Kollosionserkennung bei 3D-Objekten

Stefan M. Grünvogel
Laboratory for Mixed Realities
Institut an der Kunsthochschule für Medien in Köln

1. Einführung

Kollisionserkennung (collision detection, kurz: CD)
handelt kurz gesprochen davon, herauszufinden ob sich zwei Objekte berühren oder schneiden. Falls sich die Objekte im Raum bewegen ist oft auch noch wichtig den Zeitpunkt des Zusammentreffens zu bestimmen.

1. Ein Beispiel aus dem Leben...

Fragen

- Wie erfahre ich, ob die Spielfigur mit einem Gegenstand kollidiert?
- Wenn sich eine Kollision ereignet: Wann geschieht sie?
- Wie kann ich verhindern, dass ich alle Polygone miteinander vergleichen muss?
- Was passiert, wenn ich eine Kollision erkannt habe?

Anwendungen

- Robotik
  - Kollisionsvermeidung
  - Bewegungsplanung
- Industrieanwendungen
  - Virtual Prototyping
  - Montagetests (assembly tests)
- Haptik
- Computerspiele
- Computeranimation

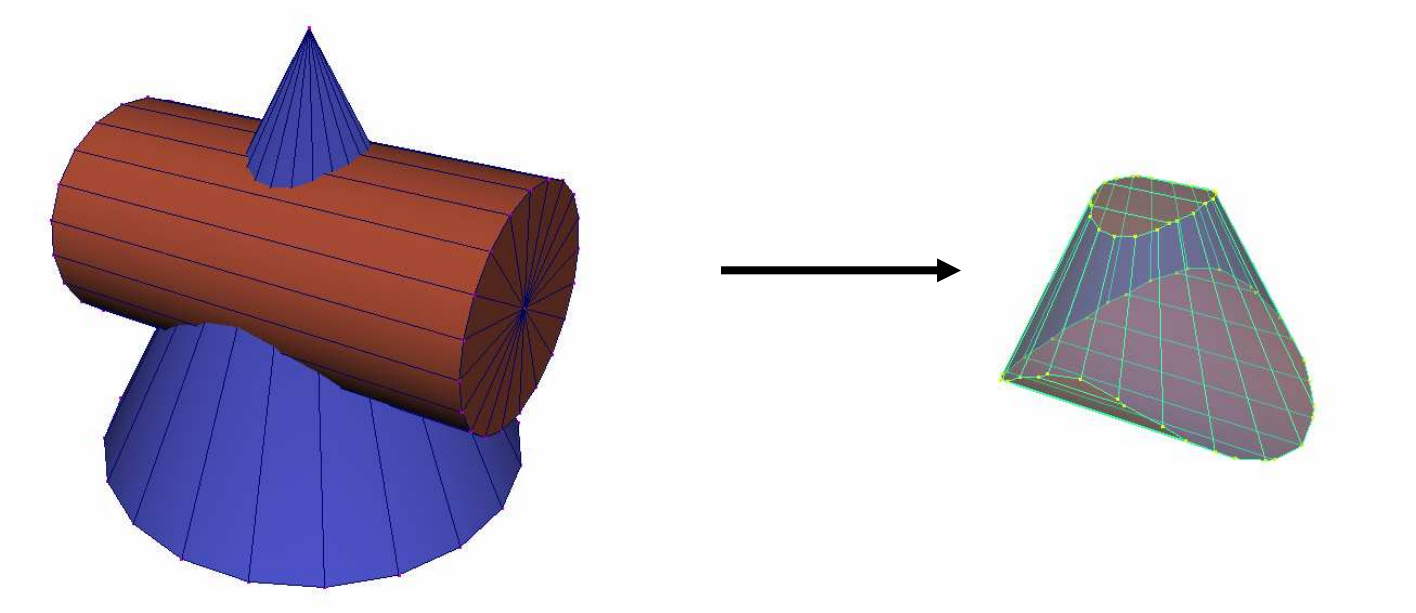
1.1 Begriffe und Aufgabenstellungen

Der Begriff Kollisionserkennung wird oft in mehreren Zusammenhängen verwendet:

- Überlappungstests (intersection detection): Finde heraus, **ob** sich Objekte schneiden – ein statisches Problem.
- Kollisionserkennung (collision detection): Finde heraus **ob und wann** sich bewegende Objekte schneiden – ein dynamisches Problem.

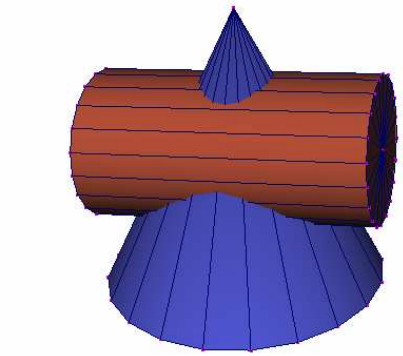
Grundaufgaben - Statisch

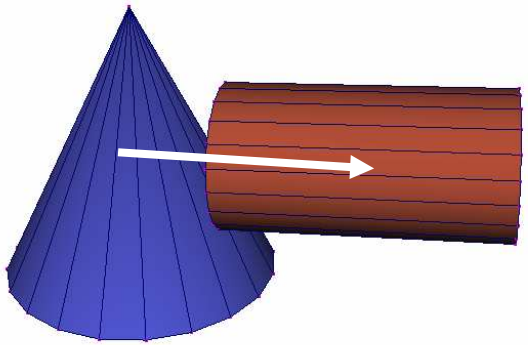
1. Schneiden sich zwei Objekte?
2. Falls sich zwei Objekte berühren /überlappen: Was sind die Kontaktpunkte?



Grundaufgaben - Statisch

3. Falls sich zwei Objekte durchdringen: Was ist die minimale Translation, so dass die Objekte wieder separiert sind.

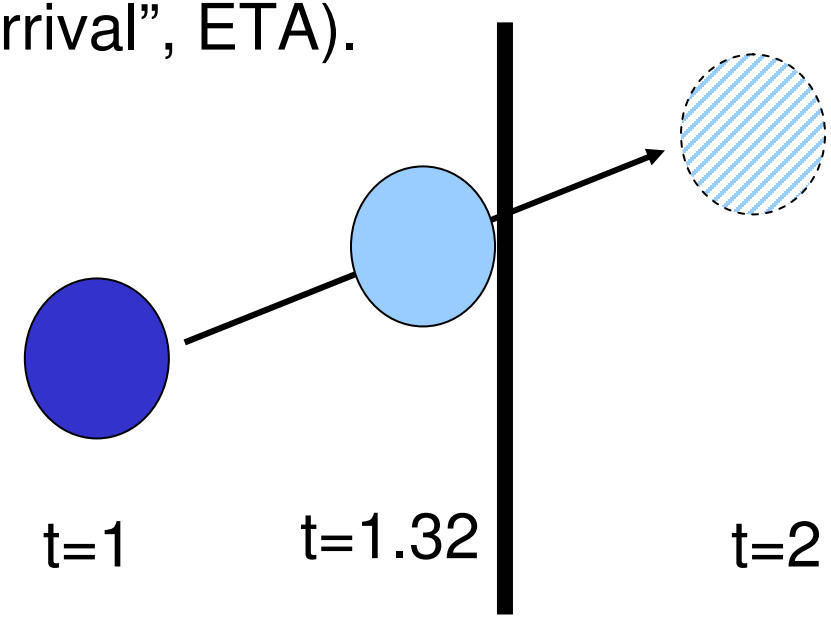




4. Falls sich zwei Objekte nicht berühren: Was ist der “minimale” Abstand zwischen den beiden Objekten?

Grundaufgaben -Dynanisch

1. Finde heraus, ob sich zwei bewegende Objekte berühren und wenn ja: wann (“estimated time of arrival”, ETA).



2. Collision Response: Was passiert mit den zwei Objekten in der Simulationsumgebung, wenn sie sich berühren?

Grundaufgaben

Fazit:

Kollisionserkennung findet im Allgemeinen in Raum und Zeit statt!

Simulationsumgebungen

Paarweise Tests vs. N-fache Tests

Sind nur Tests zwischen zwei Objekten notwendig oder sind N unabhängig sich voneinander bewegende Objekte zu testen?

Simulationsumgebung

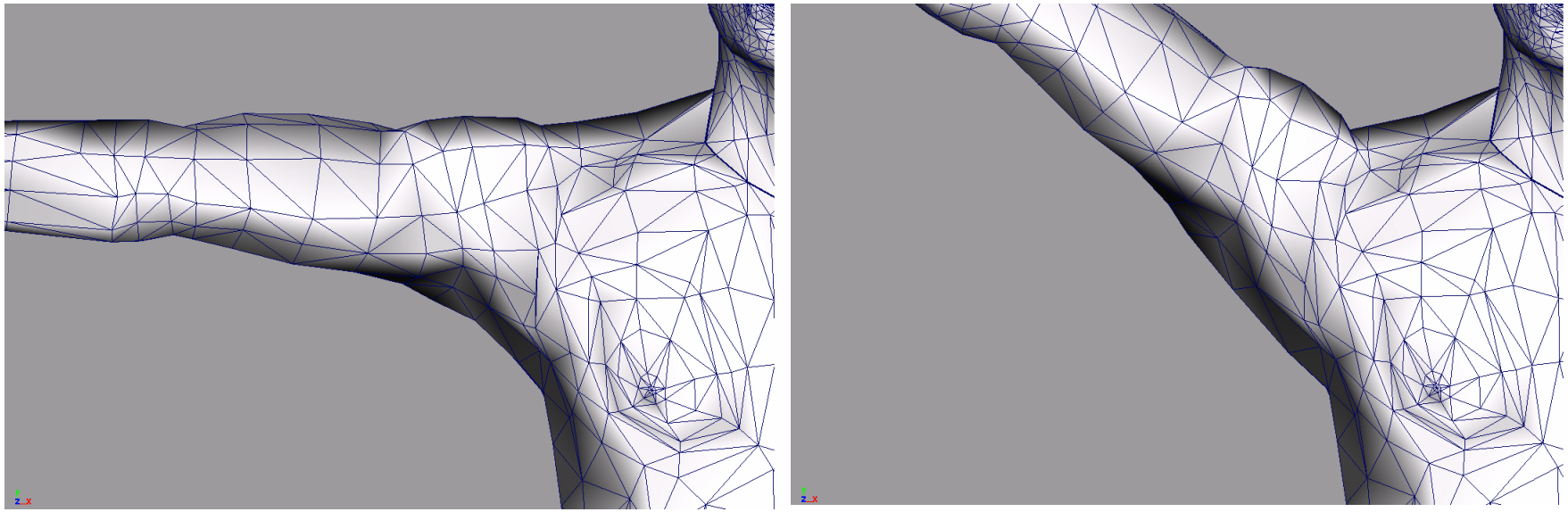
Statische vs. Dynamische Umgebung

- Statische Umgebung: Objekte bewegen sich nicht.
- Dynamische Umgebung: Objekte bewegen sich im Raum.
- Geschwindigkeit / Beschleunigung bekannt oder nur Objektposition zu aktuellen und vergangenen Zeitschritten?
- Gibt es nur einige wenige sich bewegende Körper gegenüber einer Großzahl unbeweglicher Objekte?

Simulationsumgebung

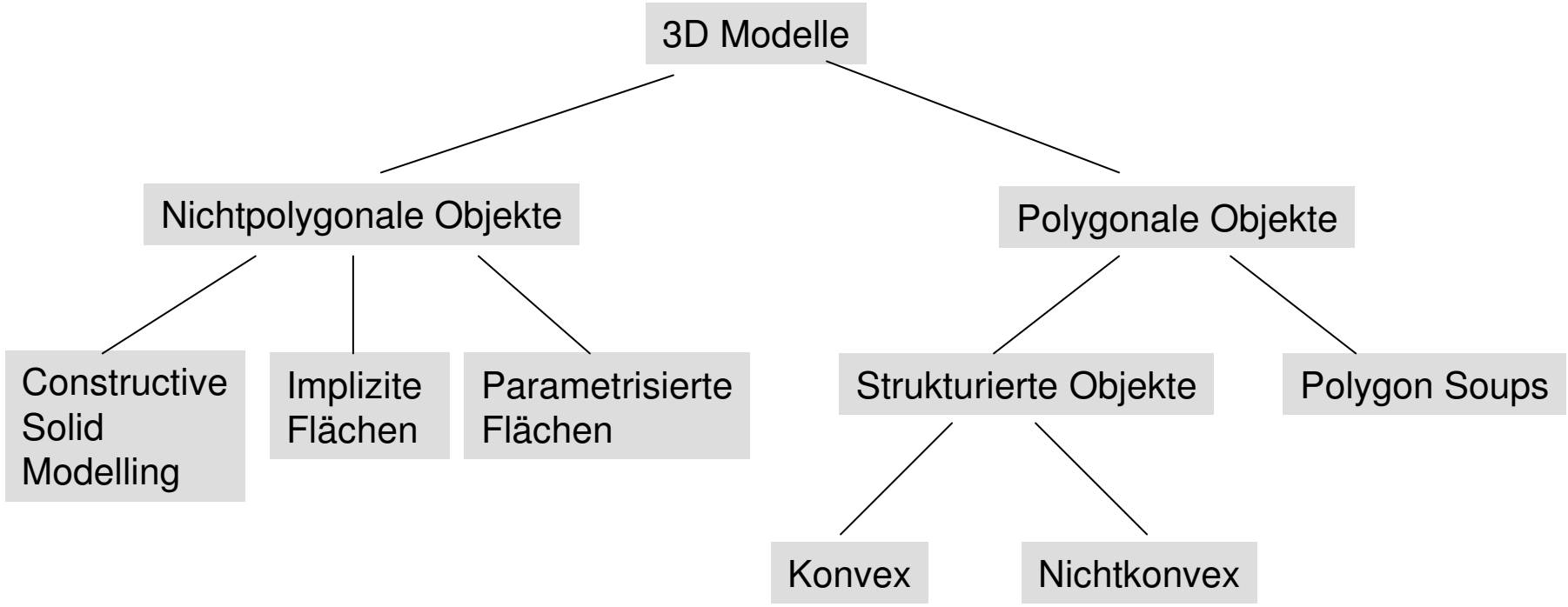
Starre vs. Deformierbare Objekte

Sind nur starre Objekte vorhanden oder können die Objekte sich auch deformieren?



Objektrepräsentation

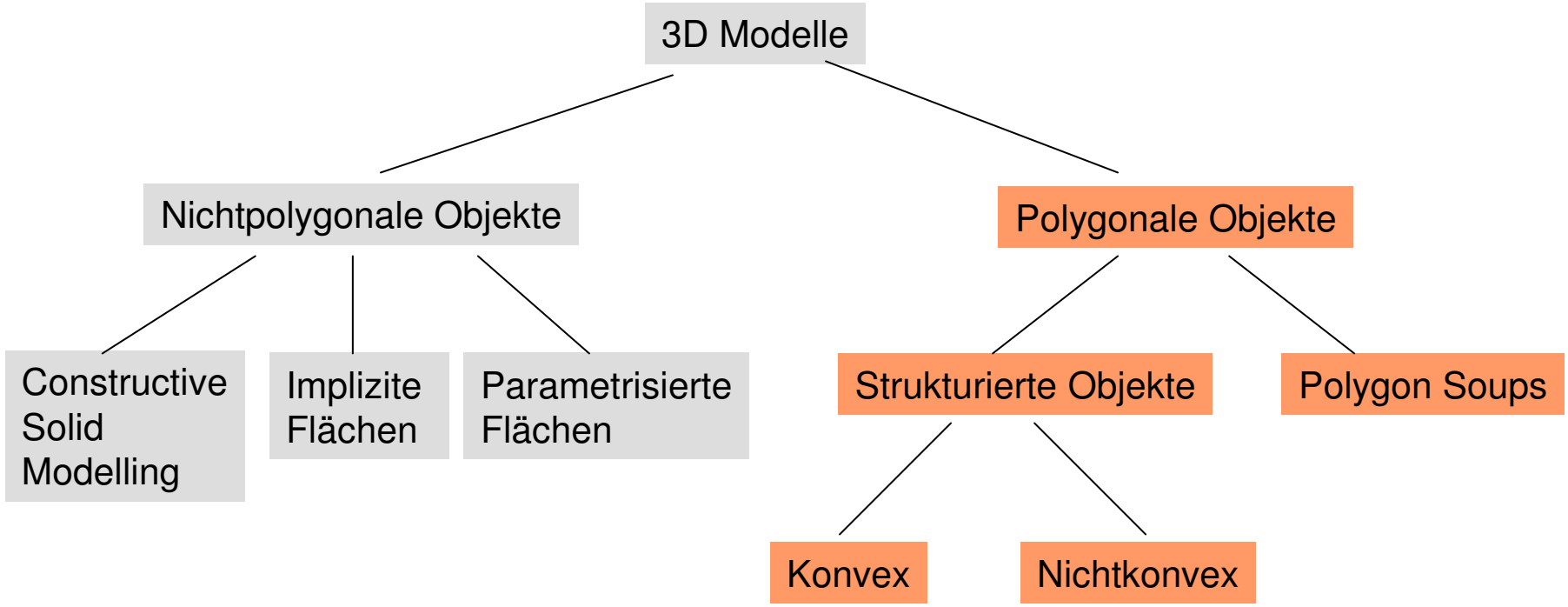
Entscheidend für die Auswahl eines Algorithmus ist auch die Repräsentation der 3D Modelle, für die Kollisionserkennung durchgeführt werden soll.



Eine Taxonomie für Repräsentationen von 3D Modellen (Lin und Gottschalk)

Objektrepräsentation

Entscheidend für die Auswahl eine Algorithmus ist auch die Repräsentation der 3D Modelle, für die Kollisionserkennung durchgeführt werden soll.



Im weiteren werden Algorithmen vor allem für Polygonmodelle vorgestellt.

Klassifikation der Verfahren

- Zweiphasige Algorithmen
  - Bounding Volumes (Sphäre, AABB, OBB, k-DOP)
  - Bounding Volume Hierarchien
  - Räumliche Zerlegungen von Objekten oder Szenen
- Einphasige Algorithmen
  - BSP-Bäume
  - Hubbard‘s Methode

1.2 Literatur

(1) Alan Watt, Fabio Policarpo; *3D Games: Real-time rendering and software technology*, ACM Press, Addison Wesley, 2001

(2) Thomas Möller, Eric Haines, *Real-time rendering*, A K Peters, 1999

(3) Lin, M.C; Gottschalk S.;*Collision detection between geometric models: a survey.* IMA Conf. On Maths and Statistics

2. Zweiphasige Verfahren

Gegeben:

Eine Anzahl von N 3D-Objekten.

Aufgabe:

Finde heraus, welche Objekte sich mit welchen schneiden.

Naiver Ansatz

Teste jedes Objekt mit jedem, um herauszufinden, welche Objekte sich berühren / schneiden.

Aufwand: $\frac{N^2 - N}{2}$ paarweise Tests

Problem:
Direkte Schnitttests zwischen Polyedern mit vielen Polygonen sind im Allg. sehr teuer.

Grundidee

Teile Kollisionserkennung in zwei Phasen auf:

1. „Broad Phase“: Sortiere zunächst die Objekte heraus, die sich auf keine Fall treffen können (Culling). Damit wird die Anzahl der Objekt-Kollisionstests verringert.
2. „Nahphase”: Teste nun die verbleibenden potentielle Kollisions-Kandidaten auf Überlappung.

2.1 Bounding Volumes

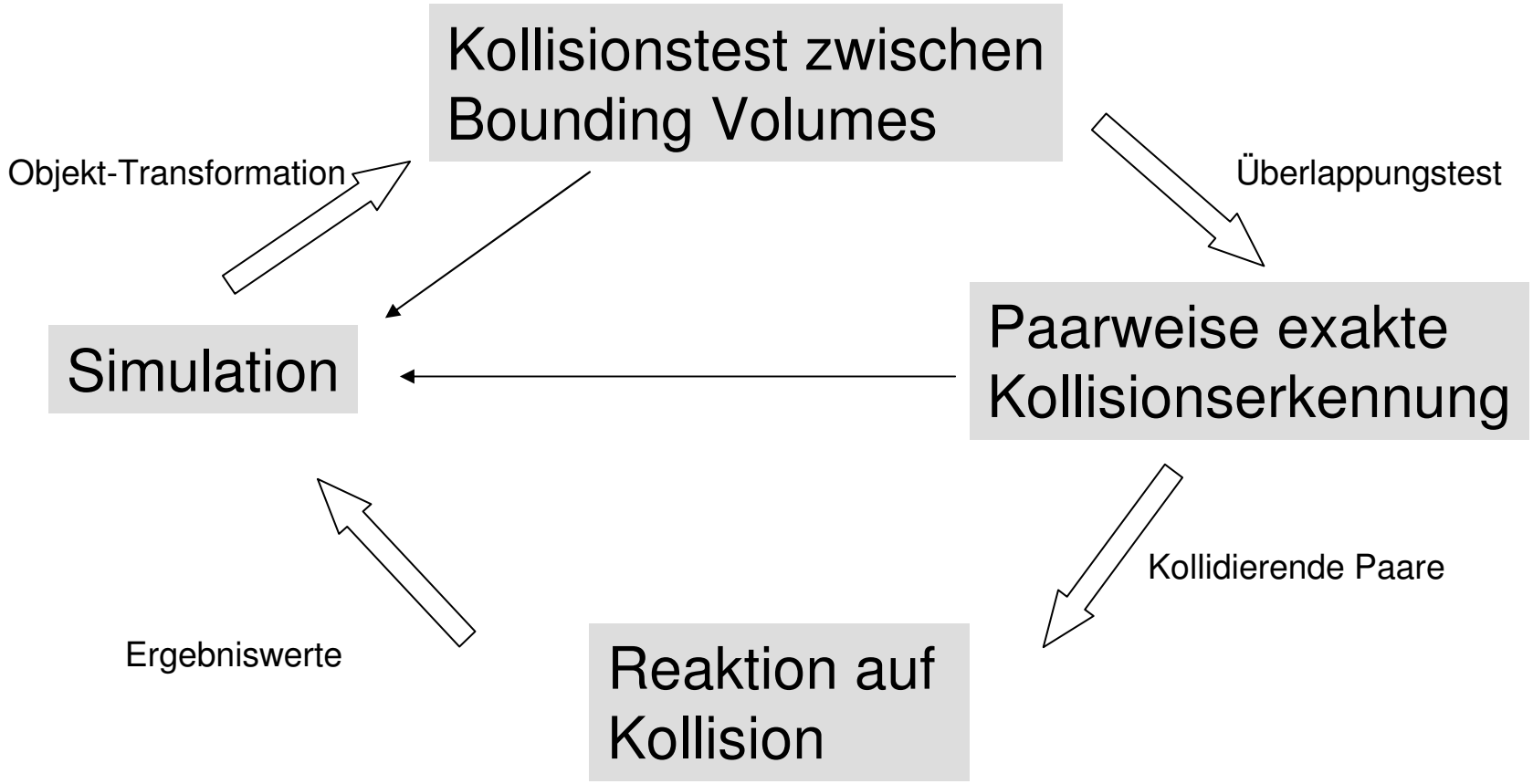
Bounding Volumes (BV, Hüllkörper) werden häufig zum Culling von Objekten verwandt.

Die Idee ist:

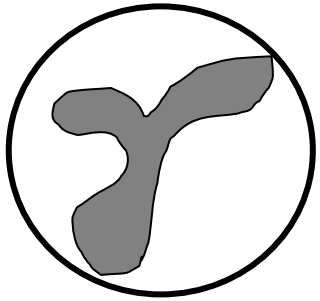
Schnitttest zwischen Objekt-Objekt sind teuer (Objekte bestehen aus vielen Polygonen).

Lege statt dessen eine einfache Geometrie um die Objekte, für die der Schnitttest billig ist.

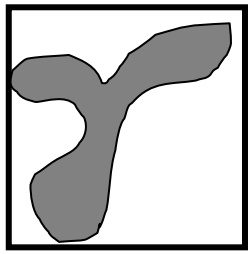
Allgemeine Architektur



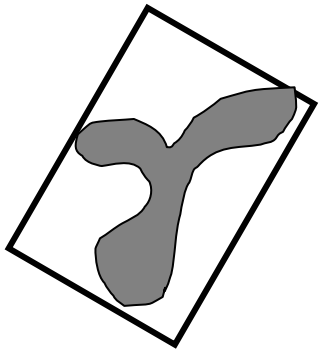
Beispiele



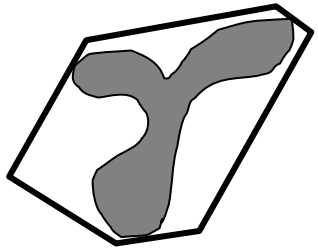
Sphäre



Axis Aligned Bounding Box (AABB)

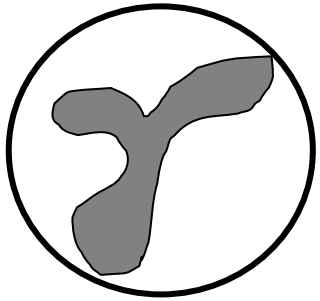


Oriented Bounding Box (OBB)

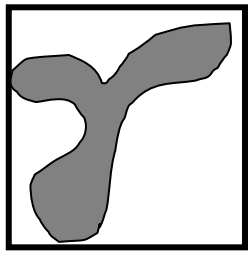


Discrete Oriented Polytopes (k-Dop)

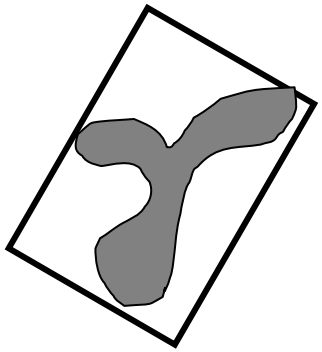
Beispiele



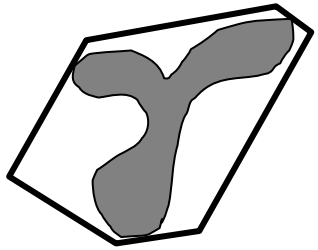
Sphäre



Axis Aligned Bounding Box (AABB)



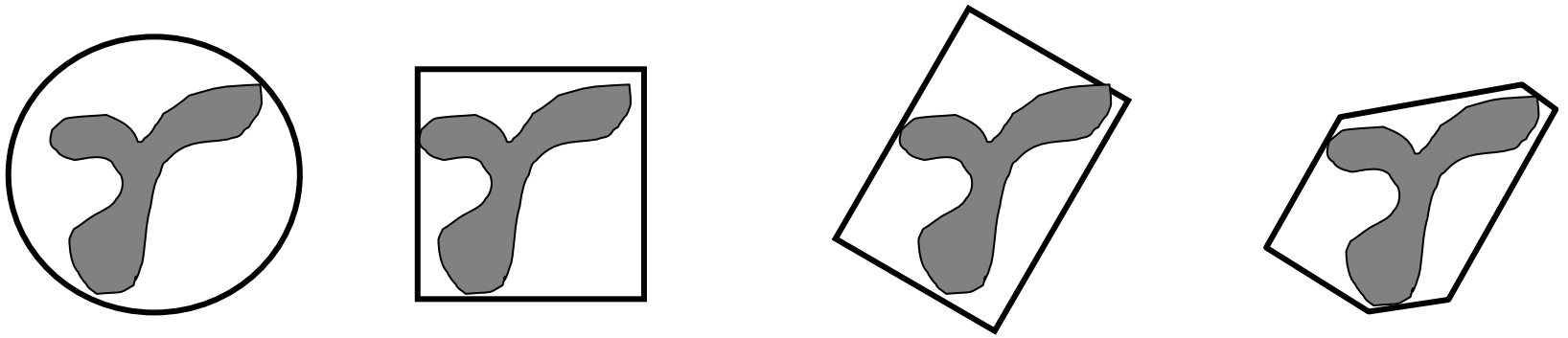
Oriented Bounding Box (OBB)



Discrete Oriented Polytopes (k-Dop)

Weniger Überlappungstests

Beispiele



Sphäre

Axis Aligned Bounding Box (AABB)

Oriented Bounding Box (OBB)

Discrete Oriented Polytopes (k-Dop)

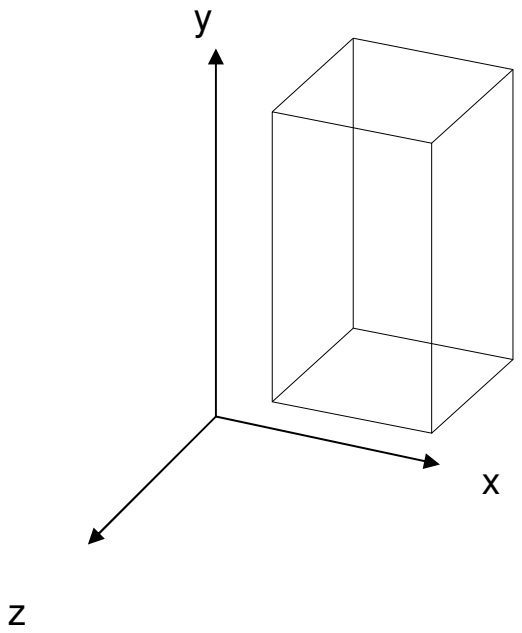
Weniger Überlappungstests

Schnellere Überlappungstests

2.2 Bounding Volume Schnitttests

2.2.1 Axis Aligned Bounding Box

Definition: (*Axis-Aligned Bounding Box*)
Eine Axis-Aligned Bounding Box (kurz AABB) ist ein Quader, dessen Oberflächennormalen mit den Achsen der Standardbasis übereinstimmen.

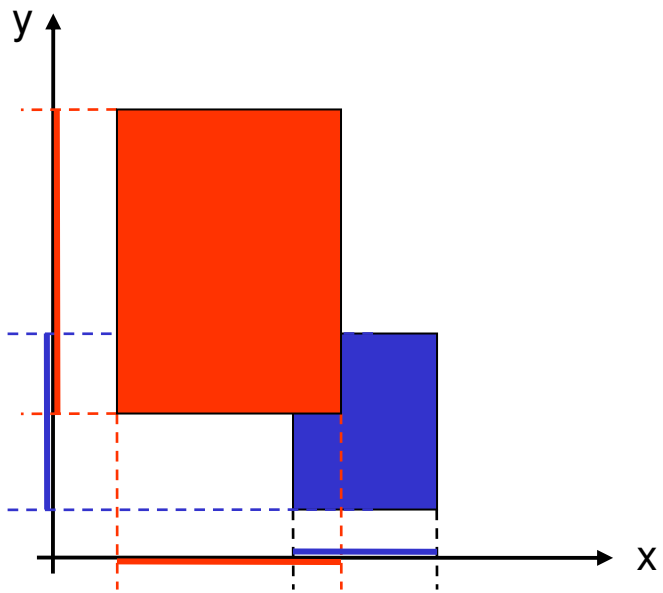


Schnitttest

Wie wird der Schnitttest zwischen zwei AABBs durchgeführt?

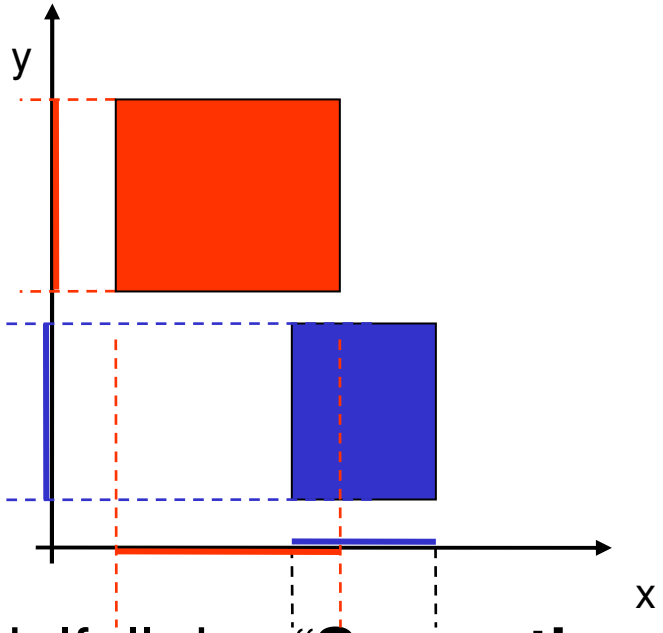
Idee: Dimensionsreduzierung

Zwei AABBs schneiden sich ⇔ Alle ihre Projektionen auf den Koordinatenachsen x,y und z überschneiden sich.



Schnitttest

Idee: Dimensionsreduzierung
Zwei AABBs sind disjunkt ⇔ Es gibt mindestens eine Koordinatenachse, auf der die Projektionen der AABBs sich nicht schneiden.



Dies ist ein Spezialfall des "**Separating Axis Theorems**" – dazu gleich mehr bei den OBBs.

... Das Beipiel aud dem Leben

Bewegliche Objekte

Was haben wir bis jetzt gewonnen?

Falls N Objekte gegeben sind, so sind nun zunächst

$$\frac{N^2 - N}{2}$$

Schnitttest zwischen AABBs durchzuführen.

Danach müssen nur von den Objekten Schnitttests durchgeführt werden, deren Bounding Box überlappen.

Muss wirklich jedes AABB mit jedem AABB verglichen werden?

Test mehrerer AABBs

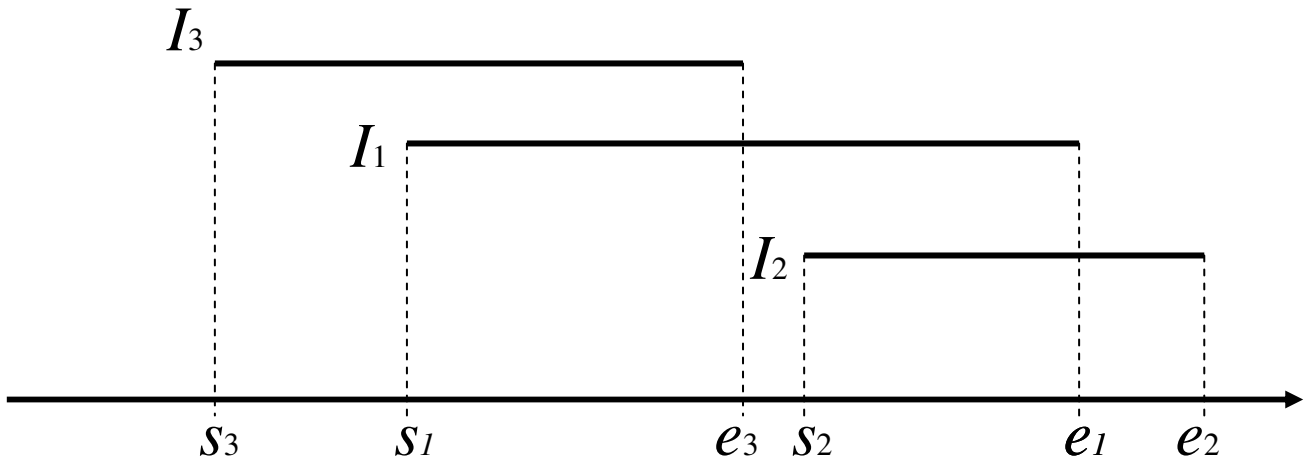
Statt alle AABBs paarweise zu vergleichen, folgende

Idee:

- Führe effizienten Überlappungstest auf jeder Koordinatenachse durch (Dimensionsreduzierung).
- Wir haben auf den Koordinatenachsen eine zusätzliche Information: Die Zahlen auf einer Achse sind durch den Vergleichsoperator **<** geordnet!
- Sortiere die Projektionen der AABBs auf jeder Achse.

Sweep and Prune Algorithmus

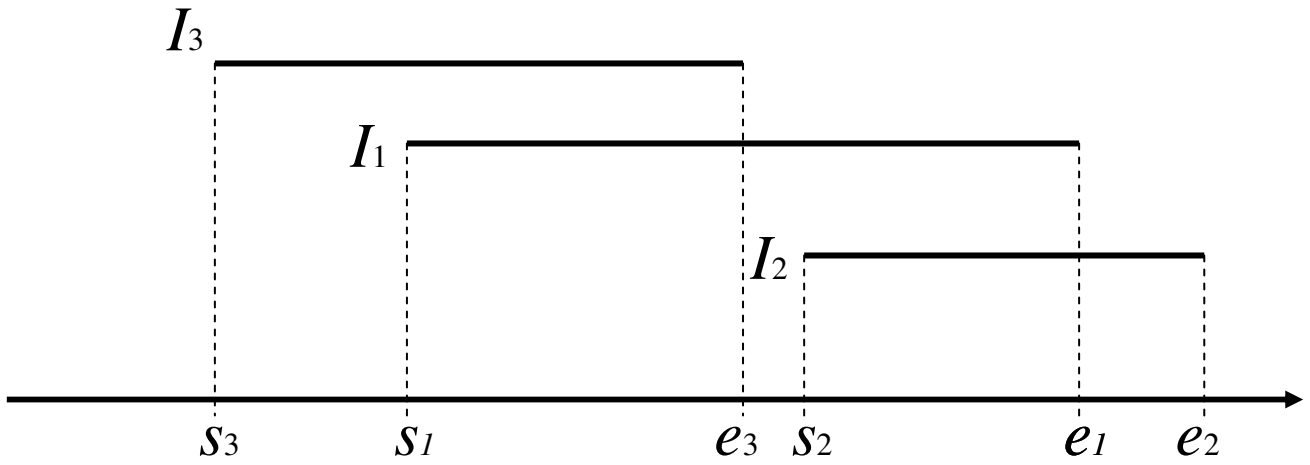
- Sortiere die Intervalle aufsteigend nach ihren linken Intervallgrenzen in einer Liste.



Sortierte Liste: L = [I_3, I_1, I_2]

Sweep and Prune Algorithmus

- Durchlaufe die Liste L mit einer Sweep-Line.
- Verwalte dabei eine zusätzliche Liste “aktiver” Intervalle.

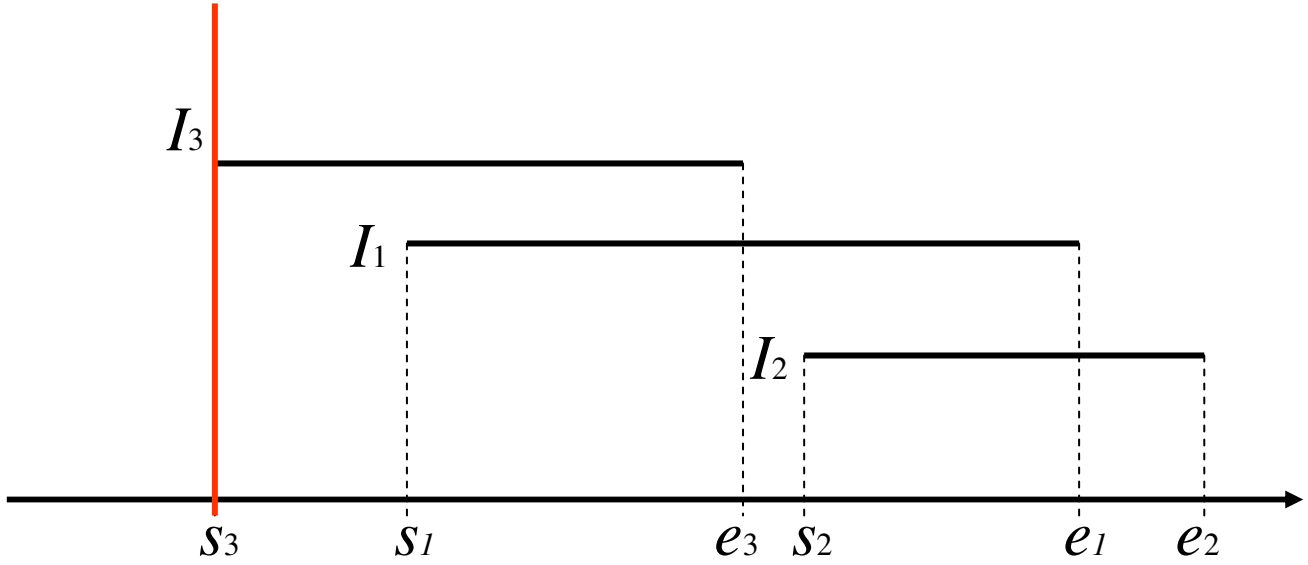


Sortierte Liste: L = [I_3, I_1, I_2]

Aktiv: A = [ ]

Sweep and Prune Algorithmus

- Beginne mit dem ersten Intervall I_3 in der Liste,
- Ein Intervall wird A zugeordnet, wenn die Sweepline die linke Seite des Intervalls erreicht.

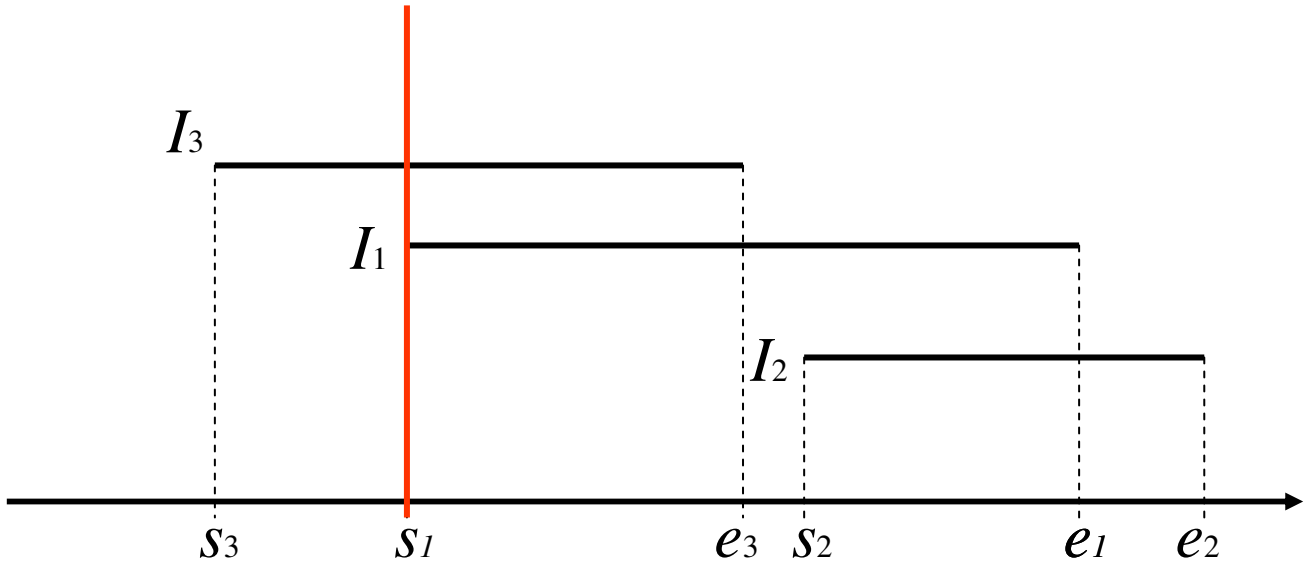


Sortierte Liste: L = [I_3, I_1, I_2]

Aktiv: A = [I_3]

Sweep and Prune Algorithmus

- Das Intervall I_1 wird zur Liste aktiver Intervalle hinzugenommen.
- Alle in A liegenden Intervalle überschneiden sich

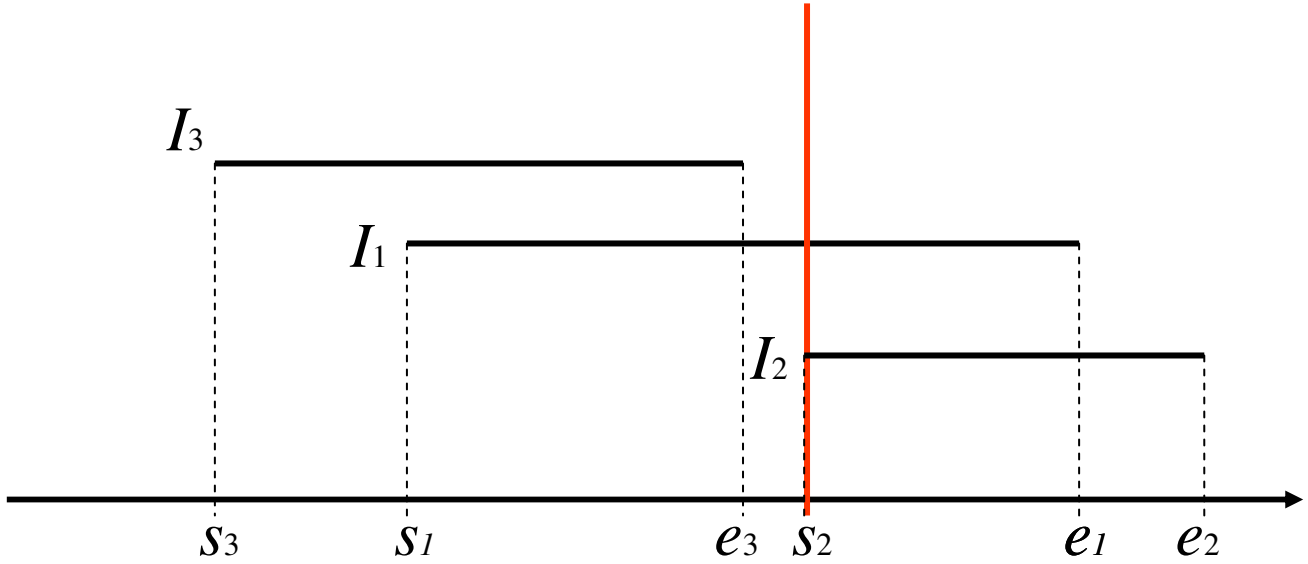


Sortierte Liste: L = [I_3, I_1, I_2]
Aktiv: A = [I_3, I_1]

Überlappung:
(I_1, I_3)

Sweep and Prune Algorithmus

- Ein Intervall wird aus A gelöscht, wenn die Sweep-Line die rechte Grenze des Intervalls überschritten hat.
- Lösche I_3 aus A, da e_3 kleiner als s_2.

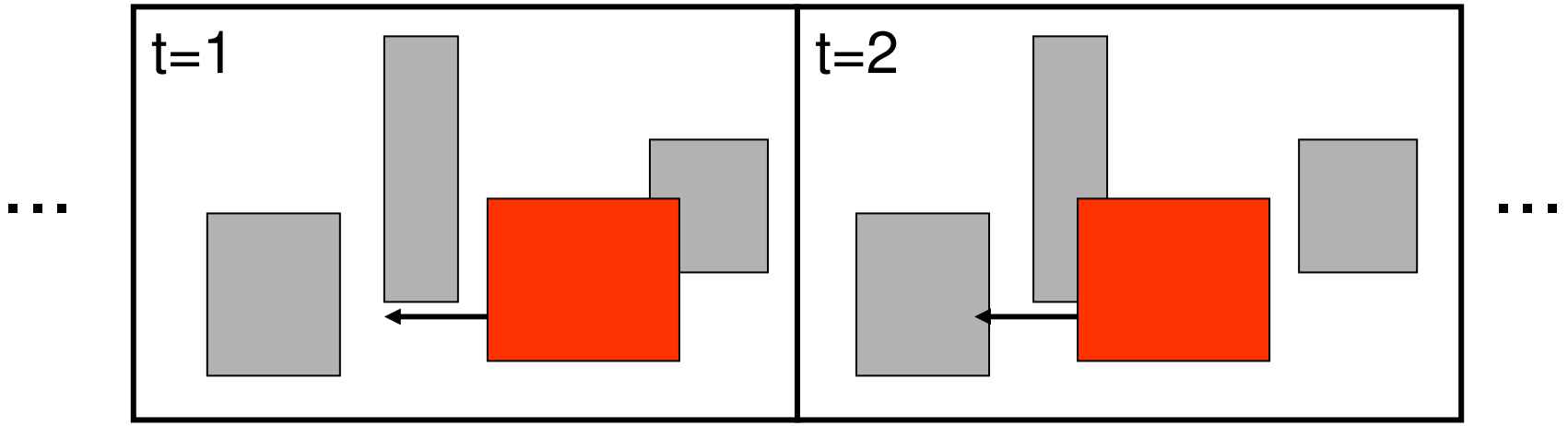


Sortierte Liste: L = [I_3, I_1, I_2]
Aktive Liste: A = [I_1 , I_2]

Überlappungen:
(I_1, I_3), (I_1, I_2)

Dynamisches Szenario

Objekte bewegen sich von Frame zu Frame.

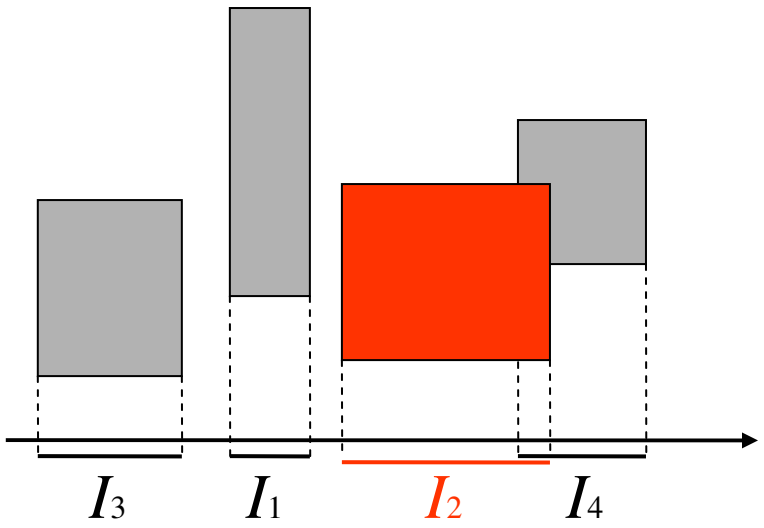


Idee:
Pro Frame bewegen sich die AABBs nur relativ wenig.
Manche AABBS bewegen sich gar nicht.
Nutze dies aus:

Zeitliche / Frame-to-Frame Kohärenz!

Dynamisches Szenario

Wiederverwendung der sortierten Listen L für jeden Zeitschritt t.



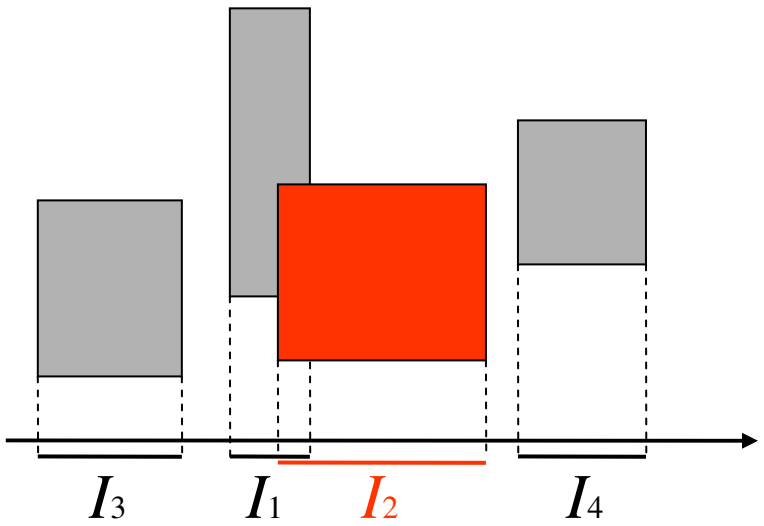
Zeit t=1

Sortierte Liste

L = [I_3, I_1, I_2 , I_4]

Dynamisches Szenario

Wiederverwendung der sortierten Listen L für jeden Zeitschritt t.

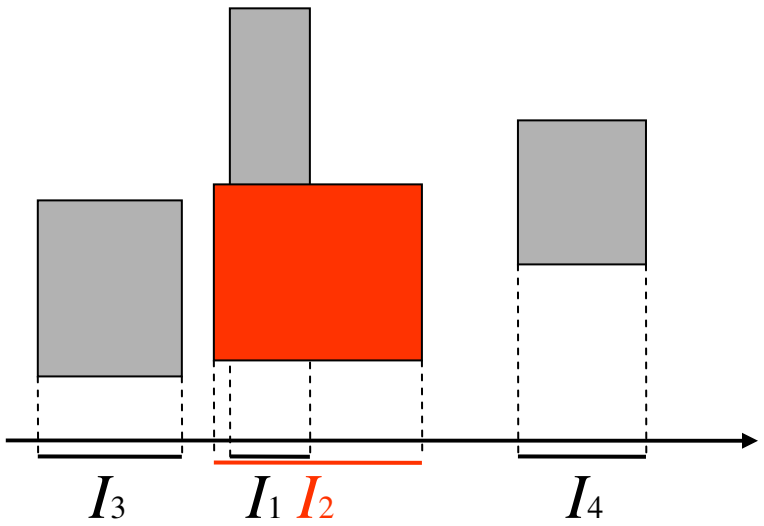


Zeit t=2:
Keine Änderung in L.

Sortierte Liste
L = [I_3, I_1, I_2 , I_4]

Dynamisches Szenario

Wiederverwendung der sortierten Listen L für jeden Zeitschritt t.



Zeit t=3:
Intervalle I_1 und I_2 werden vertauscht.

Sortierte Liste
L = [I_3, I_2 , I_1 , I_4]

Aufwand

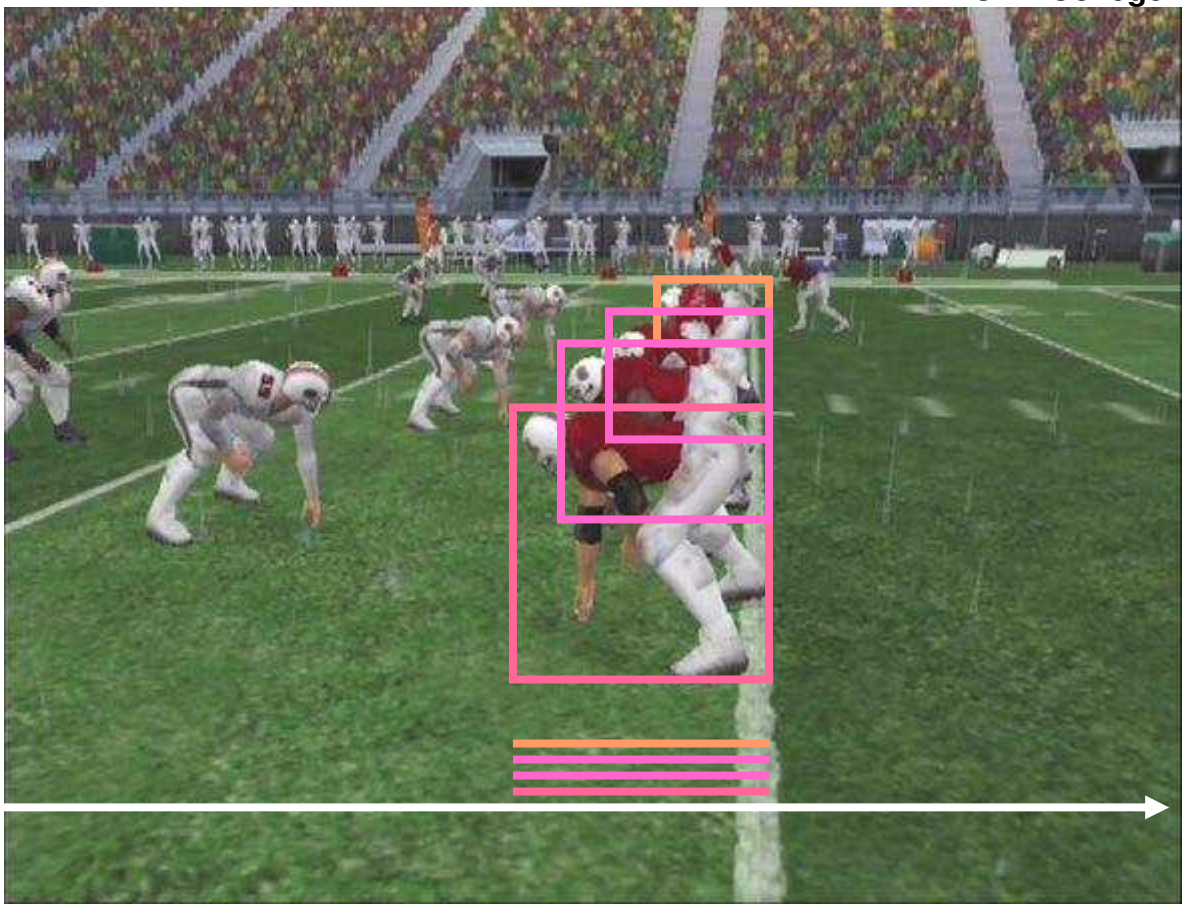
- Gegeben N AABBS.
- Aufwand die N Intervalle zu ersten mal zu sortierten: $O(N \log N)$
- Im Idealfall:
  - Erzeugte Liste beim nächsten Frame schon gut vorsortiert
  - Einzelne Vertauschungen sind notwendig.
- Dann Sortieraufwand $O(N)$.
- Es bietet sich z.B. Bubblesort an.
- Aber: Falls sich viele Intervalle an einer Stelle ballen im schlimmsten Fall doch wieder $O(N^2)$. Dann eignen sich andere Verfahren (z.B. 2D-Box Tests) besser.

Probleme

NCAA College Football 2K3

Probleme

NCAA College Football 2K3



Cluster von Intervallen. Sortieraufwand bis zu $O(N^2)$.

Literatur / Software

- J. Cohen, M.C. Lin, D. Manocha and M.Poamgi, "*I-COLLIDE: An interactive and Exact Collisions Detection System for Large Scale Environment*", Proceedings 1996 Symposium on Interactive 3D Graphics, pp.189-196, 1995
- Software: http://www.cs.unc.edu/~geom/I_COLLIDE/index.html

2.2.2 OBB Oriented Bounding Box

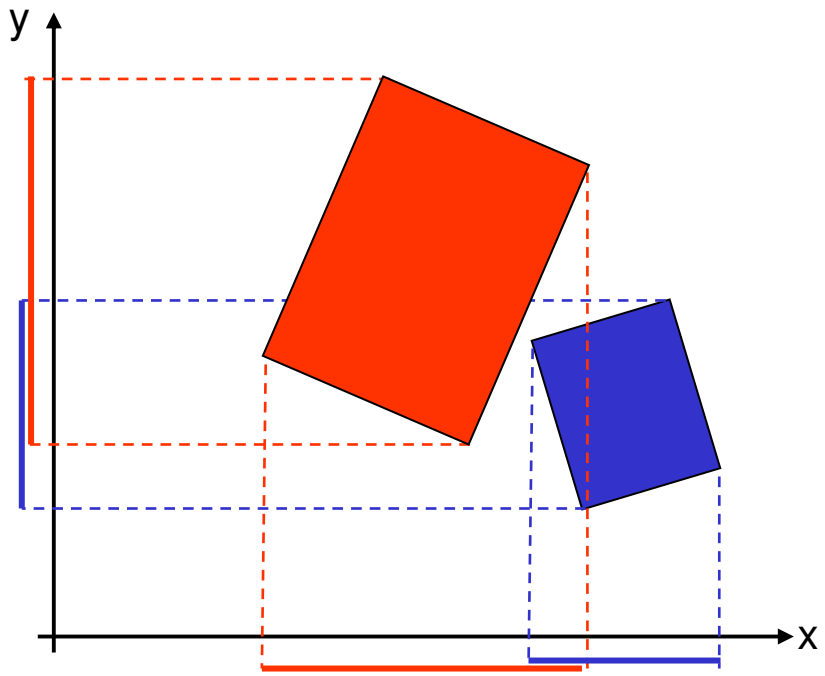
Definition: (*Oriented Bounding Box*)

Eine Oriented Bounding Box (kurz OBB) ist ein Quader.

- Wurden zunächst für Raytracing eingeführt.
- Motivation: Engere Passform um das zu umschließende Objekt führt zu besserer Identifizierung von möglichen Kollisionspaaren.
- Dimensionsreduzierung wie bei AABBs möglich?

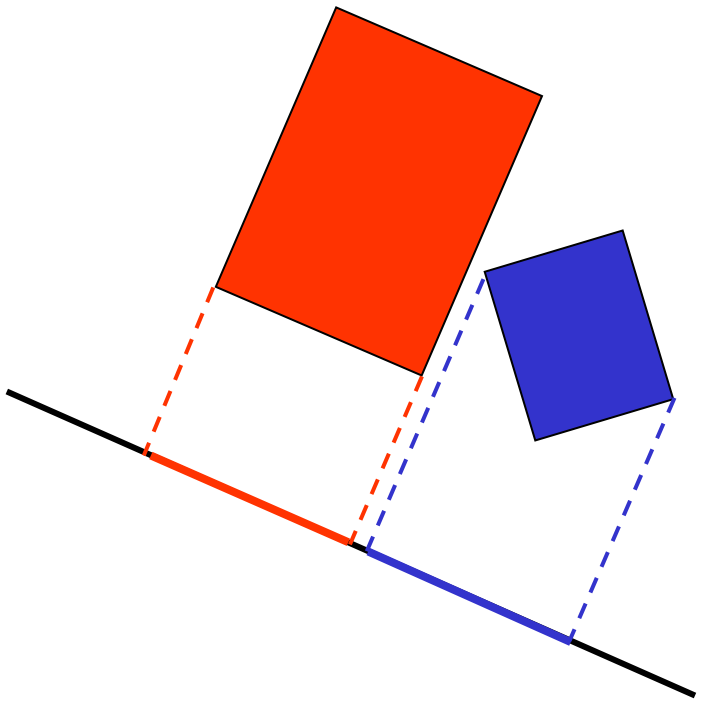
So einfach geht es nicht...

Einfacher Ansatz funktioniert nicht:



Obwohl sich jeweils die Projektionen auf die Koordinatenachsen überlappen, schneiden sich die beiden OBBs nicht!

... aber mit einer andern Achse

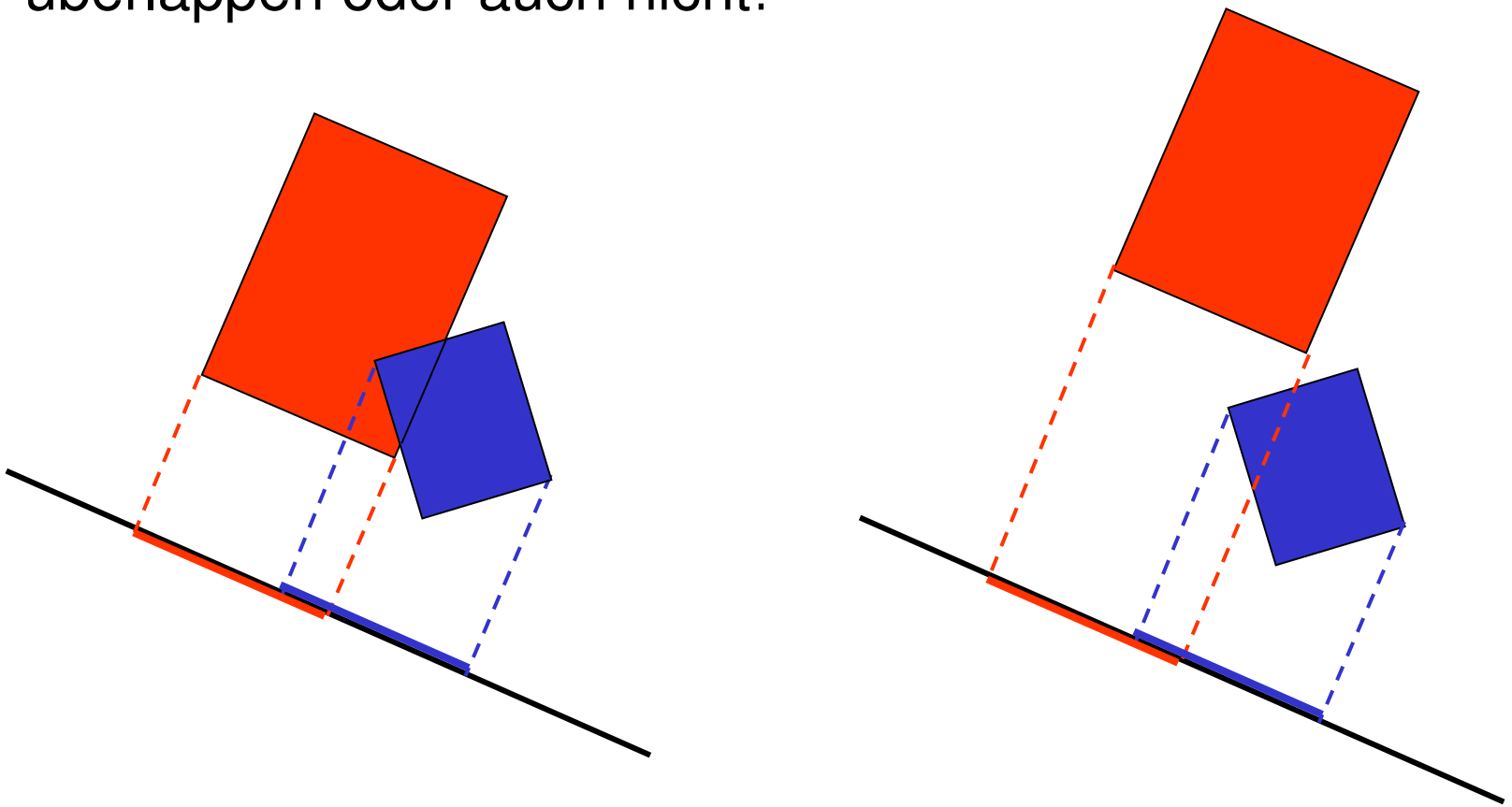


Aber:

Wenn es eine Achse im Raum gibt, so dass sich die Projektionen NICHT schneiden, so sind die beiden OBBs disjunkt!

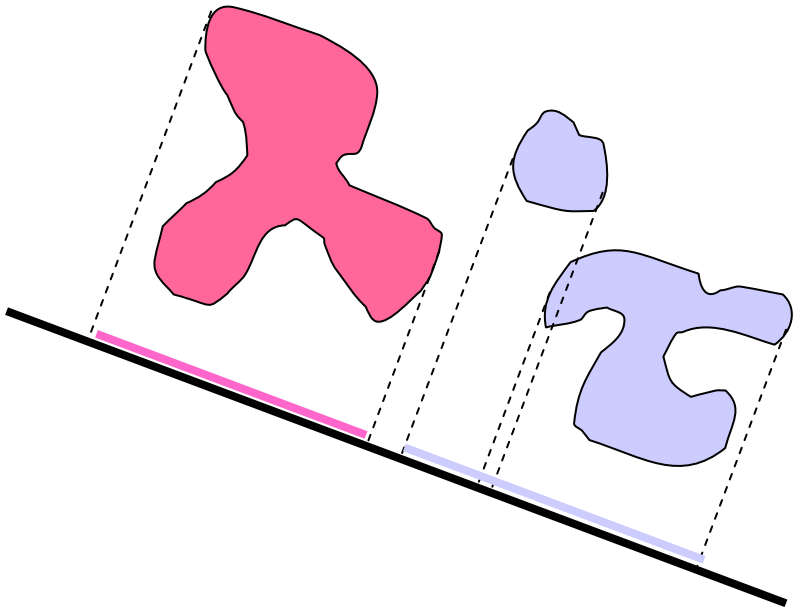
Weitere Beobachtung

Schneiden sich die Projektionen zweier OBBs auf einer beliebigen Geraden, so können sich die OBBs überlappen oder auch nicht!



Separating Axis Theorem

Definition: (*Trennachse, Separating Axis*)
Sind A und B zwei Mengen in $\boldsymbol{R^3}$ so heißt eine Gerade in $\boldsymbol{R^3}$ **Trennachse** (separating axis), falls die Projektionen der zwei Mengen auf die Achse disjunkt sind.



Separating Axis Theorem

Theorem: (*Separating Axis Theorem*)

Zwei konvexe Polyeder A und B sind genau dann disjunkt, wenn es eine Trennachse gibt (d.h die Projektionen der Polyeder auf die Trennachse ergeben zwei disjunkte Intervalle).

Falls zwei konvexe Polyeder A und B disjunkt sind, so gibt es eine Trennachse die entweder zu

1. einer Seitenfläche von A
2. einer Seitenfläche von B
3. jeweils einer Kante von A und B

orthogonal ist.

Beweis: S.Gottschalk; Collisions detection using Oriented Bounding Boxes, Dissertation, University of Carolina, Chapel Hill, 2000

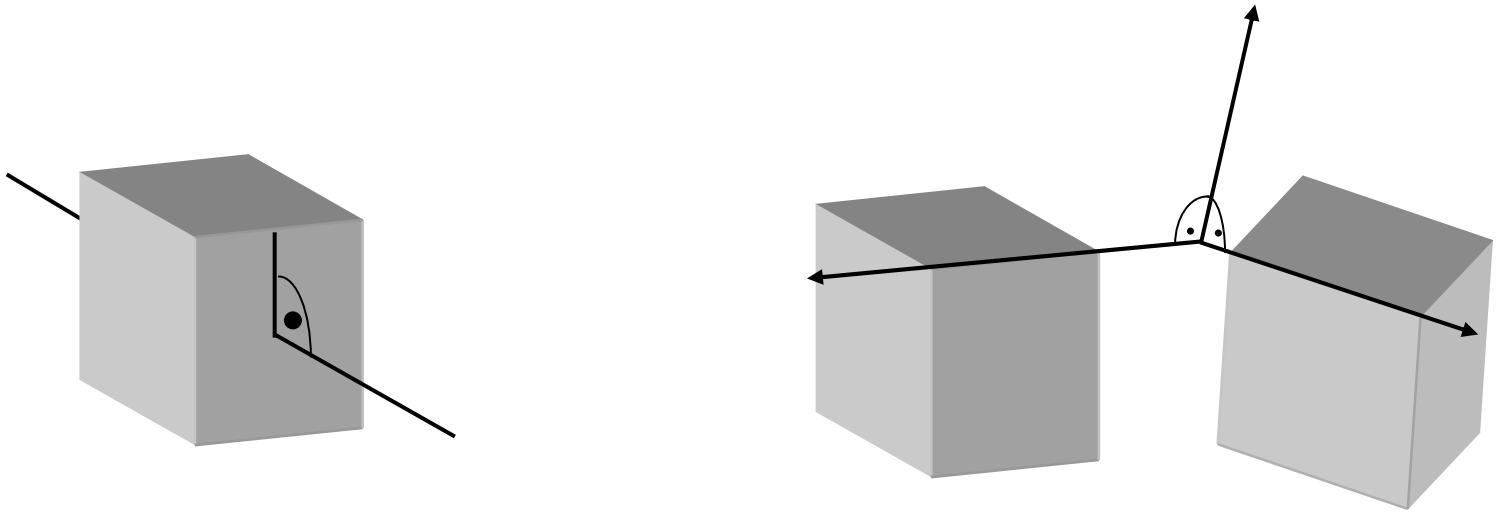
Separating Axis Theorem

Korollar: (*Spezialfall OBBs*)

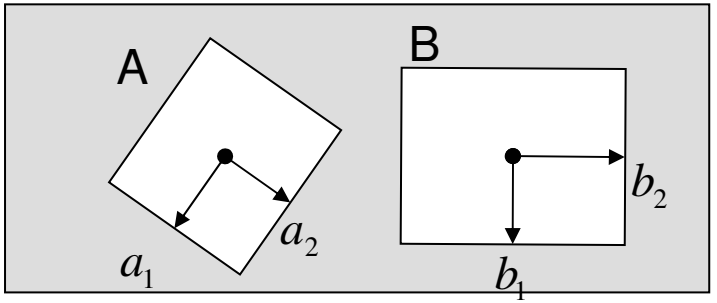
Zwei OBBs sind disjunkt, wenn es eine Trennachse gibt,

– die orthogonal zu einer Seitenfläche eines OBBs liegt, oder

– die orthogonal ist zu der Fläche, die durch zwei Kanten von den zwei OBBs aufgespannt wird.



Separating Axis Theorem



Gegeben zwei OBBs A und B.

Seien a_1, a_2, a_3 die Hauptachsen von A und b_1, b_2, b_3 die Hauptachsen von B bzgl. dem von a_1, a_2, a_3 erzeugten Koordinatensystem.

Dann gilt:

A und B sind genau dann disjunkt falls es

$$v, w \in \{a_1, a_2, a_3, b_1, b_2, b_3\} \quad \text{gibt mit } v \neq w \quad ,$$

so dass $l := v \times w$ eine Trennachse zwischen A und B ist.

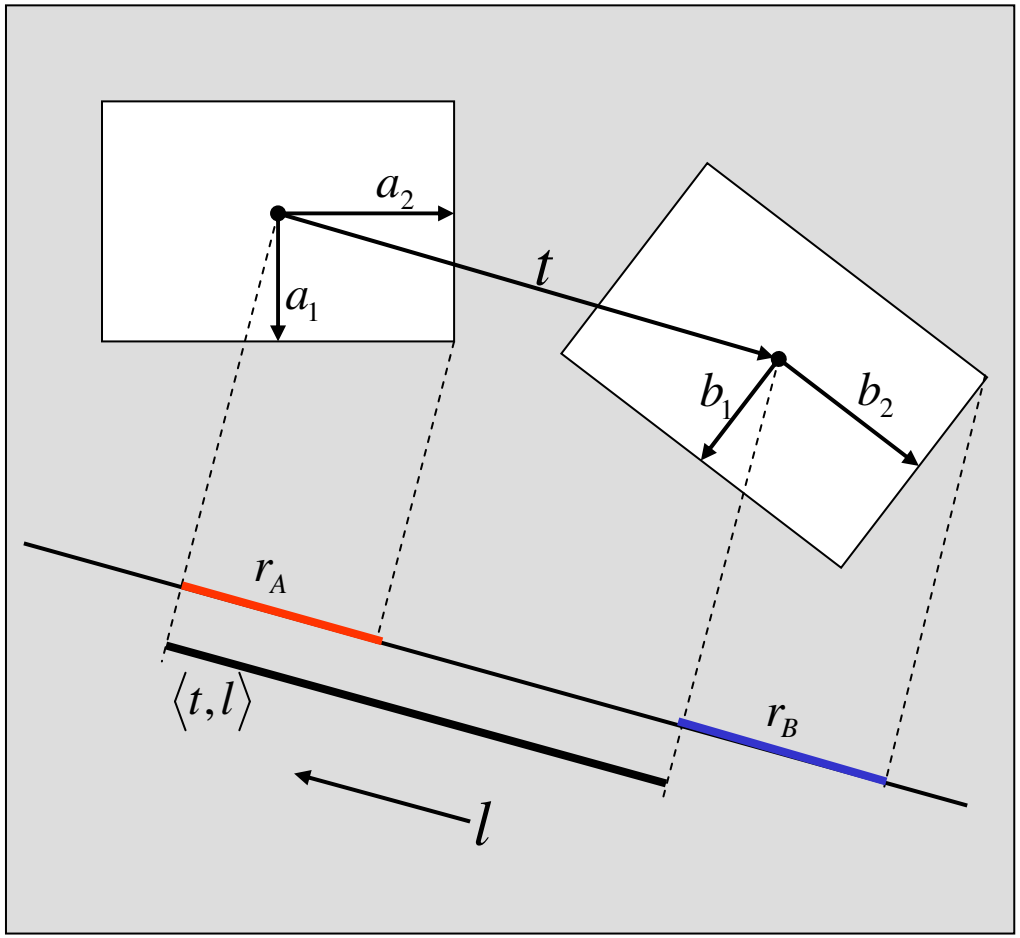
Berechnung der Projektionen

Sei $\|l\| = 1$
Die Projektionen von A und B auf l sind genau dann disjunkt, falls

$$|\langle t, l \rangle| > r_A + r_B$$

d.h.

$$|\langle t, l \rangle| > \sum_{i=1}^{3} |\langle a_i, l \rangle| + \sum_{i=1}^{3} |\langle b_i, l \rangle|$$

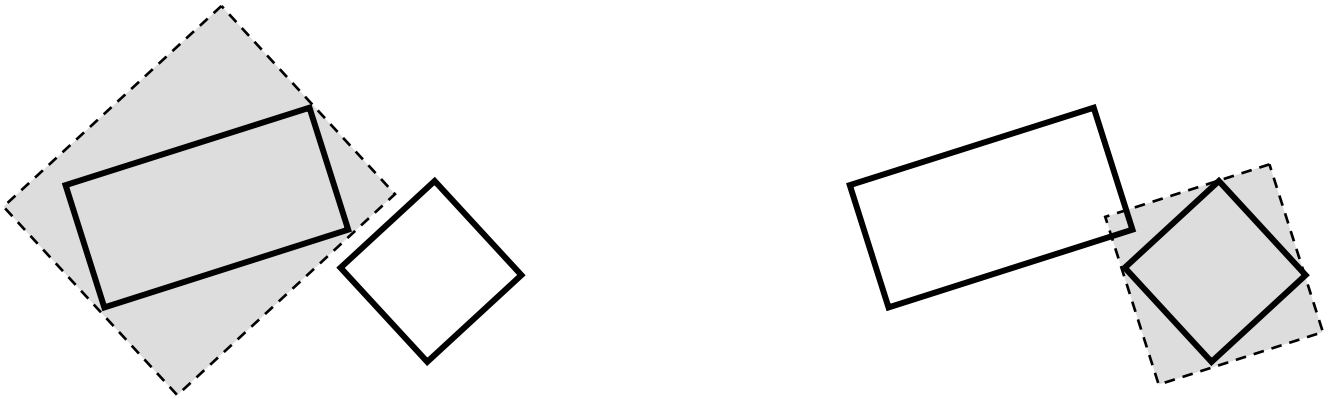


Aufwand

- Es sind maximal 15 Tests durchzuführen.
- Mit Wiederverwendung von Zwischenergebnissen kommt man mit 180 bis 240 arithmetischen Operationen aus (inklusive der Transformation von B in das Koordinatensystem von A).
- Es sollten zuerst die Test zu den Hauptachsen von A und B durchgeführt werden, dann die zu den Kombinationen von A und B.
- Numerisch stabil robust (keine Divisionen, Wurzeln,...).

Beschleunigung

- Teste nur die Hauptachsen.
- Ist konservativer Test: es werden OBBs als kollidierend markiert, obwohl sie es nicht sind.



- Falls aber hierarchische OBB-Bäume verwendet werden, kann dies wieder aufgelöst werden.

Dynamisches Szenario

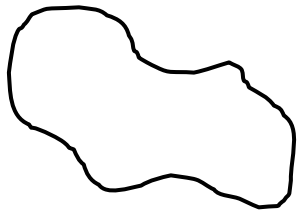
- Falls sich Objekte entlang einer Geraden bewegen, so muss nur der Translationsvektor t zwischen den OBBs berechnet werden (die Hauptachsen ändern sich nicht). Die rechte Seite in $\langle t, l \rangle > r_A + r_B$ muss nicht mehr neu berechnet werden.
- Rotationen können auch einfach auf die Achsen übertragen werden.

... Das Beipiel aus dem Leben

Bewegliche Objekte

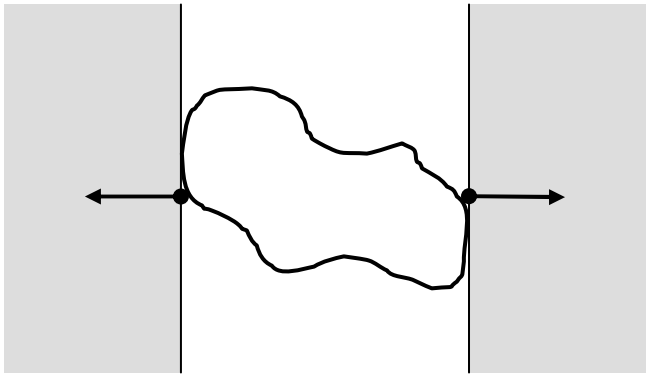
2.2.3 k-DOP Discrete Oriented Polytope

- Sind Polytope (d.h. konvexe Polyeder), deren Seitenflächen definiert werden durch Halbräume, deren Normalen durch eine Anzahl von k Orientierungen gegeben sind. (k ist dabei immer gerade).



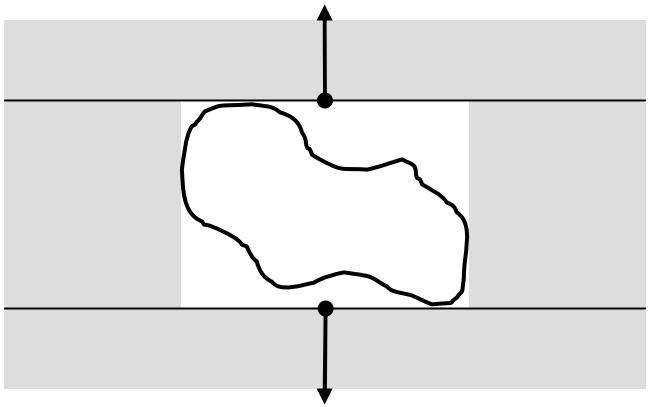
2.2.3 k-DOP Discrete Oriented Polytope

- Sind Polytope (d.h. konvexe Polyeder), deren Seitenflächen definiert werden durch Halbräume, deren Normalen durch eine Anzahl von k Orientierungen gegeben sind. (k ist dabei immer gerade).



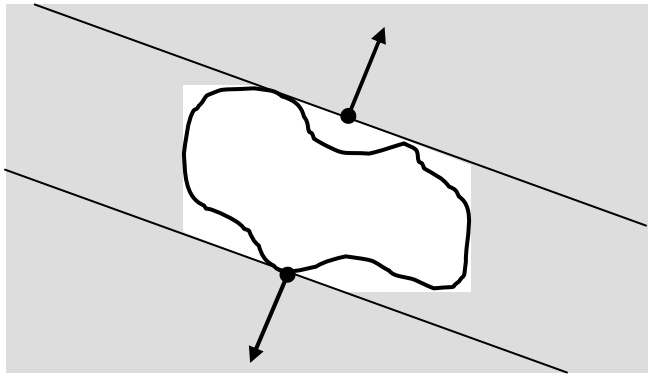
2.2.3 k-DOP Discrete Oriented Polytope

- Sind Polytope (d.h. konvexe Polyeder), deren Seitenflächen definiert werden durch Halbräume, deren Normalen durch eine Anzahl von k Orientierungen gegeben sind. (k ist dabei immer gerade).



2.2.3 k-DOP Discrete Oriented Polytope

- Sind Polytope (d.h. konvexe Polyeder), deren Seitenflächen definiert werden durch Halbräume, deren Normalen durch eine Anzahl von k Orientierungen gegeben sind. (k ist dabei immer gerade).



Definition

Definition: (*k-DOP*)
Ein k-DOP A (wobei k gerade ist) ist definiert durch $k/2$ normalisierte Normalen (Orientierungen)

$$n_i, 1 \le i \le k/2$$

sowie jeweils zwei dazugehörige skalare Werte

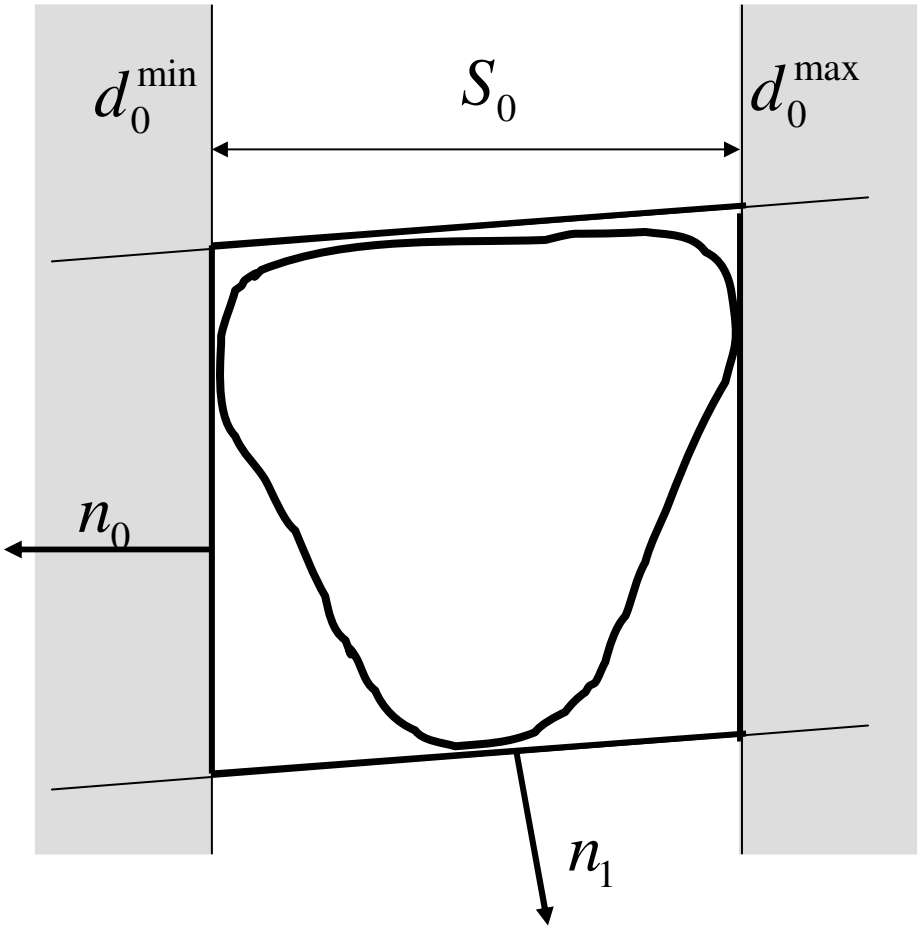
$$d_i^{\min}, d_i^{\max} \quad \text{mit} \quad d_i^{\min} < d_i^{\max}$$

Jedes Tripel $(n_i, d_i^{\min}, d_i^{\max})$ definiert ein Slab (Platte) S_i die definiert ist als das Volumen zwischen den zwei Ebenen

$$\langle n_i, x \rangle + d_i^{\min} = 0 \quad \text{und} \quad \langle n_i, x \rangle + d_i^{\max} = 0 \quad .$$

Dann ist das k-DOP definiert durch

$$A := \bigcap_{1 \le l \le k/2} S_l$$



Schnitte von k-Dops

- Verwende ähnlichen Algorithmus wie bei AABBs.
- Voraussetzung: Seien A und B k-Dops, die mit den **selben Normalen n_i definiert sind!**
- A und B sind schneiden sich genau dann, wenn

$$S_i^A \cap S_i^B \neq \{\} \qquad \text{für alle} \quad 1 \leq i \leq \frac{k}{2}.$$

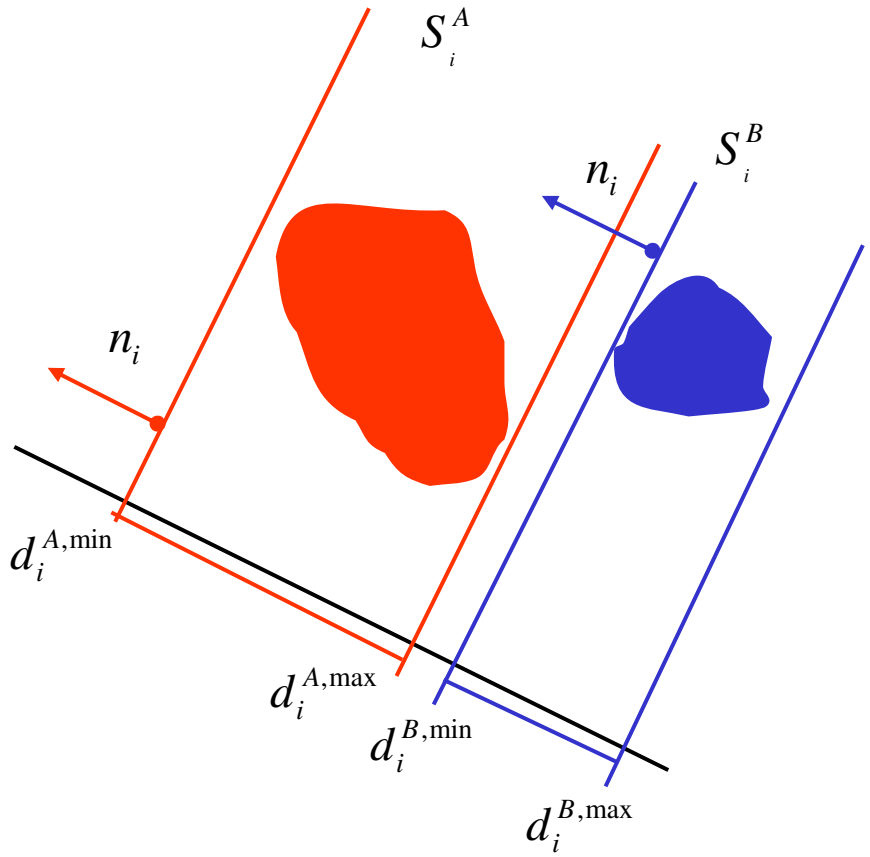
- Dies ist ein einfacher Intervalltest:

$$S_i^A \cap S_i^B = \{\}$$

genau dann, wenn

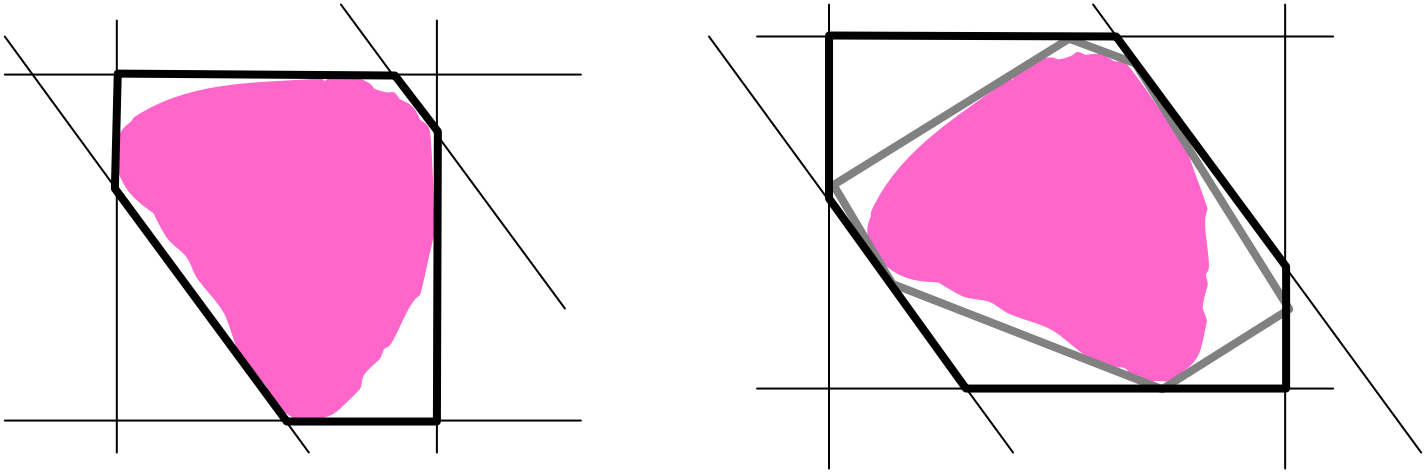
$$d_i^{B,\min} > d_i^{A,\max} \quad \text{oder} \quad d_i^{A,\min} > d_i^{B,\max}.$$

Schnitte von k-Dops



Dynamisches Szenario

- Orientierungen der k-DOPs sind fest. Falls zugrunde liegendes Objekte rotiert wird, muss k-DOP neu berechnet werden.
- Annahme: Im Allg. haben k-DOPs weniger Eckpunkte als die zugrundeliegenden Polyeder.
- Benutze die Eckpunkte des originalen k-DOPs zur Berechnung des neuen k-DOPs .



2.3 Hierarchische Kollisionserkennung mit Bounding Volumes (BV)

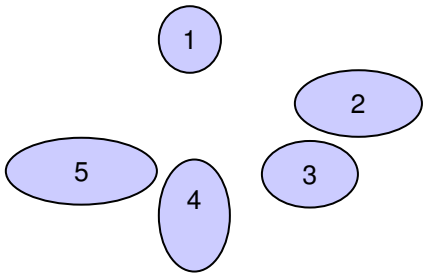
Idee:

- Hüllkörper sind im Allg. nicht optimal (z.B. Sphäre).
- Statt nur ein BV für ein Objekt zu verwenden, überdecke das Modell mit mehreren kleineren enger sitzenden Hüllkörpern.
- Erstelle dazu eine Repräsentation des 3D-Modells hierarchisch mit Bounding Volumes.
- Benutze dazu eine Baumstruktur:
  - Die Wurzel besteht aus einem BV, das alle seine Kinder und das gesamte 3D-Modell umschließt.
  - Das BV eines Knotens umschließt alle seine Kinder.
  - Die Blätter bestehen aus einem oder mehreren Polygonen des 3D-Modells.

Konstruktion von BV-Hierarchien

Bottom Up

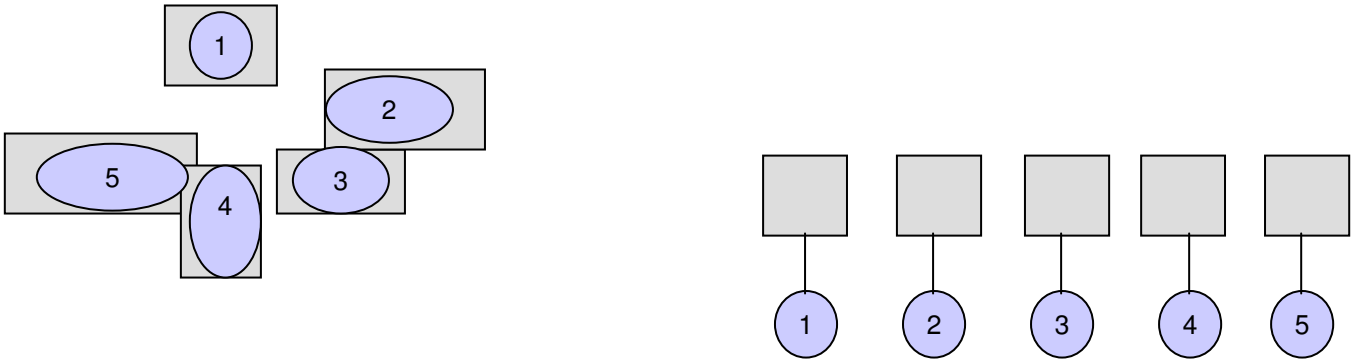
- Konstruiere BVs für Polygonmengen
- Diese BVs werden gruppiert und wieder zu einem größeren BV zusammengefasst.
- Dies wird solange gemacht, bis nur noch ein BV übrigbleibt, die Wurzel des Hierarchiebaumes.



Konstruktion von BV-Hierarchien

Bottom Up

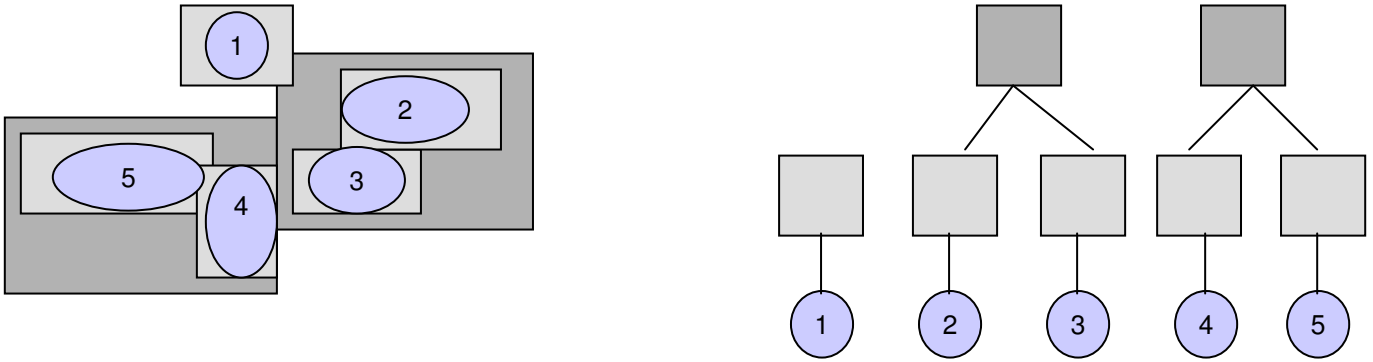
- Konstruiere BVs für Polygonmengen
- Diese BVs werden gruppiert und wieder zu einem größeren BV zusammengefasst.
- Dies wird solange gemacht, bis nur noch ein BV übrigbleibt, die Wurzel des Hierarchiebaumes.



Konstruktion von BV-Hierarchien

Bottom Up

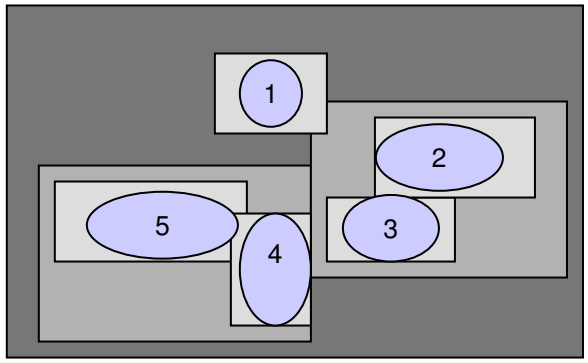
- Konstruiere BV für Polygonmengen
- Diese BVs werden gruppiert und wieder zu einem größeren BV zusammengefasst.
- Dies wird solange gemacht, bis nur noch ein BV übrigbleibt, die Wurzel des Hierarchiebaumes.

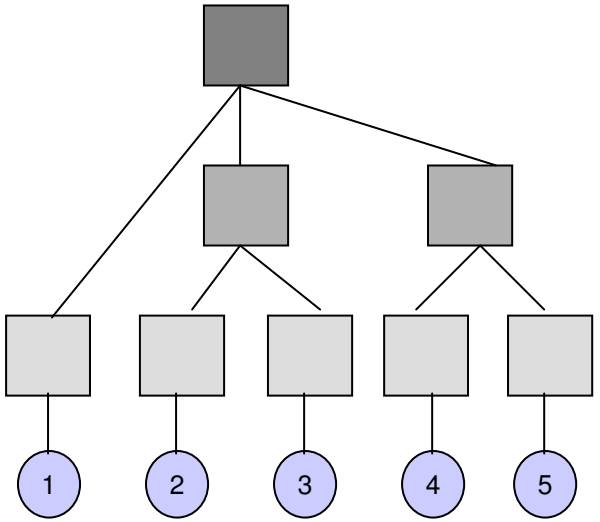


Konstruktion von BV-Hierarchien

Bottom Up

- Konstruiere BV für Polygonmengen
- Diese BVs werden gruppiert und wieder zu einem größeren BV zusammengefasst.
- Dies wird solange gemacht, bis nur noch ein BV übrigbleibt, die Wurzel des Hierarchiebaumes.

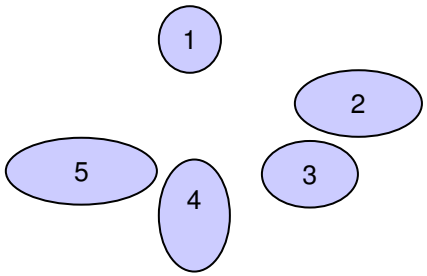




Konstruktion von BV-Hierarchien

Top Down (am verbreitetsten)

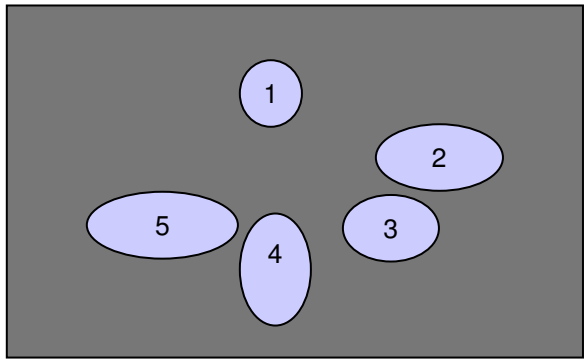
- Konstruiere BV, welches das gesamte 3D-Modell umfasst. Dies ist die Wurzel des Baumes.
- Zerlege dann das BV der Wurzel in kleinere BVs, die Kinder.
- Verfahre mit diesen Knoten genauso wie mit der Wurzel.



Konstruktion von BV-Hierarchien

Top Down (am verbreitetsten)

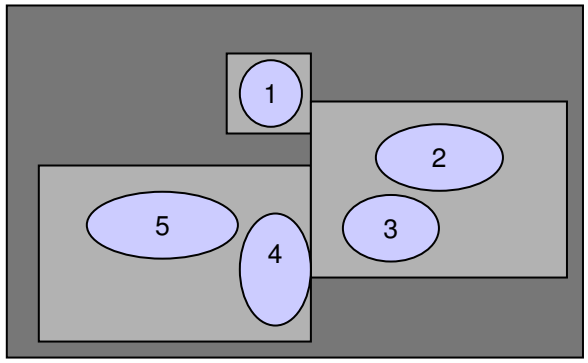
- Konstruiere BV, welches das gesamte 3D-Modell umfasst. Dies ist die Wurzel des Baumes.
- Zerlege dann das BV der Wurzel in kleinere BVs, die Kinder.
- Verfahre mit diesen Knoten genauso wie mit der Wurzel.

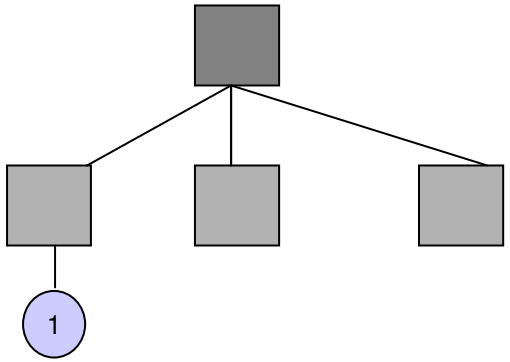


Konstruktion von BV-Hierarchien

Top Down (am verbreitetsten)

- Konstruiere BV, welches das gesamte 3D-Modell umfasst. Dies ist die Wurzel des Baumes.
- Zerlege dann das BV der Wurzel in kleinere BVs, die Kinder.
- Verfahre mit diesen Knoten genauso wie mit der Wurzel.

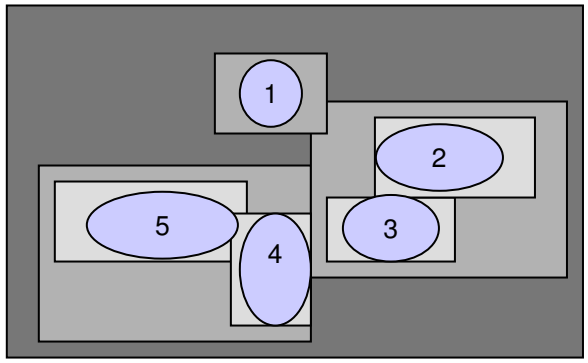


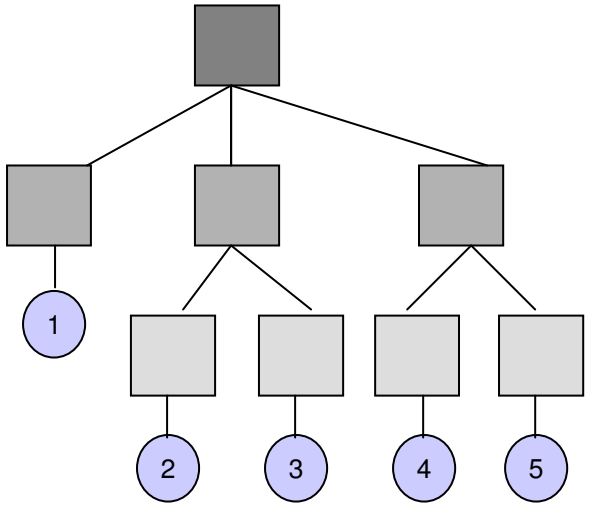


Konstruktion von BV-Hierarchien

Top Down (am verbreitetsten)

- Konstruiere BV, welches das gesamte 3D-Modell umfasst. Dies ist die Wurzel des Baumes.
- Zerlege dann das BV der Wurzel in kleinere BVs, die Kinder.
- Verfahre mit diesen Knoten genauso wie mit der Wurzel.

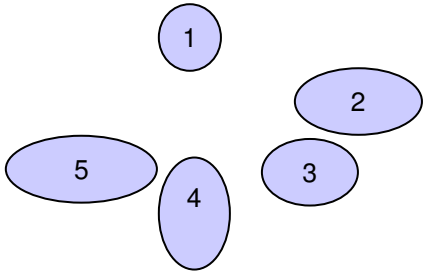




Konstruktion von BV-Hierarchien

Incremental Tree Insertion

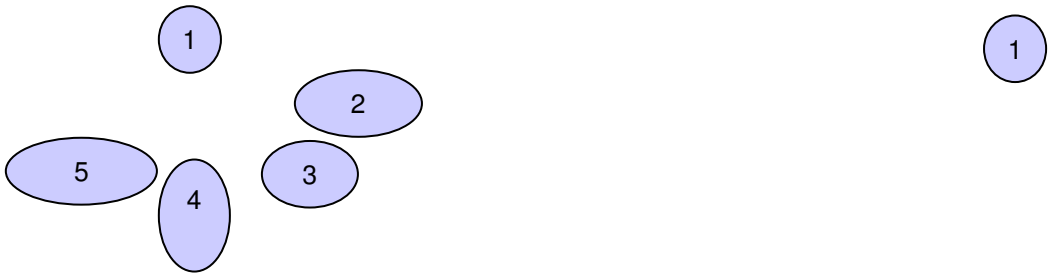
- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.



Konstruktion von BV-Hierarchien

Incremental Tree Insertion

- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.



Konstruktion von BV-Hierarchien

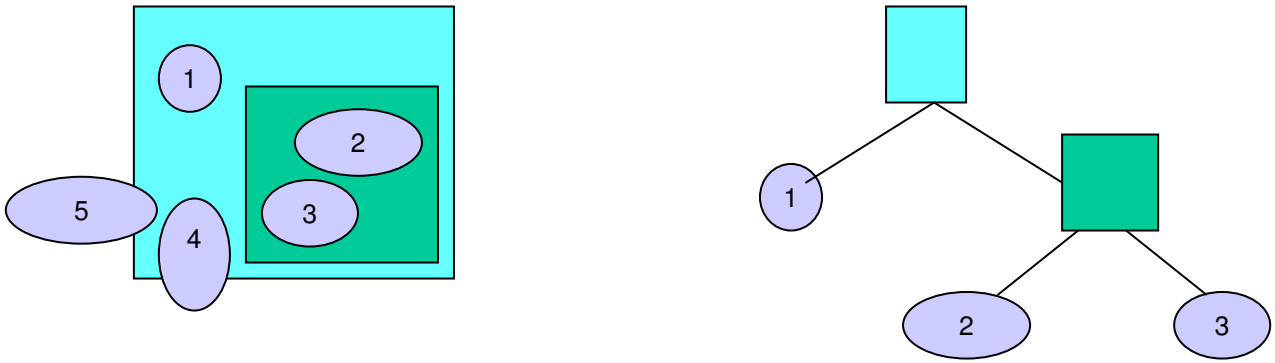
Incremental Tree Insertion

- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.

Konstruktion von BV-Hierarchien

Incremental Tree Insertion

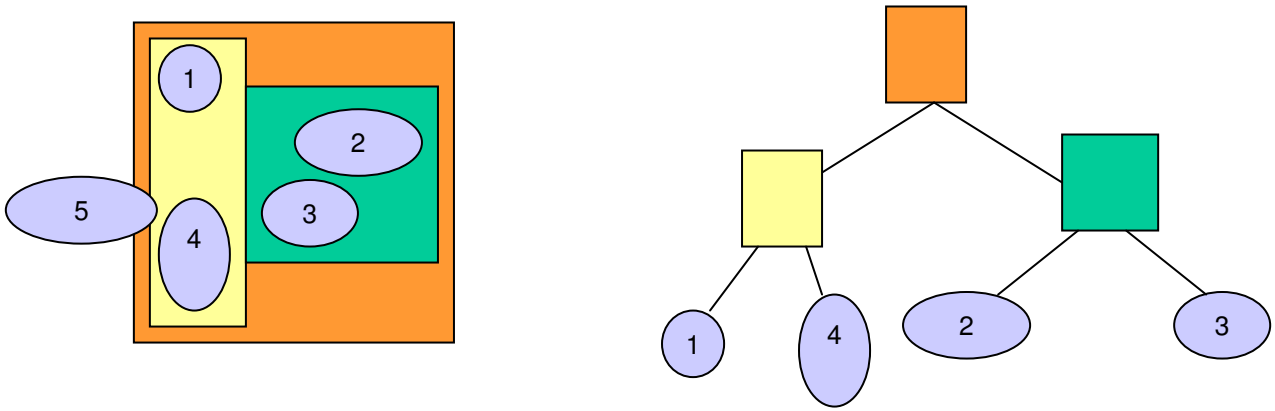
- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.



Incremental Tree Insertion

Incremental Tree Insertion

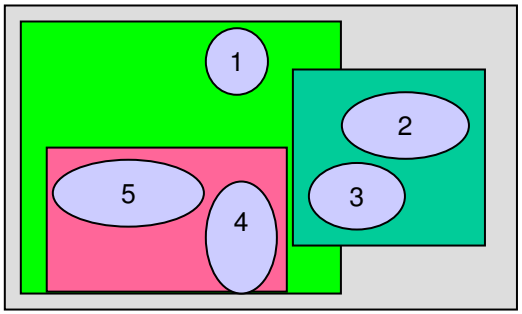
- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.

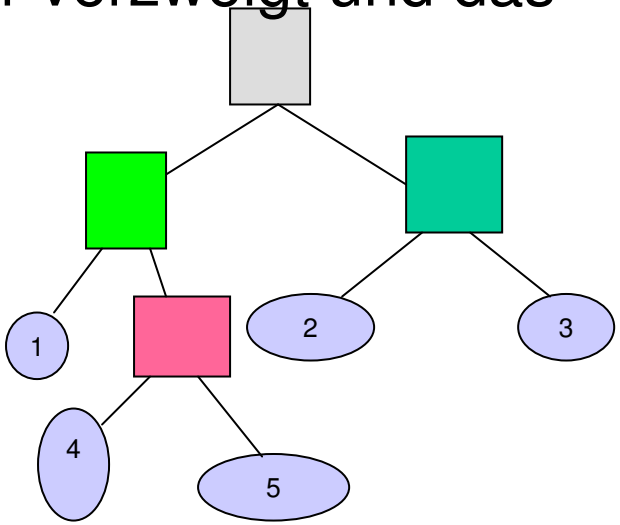


Incremental Tree Insertion

Incremental Tree Insertion

- Beginne mit leerem Baum.
- Setze inkrementell nach und nach alle Polygone in den Baum ein.
- Dabei wird der Baum von der Wurzel zu den Blättern hin durchlaufen. Beim Durchlaufen wird gegebenenfalls an einem Knoten weiter verzweigt und das zugehörige BV vergrößert.





Kollisionstest zwischen Hierarchien

```
1:  bool FindCollision(Model A, Model B)
2:  {
3:      if (! BVCollide(A.BV, B.BV)) return false;
4:      else
5:      {
6:          if (isLeaf(A))
7:          {
8:              if (isLeaf(B))
9:              {
10:                 for (jedes Paar von Primitiven A.P aus A, B.P aus D)
11:                 {
12:                     if ( PrimitiveCollide(A.P,B.P) ) return true;
13:                     else return false;
14:                 }
15:             }
16:             else
17:             {
18:                 for (jedes Kind C von B) do FindCollision (A,C);
19:             }
20:         }
21:         else
22:         {
23:             for (jedes Kind C von A) do FindCollision(C,B);
24:         }
25:     }
26: }
```

Kostenfunktion

$$t = n_v c_v + n_p c_p + n_u c_u$$

n_v : Anzahl der BV/BV-Überlappungstests

c_v : Kosten für einen BV/BV-Überlappungstest

n_p : Anzahl der Überlappungstests zwischen Primitiven

c_p : Kosten für einen Überlappungstest zwischen Primitiven

n_u : Anzahl der BVs, die wegen Modellbewegungen aktualisiert werden müssen

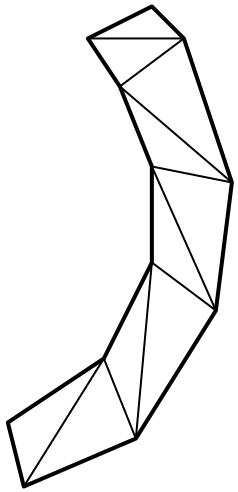
c_u : Kosten für die Aktualisierung eines BV

2.3.1 OBB-Bäume

- Top Down Entwicklung des Baumes
- Binärbaum
- Blätter enthalten ein oder mehrere Dreiecke.

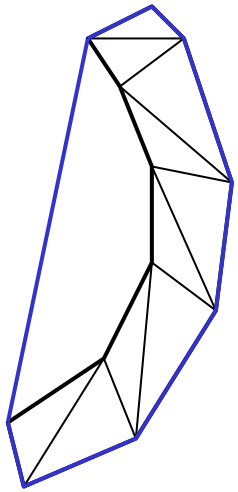
Konstruktion eines OBB-Baums

Gegeben: Polyeder, nicht notwendigerweise konvex



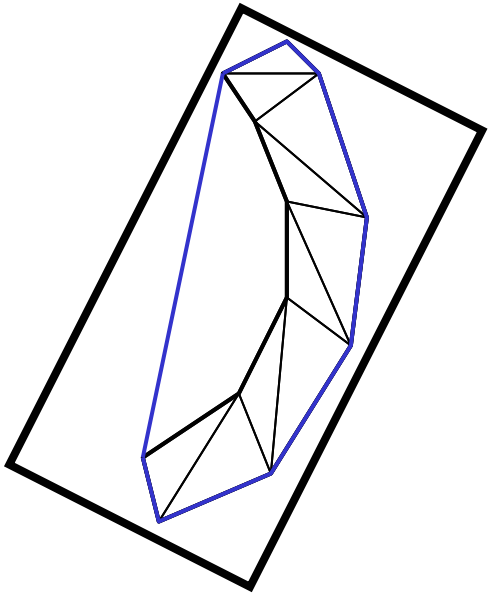
Konstruktion eines OBB-Baums

1. Bestimme konvexe Hülle des Polyeders



Konstruktion eines OBB-Baums

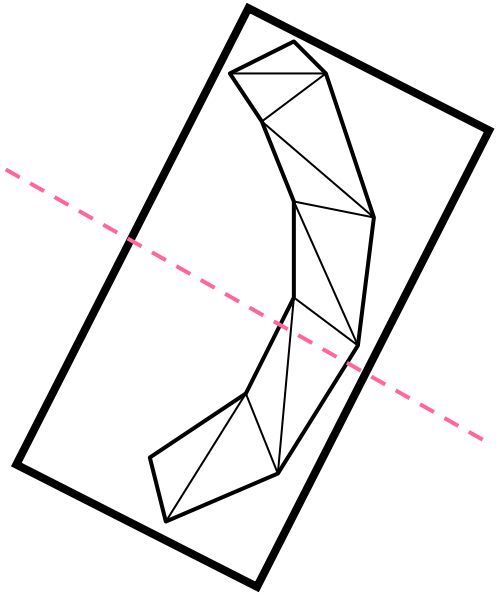
2. Bestimme ein OBB bezüglich dieser konvexen Hülle. Wie das funktioniert wird gleich gezeigt. Dies ist die Wurzel des Baumes.



1

Konstruktion eines OBB-Baums

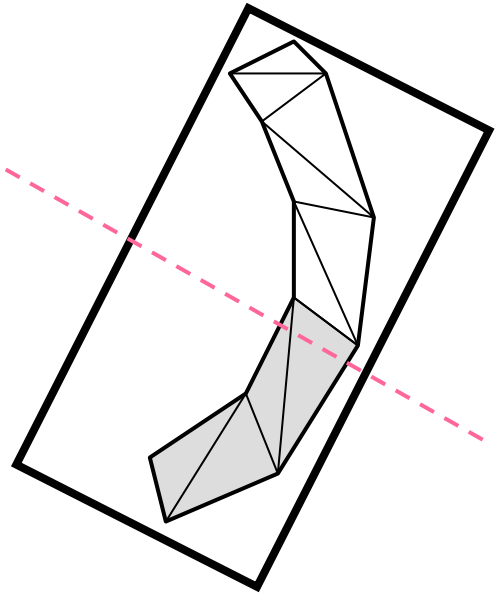
3. Halbiere das so entstanden OBB orthogonal zu seiner längsten Achse. Falls dies nicht geht (z.B. wenn alle Dreieck auf der Schnittebene liegen) so wähle die nächstgrößte Kante.



1

Konstruktion eines OBB-Baums

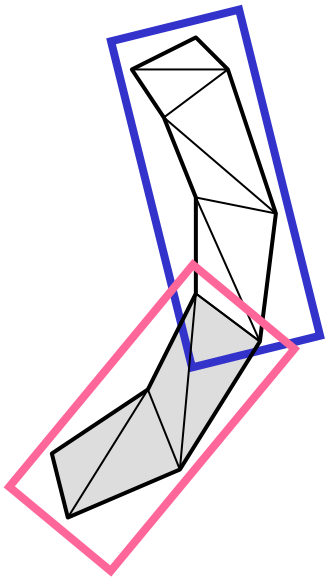
4. Ordne jeder Hälfte die Dreiecke zu, deren Schwerpunkte in der entsprechende Hälfte liegen.

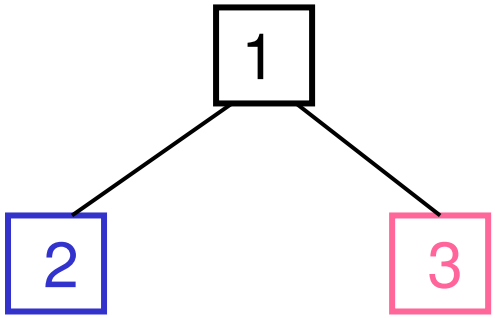


1

Konstruktion eines OBB-Baums

5. Verfahre mit den so entstandenen Gruppen von Dreiecken nun so wie in Schritt 1: Konvexe Hülle bilden, OBB berechnen, teilen,...
6. ... solange bis jede OBB nur noch ein Dreieck umschließt.

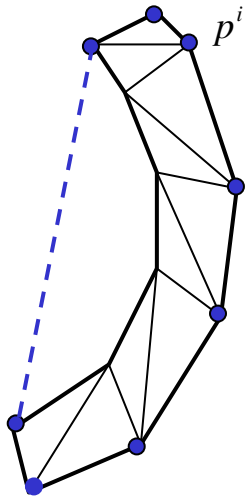




Bestimme OBB

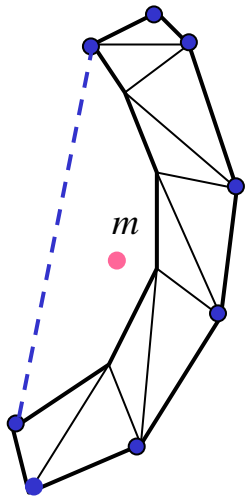
Betrachte nur die Punktwolke der Ecken der konvexen Hülle.

Eckpunkte $p^1, \ldots, p^n$ mit $p^i = (p_1^i, p_2^i, p_3^i)$



Bestimme OBB

Betrachte nur die Punktwolke der Ecken der konvexen Hülle.

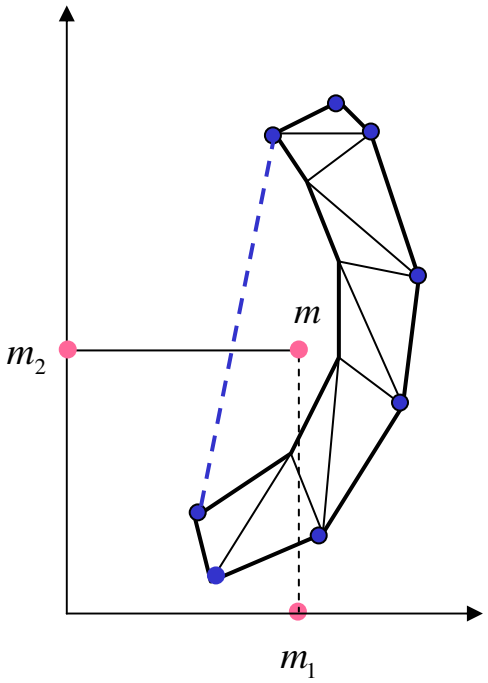


Eckpunkte $p^1, \ldots, p^n$ mit $p^i = (p_1^i, p_2^i, p_3^i)$

Schwerpunkt $m = \frac{1}{n} \sum_{k=1}^{n} p^k$

Bestimme OBB

Betrachte nur die Punktwolke der Ecken der konvexen Hülle.



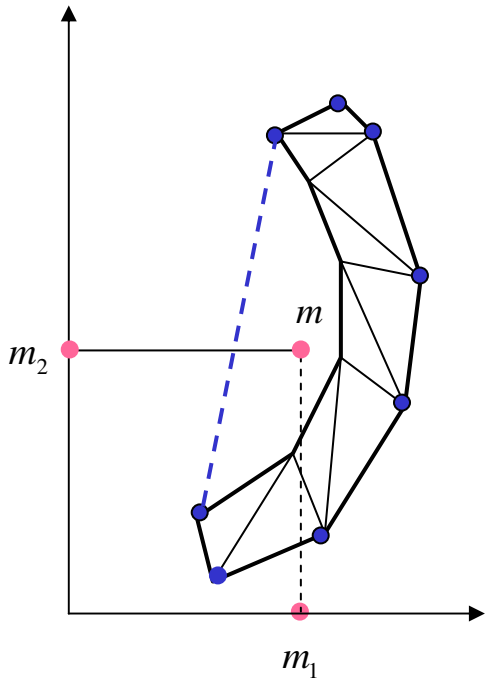
Eckpunkte $p^1, \ldots, p^n$ mit $p^i = (p_1^i, p_2^i, p_3^i)$

Schwerpunkt $m = \frac{1}{n} \sum_{k=1}^{n} p^k$

Schwerpunkt in einer Koordinantenrichtung: $m_i = \frac{1}{n} \sum_{k=1}^{n} p_i^k$

Bestimme OBB

Betrachte nur die Punktwolke der Ecken der konvexen Hülle.



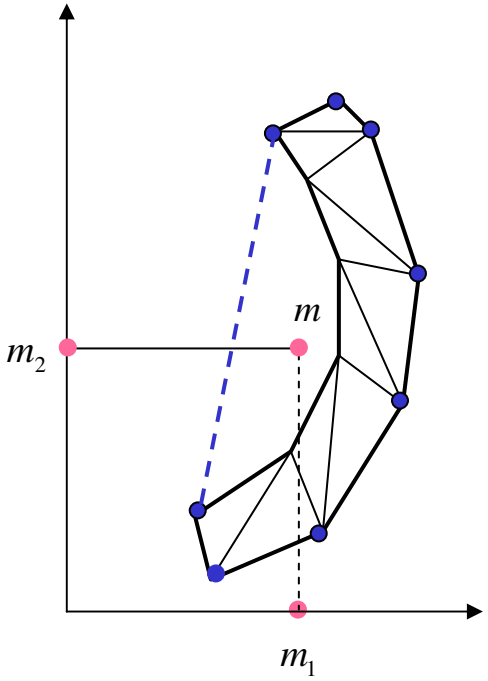
Eckpunkte $p^1, \ldots, p^n$ mit $p^i = (p_1^i, p_2^i, p_3^i)$

Schwerpunkt $m = \frac{1}{n} \sum_{k=1}^{n} p^k$

Schwerpunkt in einer Koordinantenrichtung: $m_i = \frac{1}{n} \sum_{k=1}^{n} p_i^k$

Kovarianzmatrix: $C_{ij} = \frac{1}{n} \sum_{k=1}^{n} p_i^k p_j^k - m_i m_j$

Bestimme OBB



$$C = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix}$$ ist symmetrisch

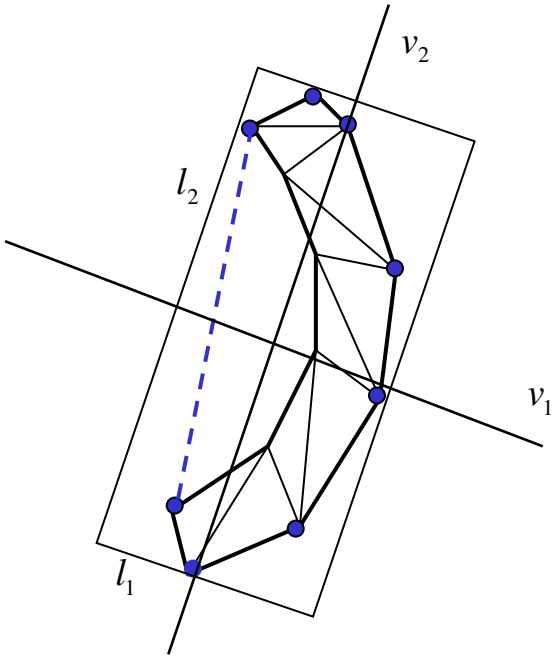
C hat drei reelle Eigenwerte r_1, r_2, r_3

mit zugehörigen Eigenvektoren

v^1, v^2, v^3 und $\|v^1\| = \|v^2\| = \|v^3\| = 1$

Bestimme OBB

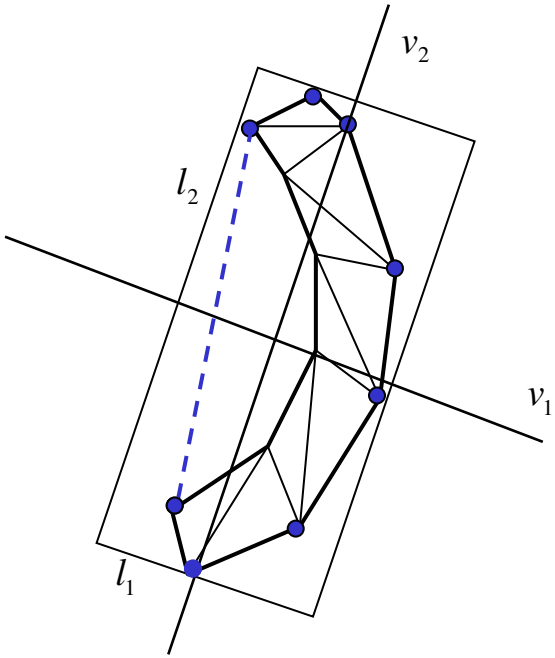
Definiere OBB folgendermaßen:



- Die Seitenflächen des neuen OBBs sind parallel zu den Eigenvektoren v_1, v_2, v_3 .
- Die Länge der Kanten des OBBS wird durch Projektion der Punkte $p^1, \ldots, p^n$ auf die Achsen des OBBs bestimmt:

$$l_i = \max_k(\langle v_i, p^k \rangle) - \min_k(\langle v_i, p^k \rangle)$$

Bestimme OBB



- Diese Technik kommt aus der Statistik und wird auch **Principal Component Analysis (PCA)** genannt. Mit Hilfe der Kovarianz-Matrix wird eine Wahrscheinlich-keitsverteilung der Punkte berechnet.
- Man kann statt der Eckpunkte auch die Kanten oder Flächen des konvexen Polyeders benutzen, um mit Hilfe einer Kovarianzmatrix ein OOB zu konstruieren. Dies sind i.Allg. bessere Näherungen.

Kosten

- Berechnung der konvexen Hülle: $O(n \log n)$
- Erzeugung des Binärbaumes: $O(\log n)$
- Insgesamt: $O(n \log^2 n)$.

Literatur / Software

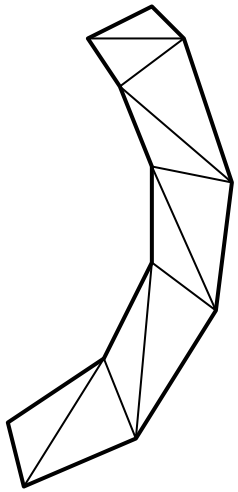
- S.Gottschalk; Collisions detection using Oriented Bounding Boxes, Dissertation, University of Carolina, Chapel Hill, 2000
- Softwarepaket RAPID (Robust and Accurate Polygon Interface Detection), freeware
- http://www.cs.unc.edu/~geom/OBB/OBBT.html

2.3.2 k-DOP- Bäume

- Top Down Ansatz
- Binärbaum enthält
  - ein Dreieck für statische Modelle
  - mehrere Dreiecke für dynamische Szenarien

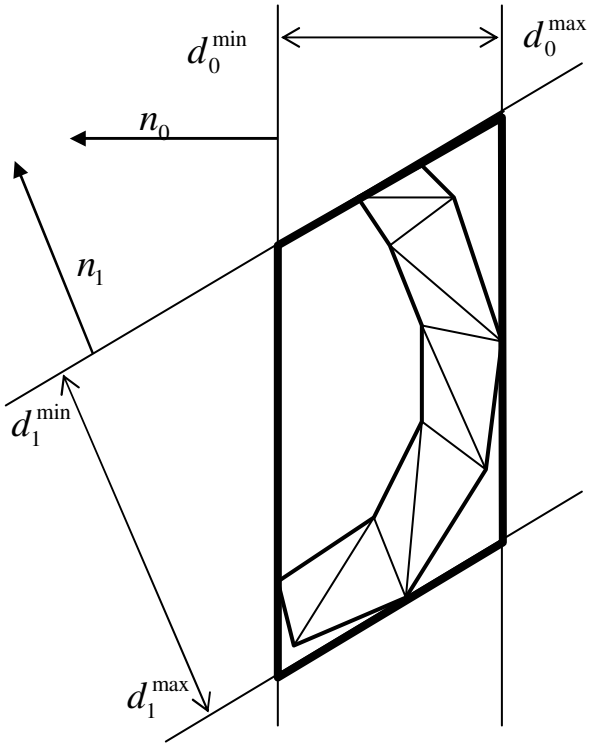
Konstruktion k-DOP Baum

1. Gegeben: Eine Menge von Dreiecken



Konstruktion k-DOP Baum

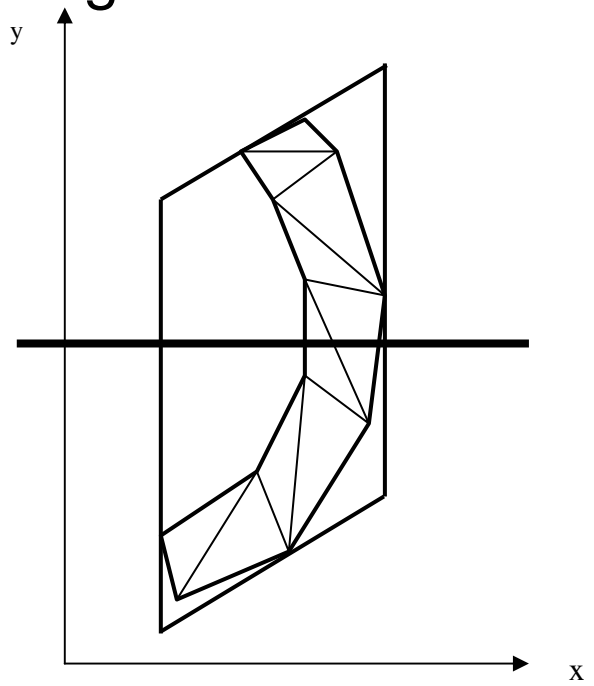
2. Bilde ein k-DOP, indem die Eckenpunkte der Dreiecke auf die Normalen projiziert werden. Die Extremwerte auf den Normalen definieren zusammen mit den Normalen das erste k-DOP.



1

Konstruktion k-DOP Baum

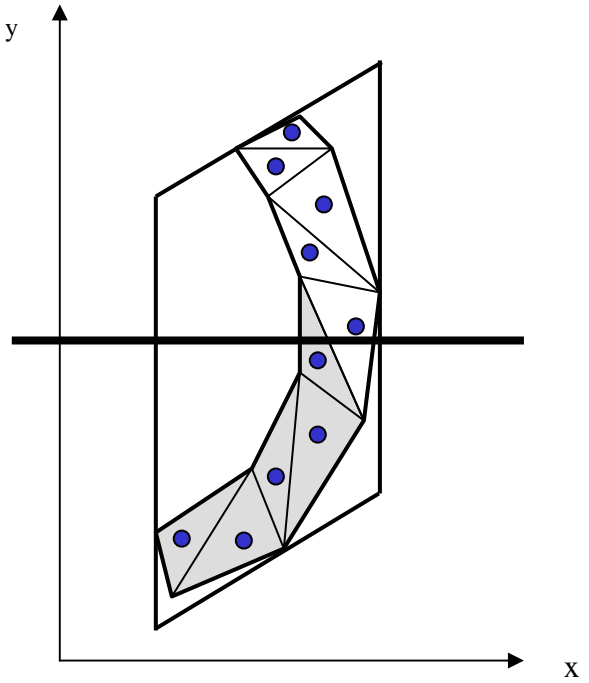
3. Wähle eine Ebene parallel zu einer Koordinatenachse aus. Dabei sollen die Kinder des momentanen k-DOPs möglichst minimales Volumen haben. Wie die Koordinatenachse ausgewählt wird, wird gleich erzählt.



1

Konstruktion k-DOP Baum

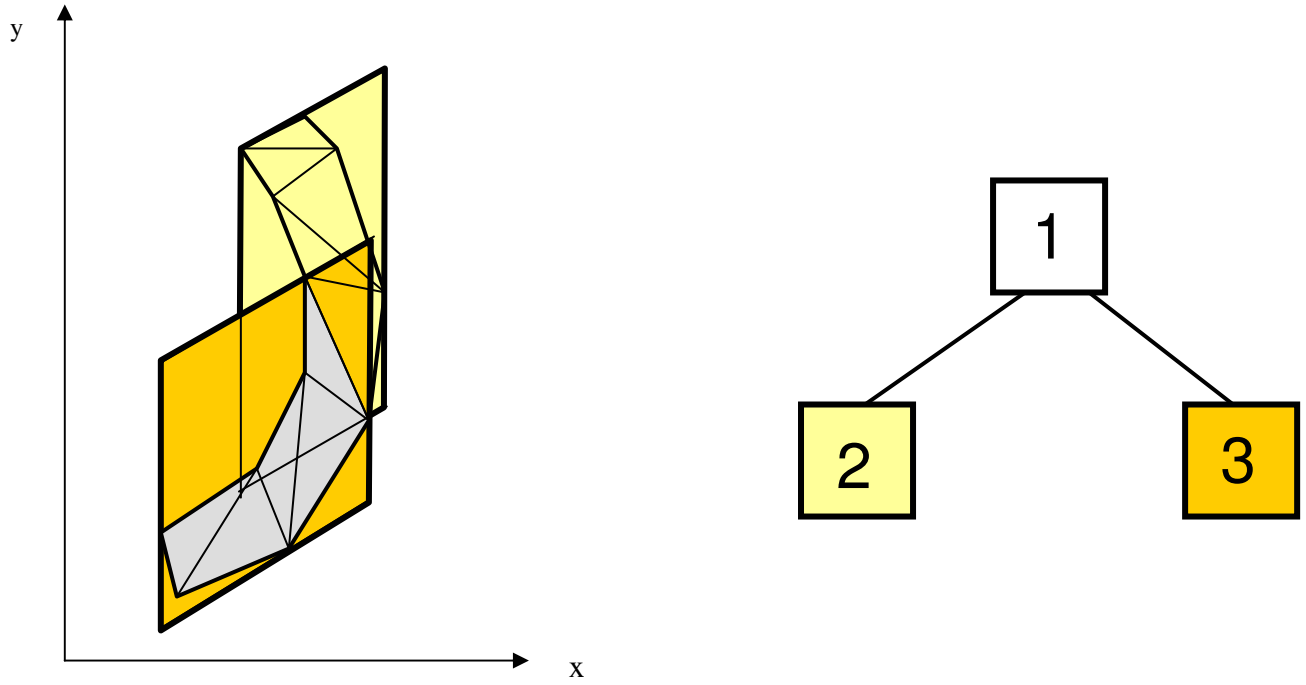
4. Ordne jedes Dreieck einem Halbraum zu. Ein Dreieck wird demjenigen Halbraum zugeordnet, in dem der Schwerpunkt des Dreiecks liegt.



1

Konstruktion k-DOP Baum

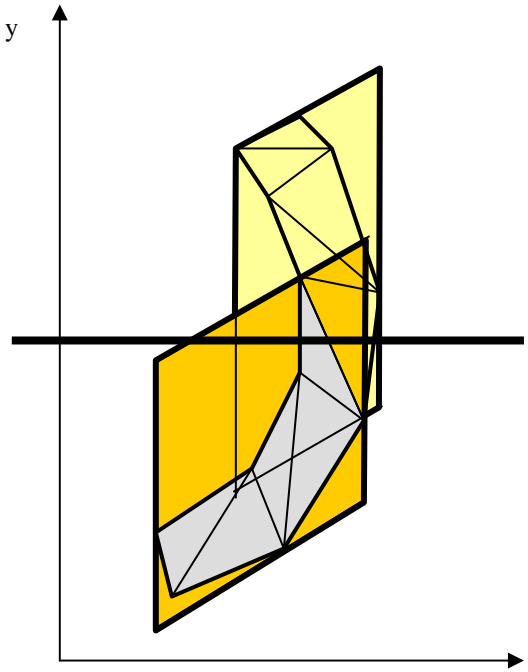
5. Konstruiere wie in Schritt 1 zu den so entstandenen zwei Dreieckmengen wieder ein k-DOP, usw.



Auswahl der Schnittebene

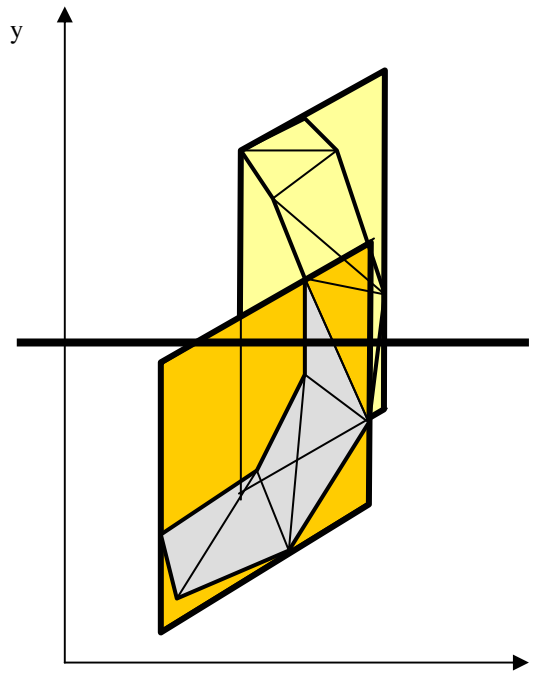
Wähle Schnittebene orthogonal zu einer Koordinaten-achse x,y oder z.

- *Min Sum*: Wählt die Achse, so dass die Summe der Volumen minimiert wird.



Auswahl der Schnittebene

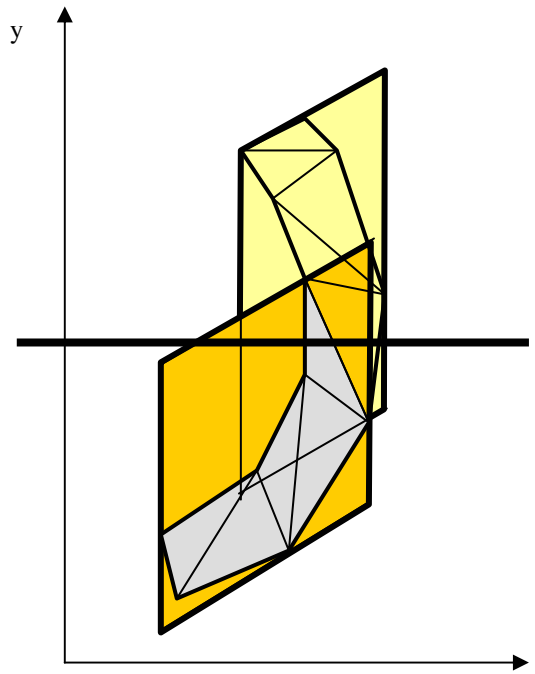
Wähle Schnittebene orthogonal zu einer Koordinaten-achse x,y oder z.



- *Min Sum*: Wählt die Achse, so dass die Summe der Volumen minimiert wird
- *Min Max*: Wählt die Achse, die die größeren der beiden Volumen minimiert.

Auswahl der Schnittebene

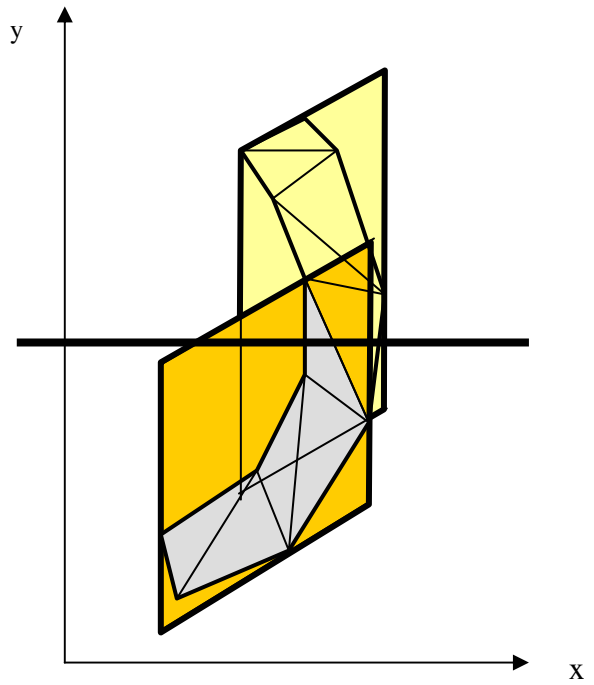
Wähle Schnittebene orthogonal zu einer Koordinaten-achse x,y oder z.



- *Min Sum*: Wählt die Achse, so dass die Summe der Volumen minimiert wird
- *Min Max*: Wählt die Achse, die die größeren der beiden Volumen minimiert.
- *Längste Seite*: Wählt die Achse entlang der das BV am längsten ist.

Auswahl der Schnittebene

Wähle Schnittebene orthogonal zu einer Koordinaten-achse x,y oder z.



- *Min Sum*: Wählt die Achse, so dass die Summe der Volumen minimiert wird
- *Min Max*: Wählt die Achse, die die größeren der beiden Volumen minimiert.
- *Längste Seite*: Wählt die Achse entlang der das BV am längsten ist.
- *Splatter*: Berechnet die Varianz der Projektion der Dreiecks-Schwerpunkte auf jede Achse, und wählt diejenige mit der größten Varianz.

Kosten

- Je größer das k, desto weniger Überlappungstests, aber umso höherer Aufwand für die Tests.
- Hohe Kosten bei dynamischen Szenarien.
- Wahl der Normalen beeinflusst numerischen Aufwand.

Literatur / Software

- James Thomas Klosovski,“Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments“,Dissertation, State Universitiy New York at Stony Brook, 1998

2.3.3 Vergleich der Verfahren

- Welche Verfahren am besten geeignet sind, ist von der Simulationsumgebung abhängig. Allgemeine Aussagen lassen sich nur schwer treffen.
- Für dynamische Szenarien wurden bessere Ergebnisse mit OBBs erzielt, für statische mit k-DOPs.
- OOB-Bäume haben gute Performance, falls OBBs fast parallel und nahe beieinander.
- Überlappungstest bei k-DOPs ist schneller als bei OBBs.
- Weitere Vergleiche der Verfahren sind in den Dissertationen von Gottschalk und Koslowski zu finden.

2.4 Zerlegung des Raumes

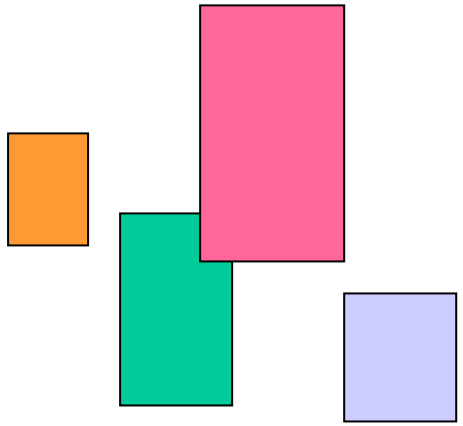
- Verfahren zum groben Aussortieren nichtkollidierender Objekte.
- Nutzt räumliche Kohärenz aus.
- Haben Probleme bei Speicherbedarf und dynamischen Szenen.

Zerlegung in Einheitszellen

- Zerlege Raum in Einheitszellen und ordne jeder Zelle die Objekte zu, die sie schneiden.
- Gut, falls alle Objekte ungefähr gleich groß sind.
- Gut, falls nur wenige Objekte vorhanden sind.

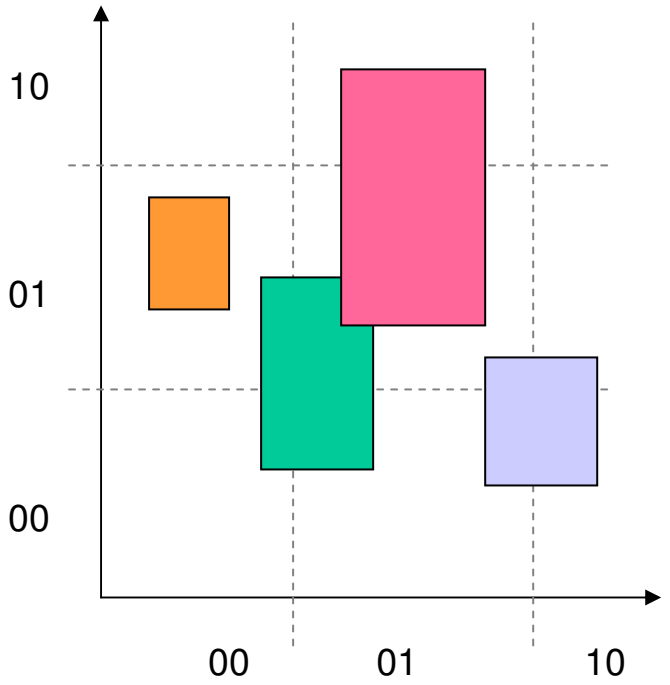
Räumliches Hashing

1. Zerlege Raum in Einheitszellen, aber speichere die Zellen nicht ab.



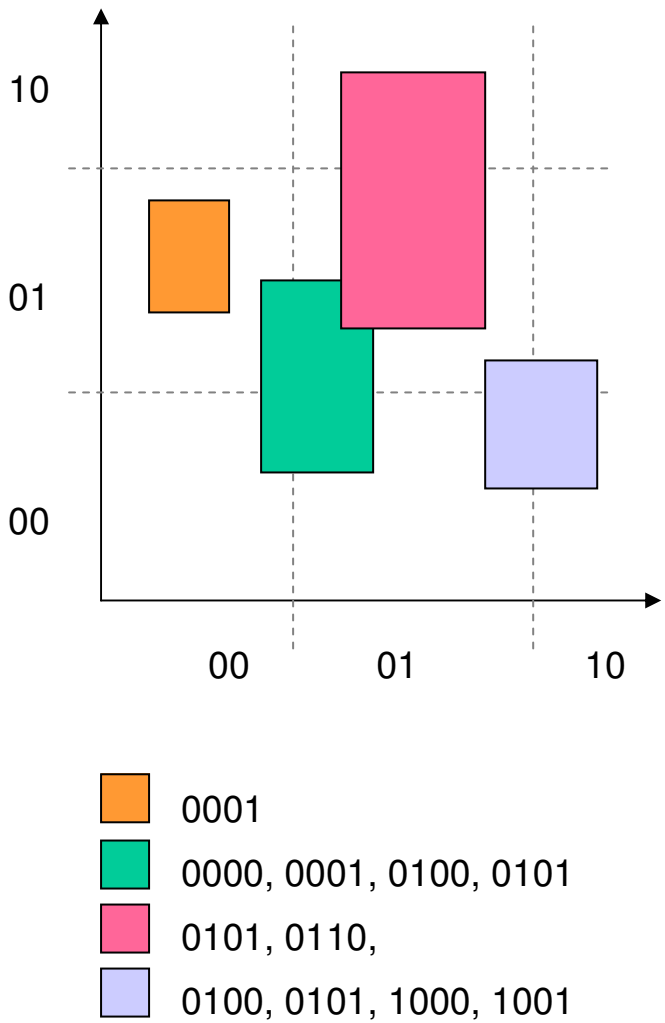
Räumliches Hashing

1. Zerlege Raum in Einheitszellen, aber speichere die Zellen nicht ab.



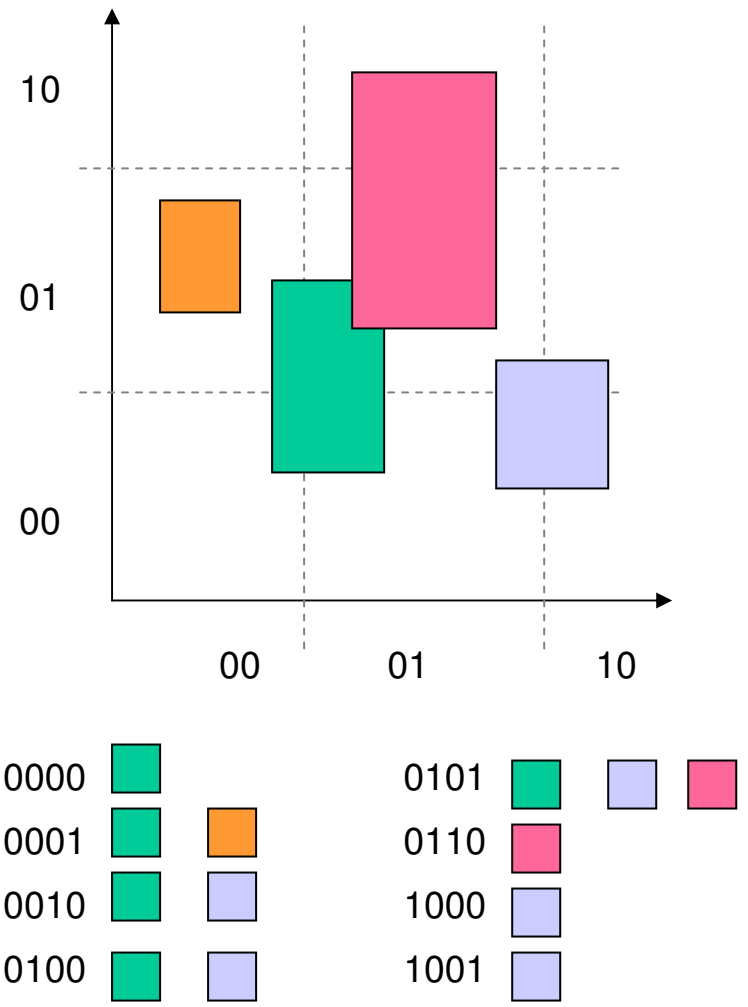
Räumliches Hashing

1. Zerlege Raum in Einheitszellen, aber speichere die Zellen nicht ab.
2. Bestimme die Indizes der Zellen, mit denen ein Objekt überlappt.
3. Wandle die Indizes in Hash-Keys um.



Räumliches Hashing

1. Zerlege Raum in Einheitszellen, aber speichere die Zellen nicht ab.
2. Bestimme die Indizes der Zellen, mit denen ein Objekt überlappt.
3. Wandle die Indizes in Hash-Keys um.
4. Füge das Objekt in die Hash-Tabelle ein.
5. Mehrere Einträge zu einem Hash-Key indizieren eine Überschneidung.

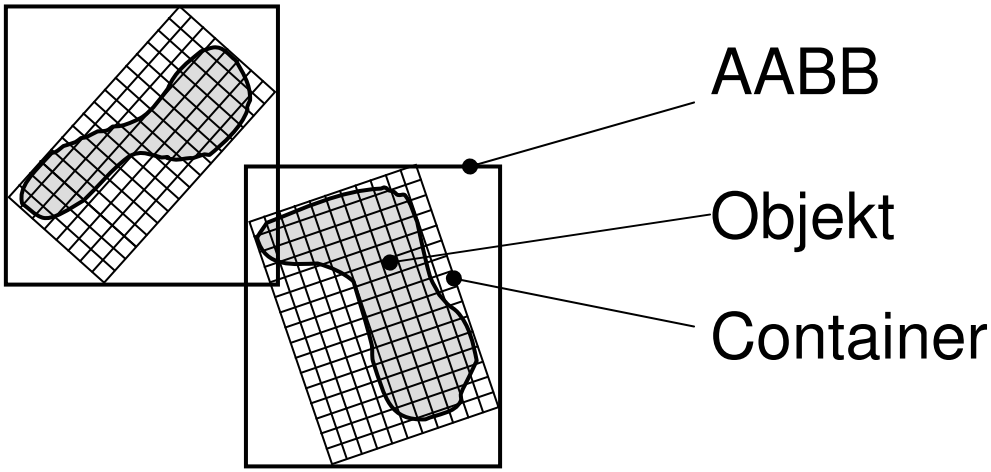


Octree Subdivision

- Statt den Raum in gleichförmige Zellen zu zerlegen: Zerlege Szene in einen Octree.
- Dies spart die a priori Wahl einer Zellengröße.
- Aber: Umordnung des Octrees nötig, falls Objekte sich bewegen, was sehr aufwendig sein kann.

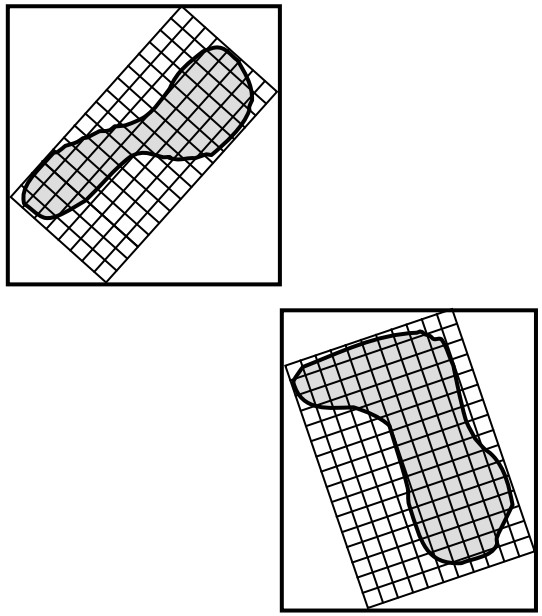
2.5 Lokale / Objekt-Zerlegung

- Nachteil der räumlichen Zerlegung einer Szene: Großer Speicherbedarf, unflexibel gegenüber dynamischen Szenen.
- Idee: Statt den ganzen Raum zu zerlegen, zerlege statt dessen nur die Objekte im Raum.
- Jedes Objekt erhält seinen eigenen voxelisierten Container, der mit dem Objekt transformiert wird.
- Jeder Container ist wieder in einer AABB enthalten.

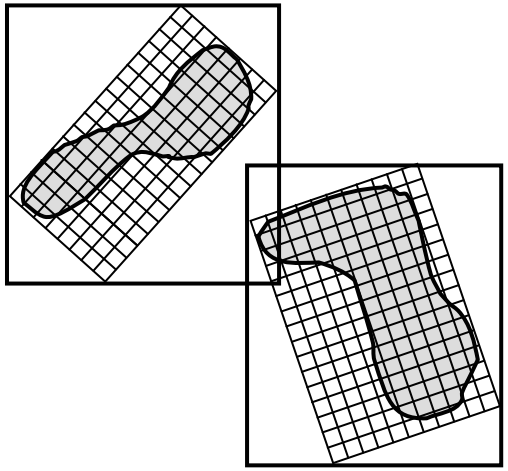


Algorithmus

1. Teste, ob AABBs der beiden Objekte A und B überlappen. Falls nicht, so sind wir fertig. Andernfalls fahre fort.



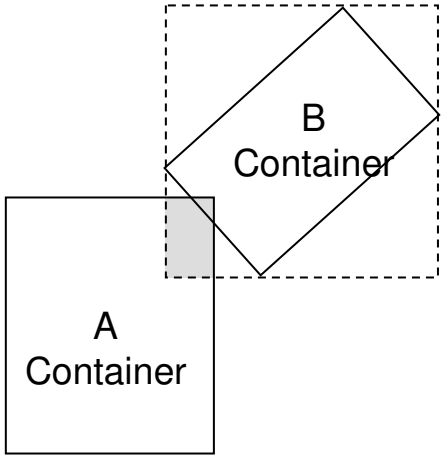
fertig



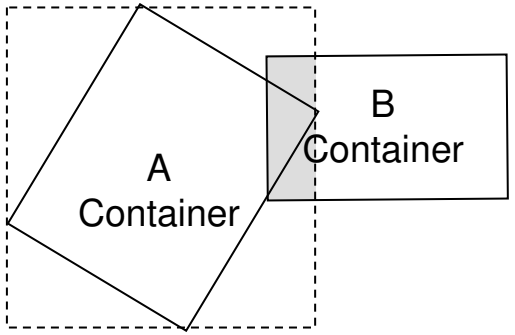
weitermachen

Algorithmus

2. Berechne die AABB von Objekt A im Koordinatensystem von Objekt B.
3. Berechne die AABB von Objekt B im Koordinatensystem von Objekt A.



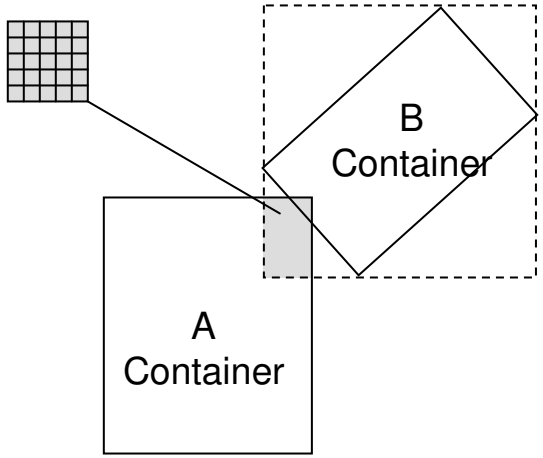
Lokaler Raum von A



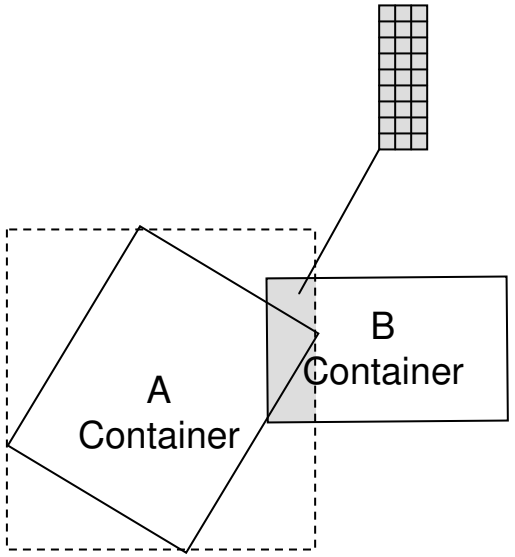
Lokaler Raum von B

Algorithmus

4. Finde die Voxels in A, die mit Voxels aus B überlappen könnten (und umgekehrt).



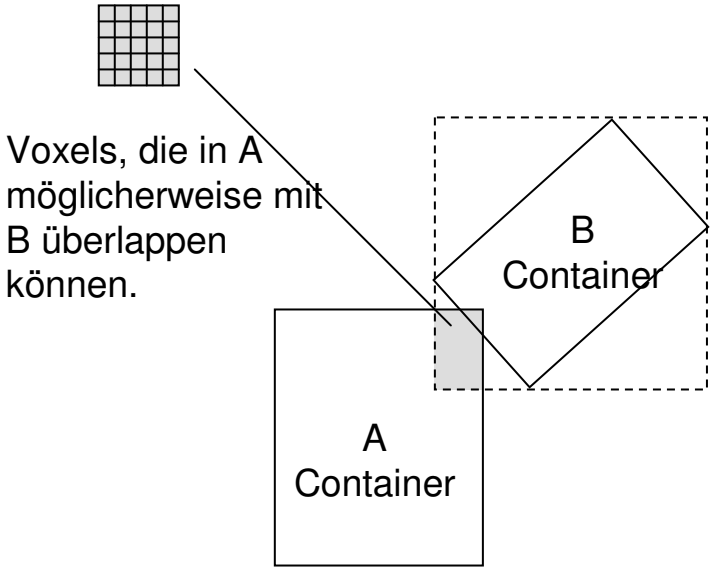
Lokaler Raum von A



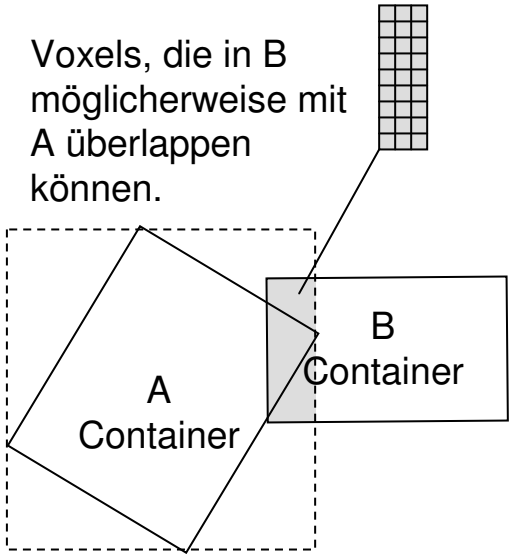
Lokaler Raum von B

Algorithmus

5. Teste nun jedes Voxel aus der Ergebnismenge von A mit jedem Voxel aus der Ergebnismenge von B auf Überschneidung.



Lokaler Raum von A



Lokaler Raum von B