathbb{R}^d$
als Punkte $p_i \in \mathbb{R}^{2d}$;
baue für diese einen k-d-Tree;
transformiere diesen in Box-Tree (AABB-):

Blatt p_i aus k-d-Tree → Blatt b_i im Box-tree (AABB);
innerer Knoten von k-d → innerer Knoten v im Box-tree
wobei $bbox(v) := bbox(c_1^v, c_2^v)$ ← Kinder von v

Der Algo tut eigtl nichts weiter,
als auf jeder Stufe die Menge der Boxes gemäß
irgend einer Koord bbox$^{\alpha}_{las/big}$ zu sortieren,
und per Median zu splitten!



Aber mit dieser Konstruktion kann man
zeigen:

Satz (o. Beweis.):
Die Rectangle-Intersection-Query im $\mathbb{R}^d$ kann man
mittels eines geeigneten Boxtrees in $O(n^{1-1/d} + k)$
Zeit im worst-case beantworten.
Man kann sie außerdem mit Boxtrees
höchstens in Zeit $\Omega(n^{1-1/d} + k)$ beantworten,
für $d \geq 3$.
(Also für $d=3$: $\Omega(n^{2/3})$!)

?

Vergl dieses Ergebnis mit Range-tree & Interval-Tree! (Tiefe & Speed)

~~Für Anwendung Ray-Tracing:~~

k-DOPs für Raytracing:

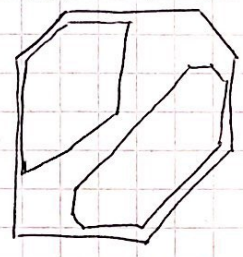
[Kay & Kajiya, Siggraph '96]

Wähle Menge $\mathcal{B} = \{B_1 \dots B_k\}$ fest, $k \geq 2d$, $B_j \in \mathbb{R}^d$,

mit setze $B_{k+i} := -B_i$, $i = 1, \dots, k$

Dazu ist DOP $D := \bigcap_{i=1}^{2k} H_i$, $H_i : x \cdot B_i - d_i \geq 0$,

$\rightarrow D = (d_1 \dots d_{2k})$ definiert Volumen



Schnitt zwischen D und Strahl $\vec{x} = \vec{p} + t\vec{v}$:

$$t_i := \frac{d_i - (B_i \cdot r)}{(B_i \cdot r)}$$ (wie Cyrus-Beck)

Beob.: 1. welches d_i ein "entering" bzw. "leaving" t_i ergibt bleibt konstant f.a. DOPs (bei gleichem Strahl).

2. die Skalarprodukte V. produkte bleiben konstant f.a. DOPs

(Die kann man durch Mult ersetzen)

(Mache den Check, ob das t-Intervall schon leer ist, abwechselnd mit Ber. des nächsten t_i.)

Bem.: Für Ray-Tracing müssen die Dri B_i nicht notwendigerweise paarweise antiparallel sein.

Für Überlappungstest zweier DOPs aber schon.

Für Coll.-Det.:

$D^i = (d_1^i, \dots, d_k^i)$

$\exists\, j = 1, \dots, \frac{k}{2}$: checke

$[d_{j+k}^1, d_j^1] \cap [d_{j+k}^2, d_j^2] = \phi$

$\Rightarrow D^1$ und D^2 schneiden sich nicht