AABB-BÄUME ALS GRUNDLAGE EFFIZIENTER ALGORITHMEN FÜR OPERATOREN EINER RÄUMLICHEN ANFRAGESPRACHE

Stefanie Schraufstetter[1] und André Borrmann[1]

[1] Lehrstuhl für Bauinformatik, Technische Universität München,

{schraufstetter | borrmann}@bv.tu-muenchen.de

Kurzfassung: *Dieses Paper beschäftigt sich mit der Entwicklung einer räumlichen Anfragesprache für 3D- Gebäudemodelle, welche metrische, topologische und direktionale Operatoren umfasst und auf Basis einer objekt-relationalen Datenbank realisiert werden soll. Es stellt einen Ansatz zur schnellen Auswertung des metrischen Operators* distance *sowie der topologischen Beziehungen* contain*,* within*,* overlap *und* disjoint *vor. Die implementierten Algorithmen sind nicht auf konvexe Objekte beschränkt und liefern exakte Ergebnisse. Grundlegende Idee ist die hierarchische Darstellung der facettierten geometrischen Objekte in Forum von AABB-Bäumen. Auf dieser Datenstruktur lassen sich die Operatoren mittels einer divide-and-conquer-Strategie effizient auswerten, indem die für das Ergebnis irrelevanten Facetten eines Objekts frühzeitig durch kostengünstige Bounding-Box-Tests ausgeschlossen werden.*

1 Einführung

In einem aktuellen Forschungsprojekt[1] entwickelt unsere Forschungsgruppe eine räumliche Anfragesprache für 3D-Gebäudemodelle. Ziel ist die Weiterentwicklung der geometrisch-topologischen Analysefähigkeit von bisher vorhandener Software im Bereich der computergestützten Bauplanung, um beispielsweise das effiziente Herausfiltern bestimmter Gebäudeinformationen oder die Partitionierung von Bauwerksmodellen nach gewissen Kriterien zu ermöglichen.

Mögliche Anfragen sind zum Beispiel gegeben durch:

- „Wie viele Heizkörper befinden sich in Raum 34?“
- „Gibt es Objekte im Umkreis von 5 Metern?“

[1] gefördert von der Deutschen Forschungsgemeinschaft (DFG)

- „Welche Stützen berühren Decke 2?“

Die Anfragesprache soll in eine objekt-relationale Datenbank, die sämtliche Bauteile enthält, integriert werden. Die Einbindung in eine objekt-relationale Datenbank ermöglicht neben der Nutzung herkömmlicher relationaler Tabellen die Definition von Datentypen samt zugehörigen Methoden. Eine entsprechende abstrakte Algebra mit räumlichen Typen und darauf anwendbaren Operatoren wurde für die räumliche Anfragesprache bereits in [4] definiert. Nach dieser lassen sich die Operatoren wie folgt gliedern:

- metrische Operatoren: *distance*, *closerThan*, *fartherThan*, ...
- topologische Operatoren: *contain*, *within*, *disjoint*, *overlap*, ...
- direktionale Operatoren: *below*, *above*, *northOf*, *southOf*, ...

Die Implementierung erfolgt mittels eines Oracle-Datenbanksystems auf Basis von SQL:1999, der jüngsten Version des ISO-Standards. Die Geometrien der Bauteile werden explizit unter Nutzung der Typdefinitionen als facettierte BRep-Objekte in der Datenbank abgelegt. Über sog. *stored procedures* lassen sich die Operatoren als Java- oder C-Methoden in die Datenbank integrieren und ggf. den entsprechenden Datentypen zuweisen.

Gewöhnlich sind zahlreiche Operatorenauswertungen zur Berechnung des Ergebnisses einer Datenbankanfrage erforderlich, so dass die Anfragezeit maßgeblich von den implementierten Algorithmen der räumlichen Anfragesprache abhängt. Effiziente Algorithmen sind deshalb für die schnelle Bearbeitung einer Datenbankanfrage von großer Bedeutung. Zur Beschleunigung eignet sich eine hierarchische Darstellung der geometrischen Objekte. Auf dieser Datenstruktur lassen sich die Operatoren mittels einer *divide-and-conquer*-Strategie effizient auswerten.

In diesem Paper konzentrieren wir uns auf metrische und topologische Operatoren und beschreiben, wie sich die Operatoren *distance* sowie *contain*, *within*, *overlap* und *disjoint* unter Verwendung einer hierarchischen Datenstruktur in Form von sogenannten AABB-Bäumen implementieren lassen. Während viele Ansätze aus der Literatur Einschränkungen unterliegen, sind die in diesem Paper vorgestellten Algorithmen nicht auf konvexe Objekte beschränkt und liefern exakte Ergebnisse.

2 AABB-Bäume

Baumdatenstrukturen sind sehr beliebt, da sie *divide-and-conquer*-Strategien ermöglichen, die im Gegensatz herkömmlichen Problemlösungsstrategien den Aufwand erheblich reduzieren können. In dem hier vorgestellten Ansatz wurden AABB-Bäume als Datenstruktur gewählt, um eine hierarchische Objektrepräsentation zu erhalten. AABB-

Bäume [3] sind Binärbäume, die rekursiv durch Aufteilung des Raumes jeweils entlang einer Koordinatenrichtung gebildet werden (vgl. Abb. 1:).

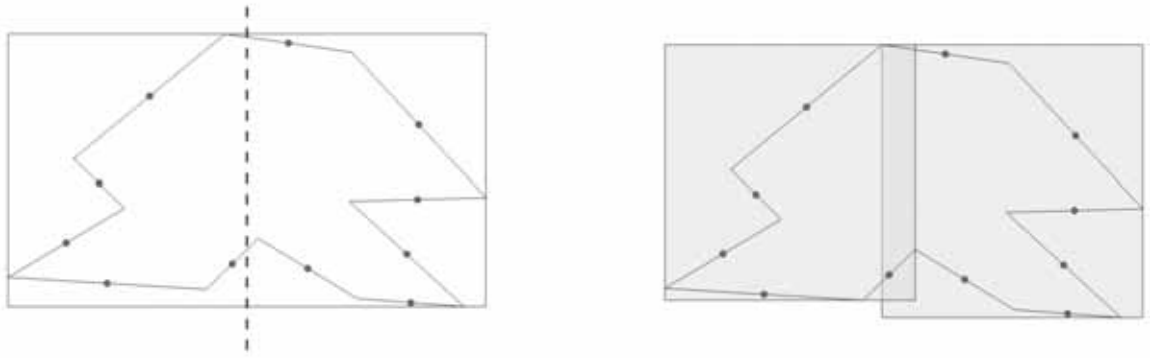

Abb. 1: Konstruktion des AABB-Baums: Aufspaltung in zwei Teilgeometrien in Richtung der längsten Achse

Jedem Knoten des Baumes ist ein Hüllvolumen in Forum eines Quaders (axis-aligned bounding box, AABB) zugewiesen, das die beiden Hüllvolumen seiner Kinder umschließt. Die Blätter des Baumes enthalten eine oder mehrere Facetten des Geometrieobjekts sowie ein Hüllquader, das diese Facetten umschließt. Ein Beispiel eines facettierten 3D-Objekts mitsamt aller Hüllvolumen des zugehörigen AABB-Baumes zeigt Abbildung Abb. 2:.

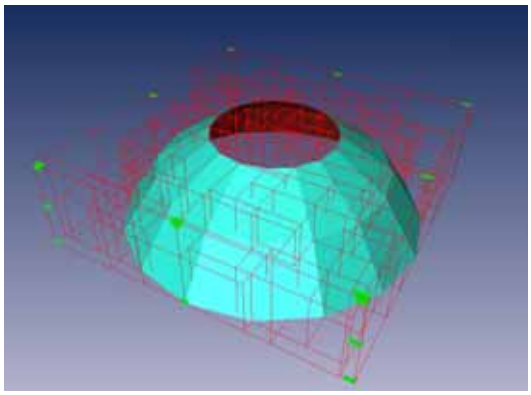

Abb. 2: Hierarchische Geometrierepräsentation mittels AABB-Bäumen in 3D.

Die Wahl der AABB-Baumdatenstruktur begründet sich in ihrer einfachen Struktur, die zum einen eine schnelle Generierung des Baumes und zum anderen extrem simple und somit kostengünstige Tests auf den AABBs, wie z.B. Schnitttests, ermöglicht.

2.1 Konstruktion von AABB-Bäumen

Die Konstruktion des AABB-Baumes zu einem gegebenen BRep-Modell erfolgt *top-down*, d.h. von oben nach unten: Zuerst wird die kleinste AABB, die alle Facetten des geometrischen Körpers enthält, bestimmt. Anschließend erfolgt die Splittung der Menge an Facetten gemäß einer Partitionierungsregel in zwei Teilmengen, im Folgenden als

linke bzw. *rechte Teilmenge* bezeichnet. Eine häufig verwendete Partitionierungsregel betrachtet die Projektion aller Objekte auf die längste Achse der AABB. Ist der Mittelpunkt der Projektion einer Facette kleiner als der Mittelpunkt der Projektion der AABB bezüglich dieser Koordinatenrichtung, so wird die Facette der linken Teilmenge zugeordnet, andernfalls der rechten. Nachdem alle Facetten klassifiziert wurden, erfolgt ein rekursives Vorgehen mit der linken und der rechten Teilmenge. Ein Abbruch der Rekursion findet statt, sobald die betrachtete AABB nur noch eine einzige Facette enthält oder – sofern vorgegeben – die maximale Baumtiefe erreicht ist. Die entsprechende AABB stellt dann im Verfeinerungsbaum ein Blatt dar.

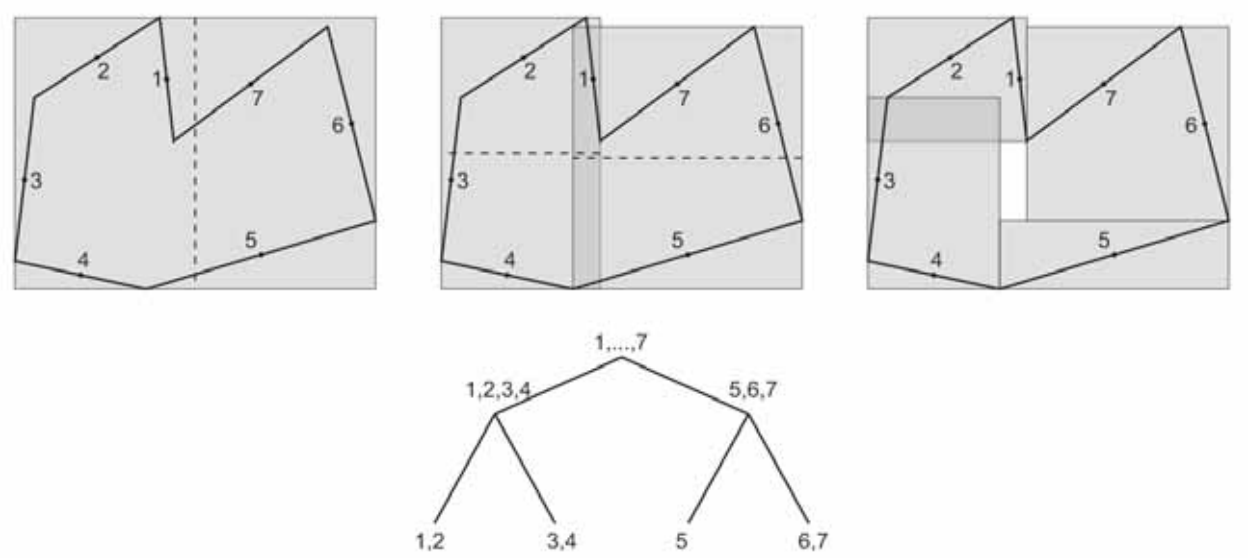

Abb. 3: Generierung eines AABB-Baums am Beispiel eines Polygons in 2D. Der Raum wird auf jeder Verfeinerungsstufe jeweils entlang der längsten Achse in zwei Hälften geteilt. Dabei werden die Kanten des Polygons über ihren Mittelpunkt einer der Hälften zugeordnet. Anschließend wird um alle zu einer Hälfte gehörenden Kanten eine Bounding Box gelegt, für die die Aufteilung rekursiv wiederholt wird.

Die Vorgabe einer maximalen Tiefe erweist sich als sinnvoll, um stark unbalancierte Bäume zu verhindern. Sie rechtfertigt sich dadurch, dass ab einem gewissen Level die Hüllvolumina ausreichend klein sind. Eine weitere Aufspaltung, d.h. Aufteilung des Körpers wäre dann nur noch von geringem Nutzen für die auf den AABB-Baum angewendeten Algorithmen, da alle Facetten bereits sehr dicht liegen und sich eine Betrachtung der einzelnen Facetten ohne Bounding Box als geeigneter erweist.

Eine alternative Partitionierungsregel ordnet die projizierten Facetten bzgl. ihres Mittelpunktes in aufsteigender Reihenfolge und teilt diese anschließend, unter Berücksichtigung ihrer Ordnung, in zwei gleich große Teilmengen. Im Gegensatz zur erst genannten Regel liefert diese Strategie einen balancierten AABB-Baum. Allerdings nimmt die ma-

ximale Größe der Hüllvolumina eines Levels mit zunehmender Baumtiefe langsamer ab, da die Streuung der Seitenlängen der Bounding Boxen stärker ist.

3 Algorithmen auf AABB-Bäumen

Die Grundstruktur der Algorithmen ist sowohl für metrische als auch für topologische Operatoren identisch. Erster Schritt ist stets die Generierung der zugehörigen AABB-Bäume der beiden facettierten Geometrieobjekte *A* und *B*, auf welche der Operator angewendet werden soll.

Anschließend werden ähnlich wie in [5] in Form einer Breitensuche auf jedem Level AABB-Paare gebildet, jeweils bestehend aus einer AABB des Objekts *A* sowie einer des Objekts *B*. Abhängig vom Prädikat lassen sich gewisse AABB-Paare eines bestimmten Levels, die nicht zum Ergebnis beitragen und damit für die Operatorenauswertung irrelevant sind, ausschließen (*operatorspezifischer Test 1*).

Alle anderen AABB-Paare müssen für eine korrekte Aussage weiter verfeinert werden. Der Algorithmus wird auf dem nächsten Level rekursiv fortgesetzt.

Stellen beide Bounding Boxen des AABB-Paares Blätter im jeweiligen AABB-Baum dar, so ist keine weitere Verfeinerung möglich. In diesem Fall erfolgt eine Paarbildung zwischen allen in den beiden AABBs enthaltenen Facetten. Auf diesen Paaren werden abhängig vom Prädikat exakte Tests wie zum Beispiel Schnitttests durchgeführt (*operatorspezifischer Test 2*).

So reduzieren sich aufwändige Berechnungen auf der expliziten Geometrie auf ein Minimum. Auf Basis der daraus resultierenden Ergebnisse lässt sich schließlich das Prädikat auswerten.

Im Fall topologischer Prädikate ist es zudem sinnvoll, zu Beginn noch einen Test mit den Hüllvolumina der beiden Objekte durchzuführen, um gegebenenfalls bereits zu diesem Zeitpunkt ohne weiteren Rechenaufwand ein Ergebnis zu erhalten. So folgt zum Beispiel aus disjunkten Bounding Boxen direkt, dass das Prädikat *disjoint(A,B)* erfüllt ist.

Die beiden operatorspezifischen Tests 1 und 2 werden im Folgenden detaillierter erläutert.

3.1 Algorithmus zur Abstandsberechnung

Zur Abstandsberechnung müssen zunächst jeweils der minimale sowie der maximale Abstand der beiden Quader eines AABB-Paares bestimmt werden. Innerhalb eines festen Levels schließt Test 1 alle AABB-Paare aus, deren minimaler Abstand größer als das Minimum aller maximalen Distanzen auf diesem Level ist. Denn zur Abstandsbe-

stimmung interessiert nur das AABB-Paar, welches das Facettenpaar mit minimalem Abstand enthält.

In Test 2, also bei Erreichen der Blätter bzw. der maximalen Baumtiefe, erfolgt die exakte Berechnung der Abstände aller im AABB-Paar enthaltenen Objekte. Hierfür eignet sich der GJK-Algorithmus [6]. Dieser Algorithmus von Gilbert, Johnson und Keerthi berechnet den Abstand zweier konvexer Objekte, in dem die Minkowski-Differenz gebildet und der Abstand des resultierenden Polytops zum Ursprung berechnet wird. Die Einschränkung auf konvexe Objekte stellt dabei kein Hindernis dar, da lediglich einzelne konvexe Facetten, die in der Regel aus einer Triangulierung resultieren, betrachtet werden. Auf Grund der geringen Anzahl an Eckpunkten fällt die Komplexität des GJK-Algorithmus von $O(N^2)$ im Gegensatz zur Anwendung auf den vollständigen Geometriekörper nicht ins Gewicht.

3.2 Algorithmus für topologische Operatoren

Eine schematische Darstellung des Algorithmus zur Bestimmung der topologischen Beziehung zweier Objekte zeigt Abb. 4:. Den ersten Teil des Algorithmus bildet der Schnitttest, welcher prüft, ob sich die beiden Objekte überlappen, und ähnlich dem im vorherigen Abschnitt beschriebenen Algorithmus für metrische Operatoren ist. Es sind lediglich sämtliche Distanzberechnungen der operatorspezifischen Tests 1 und 2 durch Schnitttests zu ersetzen. Ein Abstieg im AABB-Baum zur detaillierteren Betrachtung erfolgt immer dann, wenn sich zwei AABBs schneiden (Test 1). In den Blättern angekommen, werden wieder direkt alle Facettenpaare auf Schnitt [1] geprüft (Test 2). Im Fall eines Schnitts liegt das Prädikat *overlap(A,B)* vor.

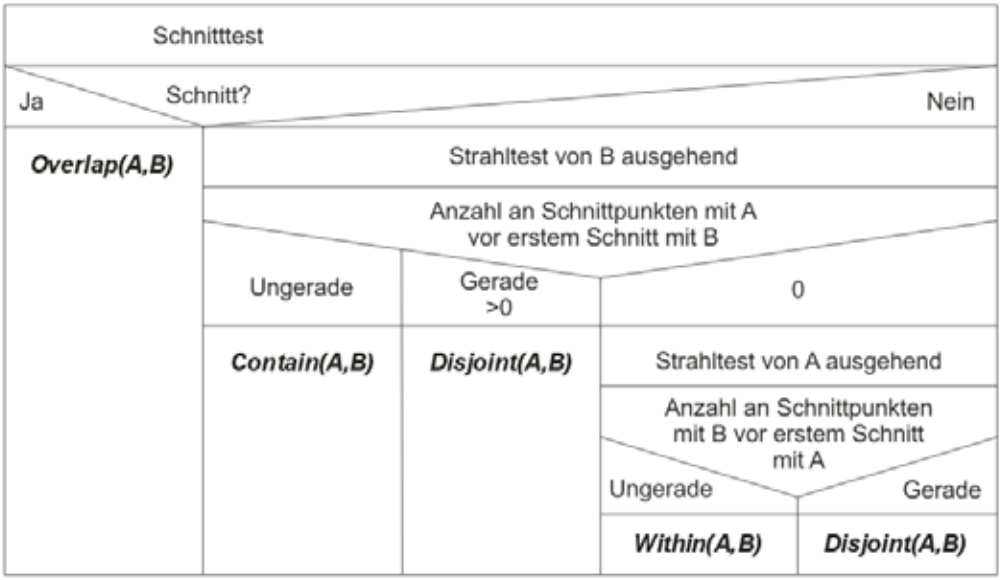

Abb. 4: Algorithmus für topologische Operatoren

Anderenfalls schneiden sich die beiden Objekte *A* und *B* nicht, sondern sind liegen ineinander oder sind disjunkt. Die genaue topologische Beziehung, d.h. welcher der drei Operatoren *contain*, *within* bzw. *disjoint* den Wert *true* zurückliefert, lässt sich anschließend mit Hilfe eines Suchstrahls anhand dessen Schnittpunkten mit den beiden Objekten bestimmen. Auch beim Strahltest ist die Implementierung als *divide-and-conquer*-Verfahren möglich, indem zunächst Schnitttests zwischen dem Strahl und den AABBs jeweils eines Objekts durchgeführt werden [8] und wiederum erst auf dem letzten Level ein Schnitttest Strahl-Facette erfolgt [7].

Das konkrete Vorgehen ist wie folgt (siehe auch Abb. 4): Zunächst werden für jedes der beiden Objekte die Schnittpunkte eines frei gewählten Strahls mit Ursprungspunkt auf Objekt *B* bestimmt und entlang der Richtung dieses Strahls sortiert. Durch die Wahl des Ursprungspunkts auf Objekt *B* ist garantiert, dass der Strahl mindestens ein Objekt trifft und nicht an beiden Objekten vorbeizielt. Nun betrachtet man die Anzahl an Schnittpunkten des Strahls mit Objekt *A*, die vor dem ersten Schnitt des Strahls mit Objekt *B* liegen (vgl. Abb. 5:). Ist diese ungerade, so liegt *B* in *A* und es gilt das Prädikat *contain(A,B)*. Im Fall einer positiven, geraden Anzahl an Schnitten sind *A* und *B* disjunkt, d.h. es gilt *disjoint(A,B).* Ist die Anzahl dagegen Null, so lässt sich keine topologische Beziehung folgern und es ist ein weiterer Strahltest – diesmal mit vertauschten Objekten *A* und *B*, d.h. ausgehend von Objekt *A* – erforderlich. Aus einer ungeraden Anzahl an Schnitten folgt dann das Prädikat *within(A,B)*, d.h. *A* liegt in *B*, bei Null oder einer geraden Anzahl folgt *disjoint(A,B).*

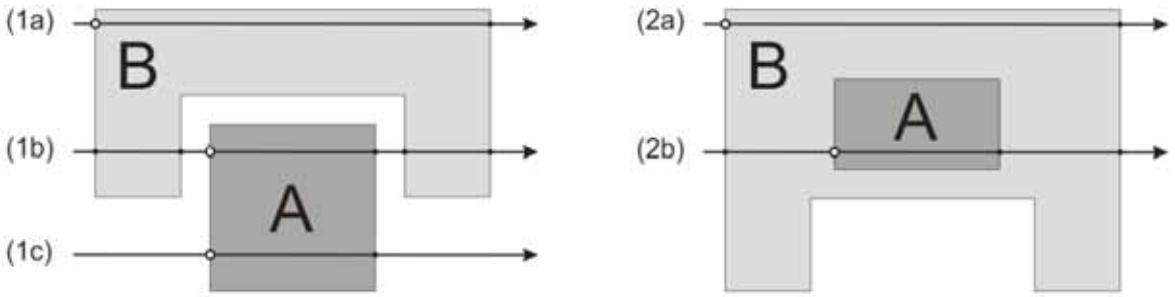

Abb. 5: Strahltest (der Ursprungspunkt des Strahls ist mit einem Kreis markiert): Der Strahl (1b) erkennt das Prädikat disjoint(A,B)*, der Strahl (2b) die Eigenschaft* within(A,B) *bzw.* contain(B,A)*. Ohne Aussage bleiben jedoch die Strahltests (1a), (1c) und (2a) mangels Schnitte mit dem zweiten Körper. Hier ist jeweils ein weiterer Strahltest vom zweiten Körper ausgehend erforderlich, z.B. in Form von (1b), (1a) bzw. (2b).*

4 Ausblick

In diesem Paper wurde beschrieben, wie sich geometrische Fragestellungen, die bei naiver Implementierung unter Nutzung von Standardalgorithmen einen enormen Re-

chenaufwand erfordern würden, mittels einer hierarchischen Geometrierepräsentation in Form von AABB-Bäumen und darauf basierender *divide-and-conquer*-Strategie auf einen geringeren Aufwand herunter brechen lassen. Zwar werden die Standardalgorithmen weiter benutzt, jedoch mit sehr einfachen Geometrien als Eingabeparameter, woraus eine geringere Komplexität resultiert.

Für die Zukunft ist zudem der Einsatz von Baumdatenstrukturen wie zum Beispiel R*-Bäume [2] bei der Indizierung innerhalb der Datenbank geplant. Die Integration dieser Baumdatenstruktur ermöglicht die Optimierung der Auswertung von Anfragen und verspricht somit eine weitere erhebliche Beschleunigung der Datenbank.

Literatur

[1] T. Akenine-Möller: A Fast Triangle-Triangle Intersection Test. *Journal of Graphical Tools*, 2(2), 25-30, 1997.

[2] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: H. Garcia-Molina, H. V. Jagadish (Hrsg.): *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, ACM Press, 322-331, 1990.

[3] G. van den Bergen: Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphical Tools*, 2(4), 1-13, 1997.

[4] A. Borrmann, C. van Treeck, E. Rank: "Towards a 3D Spatial Query Language for Building Information Models," *Proc. of Joint Int. Conf. for Computing and Decision Making in Civil and Building Engineering*, 2006.

[5] A. Borrmann, C. van Treeck: "Ein Algorithmus für die rekursive Abstandbestimmung Spacetree-codierter Rastergeometrien," 18. Forum Bauinformatik, Weimar, 2006.

[6] E. G. Gilbert, D.W. Johnson, S. S. Keerthi: A Fast Procedure for Computing the Distance between Complex Objects in Three-Dimensional Space*. IEEE Journal of Robotics and Automation*, 4(2), 193-203, 1988.

[7] T. Möller, B. Trumbore: Fast, Minimum Storage Ray-Triangle Intersection. *Journal of Graphical Tools*, 2(1), 21-28, 1997.

[8] A. Williams, S. Barrus, R. K. Morley, P. Shirley: An Efficient and Robust Ray-Box Intersection Algorithm, *Journal of Graphics Tools*, 10(1), 49-54, 2005.