Boolesche Operationen für die Visualisierung von IFC-Gebäudemodellen

Michael Theiler, Eike Tauscher, Jan Tulke, Thomas Riedel
Professur Informatik im Bauwesen, Bauhaus-Universität Weimar
michael.theiler@uni-weimar.de

Kurzfassung: Die Planung von komplexen Bauwerken erfolgt zunehmend mit Planungswerkzeugen, die den Export von Bauwerksinformationen im STEP-Format auf Grundlage der IFC (Industry Foundation Classes) erlauben. Durch die Verfügbarkeit dieser Schnittstelle ist es möglich, Bauwerksinformationen für die weiterführende Verarbeitung zu verwenden. Zur Visualisierung der geometrischen Daten stehen innerhalb der IFC verschiedene geometrische Modelle für die Darstellung von Bauteilen zur Verfügung. Unter anderem werden für das „Ausschneiden“ von Öffnungen aus Bauteilen (z.B. für Fenster und Türen) geometrische boolesche Operationen benötigt.

Gegenstand des Beitrags ist die Vorstellung eines Algorithmus zur Berechnung von booleschen Operationen auf Basis eines triangulierten B-Rep (Boundary Representation) Modells nach HUBBARD (1990). Da innerhalb von IFC-Gebäudemodellen Bauteile oft das Resultat mehrerer boolescher Operationen sind (z.B. um mehrere Fensteröffnungen von einer gegebenen Wand abzuziehen), wurde der Algorithmus von Hubbard angepasst, sodass mehrere boolesche Operationen gleichzeitig berechnet werden können. Durch diese Optimierung wird eine deutliche Reduzierung der benötigten Berechnungen und somit der Rechenzeit erreicht.

1 Einleitung

Innerhalb verschiedener Forschungsprojekte wurde an der Bauhaus-Universität Weimar ein 3D-Viewer zum Darstellen und Evaluieren von IFC-Gebäudemodellen (Industry Foundation Classes) entwickelt. Das Geometriemodell der IFC beschreibt unter anderem Bauteile als Ergebnis von geometrischen booleschen Operationen. Zu diesem Zweck wurde ein eigener Boolescher Modellierer entwickelt, der es beispielsweise ermöglicht Öffnungskörper von anderen Bauteilen abzuziehen. Er basiert auf dem Algorithmus nach HUBBARD (1990), welcher die Berechnung von booleschen Operationen auf Basis eines triangulierten Brep-Modells (Boundary Representation) beschreibt.

In IFC-Gebäudemodellen sind Bauteile oft das Resultat von mehreren booleschen Operationen. Zum Beispiel kann eine Wand eine Vielzahl von Öffnungen für Türen und Fenster enthalten. Das Endergebnis der Darstellung dieser Wand ergibt sich aus der sukzessiven Abarbeitung der booleschen Operationen. Im Verlauf der Implementierung des Booleschen Modellierers wurde jedoch erkannt, dass an dieser Stelle der Algorithmus von HUBBARD (1990) optimiert werden kann, indem man alle Abzugskörper in einem Schritt betrachtet und anschließend das Ergebnis berechnet. Dadurch wird die sukzessive Ausführung der Operationen eingespart, was zu einer deutlichen Reduzierung der benötigten Berechnungen und somit der Rechenzeit führt.

Dieser Beitrag beschreibt die generelle Vorgehensweise des implementierten Booleschen Modellierers zur Berechnung von geometrischen booleschen Operationen, sowie die Optimierung des Hubbard-Algorithmus. Im Anschluss wird die Performance zwischen dem originalen und dem verbesserten Algorithmus verglichen, um eine Aussage über die Komplexität des Algorithmus treffen zu können.

2 Der Algorithmus von Hubbard

Geometrische boolesche Operationen, auch bekannt unter dem Begriff Constructive Solid Geometry (CSG), werden in der Computergrafik und bei CAD-Programmen genutzt, um Körper durch die Verknüpfung von geometrischen Objekten mittels boolescher Mengenoperationen zu erzeugen. Auf diese Weise können mit den drei booleschen Grundoperationen (Differenz, Vereinigung und Schnittmenge) komplexere Körper erzeugt werden (*Abbildung 1*).

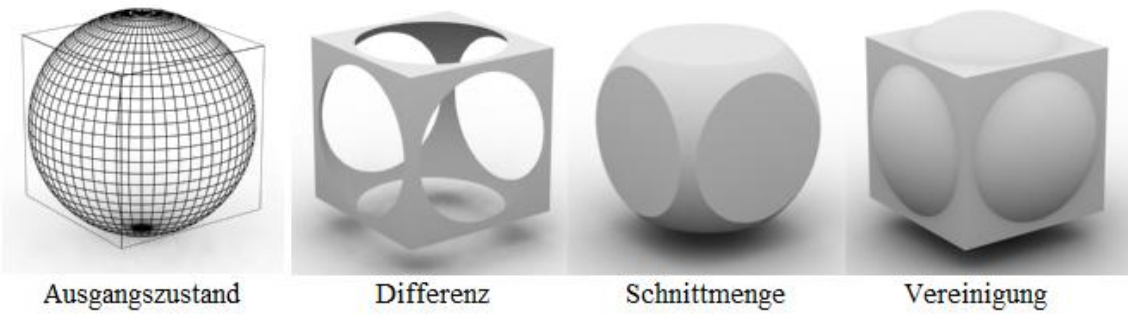

Abbildung 1: Geometrische boolesche Grundoperationen
(Bildquelle: LEHTINEN (2010))

Der an der Bauhaus-Universität Weimar entwickelte Boolesche Modellierer basiert auf dem Algorithmus von HUBBARD (1990), der den allgemeineren Ansatz von LAIDLAW et al. (1986) verbessert. Er beschränkt ihn auf den Spezialfall von triangulierten Körpern und kann somit eine Reihe von Vereinfachungen erreichen. Der Laidlaw-Algorithmus wurde ursprünglich für Körper mit beliebigen polygonal begrenzten Flächen entwickelt und weist eine schlechte Performance beim Einsatz von triangulierten Oberflächen auf. Der Hauptschwachpunkt ist der rein lokale Ansatz beim Zerlegen der Flächen, wodurch unnötig viele Dreiecke erzeugt werden. Um dieses Problem zu umgehen, fasst Hubbard benachbarte koplanare Dreiecke zu Clustern zusammen. Die Cluster werden genutzt, um eine globalere Sicht auf die gemeinsamen Schnittkanten zweier Körper zu ermöglichen. Somit wird lediglich die minimal notwendige Anzahl an Dreiecken erzeugt. Dies wird im Folgenden noch genauer betrachtet.

Die Grundidee der Algorithmen besteht darin, alle Flächen beider Körper *A* und *B* in ihrer Position relativ zu dem jeweils anderen Körper zu klassifizieren. Nach diesem Ansatz können Flächen komplett innerhalb (*inside*), komplett außerhalb (*outside*) oder auf der Oberfläche des anderen Körpers liegen. Flächen auf der Oberfläche werden unterschieden in Flächen, deren Normale in die gleiche Richtung wie die

Oberflächennormale zeigt (*same*), und Flächen, deren Normale in die entgegengesetzte Richtung zeigt (*opposite*). Anhand dieser Klassifizierungen lässt sich anschließend auf einfache Weise der Ergebniskörper zusammensetzen. Dazu werden die benötigten Flächen der Körper in Abhängigkeit der ausgeführten booleschen Operation ausgewählt. *Tabelle 1* zeigt eine Übersicht, welche Flächen zum Ergebniskörper gehören.

Tabelle 1: Übersicht der Flächenauswahl für den Ergebniskörper in Abhängigkeit der booleschen Operation (Quelle: HUBBARD (1990))

	Flächen von A				Flächen von B			
Operation	inside	outside	same	opposite	inside	outside	same	opposite
A ∪ B	nein	ja	ja	nein	nein	ja	nein	nein
A ∩ B	ja	nein	ja	nein	ja	nein	nein	nein
A \ B	nein	ja	nein	ja	ja	nein	nein	nein

Bevor die Klassifizierung der Flächen beginnen kann, müssen Flächen, die teilweise innerhalb und außerhalb oder teilweise auf der Oberfläche liegen, in Flächen zerlegt werden, die eindeutig einer der vier Klassifizierungskategorien zuzuordnen sind. Dazu muss zunächst jedes Dreieck des ersten Körpers mit jedem Dreieck des zweiten Körpers auf eine mögliche Verschneidung untersucht werden. Es reicht aus lediglich nicht koplanare Dreiecke zu untersuchen, was Hubbard ausführlich in seiner Arbeit untersucht.

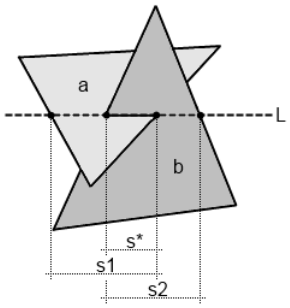

Abbildung 2: Schnittsegmente

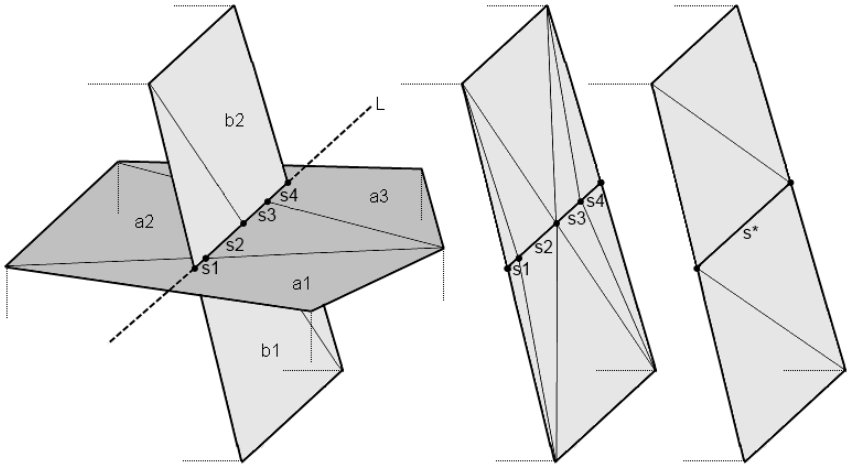

Abbildung 3: Triangulierung von Cluster b, Optimierung der Schnittsegmente

Um schnell und effizient festzustellen, ob ein Schnitt zwischen zwei Dreiecken vorliegt, wird in einem Vorabtest das Dreiecksschnittprüfverfahren von MÖLLER (1997) angewendet. Wenn ein Dreiecksschnitt festgestellt wird, kann anschließend ein Schnittsegment berechnet werden. Dieses Segment ist entweder eine Strecke oder ein Punkt auf der Schnittgeraden *L* der Dreiecke. Zunächst werden dazu zwei Schnittsegmente ermittelt, welche sich jeweils aus der Schnittprüfung von einem Dreieck zur Ebene des anderen Dreiecks ergeben. Beispielhaft sind diese zwei

Segmente *s1* und *s2* in *Abbildung 2* dargestellt. Anschließend werden die beiden Segmente miteinander verschnitten, indem man die Abstände der Segmentendpunkte zu einem Referenzpunkt auf *L* vergleicht. Als Ergebnis erhält man das Schnittsegment *s**, welches in den Clustern der jeweiligen Dreiecke als Zwangskante gespeichert wird.

Nachdem alle Dreiecke miteinander auf Verschneidung geprüft und alle Schnittsegmente ermittelt wurden, kann damit begonnen werden, die Flächen zu zerlegen. Auf diese Weise verbleiben lediglich Dreiecke, die vollständig innerhalb, außerhalb oder auf der Oberfläche des anderen Körpers liegen. Dazu werden die Cluster der Körper neu trianguliert, wobei das Triangulationsverfahren nach PREPARATA und SHAMOS (1985) verwendet wird. Zwangskanten für die Triangulierung sind die Außenkanten der Cluster sowie die ermittelten Schnittsegmente.

In *Abbildung 3* sind zwei Cluster *a* und *b* abgebildet, die sich schneiden. Der Verschneidungstest der Dreiecke ergibt insgesamt vier Schnittsegmente. Anschließend wird die Triangulierung von Cluster *b* durchgeführt, wobei einerseits zwar eindeutig klassifizierbare Dreiecke entstehen. Andererseits wird aber auch deutlich, dass mehr Dreiecke produziert werden als nötig, um die zwei Körper eindeutig an der Schnittkante voneinander zu trennen. Der Grund für die zusätzlich entstehenden Dreiecke ist, dass die Schnittsegmente durch den Schnitt der Dreiecke anstatt der polygonalen Außenkanten der Cluster berechnet wurden. Deshalb ist ein weiterer optimierender Schritt vor der Triangulierung notwendig, der alle benachbarten sowie parallelen Schnittsegmente zusammenfasst. Auf diese Weise können die vier Schnittsegmente aus *Abbildung 3* zu einem Segment *s** verbunden werden. Wenn anschließend Cluster *b* neu trianguliert wird, entsteht die minimal notwendige Anzahl von Dreiecken.

Nachdem die Phase des Zerlegens der Flächen abgeschlossen ist, kann mit der Klassifizierung der Dreiecke begonnen werden. Hierfür wird die Konstellation von Dreiecken entlang eines gemeinsamen Schnittsegmentes analysiert. Die Position eines Dreieckes des ersten Körpers wird relativ zu den zwei Dreiecken des anderen Körpers bestimmt. Beispielsweise wird für die Körper aus *Abbildung 4* die Schnittkante e_a als Startkante ausgewählt. Entlang dieser Kante liegt aus dem ersten Körper das Dreieck *A1*, sowie aus dem zweiten Körper *B1* und *B2*. Zunächst werden die jeweils nicht gemeinsamen Punkte der Dreiecke bestimmt, also hier v_{a1}, v_{b1} und v_{b2}. Als nächstes werden die Entfernungen von v_{b1} zur Ebene von *B2*, sowie von v_{a1} zu den Ebenen von *B1* und *B2* berechnet. Diese werden im Folgenden mit d_{b1b2}, d_{a1b1} und d_{a1b2} bezeichnet.

Anschließend kann durch einfache Fallunterscheidung beim Auswerten der drei berechneten Distanzen der Klassifizierungsstatus bestimmt werden. Zunächst muss bestimmt werden, ob *B1* und *B2* lokal eine konvexe oder konkave Hülle bilden. Wenn d_{b1b2} kleiner Null ist, dann bilden die Dreiecke eine konvexe Hülle – anderenfalls eine konkave Hülle (*Abbildung 5*). *A1* kann im konvexen Fall als *inside* bestimmt werden, wenn d_{a1b1} und d_{a1b2} kleiner Null sind. Im konkaven Fall reicht es aus, wenn eine der beiden Distanzen kleiner Null ist. Ist eine der beiden Distanzen gleich Null, das heißt *A1* liegt in der Ebene von *B1* bzw. *B2*, dann wird *A1* nach einem Vergleich der

Normale von *A1* mit der Normale von *B1* bzw. *B2* der Status *same* bzw. *opposite* zugewiesen. In allen anderen Fällen liegt *A1* außerhalb (*outside*).

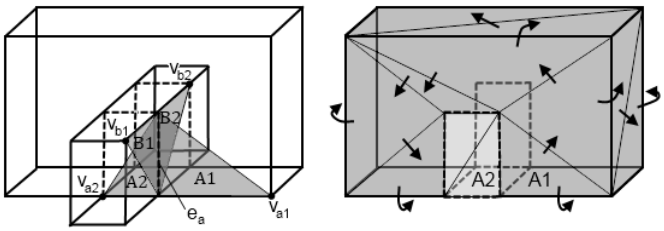

Abbildung 4: Dreiecke entlang eines Schnittsegmentes, Ausbreitung des Klassifizierungsstatus

Sobald der Klassifizierungsstatus eines Dreiecks ermittelt wurde, kann er an die Nachbardreiecke weitergegeben werden. Diese Ausbreitung erfolgt solange bis eine Schnittkante erreicht wird, welche hierbei nicht überschritten werden dürfen. Nachdem beispielsweise *A1* aus *Abbildung 4* als *outside* bestimmt wurde, kann der Status an alle Nachbardreiecke weitergegeben werden, die nicht durch ein Schnittsegment (gestrichelte Linien) davon getrennt sind. Das bedeutet alle dunkelgrau eingefärbten Dreiecke würden ebenfalls als *outside* eingestuft werden.

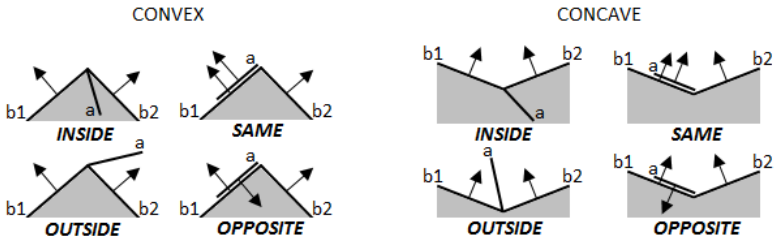

Abbildung 5: Bestimmung des Klassifizierungsstatus

Anschließend wird die Klassifizierung mit einem noch nicht klassifizierten Dreieck entlang eines Schnittsegmentes fortgesetzt. Der Algorithmus terminiert, sobald alle Dreiecke von beiden Körpern klassifiziert wurden.

Der vorgestellte Klassifizierungsalgorithmus funktioniert nur, wenn Schnittsegmente ermittelt werden konnten. Ist dies nicht der Fall, das heißt die Körper durchdringen sich nicht gegenseitig, muss einmalig mittels Ray-Casting ein Punkt-in-Polyhedron-Test ausgeführt werden. Damit kann festgestellt werden, ob ein Körper komplett innerhalb des anderen Körpers liegt. Liegt keiner der beiden Körper vollständig innerhalb des anderen Körpers, können alle Dreiecke als *outside* klassifiziert werden.

3 Gleichzeitiges Berechnen von mehreren booleschen Operationen

Innerhalb des IFC-Modells sind Bauteile oft das Resultat von mehreren booleschen Operationen (z.B. eine Wand mit mehreren Öffnungen für Fenster und Türen). Das Endergebnis der Darstellung ergibt sich aus der sukzessiven Abarbeitung der einzelnen Operationen. Das heißt, der gesamte Algorithmus zum Berechnen der booleschen

Operationen wird so oft wiederholt, wie es Abzugskörper gibt. In der Regel nimmt dabei die Anzahl der Dreiecke des Zwischenergebnisses zu, da der Körper komplexer wird. Dies bedeutet wiederum, dass in jedem Schritt mehr Dreiecke miteinander auf Verschneidung geprüft werden müssen und dass mehr Dreiecke klassifiziert werden müssen. Zusätzlich muss in jedem Schritt der Zwischenergebniskörper neu trianguliert werden.

Der Algorithmus von Hubbard kann für dieses Problem angepasst werden, sodass gleichzeitig mehrere Körper von einem anderen Körper abgezogen werden können oder auch mit ihm vereinigt werden können. Dazu wird nicht der gesamte Algorithmus immer wieder vollständig durchlaufen, sondern es werden in einem ersten Schritt alle Schnittsegmente des ersten Körpers mit den Abzugs- bzw. Vereinigungskörpern ermittelt. Anschließend wird jeder Körper unter Zuhilfenahme der ermittelten Schnittsegmente genau einmal neu trianguliert.

Danach folgt die Klassifizierung der Dreiecke nach dem vorgestellten Algorithmus, wobei jeweils nur gemeinsame Schnittsegmente des ersten Körpers und des gerade betrachteten Abzugskörpers zur Klassifizierung herangezogen werden. Der Klassifizierungsstatus kann solange an die jeweiligen Nachbardreiecke weitergegeben werden, solange kein Schnittsegment überschritten wird. Auch Schnittsegmente, die durch die Verschneidung mit anderen Abzugskörpern erzeugt wurden, dürfen nicht überschritten werden. Abschließend wird das Gesamtergebnis unter Zuhilfenahme von *Tabelle 1* zusammengestellt, wobei die Abzugskörper alle gemeinsam als Körper *B* anzusehen sind.

Diese Verbesserung ist beschränkt auf Abzugskörper, die sich nicht gegenseitig berühren oder durchdringen. Diese Einschränkung ist notwendig, da sonst zu viele Spezialfälle auftreten könnten, die im eigentlichen Algorithmus nicht vorgesehen sind. So müssten dann auch die Abzugskörper gegeneinander auf Verschneidung geprüft und gegeneinander klassifiziert werden. Dies würde aber die Aufwandseinsparung wieder zunichtemachen und den Algorithmus deutlich verkomplizieren.

Um anfangs schnell und effizient feststellen zu können, ob sich die Abzugskörper gegenseitig durchdringen oder berühren, eignet sich ein Bounding-Box-Test. Bei einem positiven Ergebnis kann die Verbesserung nicht angewendet werden. Stattdessen muss dann das Ergebnis auf dem herkömmlichen Weg sukzessive berechnet werden.

4 Vergleich der Komplexität der Algorithmen

Die vorgestellte Verbesserung des Algorithmus von HUBBARD (1990) verringert den Rechenaufwand und somit die Rechenzeit bei der Berechnung von booleschen Operationen von Bauteilen mit mehreren Abzugskörpern. Der Rechenaufwand ist abhängig von mehreren Faktoren, wie beispielsweise der Anzahl der sich schneidenden Dreiecke oder der Anzahl der auszuführenden Operationen. Im Folgenden wird der originale Hubbard-Algorithmus mit *Hubbard_Suc* (successively) und der verbesserte Algorithmus mit *Hubbard_Sim* (simultaneously) bezeichnet.

Der Hauptvorteil des *Hubbard_Sim* besteht darin, dass der primäre Körper, von dem mehrere Körper abgezogen werden sollen, nur einmal im Laufe der Berechnung neu trianguliert werden muss. Hingegen wird beim *Hubbard_Suc* der Primärkörper nach jeder Verschneidung mit einem Abzugskörper neu trianguliert. Das Zwischenergebnis nach jedem Schritt wird in der Regel immer komplexer, das heißt es werden in jedem Schritt zusätzliche Dreiecke produziert. Diese Dreiecke müssen danach zusätzlich mit dem nächsten Abzugskörper verschnitten werden. Somit werden insgesamt mehr Schnittsegmente produziert, was zu einem erhöhten Aufwand bei der Optimierung der Segmente führt.

Ein weiterer Punkt, durch den Rechenaufwand eingespart wird, besteht darin, dass die Klassifizierungsphase nur einmal durchlaufen wird. Bei der sukzessiven Subtraktion wird hingegen so oft klassifiziert, wie es Abzugskörper gibt. Auch die Erzeugung der Cluster-Struktur zu Beginn des Algorithmus muss nur einmal für alle Körper durchgeführt werden, wohingegen sonst die Cluster-Struktur des Zwischenergebniskörpers immer wieder neu generiert werden muss.

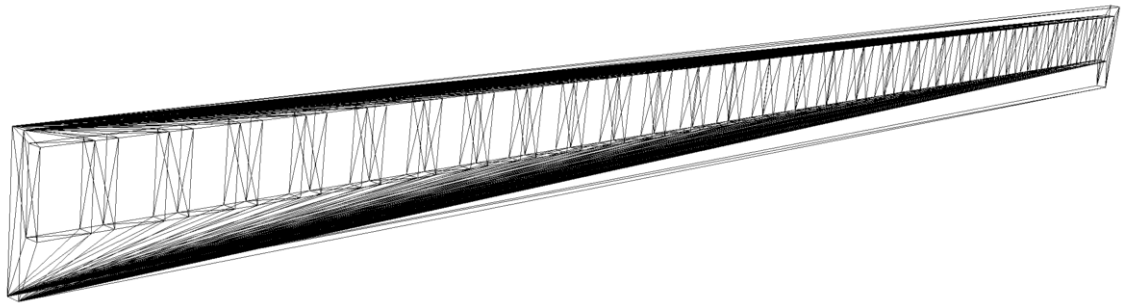

Abbildung 6: Beispielwand mit 28 Öffnungen

Im Rahmen dieser Arbeit wurde untersucht, wie sich die Rechenzeit in Abhängigkeit von der Anzahl der Öffnungskörper für beide Algorithmen verhält. Dabei wurde die Zeit gemessen, die benötigt wird, um das Endergebnis einer Wand mit unterschiedlicher Anzahl von Öffnungskörpern zu berechnen. Angefangen mit einer Wand mit nur einer Öffnung, wo keine Verbesserung oder Verschlechterung zu erwarten war, wurden Wände mit bis zu 28 Öffnungen getestet (*Abbildung 6*).

Tabelle 2: Vergleich der Rechenzeiten (Mittelwerte) der Algorithmen in Abhängigkeit der Anzahl der Öffnungskörper

Öffnungen	1	2	3	5	10	15	20	25	28
Hubbard_Sim [ms]	3,4	5,3	7,5	15,3	32,6	59,5	86,0	112,9	132,9
Hubbard_Suc [ms]	3,4	7,9	13,9	35,4	130,1	321,7	666,7	1152,8	1558,8
Faktor	1,0	1,5	1,8	2,3	4,0	5,4	7,8	10,2	11,7

In *Tabelle 2* sind die Mittelwerte der gemessenen Zeiten für beide Algorithmen dargestellt. Mit steigender Anzahl der Öffnungskörper reduziert sich die Rechenzeit gegenüber dem sukzessiven Vorgehen maßgeblich. Aus *Abbildung 7* geht hervor, dass der stärker als quadratisch anwachsende Anstieg der Rechenzeit mit dem verbesserten Algorithmus auf einen flachen linearen Anstieg reduziert wird.

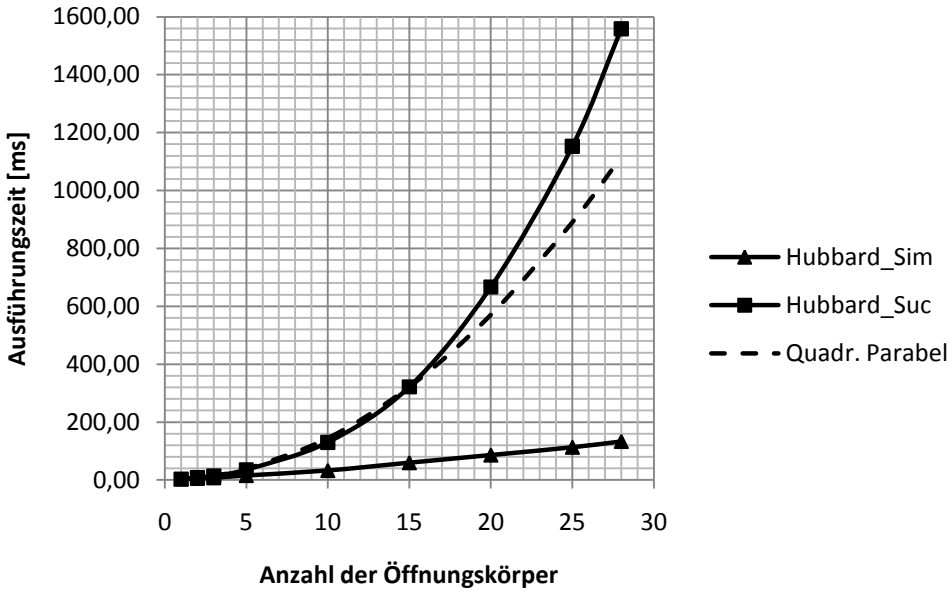

Abbildung 7: Vergleich der Rechenzeiten der Algorithmen in Abhängigkeit von der Anzahl der Öffnungskörper

5 Ergebnis und Ausblick

Der in diesem Beitrag vorgestellte verbesserte Algorithmus zum Berechnen von geometrischen booleschen Operationen reduziert die Rechenzeit erheblich. Die quadratische Abhängigkeit des sukzessiven Hubbard-Algorithmus in Bezug auf die Anzahl der Öffnungskörper, konnte durch die gleichzeitige Betrachtung von allen Abzugskörpern auf eine flach linear ansteigende Abhängigkeit reduziert werden.

Der vorgestellte Boolesche Modellierer findet Anwendung im Open-Source-Projekt Open IFC Tools und wird für die Darstellung von IFC Gebäudemodellen genutzt. Internetadresse: *www.openifctools.org*.

6 Referenzen

HUBBARD, (1990). Constructive Solid Geometry for Triangulated Polyhedra. Department of Computer Science, Brown University, Providence, Rhode Island 02912, CS-90-07.

LAIDLAW, TRUMBORE und HUGHES (1986). Constructive Solid Geometry for Polyhedral Objects. Proceedings of SIGGRAPH ’86, Computer Graphics, Vol. 2, ACM, New York, USA.

LEHTINEN, (2010). 3D Graphics - What is 3D Computer Graphics. Artikel über Computergrafiken: http://knol.google.com/k/3d-graphics, Version: 39, Stand vom 02.05.2010.

MÖLLER, (1997). A Fast Triangle-Triangle Intersection Test. Journal of Graphics Tools, 2 (2), pp. 25-30.

PREPARATA und SHAMOS, (1985). Computational Geometry: An Introduction. Springer-Verlag, New York, USA.