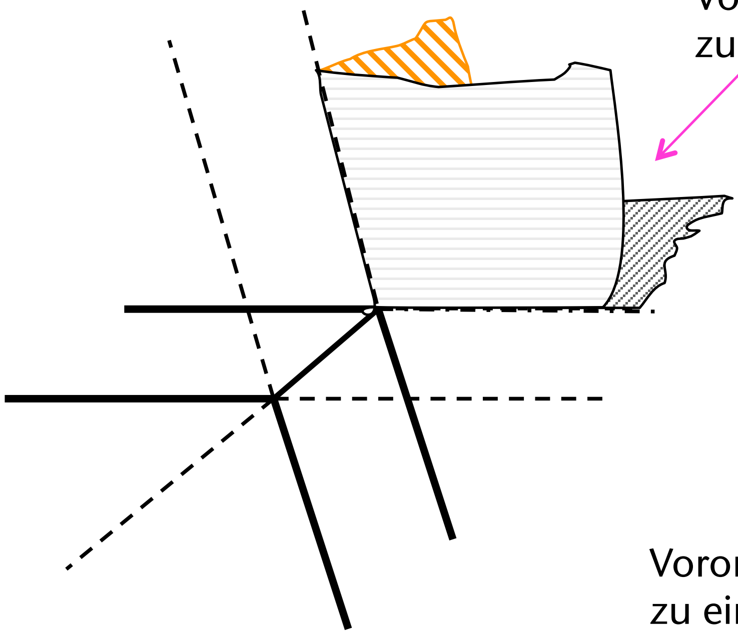
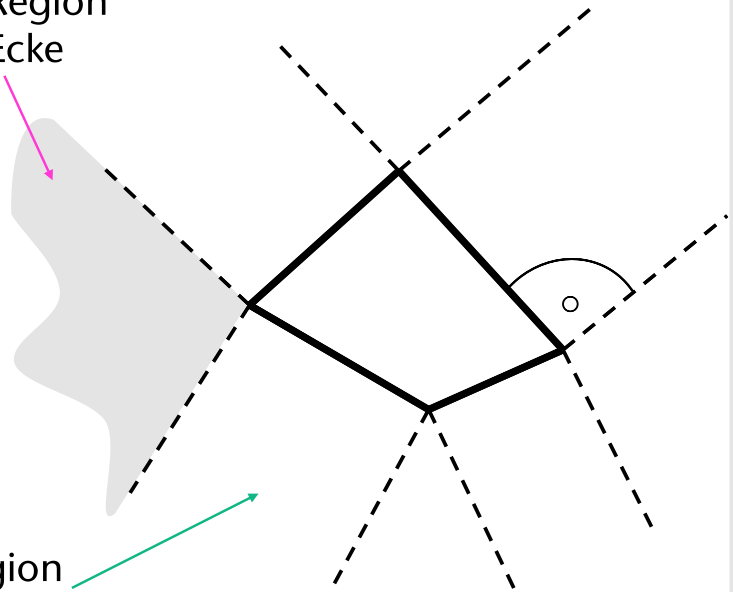
Voronoi-Diagramme zu Polyedern

Voronoi-Regionen in 3D



Voronoi-Regionen in 2D



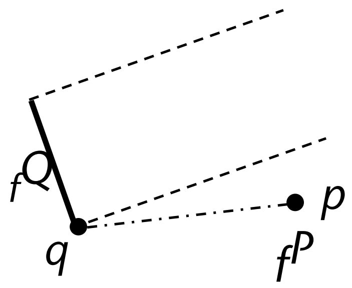
Äußere Voronoi-Regionen sind für konvexe Objekte
sehr einfach zu konstruieren!
(Innere Voronoi-Regionen brauchen wir nicht.)

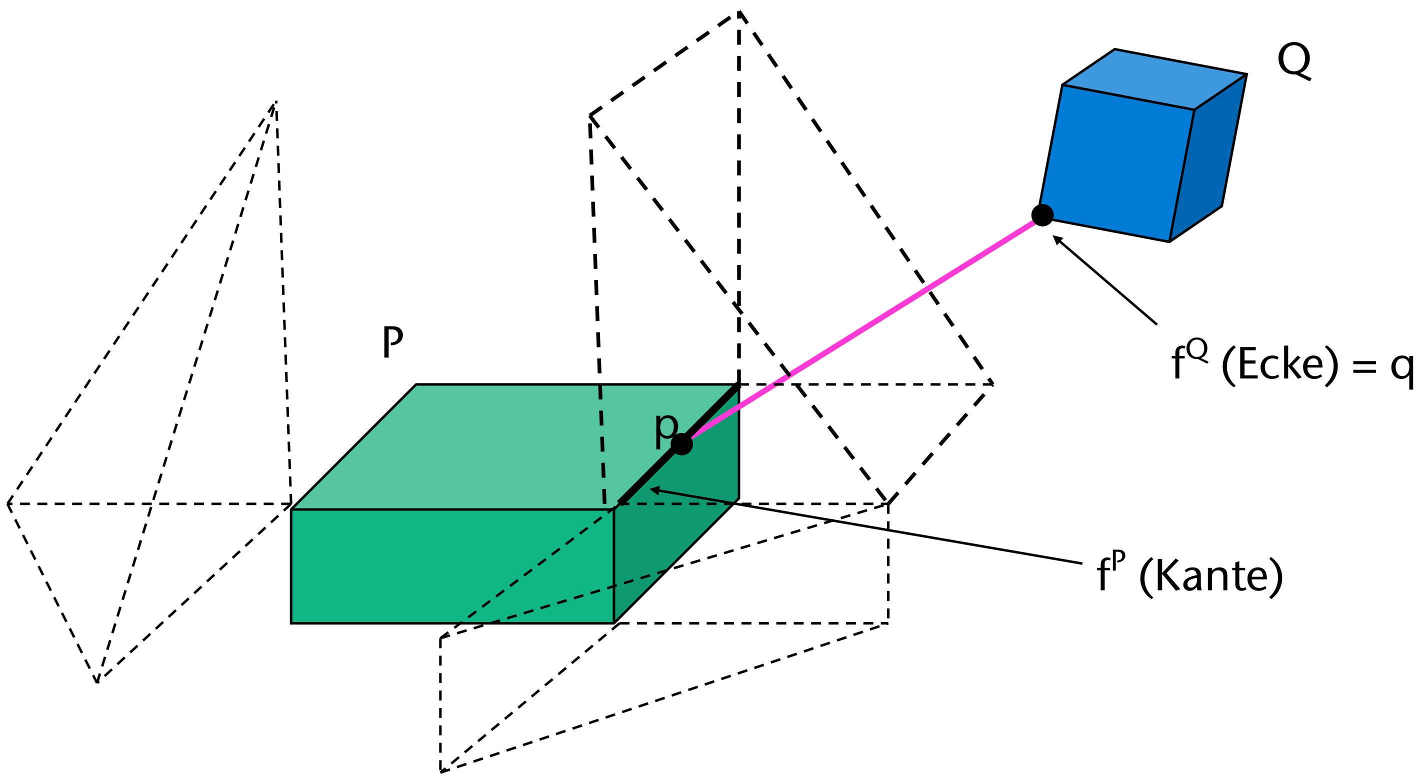
Closest Features

- Definition *Feature* f^P :=
  Ecke, Kante oder Polygon eines Polyeders *P*.

- Definition "*Closest Feature*":
  Seien f^P und f^Q zwei Features auf *P* bzw. *Q*, und seien *p*, *q* Punkte auf f^P bzw. f^Q die den minimalem Abstand von *P* und *Q* realisieren, d.h., $d(P,Q) = d\left(f^P, f^Q\right) = ||p-q||$.
  Dann heißen "*closest features*".

- Lemma:
  Sei *V*(*f*) die Voronoi-Region zu einem Feature *f*;
  f^P, f^Q sind "closest features" :⇔
  p liegt in $V(f^Q)$, *q* liegt in $V(f^P)$.





Q
P
p
f^Q (Ecke) = q
f^P (Kante)

Algorithmus

starte mit zwei beliebigen Features f^P, f^Q auf P bzw. Q

while (f^P, f^Q) sind noch nicht closest features && d(f^P, f^Q) > 0

  if (f^P,f^Q) wurde schon einmal betrachtet

    return "Kollision" (weil Zyklus)

  bestimme p und q, die den Abstand zwischen f^P,f^Q realisieren

  if $p \in V_q$ und $q \in V_p$

    return "keine Kollision", (f^P,f^Q) sind closest features

  if ex. eine Seite von V_q bzgl. der p auf der falschen Seite liegt

    $f^P \leftarrow$ das Feature der "dahinter" liegenden Voronoi-Region

  analog für q, falls $q \notin V_p$

if d(f^P, f^Q) > 0

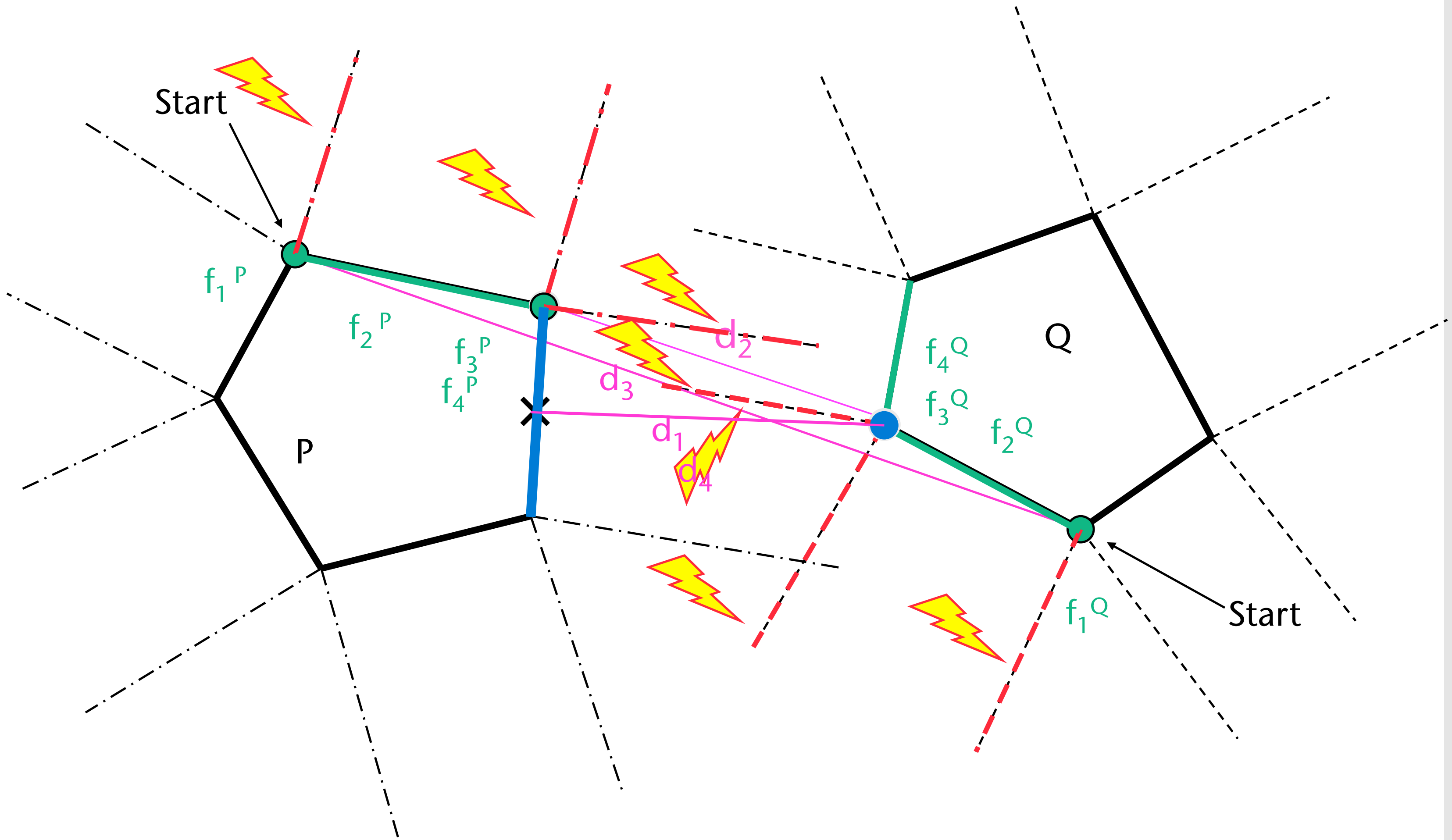
  return "keine Kollision"

else

  return "Kollision"

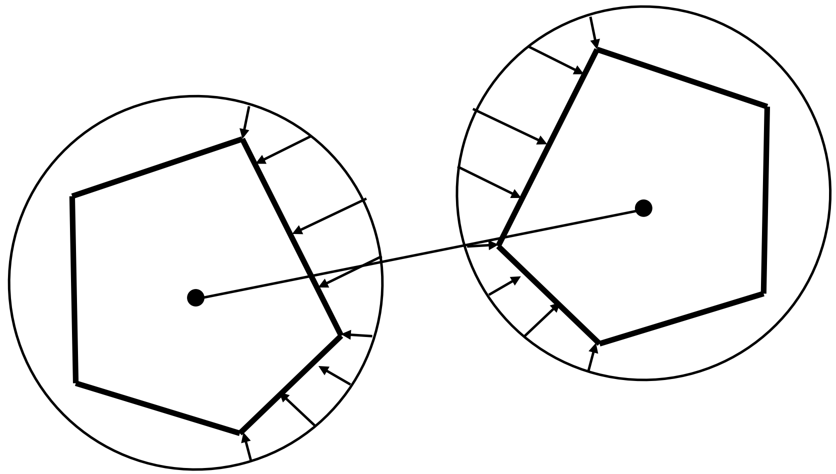
Achtung: bei Kollision befinden sich einige Features im Innern des anderen Objektes, aber im Innern ex. keine Voronoi-Regionen!

Visualisierung des Algorithmus'



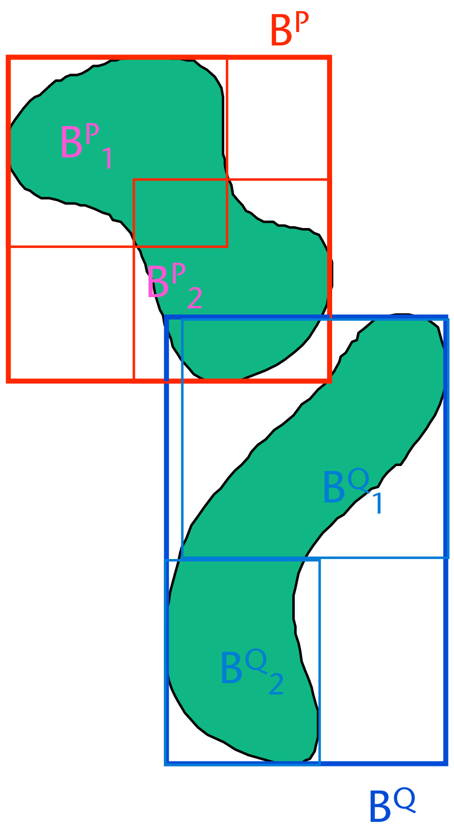
Anmerkungen

- Kleine Denkaufgabe:
  Das *Voronoi-Diagramm* braucht man eigentlich nicht!
  (aber mit *Voronoi-Diagramm* ist der Algo schneller)
- Berechnungsdauer hängt ab vom "Maß" der zeitlichen Kohärenz
- Verbesserung durch *Lookup-Table*:
  trage sphärische Koordinaten der Features
  in Tabelle ein



Hierarchische Kollisionserkennung

- Für *"Polygon soups"*
- Algorithmentechnik: Divide & Conquer



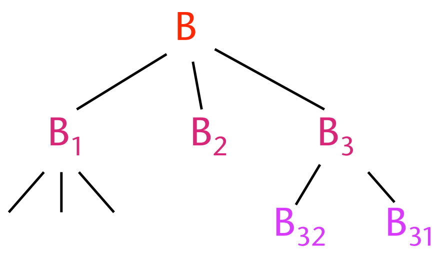
Bounding Volume Hierarchy (BVH)

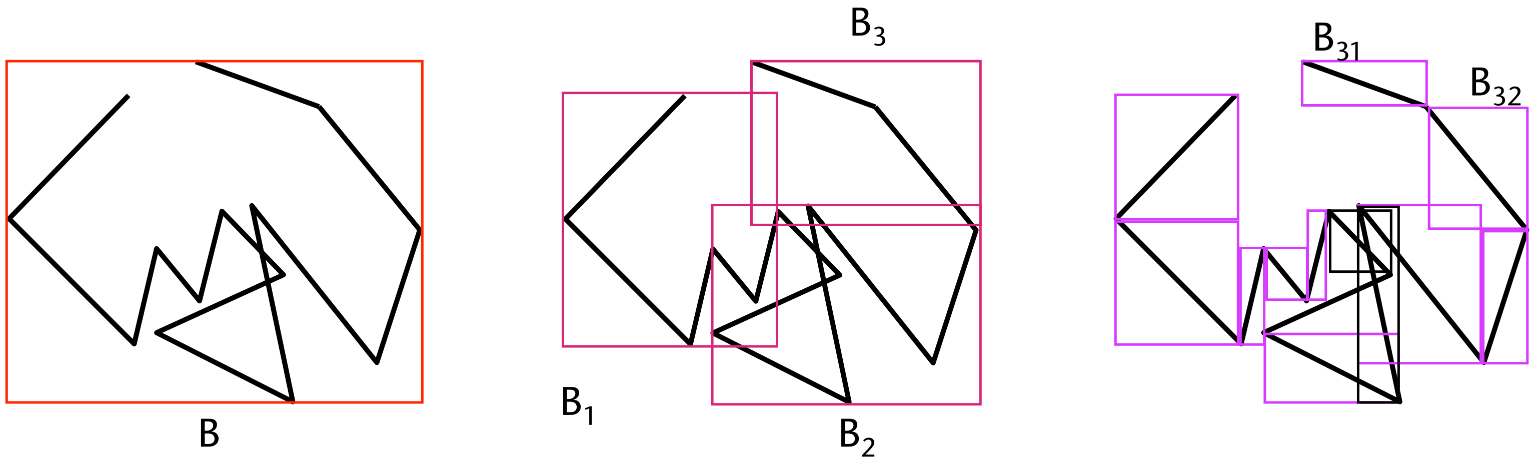
- Schließe alle Polygone aus P in ein Hüllvolumen (*bounding volume*) BV(P) ein
- Teile P auf in $P_1, P_2, P_3, \ldots, P_n$, mit

$$P_1 \cup P_2 \cup P_3 \cup \ldots \cup P_n = P$$

- Rekursiv für die P_i .

→ *bounding volume hierarchy*





Simultane Traversierung

<u>traverse(X, Y)</u>

if X,Y do not overlap **then**

  return

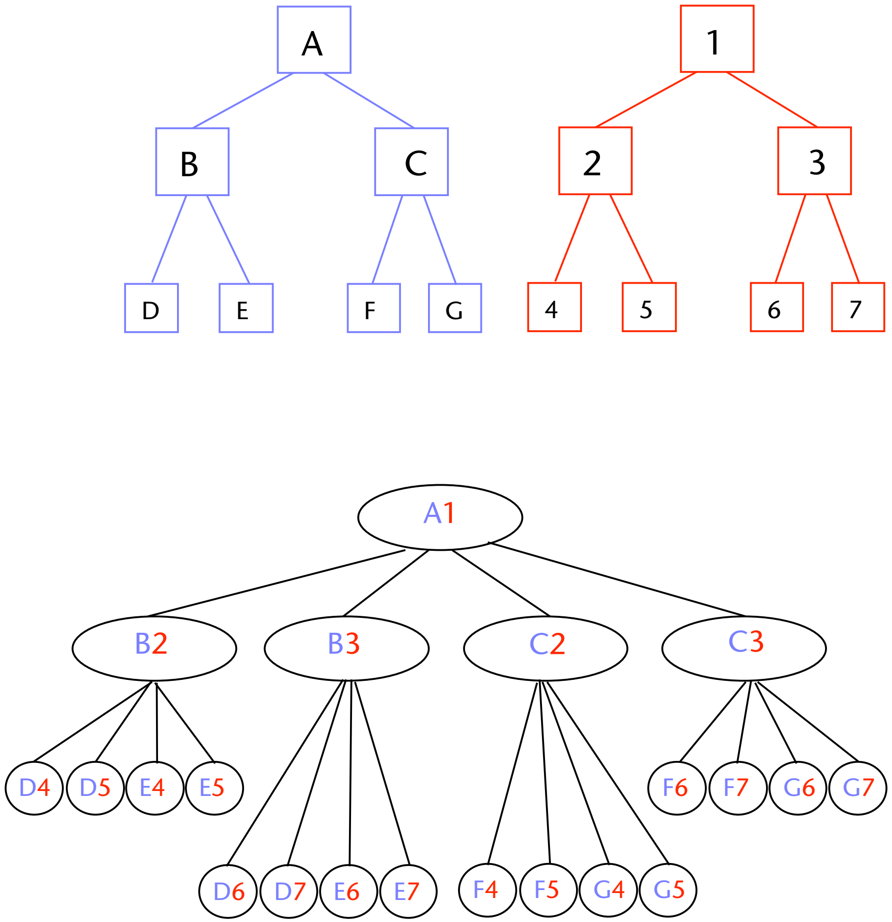
if X,Y are leaves **then**

  check polygons

else

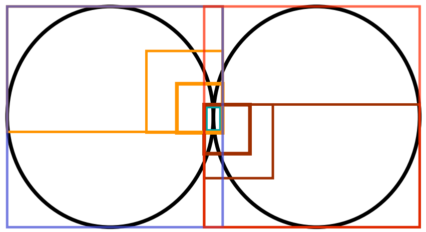
  for all children pairs **do**

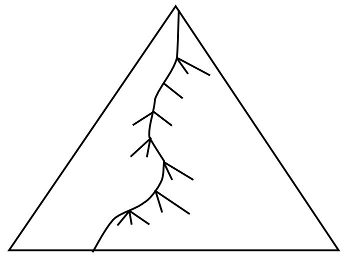
    traverse(X_i, Y_j)



Einfache Laufzeit-Abschätzung

- *Best-case*: $O(\log n)$





Bounding Volume
Test Tree (BVTT)

- Einfache *average-case* Abschätzung:
  - P[k] = Wahrsch.keit daß genau k Kinderpaare überlappen, $k \in [0,\ldots,4]$

$$P[k] = \binom{4}{k}, \quad P[0] = \frac{1}{16}$$

  - Annahme: alle Ereignisse sind gleich wahrscheinlich
  - Erwartete Laufzeit :

$$T(n) = \frac{1}{16}\cdot 0 + \frac{4}{16}\cdot T(\tfrac{n}{2}) + \frac{6}{16}\cdot 2T(\tfrac{n}{2}) + \frac{4}{16}\cdot 3T(\tfrac{n}{2}) + \frac{1}{16}\cdot 4T(\tfrac{n}{2})$$

$$T(n) = 2T(\tfrac{n}{2}) \in O(n)$$

- In der Praxis besser

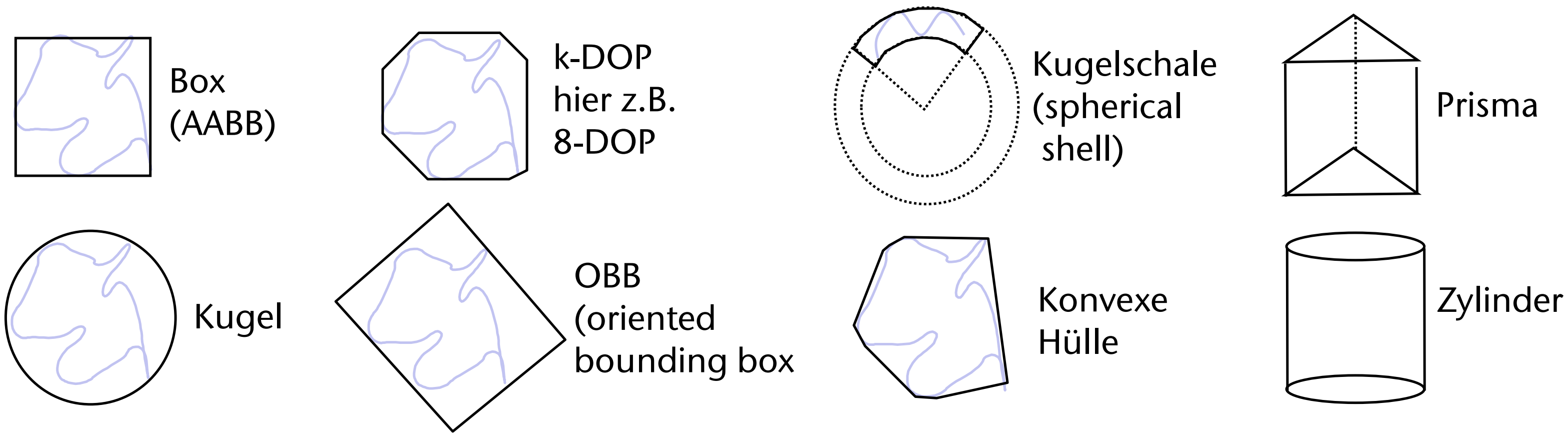
Bounding Volumes

Anforderungen:

- *sehr* schneller Überlappungstest
- auch dann, wenn die *Bounding Volumes* rotiert oder transl. sind!

  → "einfache" *Bounding Volumes*
- eine Überdeckung des ganzen Raumes sollte möglichst wenig mehrfach belegten Raum haben → "*tight BVs*"

Einige mögliche *Bounding Volumes*:



Die Minkowski-Summe

- Hermann Minkowski (22. 6. 1864 – 12. 1. 1909), deutscher Mathematiker und Physiker
- Definition (*Minkowski-Summe*):
  Seien *A* und *B* Teilmengen eines Vektorraums;
  die Minkowski-Summe von *A* und *B* ist

$$A \oplus B = \{\mathbf{a} + \mathbf{b} \,|\, \mathbf{a} \in A,\ \mathbf{b} \in B\}$$

- Entsprechend die *Minkowski-Differenz*:

$$A \ominus B = \{\mathbf{a} - \mathbf{b} \,|\, \mathbf{a} \in A,\ \mathbf{b} \in B\}$$

- Zusammenhang zwischen *Minkowski-Summe* und *-Differenz*:

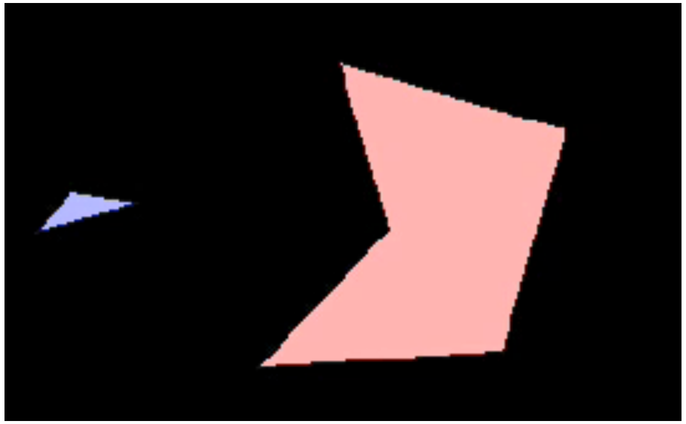
$$A \ominus B = A \oplus (-B)$$

- Anwendungen: Computergraphik, Bildverarbeitung, Lineare Optimierung, Roboter-Pfadplanung, ...

Eigenschaften

- *Minkowski-Summen* sind:
  - Kommutativ: $A \oplus B = B \oplus A$
  - Assoziativ: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
  - Distributiv bzgl. Vereinigung: $A \oplus (B \cup C) = (A \cup B) \oplus (A \cup C)$
  - Invariant (in gewissem Sinne) gegenüber Translation: $T(A) \oplus B = T(A \oplus B)$

- Intuitive "Berechnung" der Minkowski-Summe:



  - Achtung: das gelbe Polygon zeigt die Minkowsi-Summe **modulo**(!) eventueller Translationen!

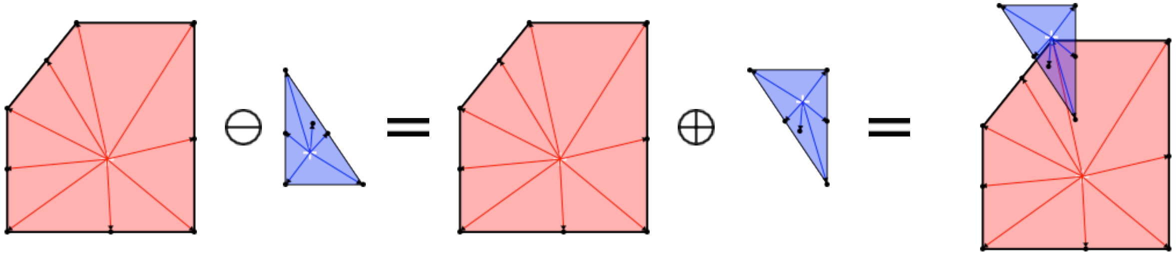
Komplexität

- Seien *A* und *B* Polygone mit *n* bzw. *m* Ecken
  - Sind *A* und *B* konvex, so ist $A \oplus B$ konvex und hat Komplexität $O(mn)$
  - Ist nur *B* konvex, so hat $A \oplus B$ die Komplexität $O(mn)$
  - Ist keines der beiden konvex, so hat $A \oplus B$ die Komplexität $O(m^2 n^2)$
- Algorithmische Komplexität des Problems mit Divide & Conquer:
  - Sind *A* und *B* konvex, so kann $A \oplus B$ in Zeit $O(m+n)$ berechnet werden
  - Ist nur *B* konvex, so kann $A \oplus B$ randomisiert in Zeit $O(mn\, log^2(mn))$ berechnet werden
  - Ist keines der beiden konvex, so hat $A \oplus B$ die Komplexität $O(mn^2\, log(mn))$

Schnitttest für zwei konvexe BVs

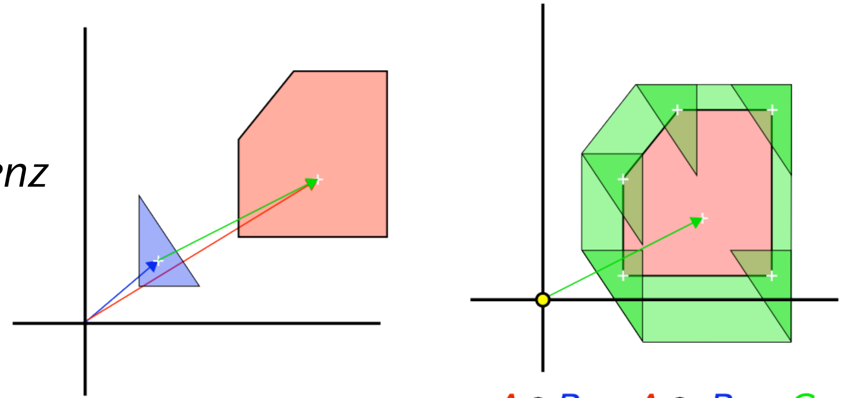
- Erkennen von Kollisionen durch die *Minkowski-Differenz:*

$$A \ominus B = A \oplus -B = C$$



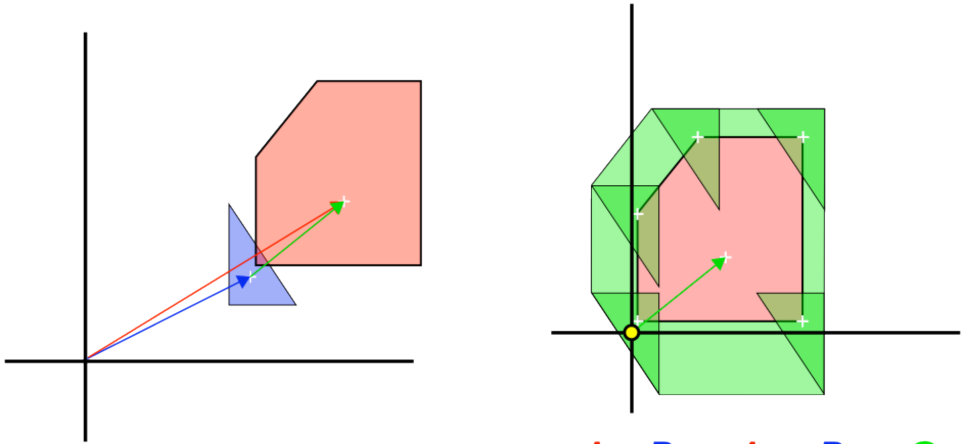
- Für zwei Objekte ergibt sich somit:
  - Verschiebe beide Objekte mit derselben Translation, so daß der Urpsrung in *B* liegt
  - Berechne die *Minkowski-Differenz*
  - A und B schneiden sich gdw.

$$0 \in A \ominus B$$



$$A \ominus B = A \oplus -B = C$$

- Beispiel, in dem sich A und B schneiden:



$A \ominus B = A \oplus -B = C$

  - Der Koordinatenursprung befindet sich in der *Minkowski-Differenz C*

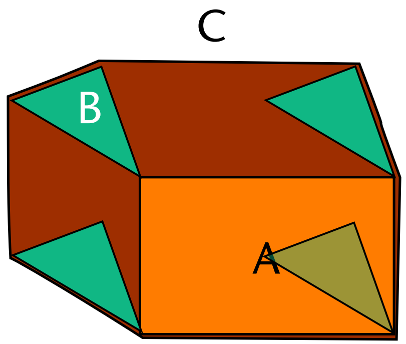
Oriented Bounding Boxes (OBB)

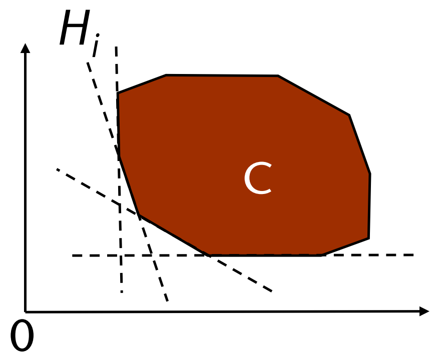
- Lemma "*Separating Axis Test*" (SAT):
  Seien A, B zwei konvexe Polytope (Polyeder).
  Wenn es eine separierende Ebene gibt,
  dann auch eine, die parallel zu einer Seite von A oder B ist,
  oder parallel zu mindestens einer Kante von A und einer von B.
  [Gottschalk, Lin, Manocha; 1996]

- Abwandlung des "*separating plane*" Lemmas
  ("*separating axis*" Lemma):
  Zwei konvexe Polyeder überlappen sich nicht $\Leftrightarrow$
  es gibt eine Gerade, so daß die Projektion der beiden
  Objekte auf dieser Geraden sich nicht überlappen.
  Diese Achse heißt "separierende Achse".

Beweis des SAT-Lemmas

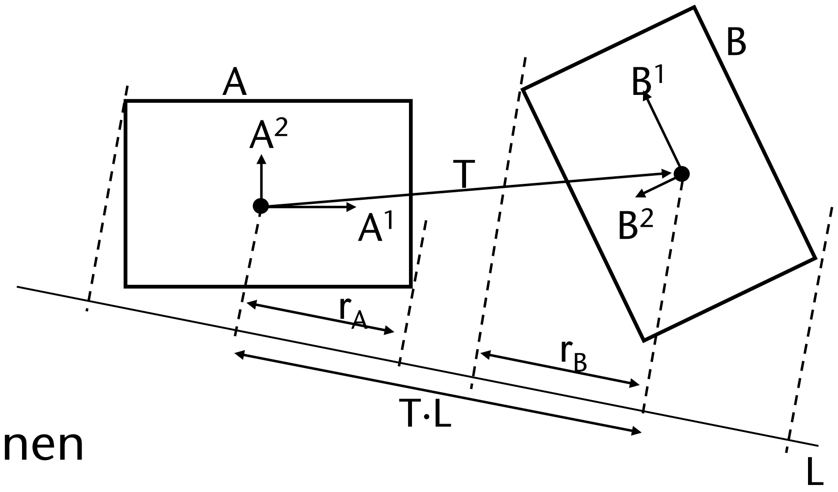
1. Annahme: A und B sind disjunkt
2. Betrachte Minkowski-Summe
3. Alle Faces von C sind entweder parallel zu einem Face von A, oder einem Face von B, oder parallel zu einer Kante von A *und* einer Kante von B
4. C ist konvex
5. $C = \bigcap_{i=1}^{m} H_i$
6. $A \cap B = \varnothing \Leftrightarrow (0,0,0) \notin C$
7. $\exists i : 0 \notin H_i$ (0 liegt außerhalb eines H_i)
8. Es gibt eine separierende Ebene für A und B, die parallel zu diesem H_i ist.





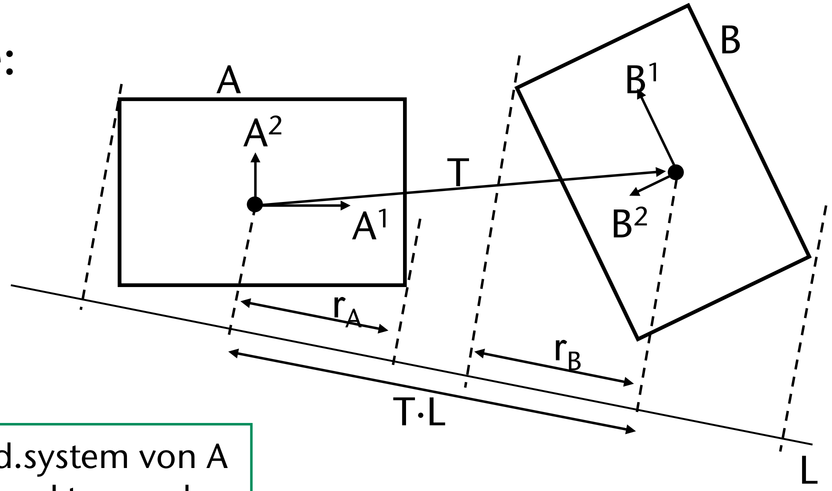
Der SAT für OBBs

- OBdA: rechne im Koord.system von Box A
- Box A definiert durch: C, a^1A^1, a^2A^2, a^3A^3
- Position von B relativ zu A ist definiert durch R & T



- Im Koord.system von A: B^i sind Spalten von R
- Gemäß Lemma müssen wir nur einige spezielle Ebenen betrachten, um die Separierung festzustellen
- A,B überlappen, wenn $|T \cdot L| < r_A + r_B$ für jede dieser Ebenen
  - L = Normale der Ebene
- Anzahl solcher "spezieller" Achsen bei Boxes = 15

- Bsp.: $L = A^1 \times B^2$
- Zu berechnen: $r_A = \sum_i a_i |A^i \cdot L|$ (und analog r_B)
- Bsp. 2-ter Term der Summe:



$$
\begin{aligned}
& a_2 A^2 \cdot (A^1 x B^2) \\
&= a_2 B^2 \cdot (A^2 x A^1) \\
&= a_2 B^2 \cdot A^3 \\
&= a_2 R_{32}
\end{aligned}
$$

Wir rechnen in Koord.system von A
→ A^3 ist 3-ter Einheitsvektor, und
B^2 ist 2-te Spalte von R

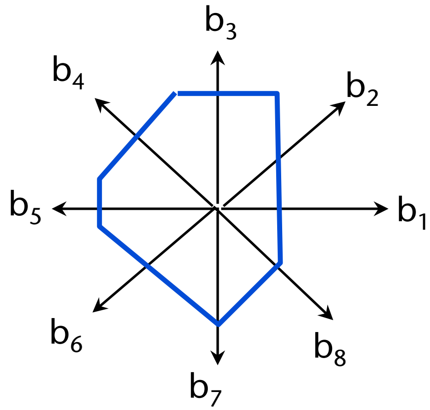
- Für jede der 15 Achsen hat man einen Test der Form

$$|T \cdot L| < a_2|R_{32}| + a_3|R_{22}| + b_1|R_{13}| + b_3|R_{11}|$$

Discretely Oriented Polytopes (k-DOPs)

- Definition:
  Wähle k Vektoren $\mathbf{b}_i \in \mathbb{R}^3$ fest, k gerade,
  mit b_i antiparallel zu $b_{i+k/2}$.
  k-DOPs sind als Volumen beschrieben durch

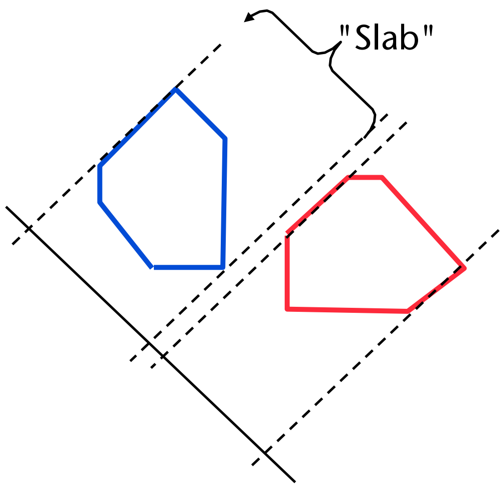
$$D = \bigcap_{i=1..k} H_i \quad , \quad H_i : \mathbf{b}_i \cdot x - d_i \leq 0$$



- Beschreibung eines k-DOP: $D = (d_1 \dots d_k) \in \mathbb{R}^k$
- Überlappungstest:

$$D^1 \cap D^2 = \emptyset \Leftrightarrow$$

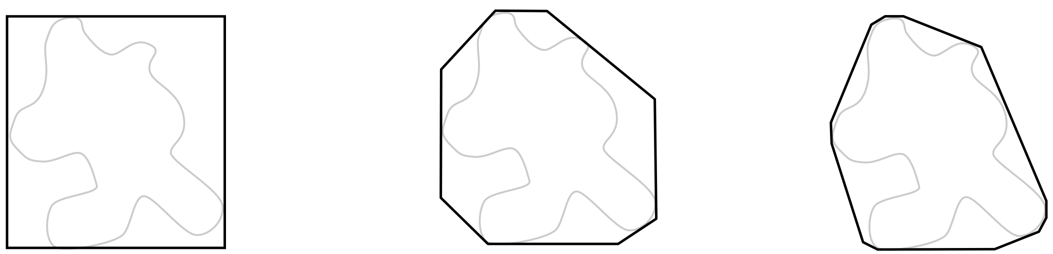
$$\forall i = 1, .., \frac{k}{2} : \left[d_i^1, d_{i+\frac{k}{2}}^1 \right] \cap \left[d_i^2, d_{i+\frac{k}{2}}^2 \right] = \emptyset$$



→ $k/2$ Intervall-Tests

Eigenschaften

- AABBs sind spezielle DOPs
- Überlappungstest $\in O(k)$, k = Anzahl Orientierungen
- Beliebig genaue Approximation der konvexen Hülle



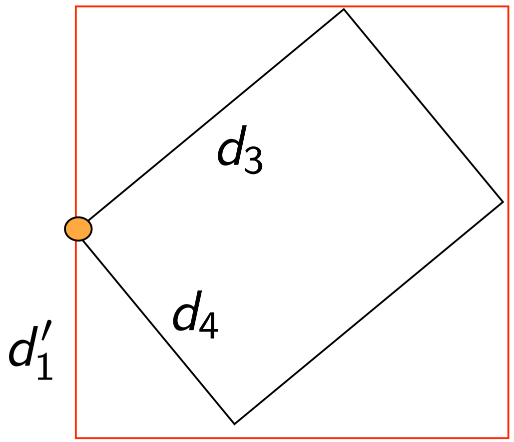
k=4 k=8 k=12

Overlap test of DOPs

- Algorithmus für "schiefe" DOPs:
  - Objektbewegung: Rotation R & Translation T
  - Neuer DOP nach affiner Transformation des Objektes:

$$d'_i = \boxed{\mathbf{B}_i \begin{pmatrix} \mathbf{b}_{j'_1} \\ \mathbf{b}_{j'_1} \\ \mathbf{b}_{j'_1} \end{pmatrix}^{-1}} \begin{pmatrix} d_{j'_1} \\ d_{j'_1} \\ d_{j'_1} \end{pmatrix} + \boxed{\mathbf{B}_i T},$$

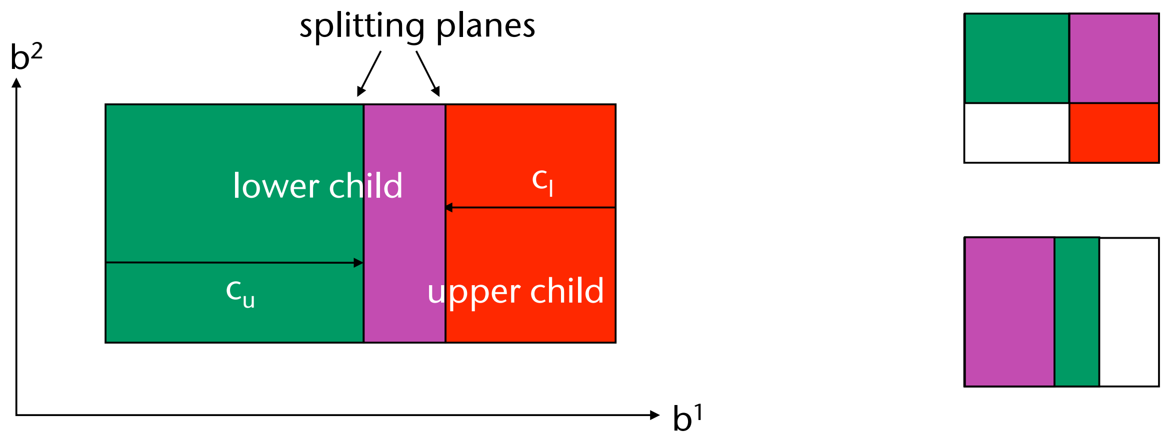
$$\mathbf{b}_j = \mathbf{B}_i R^{-1}$$



  - Korrespondenz j^i_l identisch für alle DOPs einer Hierarchie
  - Aufwand: O(k), früher O(k²)

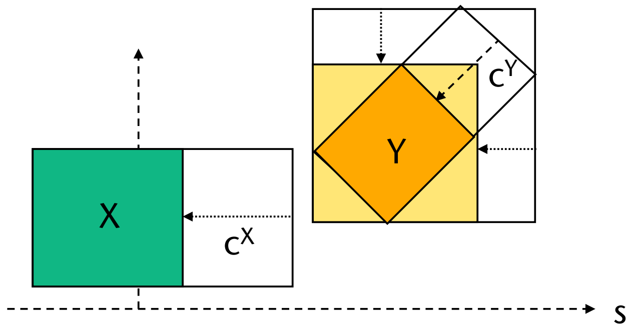
Restricted Boxtrees (Variante von k-d Trees)

- Kombination von k-d tree und AABB:



- Speicher: 1 Float, 1 Achsen-ID, 1 Pointer (= 9 Bytes)
- Weitere Namen:
  - BIH (Bounding Interval Hierarchy)
  - SKD-Tree (spatial kd-Tree)

- Overlap Tests durch Re-Alignment:
  12 FLOPs (mit kleinem Trick: 8.5)



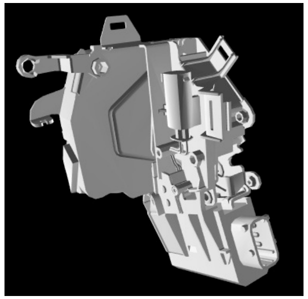
- SAT: 82 FLOPs
- SAT lite: 24 FLOPs
- Sphere test: 29 FLOPs

Mehr dazu in

http://zach.in.tu-clausthal.de/papers/vrst02.html

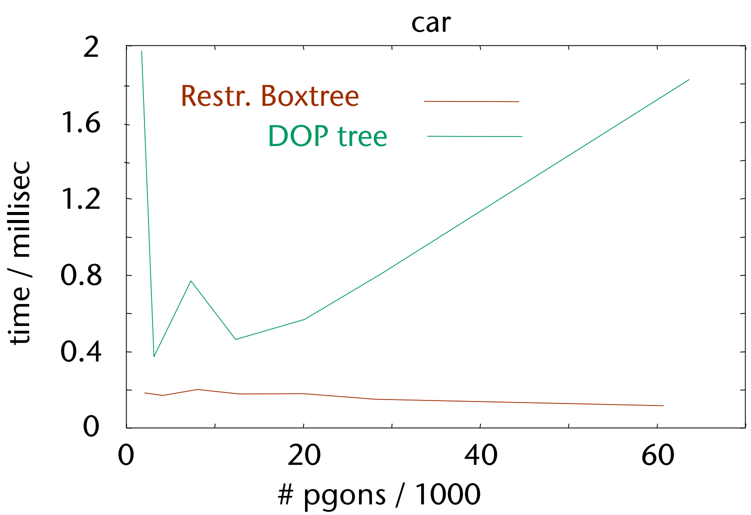
Performance

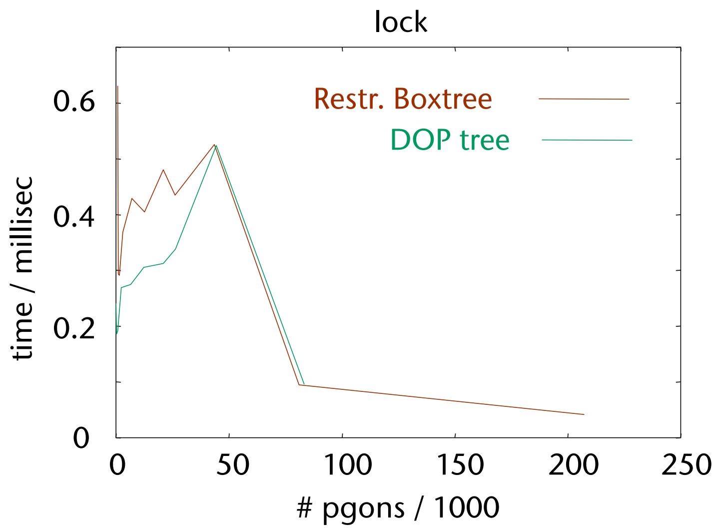
- Bsp. *Boxtree*:



Door lock (BMW)

Car (courtesy VW)





Konstruktion von BV-Hierarchien

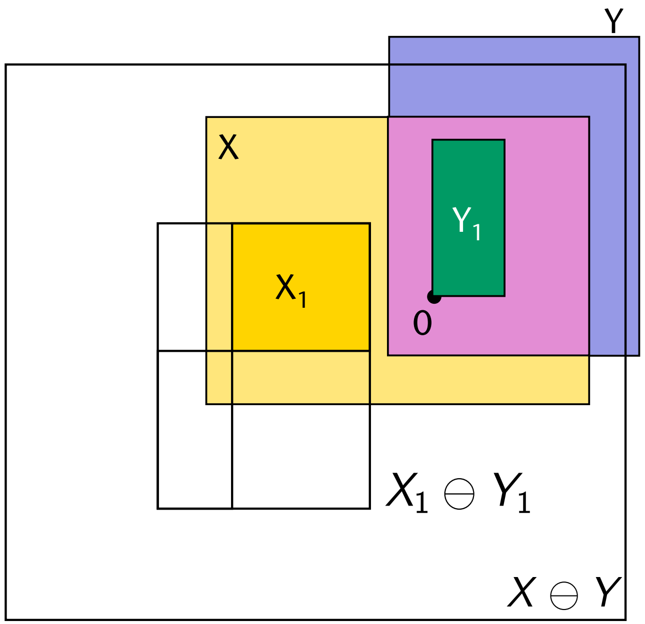
- Hierarchie schlecht → Kollisionserkennung dauert lange.
- Algorithmus: *top-down*
  1. Berechne BV um gegebene Polygon-Menge
  2. Splitte Polygon-Menge
- Split-Kriterium?
- Erwartete Traversierungskosten:

$$C(X,Y) = 4 + \sum_{i,j=1,2} P(X_i,Y_j)\, C(X_i,Y_j)$$

$$\approx 4\,(1 + P(X_1,Y_1) + \cdots + P(X_2,Y_2))$$

- Eine Abschätzung von $P(X_i,Y_j)$
- Hilfsmittel dabei:
  die Minkowski-Summe
- Erinnerung:

$$X_i \cap Y_j = \varnothing \;\Leftrightarrow\; 0 \notin X_i \ominus Y_j$$



- Die Wahrscheinl'keit ist somit

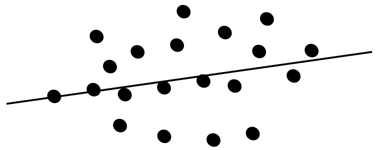
$$P(X_i, Y_j) = \frac{|\text{günstige Fälle}|}{|\text{mögliche Fälle}|} = \frac{\text{vol}(X_i \ominus Y_j)}{\text{vol}(X \ominus Y)} = \frac{\text{vol}(X_i \oplus Y_j)}{\text{vol}(X \oplus Y)}$$

$$\approx \frac{\text{vol}(X_i) + \text{vol}(Y_j)}{\text{vol}(X) + \text{vol}(Y)}$$

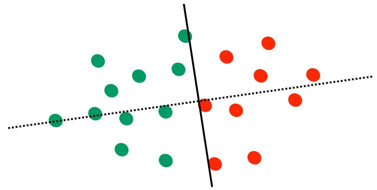
- Fazit: Minimiere Summe der Volumen der Kind-BVs.

Algorithmus

1. Orientierung für "gute" *Splitting-Ebene* aus PCA



2. Suche Minimum gemäß Volumen-Kriterium



- Komplexität bei *Plane-Sweep*:

$$T(n) = cn + T(\alpha n) + T((1-\alpha)n) \in O(n \log n)$$