Technology
Arts Sciences
TH Köln

Entwicklung und Evaluation einer auf der Hauptkomponentenanalyse basierenden Bounding Volume Hierarchy

Bachelorarbeit

ausgearbeitet von

Matthias Sauer

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang
Medieninformatik

Erster Prüfer/in: Prof. Dr.-Ing. Martin Eisemann
Technische Hochschule Köln

Zweiter Prüfer/in: Prof. Dr. rer. nat. Wolfgang Konen
Technische Hochschule Köln

Gummersbach, im Juli 2019

Kurzfassung

Die vorliegende Arbeit beschäftigt sich im übergeordneten Kontext mit den Bounding Volume Hierarchies zur Veinfachung des Intersection Testings beim Raytracing. Die derzeitige Problematik besteht vor allem in der immer noch zu optimierenden Laufzeit. Dementsprechend wird trotz der bereits bestehenden Beschleunigungsdatenstrukturen wie unter anderem der Bounding Volume Hierarchy versucht, effizientere Strukturen oder Erstellungsprozeduren zu entwickeln. Für die Bounding Volume Hierarchy bedeutet dies, dass vor allem hinsichtlich verschiedener Splitting-Methoden und Möglichkeiten für die Baumoptimierung geforscht wird. Explizit wird daher innerhalb dieser Arbeit untersucht, wie die Bounding Volume Hierarchy durch die Verwendung der Hauptkomponentenanalyse bei der Erstellung optimiert werden kann und wie effizient der daraus resultierende Ansatz gegenüber der klassischen Bounding Volume Hierarchy sowie deren Splitting-Methoden ist. Eine Evaluation anhand 12 verschiedener Szenen zeigte, dass der vorliegende Ansatz unter Verwendung der SAH-Methode wie auch mit der Middle-Methode 17.70 % respektive 13.14 % geringere Renderlaufzeiten als der distanzbasierte Ansatz aufweist. Des Weiteren konnte mittels der kombinierten Verwendung aus klassischer SAH-Methode und PCA-basierter SAH-Methode eine weitere Verbesserung um 6.65 % gegenüber der SAH-Methode der PCA-BVH erreicht werden.

Inhaltsverzeichnis

1. Einleitung

Das Raytracing beschreibt einen Algorithmus zur Erzeugung eines photorealistischen Bildes (Glassner, 1989, S. 1), wobei das Verfahren gerade in jüngster Vergangenheit hinsichtlich neuer, den Algorithmus unterstützender Hardware (Boksansky u. a., 2019) einen Aufschwung hinsichtlich neuer Einsatzfelder, vor allem im Bereich von Cinematics (Liu u. a., 2019) wie auch in Echtzeit-Systemen im Rahmen von Hybrid-Verfahren erfährt (Barré-Brisebois u. a., 2019). Ein wichtiger Schritt beim Raytracing ist das Intersection Testing, bei dem die vom Augpunkt aus emittierten Strahlen gegen die innerhalb der Szene befindlichen Primitive getestet werden, um feststellen zu können, an welchen Bildpunkt welcher Teil eines Primitivs der Szenengeometrie getroffen wird. Dieser Schritt macht einen Großteil des Berechnungsaufwands aus (Glassner, 1989, S. 203), da jeder Strahl mit jedem Primitiv getestet werden müsste, obwohl ein Großteil dieser Strahlen nur wenige der Primitive trifft (Pharr u. a., 2016, S. 247-248).

Um dieser Problematik zu entgegnen, wurden die Beschleunigungsdatenstrukturen entwickelt, welche die Komplexität der Berechnungen sowie damit einhergehend auch den allgemeinen Zeitaufwand verringern können (Pharr u. a., 2016, S. 247-248). Die zwei übergeordneten Konzepte für derartige Strukturen sind zum einen die räumliche Aufteilung und zum anderen die primitivbezogene Aufteilung (Pharr u. a., 2016, S. 254). Das letztgenannte Konzept umfasst auch die für diese Arbeit relevante Bounding Volume Hierarchy (BVH). Hier wird mittels rekursiver Unterteilung der Primitive ein Binärbaum gebildet, der die Bounding Volumes enthält, welche die unterteilten Primitive umfassen. Bevor ein Strahl gegen ein einzelnes Primitiv getestet werden muss, kann er zunächst gegen den Binärbaum getestet werden und falls die Traversierung nicht bis zu einem Blattknoten positiv ausfällt, kann ausgeschlossen werden, dass dieser Strahl ein Primitiv der Szene trifft. Des Weiteren muss der Strahl bei einer positiven Traversierung nur gegen die Primitive des Blattknotens getestet werden, für den die Intersektion positiv verlief. Für die Erstellung der Bounding Volumes, die als Knoten im Binärbaum eingegliedert werden, existieren verschiedene Methoden, sogenannte Splitting-Methoden, wie unter anderem die Surface Area Heuristic-Methode oder die Middle-Methode. Daher soll Gegenstand dieser Arbeit sein, wie die Hauptkomponentenanalyse für die Erstellung einer BVH genutzt werden kann und wie effizient eine derartige BVH im Vergleich zur klassischen BVH von Pharr u. a. (2016) mit der in Sauer (2019) implementierten, distanzbasierten Splitting-Methode sowie den anderen aus Pharr u. a. (2016) bekannten Methoden ist.

Hierzu werden im Grundlagenkapitel neben den aus Sauer (2019) bereits bekannten Fakten tiefergehende Grundlagen geschaffen. Dies betrifft unter anderem die Darstellung verschiedener Raytracing-Techniken, die BVH mit den mathematischen Intersektionsgrundlagen sowie die verschiedenen Splitting-Methoden. Abschließend wird die Hauptkomponentenanalyse anwendungsbezogen sowie mathematisch erläutert.

Im Kapitel 4 wird die theoretische Entwicklung der PCA-basierten BVH (Abschnitt 4.1), die Implementierung (Abschnitt 4.2) sowie die Evaluation (Abschnitt 4.3) dieser näher betrachtet. Hierzu wird im Unterabschnitt 4.1.1 zunächst der Grundgedanke des Ansatzes skizziert, um im Folgenden in Unterabschnitt 4.1.2 die Anwendung der PCA im vorliegenden Kontext sowie in Unterabschnitt 4.1.3 die zu verwendende Splitting-Methode zu diskutieren.

Innerhalb des darauffolgenden Abschnitts 4.2 wird die Verwendung des Physically Based Raytracer im Unterabschnitt 4.2.1 thematisiert. Die folgenden Unterabschnitte 4.2.2 und 4.2.3 gehen dabei auf die Implementierung der PCA und der zu verwendenden Splitting-Methoden ein.

Der Abschnitt 4.3 beschreibt die Evaluation des zuvor theoretisch und praktisch entwickelten Ansatzes für die Bounding Volume Hierarchy. Hierzu werden zunächst in Unterabschnitt 4.3.1 die Testzenen dargestellt und die der Evaluation zugrundeliegenden Metriken näher erläutert. In Unterabschnitt 4.3.2 und 4.3.3 werden dann die Ergebnisse der Evaluationsschritte präsentiert und interpretiert.

Das Kapitel 5 fasst die Arbeit sowie die erbrachten Erkenntnisse zusammen und Kapitel 6 enthält eine Reflektion über die methodische Vorgehensweise sowie entsprechendes Verbesserungspotential.

2. Grundlagen

In diesem Kapitel werden für die Bachelorarbeit relevante Grundlagen wie die verschiedenen Techniken für das Raytracing, die Grundlagen für das Intersection Testing mittels der Bounding Volume Hierarchies und deren Splitting-Methoden sowie die Hauptkomponentenanalyse dargestellt.

2.1. Techniken im Kontext von Raytracing

In diesem Abschnitt werden Techniken im Kontext des Raytracings vorgestellt. Dies umfasst unter anderem das standardmäßige Backward Raytracing, das Forward Raytracing, das rekursive sowie das diffuse Raytracing, Path Tracing und Photon Mapping.

2.1.1. Backward Raytracing

Bei dieser Technik handelt es sich um die traditionelle Umsetzung des Verfahrens (Arvo u. a., 1986). Problematisch für diese Methode sind allerdings optische Effekte wie Reflektion und Brechung durch transparente Objekte, da lediglich an den Intersektionspunkten einzelne Strahlen zu den Lichtquellen erzeugt werden, um zu überprüfen, ob der Ursprungspunkt im Licht liegt oder durch ein anderes Objekt verdeckt wird (Arvo u. a., 1986).

2.1.2. Forward Raytracing

Im Rahmen des Forward Raytracings (Abbildung 2.1) werden entgegen des klassischen Ansatzes vom Lichtpunkt ausgehend Strahlen versendet und entsprechend beim Eintritt in die Kamera als Augpunkt wiedergegeben. Problematisch bei dieser Technik ist allerdings die geringe Effizienz, da zum einen sehr viele Strahlen von der Lichtquelle verschossen werden müssen und zum anderen viele Strahlen nicht am Augpunkt eintreffen (Glassner, 1989, S. 7-8) und somit keinen Mehrwert, sondern nur Mehraufwand produzieren.

2.1.3. Rekursives Raytracing

Beim rekursiven Raytracing handelt es sich um eine erweiterte Version des Raytracings, welche photorealistische Effekte wie Reflektion oder Brechung ermöglicht (Jensen u. Christensen, 2007, S. 25). Diese Effekte werden durch die Erweiterung des Verfahrens realisierbar, indem am Intersektionspunkt des Strahls jeweils ein weiterer Strahl für die Brechungs- und Reflektionseffekte gemäß der zugrundeliegenden physikalischen Gesetzmäßigkeiten erzeugt wird (Abbildung 2.2) und diese bei weiteren Treffern mit entsprechenden Oberflächen selbst wiederum weitere Strahlen erzeugen (Jensen u. Christensen, 2007, S. 25).

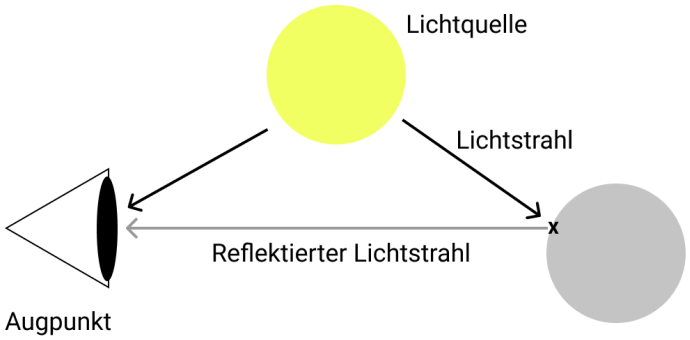


Abbildung 2.1.: Visualisierung des Forward Raytracings

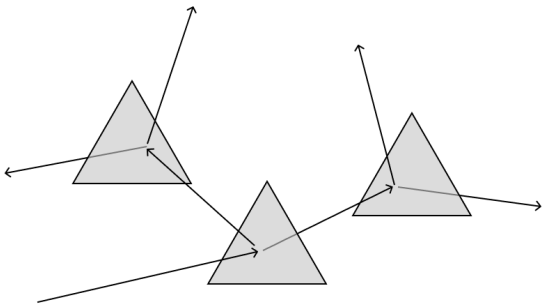


Abbildung 2.2.: Ray-Tree des rekursiven Raytracing (in Anlehnung an Jensen u. Christensen (2007), S. 25)

2.1.4. Diffuses Raytracing

Beim diffusen Raytracing handelt es sich um eine Unterart des Monte-Carlo-Raytracings, bei dem im Gegensatz zu den vorherigen Methoden, welche die Strahlenrichtung deterministisch bestimmten, unter anderem Strahlenursprung und Richtung non-deterministisch durch Zufallszahlen bestimmt werden (Jensen u. Christensen, 2007, S. 29).

Somit werden ähnlich wie beim rekursiven Raytracing (Unterabschnitt 2.1.3) für jeden Oberflächentreffer Strahlen erzeugt, wobei allerdings nicht nur ein Strahl, sondern mehrere Strahlen erzeugt werden (Abbildung 2.3). Dies führt zwar zu guter Qualität, allerdings kann die Anzahl der Strahlen bei mehrfachen Reflektionen und Brechungen rapide ansteigen, weswegen die Anzahl der Strahlen ab einer gewissen Reflektionstiefe reduziert wird, um der ansteigenden Komplexität entgegenzuwirken (Jensen u. Christensen, 2007, S. 29).

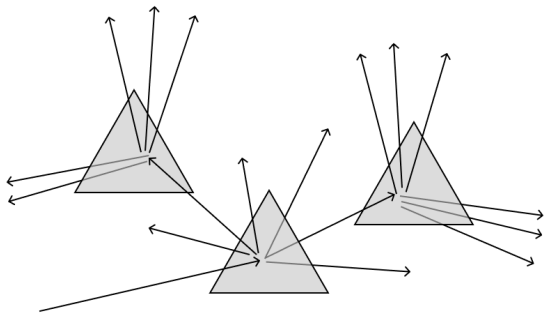


Abbildung 2.3.: Ray-Tree des diffusen Raytracings mit jeweils drei Strahlen für Reflektion und Refraktion (in Anlehnung an Jensen u. Christensen (2007), S. 29)

2.1.5. Path Tracing

Das Path Tracing beschreibt eine Variation des diffusen Raytracings, bei der statt mehreren Strahlen für Reflektion und Brechung für jeden Punkt nur je ein Strahl für Reflektion und Brechung erzeugt wird (Jensen u. Christensen, 2007, S. 29). Um trotzdem das Rauschen gering zu halten und eine gute Bildqualität zu gewährleisten, werden eine Vielzahl an Sichtbarkeitsstrahlen verschossen, wodurch auch zusätzlich noch visuelle Effekte wie Motion Blur und Tiefenunschärfe mit geringem Mehraufwand umgesetzt werden können (Jensen u. Christensen, 2007, S. 29).

Als Erweiterung zum Path Tracing existiert das bidirektionale Path Tracing (Abbildung 2.4). Der Gedanke hinter dem bidirektionalen Path Tracing basiert darauf sowohl den Augpunkt wie auch die Lichtquelle Strahlen emittieren zu lassen und deren Trefferpunkte mittels Schattenstrahlen zu verbinden (Lafortune u. Willems, 1993). Dadurch wird nicht nur die Beleuchtung der primären Lichtquellen am Trefferpunkt eines Primärstrahls für die Helligkeit genutzt, sondern auch reflektierende Flächen wie Wände miteinbezogen, wodurch die globale Beleuchtung simuliert werden kann (Lafortune u. Willems, 1993).

2.1.6. Photon Mapping

Beim Photon Mapping handelt es sich um eine Technik, die es ermöglicht, globale Illumination sowie photorealistische Effekte wie Kaustiken zu simulieren (Jensen u. Christensen, 2007, S. 10). Nach Jensen u. Christensen (2007) besteht dieser Algorithmus aus zwei Phasen. Innerhalb der ersten Phase werden von der Lichtquelle aus die Photonen gesendet und bei Treffern auf einer Objektoberfläche innerhalb der namensgebenden Photon Map (Abbildung 2.5) gespeichert (Jensen u. Christensen, 2007, S. 10). In der zweiten Phase wird die Photon Map mittels statistischer Methoden ausgewertet, um Informationen über den eintretenden Lichtfluss und reflektierte Strahlen zu erhalten (Jensen u. Christensen, 2007, S. 10). Im Vergleich zu anderen Methoden erbringt das Photon Mapping neben der höheren Geschwindigkeit vor allem Vorteile durch bessere Skalierbarkeit für komplexere Szenen und qualitativ weniger auffällige Artefakte

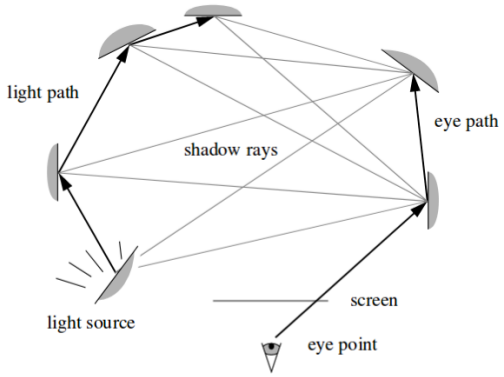


Abbildung 2.4.: Beispiel des bidirektionalen Path Tracings (Lafortune u. Willems, 1993)

(Jensen u. Christensen, 2007, S. 10). Das weniger auffällige Rauschen ergibt sich aus dem Fakt, dass das Rauschen beim Photon Mapping eine niedrigere Frequenz besitzt, während das Rauschen bei den Monte-Carlo-Methoden wie beim Path Tracing oder dem bidirektionalen Path Tracing mit hoher Frequenz auftritt, wodurch die allgemeine Bildqualität beim Photon Mapping höher ist (Jensen u. Christensen, 2007, S. 10).

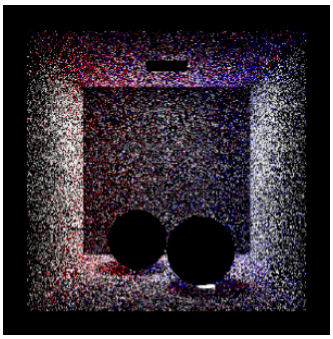


Abbildung 2.5.: Beispiel einer Photon Map (Jensen u. Christensen, 2007, S. 56)

2.2. Bounding Volume Hierarchy

Neben der Darstellung der mathematischen Grundlagen für das Intersection Testing im Allgemeinen wie für die Bounding Boxes innerhalb der Bounding Volume Hierarchy im Speziellen wird auch auf die verschiedenen Splitting-Methoden eingegangen.

2.2.1. Mathematische Grundlagen des Intersection Testings

Das Intersection Testing selbst benötigt einen großen Teil des gesamten Zeitaufwands für das Rendering eines Bildes mittels Raytracing (Glassner, 1989, S. 203). Im Rahmen des Intersection Testing werden die Schnitte der Strahlen mit den Primitiven innerhalb der Szene getestet. Nur mittels dieses Schnittes kann eine Färbung des entsprechenden

Bildpunktes vorgenommen werden, da der Schnitttest die Informationen der Oberfläche zurückliefert (Pharr u. a., 2016, S. 116). So werden dadurch unter anderem das Material sowie die Normale der Oberfläche an der entsprechenden Stelle bestimmt (Pharr u. a., 2016, S. 116).

Der klassische Strahl kann hierbei nach Pharr u. a. (2016) sowie Jensen u. Christensen (2007) mathematisch wie folgt ausgedrückt werden:

$$r(t) = o + td$$

Hierbei beschreibt, wie auch in der Abbildung 2.6 zu sehen, r den ganzen Strahl, die Variable o den Ursprung des Strahles im Raum, die Variable t die Veränderung im Intervall $(0, \infty)$ und die Variable d den Richtungsvektor (Pharr u. a., 2016, S. 73).

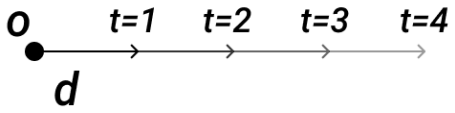


Abbildung 2.6.: Beispielhafte Skizze eines Strahls

Der dargestellte Strahl wird im Rahmen des Intersections Testings gegen die Objekte getestet. Dies kann mit der impliziten Funktion $F(x, y, z) = 0$ aufgelöst werden, wodurch für elementare Formen entsprechende Gleichungen aufgestellt werden (Pharr u. a., 2016, S. 7). Für eine Kugel am Ursprung kann somit nach Pharr u. a. (2016) die folgende Gleichung formuliert werden:

$$x^2 + y^2 + z^2 - r^2 = 0$$

In diese Gleichung, bestehend aus dem Radius r und den Komponenten für x, y und z, kann die bereits bekannte Gleichung des Strahls komponentenweise eingesetzt werden, wodurch nach Pharr u. a. (2016) die folgende Gleichung entsteht:

$$(o_x + td_x)^2 + (o_y + td_y)^2 + (o_z + td_z)^2 - r^2 = 0$$

Diese Gleichung enthält mit t nur eine unbekannte Variable, die im Folgenden durch Auflösung einer quadratischen Gleichung ermittelt werden kann. Falls die Wurzel von t eine oder mehrere reelle Zahlen darstellt, so ist die kleinste, positive Zahl der Wert der Variable t am nähsten Intersektionspunkt (Pharr u. a., 2016, S. 7).

Da die meisten Szenen allerdings einen deutlich komplexeren Aufbau besitzen (Pharr u. a., 2016, S. 152), reichen die bisherigen Erkenntnisse über die Abbildung des Intersection Testings für Kugelformen nicht aus. Im Normalfall sind die Szenen aus einzelnen Dreiecken in einem Mesh zusammengesetzt (Pharr u. a., 2016, S. 152). Dementsprechend muss auch für Strahl-Dreiecks-Konstellationen ein passender Schnitttest durchgeführt werden, wobei die Anzahl an Intersektionstests beim Brute-Force-Verfahren zu entsprechend hoher Laufzeit führt (Pharr u. a., 2016, S. 248).

Diese Problematik wird von den Beschleunigungsdatenstrukturen adressiert. Eine Art der Beschleunigungsdatenstrukturen ist die sogenannte Bounding Volume Hierarchy, welche einen Binärbaum aus Bounding Volumes aufspannt, die die jeweiligen

Primitive gruppieren. Die Erzeugung der Volumes wird durch die Vereinigung der Bounding Boxes der Primitive erreicht. Die Gruppierung wird durch die später vorgestellten Splitting-Methoden realisiert, welche abhängig von selbiger die Primitive in genau zwei Gruppen einteilen. Die Abbildung 2.7 visualisiert hierbei auch die Traversierung der BVH.

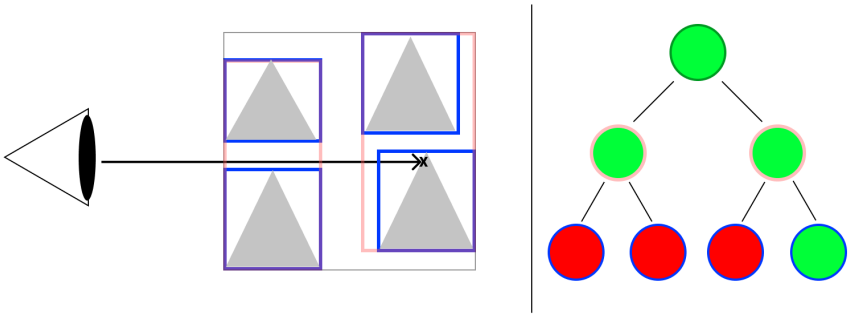


Abbildung 2.7.: Beispiel einer Bounding Volume Hierarchy mit einem Strahl, der eine Szene trifft und die zugehörige BVH traversiert, wobei grün dargestellte Knoten einen positiven Schnitttest und rot dargestellte Knoten einen negativen Schnitttest darstellen

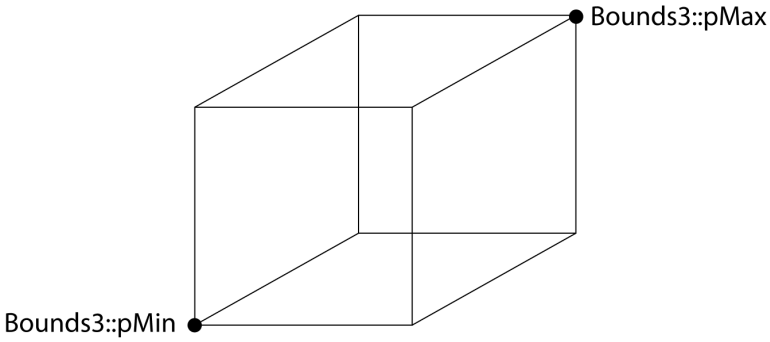


Abbildung 2.8.: Beispiel für eine Axis-Aligned Bounding Box (Online-Abbildung: Axis-Aligned Bounding Box nach Pharr u. a. (2016), S. 77; URL: `http://www.pbr-book.org/3ed-2018/Geometry_and_Transformations/Bounding_Boxes.html`)

Wie bereits für die Kugel können für die Bounding Volumes auch derartige Gleichungen aufgestellt werden. Eine Variante der Bounding Volume stellt die Axis Aligned Bounding Box (AABB) dar, deren Repräsentation auch in der Abbildung 2.8 zu sehen ist. Bei dieser handelt es sich um eine Bounding Box, die entlang der Achsen des Koordinatensystems ausgerichtet ist. Durch diese Eigenschaft kann die AABB im Gegensatz zu normalen Boxen, welche mit einem Ursprung und drei Vektoren für die jeweiligen Ausdehnungen ausgedrückt werden müssen, mit den beiden Vektoren p_{min} und p_{max} beschrieben werden, die den kleinsten und den größten Eckpunkt der Box darstellen (Pharr u. a., 2016, S. 77). Andere Parameter können aus der Bedingung

der Achsenausrichtung und den beiden Vektoren errechnet werden (Pharr u. a., 2016, S. 77). Die Achsenausrichtung ist allerdings auch gleichzeitig ein Nachteil der AABB, welcher beim Splitting der Primitive auffällt. So führen eng aneinander liegende Primitive schnell zu relativ starken Überlappungen bei AABBs (Abbildung 2.9).

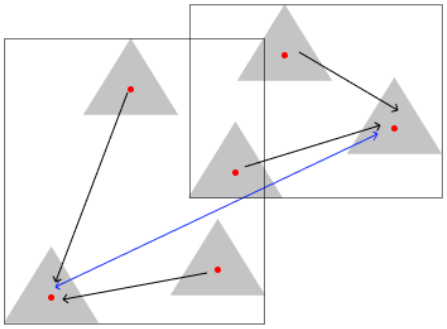


Abbildung 2.9.: Überlappungen bei Axis-Aligned Bounding Boxes im Rahmen des distanzbasierten Splitting-Ansatzes (Sauer, 2019)

Ein Schnitttest wird bei den AABBs im Physically Based Raytracer über die sogenannten Slabs durchgeführt. Die Slabs stellen parallele, achsenorientierte Ebenen dar, welche sich an den die Bounding Box beschreibenden Vektoren ausrichten. Entsprechend müssen für alle Slabs jeweils zwei Berechnungen durchgeführt werden, welche den Minimal- und Maximalwert für den Parameter t für die jeweilige Dimension liefern.

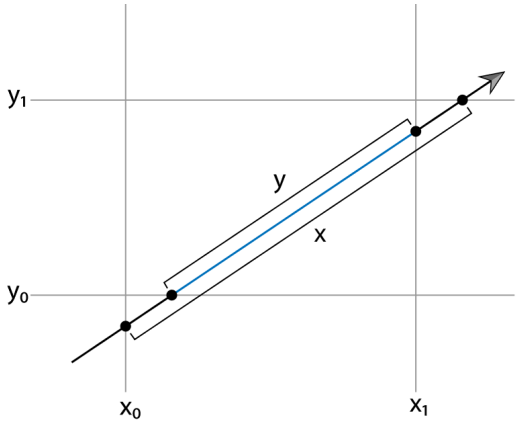


Abbildung 2.10.: Slabs einer Axis Aligned Bounding Box im zweidimensionalen Raum mit einem schneidenden Strahl (Online-Abbildung: Slabs einer AABB nach Pharr u. a. (2016), S. 126; URL: `http://www.pbr-book.org/3ed-2018/Shapes/Basic_Shape_Interface.html`)

Ähnlich wie für die Kugel kann in die Formel der Ebenen eines Slabs (Abbildung 2.10) komponentweise die Gleichung des Strahls eingesetzt werden. Hierdurch wird die Formel $ax + by + cz + d = 0$ der Ebenengleichung in die Formel $0 = a(o_x + td_x) + b(o_y + td_y) + c(o_z + td_z) + d$ umgewandelt (Pharr u. a., 2016, S. 127). Da als

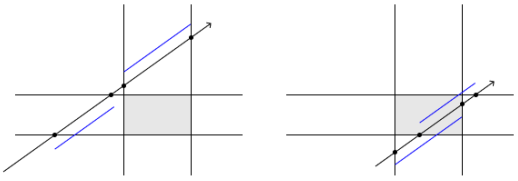


Abbildung 2.11.: Beispiel einer nicht getroffenen Bounding Box und einer getroffenen Bounding Box mit der Kalkulation von t für die Slabs mit blauen Hilfslinien zur Verdeutlichung des Intervalls der beiden Intersektionspunkte der jeweiligen Dimension (in Anlehnung an Pharr u. a. (2016), S. 126)

Ziel nach dem Parameter t aufgelöst werden soll, kann weiter vereinfacht werden indem die Komponenten als Vektoren zusammengefasst werden, wodurch die Gleichung $0 = (a, b, c) * o + t(a, b, c) * \boldsymbol{d} + d$ entsteht (Pharr u. a., 2016, S. 127). Die Variable $\boldsymbol{d}$ steht hierbei für den Richtungsvektor des Strahls, während d die Variable aus der Ebenengleichung widerspiegelt. Im Folgenden kann nach t aufgelöst werden, unter der Annahme dass jeweils zwei der Komponenten von a, b und c gleich 0 sind, da die Normale der Ebene eines Slabs in der jeweiligen Ebene 1 aufweist (Pharr u. a., 2016, S. 127), sodass schließlich die Formel $t = \frac{-d-o_x}{d_x}$ entsteht. Die letzte Modifikation kann vorgenommen werden, indem für den Koeffizenten d aus der Ebenengleichung $-x_1$, somit also der negative x-Wert des Punktes x genutzt wird, wodurch die Gleichung $t_1 = \frac{x_1-o_x}{d_x}$ ensteht (Pharr u. a., 2016, S. 127). Analog kann dies für alle anderen Ebenen der Slabs durchgeführt werden, so dass für 3 Dimensionen ingesamt 6 Werte für den Parameter t ermittelt werden. Mittels dieser Werte für t kann, wie in Abbildung 2.11 visualisiert ist, die Intersektion bestimmt werden. Hierzu gilt die Maßgabe, dass zwischen den Intervallen von t einer Dimension ein Wert für t der jeweiligen anderen Dimensionen liegen muss wie es beispielsweise in der Abbildung 2.11 zu sehen ist.

Eine andere Variante der Bounding Boxes entgegnet dem Problem von überlappenden Bounding Boxes, indem die Eigenschaft der Achsenausrichtung entfällt und stattdessen eine Orientierung im Raum genutzt wird. Diese Bounding Boxes werden als Oriented Bounding Boxes (OBB) bezeichnet. Ein Nachteil dieser Methode gegenüber den AABBs ist der durch die Rotationstransformation entstehende Mehraufwand in der Berechnung (Glassner, 1989, S. 210).

2.2.2. Splitting-Methoden

Ingesamt sind im PBRT bereits vier verschiedene Splitting-Methoden implementiert, welche durch die distanzbasierte Methode aus Sauer (2019) erweitert werden. Bei den anderen vier Methoden handelt es sich um die Surface Area Heuristic-, die Middle-, die EqualCounts-Methode und die Hierarchical Bounding Volume Hierarchy.

SAH-Methode Bei der SAH-Methode werden entlang der Achse mit der längsten Ausdehnung mehrere kleinere Teilpartitionierungen erstellt, indem die Bounding Box

zunächst durch die möglichen Split-Positionen in gleich große Abschnitte eingeteilt wird und die Mittelpunkte entsprechend zugeordnet werden (Pharr u. a., 2016, S. 265). Im Folgenden können iterativ aus den Buckets an der jeweiligen Split-Position zwei Bounding Boxes erstellt werden, welche mit der Anzahl der enthaltenen Primitive nach der SAH von (Goldsmith u. Salmon, 1987) evaluiert werden, sodass die Split-Positionierung mit den günstigsten Kosten für die Unterteilung gewählt wird. Diese Kosten der Surface Area Heuristic errechnen sich nach den jeweiligen prozentualen Größen der beiden Kindknoten gegenüber des Oberknotens, wobei die Kosten für die Intersektion eines Primitivs für alle Primitive innerhalb eines Knotens jeweils für beide Kindknoten aufsummiert werden, dann mit dem zuvor errechneten Prozentanteil multipliziert werden und schließlich die beiden Werte für die Kindknoten addiert werden.

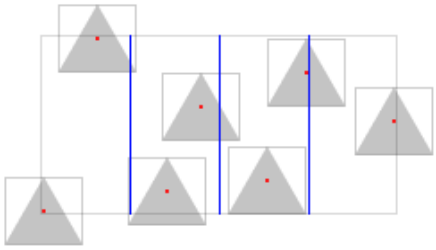


Abbildung 2.12.: Vereinfachtes Beispiel für die SAH-Methode mit vier Buckets und drei möglichen Split-Positionen (in Anlehnung an Pharr u. a. (2016), S. 266)

Als Anwendungsbeispiel kann die in der Abbildung 2.12 dargestellte Primitivverteilung dienen. Entlang der Achse mit der längsten Ausdehnung sind die möglichen Split-Positionen mit blauen Linien eingezeichnet. Wie die Methode mit den entstehenden Buckets verfährt, ist in Abbildung 2.13 dargestellt. Allerdings wurden innerhalb dieser Abbildung aus Gründen der Übersichtlichkeit die Bounding Boxes der einzelnen Buckets nicht dargestellt.

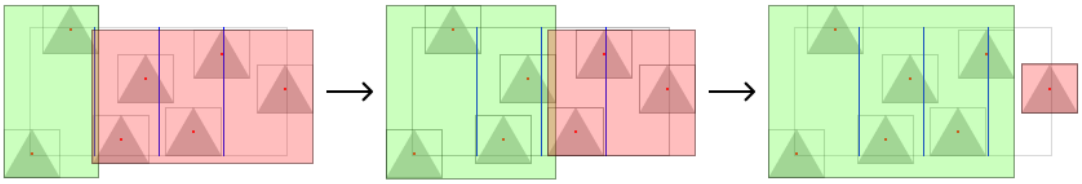


Abbildung 2.13.: Verfahren der SAH-Methode für das Anwendungsbeispiel

Die SAH-Methode gilt gemeinhin als die Splitting-Methode, welche die qualitativ besten Binärbäume generiert und dementsprechend trotz der kostenintensiven Vielzahl an Kostenberechnungen einen zeitlichen Vorsprung vor anderen Methoden bietet (Pharr u. a., 2016, S. 263 - 264). Diese Methode wird im weiteren Verlauf dieser Arbeit zur besseren Differenzierung auch als „klassische SAH-Methode“ bezeichnet.

Middle-Methode

Die Middle-Methode teilt den Knoten entlang des Mittelpunktes der längsten Achse und führt anhand dieses Mittelpunktes die Unterteilung der Primitive durch (Pharr u. a., 2016, S. 262).

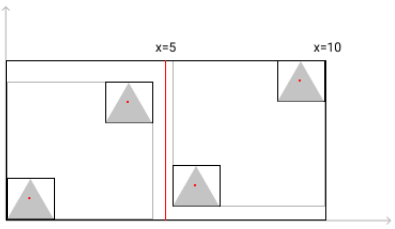


Abbildung 2.14.: Anwendungsbeispiel des Middle-Splittings

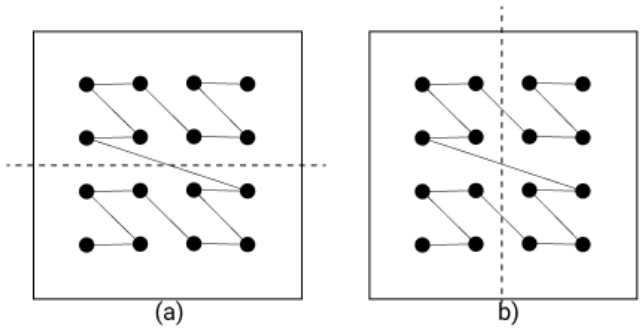
Ein Anwendungsbeispiel (Abbildung 2.14) hierfür könnte eine Bounding Box mit den Ausdehnungen $x = 10$ und $y = 5$ darstellen. Auf der x-Achse liegt hierbei die längste Ausdehnung, wodurch alle Primitive deren Mittelpunkte der Bounding Box mit $x < 5$ dem linken und alle Mittelpunkte mit $x \geq 5$ dem rechten Subset hinzugefügt werden. Analog gilt dies auch für Bounding Boxes mit drei Dimensionen.

EqualCounts-Methode

Bei der EqualCounts-Methode wird das aktuelle Subset der Primitive in genau gleich große Bereiche unterteilt, die daraufhin hinsichtlich der längsten räumlichen Achsenausdehnung gegeneinander sortiert werden, sodass mögliche, nachteilige Primitivverteilungen vermieden werden soll. Hierbei ist zu erwähnen, dass diese als alleinige Splitting-Methode wegen der schlechten Ergebnisse kaum Relevanz bei der späteren Evaluation hätte und sie daher für die Evaluation nicht betrachtet wird und hier nur aus Gründen der Vollständigkeit erwähnt wird. Für die verschiedenen anderen Splitting-Methoden wie die Middle- oder die SAH-Methode kann der Ansatz der EqualCounts-Methode für wenige innerhalb einer Bounding Box verbleibende Primitive angewandt werden (Pharr u. a., 2016, S. 262).

Hierarchical Bounding Volume Hierarchy (HLBVH)

Die HLBVH ist durch ihre Abweichung vom Top-Down-Verfahren der anderen Methoden und mit Parallelisierung vor allem für Umsetzungen auf der Graphikkarte relevant (Pharr u. a., 2016, S. 268). Die Methode wird hierbei durch die Morton-Codes realisiert, welche es ermöglichen, die mehrdimensionalen Koordinaten als Sortierproblem abzubilden (Pharr u. a., 2016, S. 268). Die Morton-Codes stellen die Koordinaten der relevanten Punkte durch Binärcodierung dar, wodurch eine entsprechende z-förmige Kurve entsteht (Pharr u. a., 2016, S. 268). Auf Basis dieser Kurve können dann die Split-Ebenen ermittelt werden (Abbildung 2.15).



(a) Horizontale Splitting-Ebene (b) Vertikale Splitting-Ebene

Abbildung 2.15.: Beispiel für die Splitting-Ebenen bei der Verwendung von Morton-Codes (Online-Abbildung: Morton-Code basiertes Splitting nach Pharr u. a. (2016), S. 270; URL: `http://www.pbr-book.org/3ed-2018/Primitives_and_Intersection_Acceleration/Bounding_Volume_Hierarchies.html`)

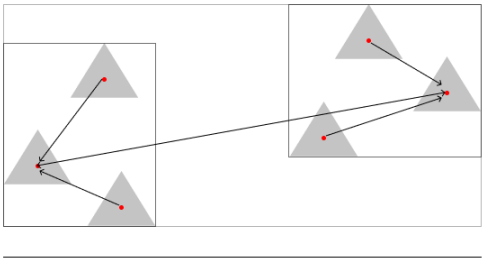


Abbildung 2.16.: Beispiel für den distanzbasierten Ansatzes (Sauer, 2019)

Distanzbasierter Ansatz

Der distanzbasierte Ansatz (Abbildung 2.16) wurde innerhalb der Praxisprojektarbeit (Sauer, 2019) als alternative Splitting-Methode für die klassische, aus Pharr u. a. (2016) bekannte Bounding Volume Hierarchy entwickelt. Hierbei werden die zwei Referenzprimitive mit der größten Entfernung zueinander ermittelt (Sauer, 2019). Danach werden die restlichen Primitive dem jeweils näheren Referenzprimitiv zugeordnet, um letztlich die Bounding Boxes für den derzeitigen Split zu erstellen und in den Binärbaum einordnen zu können (Sauer, 2019).

Allerdings wurde in dieser Arbeit eher primitiven Prinzipien gefolgt und weniger auf Techniken eingegangen, die durch ihre mathematischen Eigenschaften Vorteile hinsichtlich der Qualität der entstehenden Ergebnisse erbracht hätten. So wurden, um möglichst kurze Erstellungszeiten gewährleisten zu können, für die Ermittlung der entferntesten Primitive die Minimal- und Maximalwerte in den jeweiligen Dimensionen (Abbildung 2.17) ermittelt und diese hinsichtlich des längsten Distanzvektors mittels Brute-Force-Verfahren verglichen (Sauer, 2019). Dies erbringt im Vergleich zu anderen Methoden zwar Vorteile in der Komplexität, aber einen geringen, wenn auch ver-

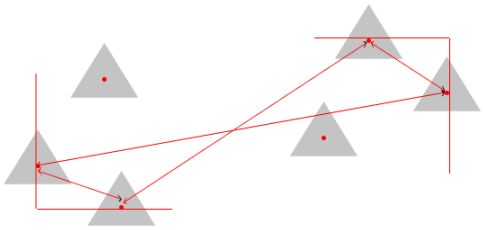


Abbildung 2.17.: Ermittlung der entferntesten Primitive (Sauer, 2019)

nachlässigbaren Fehler (Sauer, 2019). Bei der Zuordnung der restlichen Primitive wurde ein naiver Ansatz (Abbildung 2.18) gegenüber einem Ansatz der Raumpartitionierung mittels der Auswertung der jeweiligen Determinante zur Split-Ebene vorgezogen, da ein entsprechender Proof-of-Concept für die erstgenannte Variante kürzere Erstellungszeiten ergab (Sauer, 2019). Somit wurden für das aktuelle Primitiv die Distanzvektoren zu beiden Referenzprimitiven ermittelt und das Primitiv dem Referenzprimitiv zugeordnet, zu dem der jeweilige Distanzvektor längenmäßig kürzer war (Sauer, 2019).

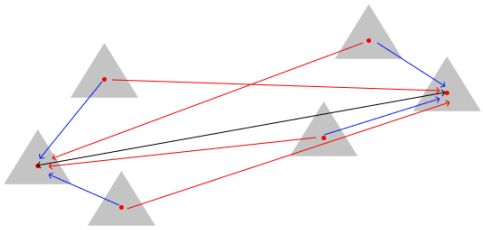


Abbildung 2.18.: Ermittlung des näheren Referenzprimitives (Sauer, 2019)

Die Evaluation ergab, dass sich der Ansatz eher für den Einsatz bei Landschafts- oder Raumszenen eignet und weniger gut für einzelne 3D-Modelle funktioniert. Ein Grund hierfür lag in den für die einzelnen Modelle stärker auftretenden Überlappungen der Bounding Boxes und der damit verbundenen erhöhten Anzahl an Traversierungsschritten sowie den entsprechenden mathematischen Berechnungen (Sauer, 2019).

2.3. Hauptkomponentenanalyse

Innerhalb dieses Abschnitts werden die anwendungsbezogene Vorgehensweise sowie der mathematische Hintergrund der Hauptkomponentenanalyse dargestellt. Dabei wird bei der Vorgehensweise hauptsächlich darauf eingegangen wie die Daten verarbeitet werden, um daraus mittels der Eigenvector Decomposition die Transformationsmatrix zu generieren und die Daten in der neuen Basis bewerten zu können. Der mathematische Hintergrund, welcher unter anderem auch die Eigenvector Decomposition umfasst, wird durch die verschiedenen zugrundeliegenden Konzepte dargestellt. Es gilt festzuhalten, dass der mathematische Hintergrund nicht zwingend notwendig für das weitere Verständnis der folgenden Arbeit ist, allerdings aus Gründen der Vollständigkeit mit-

einbezogen wird. Einen noch umfassenderen Einblick in die mathematischen Grundlagen bieten hierbei unter anderem Shlens (2014) sowie Wold u. a. (1987).

Die Hauptkomponentenanalyse, auch PCA (Principal Component Analysis), ist ein statistisches Verfahren, welches mittels einer orthogonalen Transformation Daten in eine neue Basis überführt, welche möglicherweise zugrundeliegende Strukturen mittels der Hauptkomponenten ausdrücken soll (Shlens, 2014). Eines der Ziele der Hauptkomponentenanalyse ist die Dimensionsreduktion, wodurch auch die Bedingung gegeben ist, dass die Anzahl der entstehenden Hauptkomponenten kleiner ist als die Anzahl der ursprünglichen Variablen (Shlens, 2014).

2.3.1. Anwendungsbezogenes Vorgehen

Um die PCA durchführen zu können, müssen zunächst entsprechende Daten vorhanden sein (Smith, 2002). Diese Daten können abhängig von dem jeweiligen Anwendungsgebiet durch Messungen oder Untersuchungen erhoben werden, aber auch innerhalb von Bilder für Gesichtserkennung (Yang u. a., 2004) vorliegen.

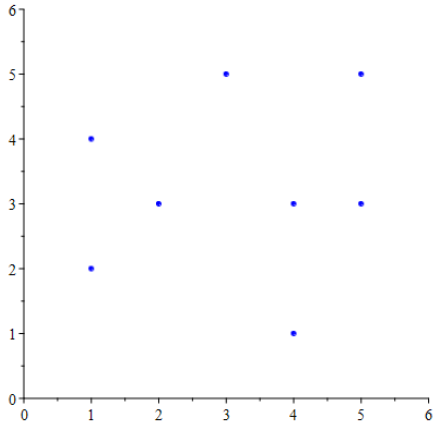


Abbildung 2.19.: Ursprüngliche Beispieldaten für die Visualisierung der PCA

Bevor die PCA durchgeführt werden kann, gilt es die Originaldaten, wie sie in Abbildung 2.19 vorliegen, noch vorzuverarbeiten. Dies umfasst zunächst die Zentrierung (Abbildung 2.20 & 2.21) sowie die Normalisierung der Input-Daten. Die Zentrierung dient dazu, die Daten um den Ursprung auszurichten und so die spätere Transformation zu vereinfachen. Die Zentrierung erfolgt, indem für jede Variable das arithmetische Mittel der selbigen gebildet wird und von den einzelnen Ergebniswerten abgezogen wird.

Des Weiteren wird durch die Zentrierung die Normalisierung ermöglicht. Die Normalisierung dient dazu, um Werte aus unterschiedlichen Skalen vergleichbar zu machen (Shlens, 2014). Als Beispiel können Betrachtungen von Fahrzeugen mit der Korrelation von der Anzahl der Reifen und der maximalen Geschwindigkeit genutzt werden. So bewegt sich die Skala der Anzahl der Reifen von zwei Reifen für Motorräder über vier Reifen für klassische PKWs oder Rennwägen bis hin zu 18 oder mehr Reifen bei LKWs. Die Geschwindigkeit hingegen kann sich im Rahmen von 0 bis 350 km/h bewegen. Klassischerweise sind somit beide Skalen nicht direkt miteinander vergleichbar.

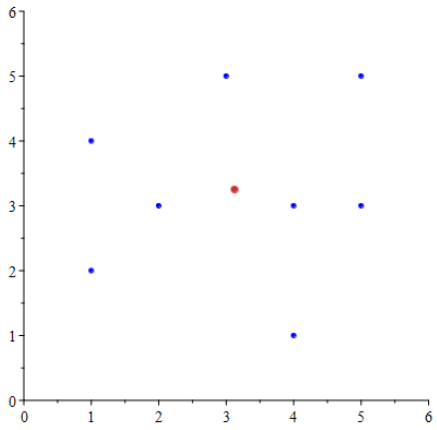


Abbildung 2.20.: Errechneter Mittelwert aus den Originaldaten als orangener Punkt dargestellt

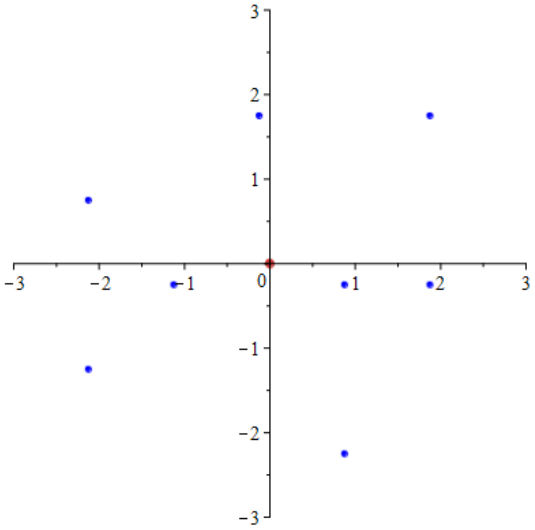


Abbildung 2.21.: Im Ursprung zentrierte Beispieldaten

Die Normalisierung allerdings macht die beiden Skalen vergleichbarer (Shlens, 2014). Für Variablen, die in gleichen Skalen vorliegen, kann der Schritt der Normalisierung entfallen (Shlens, 2014). Ein Beispiel hierfür wären dreidimensionale Koordinaten aus dem karthesischen Koordinatensystem.

Nach der Vorverarbeitung der Eingabedaten, wie sie in Abbildung 2.21 vorliegen, kann die Hauptkomponentenanalyse durchgeführt werden, um so die Basis zu ermitteln, welche die Daten am präzisesten repräsentiert (Shlens, 2014).

Hierzu kann die Kovarianzmatrix ermittelt werden, welche die Kovarianzen der Variablen für den kompletten Datensatz enthält (Smith, 2002). Mittels dieser Kovarianzmatrix können sowohl die Eigenvektoren (Abbildung 2.22) wie auch die Eigenwerte ermittelt werden. Die Eigenvektoren richten sich hierbei entlang der bestpassendsten

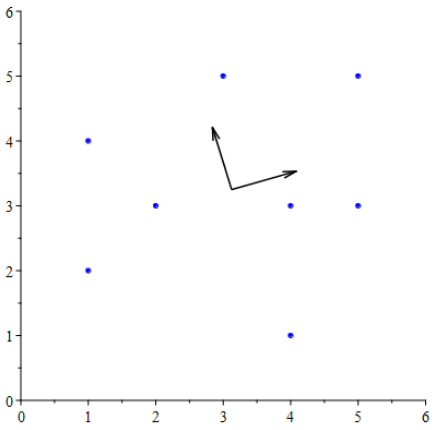


Abbildung 2.22.: Beispielhafte Originaldaten, in denen die durch die Kovarianzmatrix ermittelten Eigenvektoren an Mittelpunkt der Daten visualisiert sind.

Linien innerhalb der Daten aus (Smith, 2002). Die Hauptkomponenten entlang der Eigenvektoren sind in Abbildung 2.23 visualisiert dargestellt.

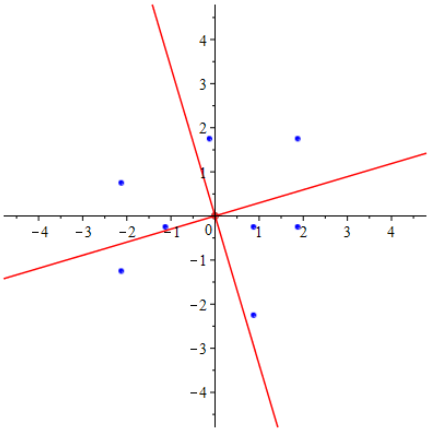


Abbildung 2.23.: Zentrierte Daten mit den Hauptkomponentenachsen

Eine wichtige Eigenschaft im Zusammenhang von Eigenvektor und Eigenwert ergibt sich vor allem für eine möglicherweise angestrebte Dimensionsreduktion, denn die Eigenwerte bestimmen durch ihre Größe die Rangfolge der Eigenvektoren als Hauptkomponenten (Smith, 2002). Der Eigenvektor mit dem höchsten Eigenwert ist die erste Hauptkomponenten, der Eigenvektor mit dem zweithöchsten Eigenwert ist die zweite Hauptkomponenten und so weiter. Die Dimensionsreduktion kann nun herbeigeführt werden, indem die Hauptkomponenten ab dem Rang, der als Zieldimension ausgewählt ist, verworfen werden. Wenn somit n Variablen in den Ursprungsdaten vorliegen, als Anzahl der Zielvariablen allerdings nur drei Variablen erhalten bleiben sollen, so werden nur die drei Hauptkomponenten mit den drei höchsten Eigenwerten genutzt. Die Eigenvektoren werden dann als Reihenvektoren in der Transformationsmatrix genutzt.

Im letzten Schritt werden die Daten transformiert, indem die Matrizen miteinander entsprechend der Formel $PX = Y$ nach Shlens (2014) multipliziert werden. X beschreibt die Ursprungsdaten in einer $m \times n$-Matrix, P die Transformationsmatrix und Y die Ergebnismatrix mit den durch die Hauptkomponentenanalyse transformierten Werte.

2.3.2. Mathematischer Hintergrund

Für die im Folgenden dargestellten Zusammenhänge werden die drei Annahmen festgelegt, dass Linearität vorliegen muss, größere Varianz auch gleichzeitig eine hohe Aussagekraft besitzt sowie Orthogonalität zwischen den Hauptkomponenten besteht (Shlens, 2014).

Hierfür können noch verschiedene Aussagen getroffen werden. Für die Eingabedaten dient die PCA dazu, um die ursprünglich in der naiven, orthogonalen Basis aufgezeichneten Daten in eine neue Basis zu überführen, die den Zusammenhang der Daten besser wieder gibt (Shlens, 2014). Die ursprüngliche Basis lässt sich jeweils als Identitätsmatrix in m-Dimensionen ausdrücken. Folglich ergibt sich nach Shlens (2014) die folgende $m \times m$-Identitätsmatrix:

$$\boldsymbol{B} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} = \boldsymbol{I}$$

Hierbei bilden die jeweiligen Reihen-Vektoren die orthonormalen Basis-Vektoren (Shlens, 2014). Dies lässt sich unter anderem auch an der 3×3-Identitätsmatrix feststellen, welche die orthogonalen Vektoren $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ und $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ des karthesischen Koordinatensystems beinhaltet.

Da ein Ziel der PCA die Transformation von Daten in eine neue Basis ist, muss zur Transformation für die 3×3-Identitätsmatrix ebenfalls eine 3×3-Matrix ermittelt werden, welche die orthonormalen Vektoren der neuen Basis enthält (Shlens, 2014).

Die Zielgleichung der PCA lautet somit nach Shlens (2014):

$$\boldsymbol{PX} = \boldsymbol{Y}$$

Hierbei beschreibt $\boldsymbol{X}$ die $m \times n$-Matrix der Eingabedaten, wobei m die Zahl der Variablen und n die Anzahl der vorhandenen Datensätze ist, $\boldsymbol{Y}$ stellt die Eingabedaten in der neuen, durch Transformation geänderten Basis dar und $\boldsymbol{P}$ bezeichnet die Transformationsmatrix, welche die Eigenvektoren, die die Basisvektoren des neuen Koordinatensystems darstellen, als Reihenvektoren beinhaltet (Shlens, 2014). Für die Matrizen $\boldsymbol{X}$ und $\boldsymbol{Y}$ bedeutet dies im vorliegenden Anwendungskontext, dass $m = 3$ für die drei Dimensionen x, y und z sowie $n =<$ Anzahl der Primitive $>$ gilt. Die Matrix $\boldsymbol{P}$ stellt somit eine 3×3-Matrix für die drei später entstehenden Eigenvektoren dar.

Für Betrachtungsfälle mit zwei Variablen respektive zwei dadurch gegebenen Dimensionen kann der Anstieg der bestpassendsten Linie auch graphisch bestimmt werden (Shlens, 2014). Um allerdings auch für höherdimensionale Ergebnisse eine Annäherung erzielen können, müssen die Varianz sowie die Kovarianz betrachtet werden(Shlens, 2014). Die Varianz der Ergebnissets von A und B sind $\sigma_a^2 = \frac{1}{n}\sum_i a_i^2$ respektive $\sigma_b^2 = \frac{1}{n}\sum_i b_i^2$. Dementsprechend setzt sich nach (Shlens, 2014) die Kovarianz von A und B wie folgt zusammen:

$$\sigma_{AB}^2 = \frac{1}{n}\sum_i a_i b_i$$

Durch Umformung der Ergebnissets A und B in die Vektoren a und b kann die Kovarianz auch als Skalarprodukt entsprechend ausgedrückt werden:

$$\sigma_{AB}^2 = \frac{1}{n}\boldsymbol{a}\boldsymbol{b}^{\boldsymbol{T}}$$

Basierend auf dieser Erkenntnis im Rahmen der Verwendung zweier Vektoren können weitere Vektoren und somit weitere Variablen betrachtet werden. Hierzu kann eine neue $m \times n$-Matrix X erstellt werden, deren Zeilen allen gemessenen Werten einer Variable entspricht. Dadurch können in der zuvor genannten Gleichung für die Kovarianz die beiden Vektoren a und b jeweils durch die Matrix X ersetzt werden, wodurch nach (Shlens, 2014) die Definition der Kovarianzmatrix C_X wie folgt lautet:

$$C_X = \frac{1}{n}XX^T$$

Diese Kovarianzmatrix drückt hierbei aus wie die einzelnen Variablen miteinander korrelieren (Shlens, 2014).

Um im weiteren Verfahren die PCA zu lösen, bieten sich zwei Verfahren an. Bei der ersten Variante handelt es sich um die Singular Value Decomposition (SVD), bei der zweiten Variante um die Eigenvector Decomposition (EVD). Da innerhalb des vorherigen Unterabschnitts sowie in der späteren Verwendung hauptsächlich die Eigenvector Decomposition Verwendung findet, wird zunächst die Singular Value Decomposition nur sehr kurz aus Gründen der Vollständigkeit algebraisch dargestellt. Im Anschluss wird die Eigenvector Decomposition ausführlicher dargestellt.

Bei der Singular Value Decomposition, auch Singulärwertzerlegung, wird durch verschiedene Annahmen (Shlens, 2014) die folgende Gleichung abgeleitet:

$$X = U\Sigma V^T$$

Diese besagt, dass jegliche Matrix X in einer orthogonalen Matrix U, einer diagonalen Matrix Σ aus den Wurzeln der Eigenwerte und einer weiteren orthogonalen Matrix V^T aus den Eigenvektoren abgebildet werden kann (Shlens, 2014). Mittels weiterer Umformung kann die Gleichung in $U^T X = Z$ umgewandelt werden, wobei Z die Multiplikation aus Σ und V^T darstellt und U^T die selbe Rolle spielt wie die Matrix P zuvor, wodurch die Reihenvektoren von U^T die Eigenvektoren für die neue Basis und somit auch die Hauptkomponenten darstellen (Shlens, 2014).

Bei der Eigenvector Decomposition besteht das Ziel in der Findung einer orthonormalen Matrix P aus der bereits bekannten Gleichung $Y = PX$, für welche die Matrix

$C_Y = \frac{1}{n} YY^T$ eine diagonale Matrix ist (Shlens, 2014). Die Reihenvektoren der Matrix P stellen dann die Hauptkomponenten der Ursprungsmatrix X dar.

Zum Beweis kann die Matrix C_Y durch Äquivalenzumformung nach (Shlens, 2014) wie folgt umgeformt werden:

$$\begin{aligned} C_Y &= \frac{1}{n} YY^T \\ &= \frac{1}{n} (PX)(PX)^T \\ &= \frac{1}{n} PXP^T X^T \\ &= P(\frac{1}{n} XX^T)P^T \\ C_Y &= PC_X P^T \end{aligned}$$

Dabei wird im ersten Schritt die Matrix Y durch die bereits bekannte Gleichung $Y = PX$ ersetzt. Im zweiten Schritt kann der Teil der transponierten Matrix auf die Matrizen P^T und X^T sowie die sonstigen vorhandenen Klammern aufgelöst werden. Der dadurch entstehende Term kann im dritten Schritt so vereinfacht werden, dass die für die Kovarianzmatrix C_X substituierbare Matrix X sowie deren transponierte Matrix X^T und der Kehrbruch zur Anzahl n zusammengezogen werden und im Folgenden durch die Kovarianzmatrix C_X ersetzt wird. Des Weiteren entspricht die Inverse der Matrix P der Transponierten der Matrix P, wenn die Matrix P selbst orthogonal ist, sodass die Gleichung $P^T = P^{-1}$ gilt (Shlens, 2014).

Shlens (2014) führt im Folgenden die Eigenschaft auf, dass jede symmetrische Matrix A aus der diagonalen sowie orthogonalen Matrix der Eigenvektoren zusammengesetzt werden kann. Demnach ergibt sich als weitere Gleichung für die Substitution im Folgenden $A = EDE^T$, wobei D die diagonale Matrix darstellt und die Matrix E die Eigenvektoren der Matrix A als Spalten beinhaltet (Shlens, 2014). Diese Gleichung wird genutzt, um die Kovarianzmatrix C_X zu substituieren. Außerdem kann für die Reihen der Matrix P festgelegt werden, dass diese die Eigenvektoren der Kovarianzmatrix von X darstellen, wodurch sich die Gleichung $P \equiv E^T$ ergibt (Shlens, 2014). Dadurch lässt sich die folgende Umformung durchführen (Shlens, 2014):

$$\begin{aligned} C_Y &= PC_X P^T \\ &= P(E^T DE)P^T \\ &= P(P^T DP)P^T \\ &= (PP^T)D(PP^T) \\ &= (PP^{-1})D(PP^{-1}) \\ C_Y &= D \end{aligned}$$

Zunächst kann die zuvor genannte Substitution der Kovarianzmatrix C_X durch $E^T DE$ erfolgen. Die Eigenvektorenmatrix E kann durch die Transponierte der Matrix P ersetzt werden, genauso wie die Transponierte E^T durch die Matrix P ersetzt

werden kann. Darauffolgend können die beiden Matrizen P sowie die Transponierten von P als Terme zusammengefasst werden, wobei die Transponierte nach der zuvor genannten Gleichung bei orthonormalen Matrizen der Inversen entspricht und somit P^T durch P^{-1} ersetzt wird, wodurch bei der Multiplikation der Matrix P mit der inversen Matrix P^{-1} die Identitätsmatrix I entsteht, welche im vorliegenden Fall entfallen kann. Somit wird durch die Wahl der Matrix P die Kovarianzmatrix C_Y diagonalisiert (Shlens, 2014).

3. Related Work

Innerhalb dieses Kapitels werden verwandte wissenschaftliche Beiträge präsentiert.

Raytracing und Hauptkomponentenanalyse Das Raytracing wurde zuerst von Whitted (1979) beschrieben und wurde durch Glassner (1989) vor allem theoretisch behandelt sowie durch Pharr u. a. (2016) innerhalb eines beispelhaften Raytracers praktisch implementiert. Die Hauptkomponentenanalyse stellt ein statistisches Verfahren dar, welches unter anderem von Shlens (2014), Smith (2002), Wold u. a. (1987) und Jolliffe (2011) behandelt wird. Während Shlens (2014) den mathematischen Hintergrund sowie entsprechende Beweise und Annahmen darlegt, beschreibt Smith (2002) die Hauptkomponentenanalyse im Anwendungsbezug mit detaillierter Beschreibung der einzelnen Schritte zum Erhalt der transformierten Daten.

Weiterentwicklungen der Bounding Volume Hierarchy Eine Möglichkeit in der Weiterentwicklung der BVH besteht, anders als beispielsweise in Sauer (2019) durch eine neue Splitting-Variante oder innerhalb dieser Arbeit durch die Transformation der Eingabedaten für das Splitting, auch darin, die Sichtbarkeit der einzelnen Knoten und Primitive für die BVH miteinzubeziehen und somit eine entsprechende Beschleunigung des Verfahrens zu bewirken (Vinkler u. a., 2012). Auch Bounding Volume Hierarchies für GPU-gestütztes Raytracing werden mit den für die Randbedingungen von GPUs angepassten Optimierungen verbessert wie beispielsweise durch die Grid-based BVHs von Garanzha u. a. (2011), welche neben der zur Erstellung der Bounding Volumes auch einmalig ein Grid erstellen, welches die Primitive wiederum bezüglich der Bounding Boxes an das Raster anpasst. Für speziellere Anwendungsfälle wie das anspruchsvolle Rendering von fluiden Materialien bei animierten Szenen können die von Gourmel u. a. (2010) entwickelten BVHs genutzt werden. Eine Verbesserungsmöglichkeit für Bounding Volume Hierarchies mit einer durch große Überlappungen hervorgerufenen, erhöhten Anzahl an Traversierungsschritten ist die Verwendung des Early-Split-Clippings (Ernst u. Greiner, 2007) oder der Weiterentwicklung des Verfahrens in Form der Split-BVH (Stich u. a., 2009). Mittels des Approximate Agglomerative Clusterings kann statistisch ähnlich wie mit der PCA eine BVH erstellt werden, wobei die Zuteilung der Primitive in die Bounding Volumes über die räumliche Nähe zueinander erfolgt (Gu u. a., 2013).

Klassische Anwendungsgebiete der Hauptkomponentenanalyse Der Verwendungszweck der PCA bezieht sich auf die Dimensionsreduktion von mehrdimensionalen Daten und damit einhergehend auch auf die Findung wichtiger Strukturen und Zusammenhänge, die den Ursprungsdaten im Verborgenen zugrundeliegen. Daher findet sich die PCA vor allem in den Experimentalwissenschaften der Biologie (Novembre u.

Stephens, 2008), Chemie (Christensen u. a., 2005) und Physik (Marinetti u. a., 2004) wieder, aber auch in Anwendungsbereichen der Informatik wie beispielsweise der Gesichtserkennung aus Gottumukkal u. Asari (2004) und aus Yang u. a. (2004).

4. Entwicklung und Evaluation

Innerhalb dieses Kapitels werden die theoretische und praktische Entwicklung sowie die Evaluation der auf der Hauptkomponentenanalyse basierenden Bounding Volume Hierarchy dargestellt.

4.1. Theoretische Skizzierung

Zunächst wird in diesem Abschnitt der Ansatz, eine Bounding Volume Hierarchy mittels der Hauptkomponentenanalyse zu erstellen, theoretisch skizziert. Hierzu wird auf den Grundgedanken des Ansatzes verwiesen. Daraufhin wird die Verwendung der Hauptkomponentenanalyse erläutert sowie die nach der Transformation zu verwendenden Splitting-Methoden diskutiert.

4.1.1. Grundgedanke des Ansatzes

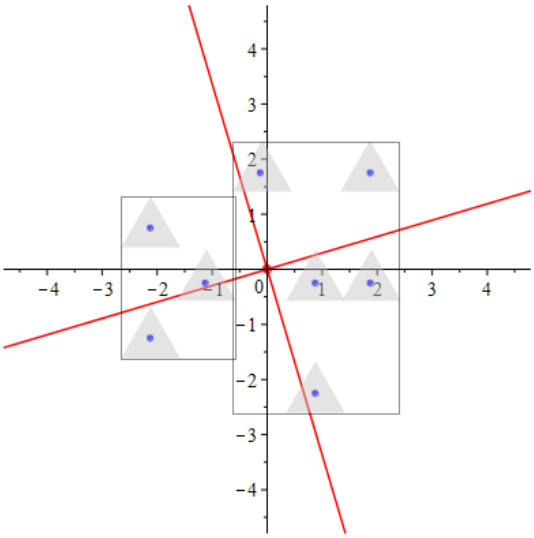


Abbildung 4.1.: Visualisierung des Ergebnisses der Hauptkomponentenanalyse mit beispielhaften Bounding Boxes

Der Grundgedanke dieses Ansatzes (Abbildung 4.1) basiert auf der Verwendung der Hauptkomponentenanalyse für die Transformation der Eingabepunkte in eine neue Basis. Für die Verwendung bieten sich zwei Möglichkeiten an. Diese umfassen zum einen die einmalige Verwendung der PCA vor der Erzeugung des Binärbaums und zum anderen die mehrmalige Verwendung im rekursiven Splitting während der Erstellung des

Binärbaums. Bei der einmaligen Verwendung der PCA werden die Eingabemittelpunkte entlang der sich ergebenden Eigenvektoren ausgerichtet, wodurch sich eine distanzorientierte Verteilung ergibt, welche unter Verwendung klassischer Splitting-Methoden starke Analogien zum distanzbasierten Ansatz (Sauer, 2019) aufweist. Die mehrmalige Verwendung hingegen kann eine neue Perspektive in dieser Hinsicht schaffen, indem für jedes entstehende Subset die PCA durchgeführt wird. Hierdurch wird der Mittelpunkt jeder Bounding Box eines Primitivs in einer neuen Basis beschrieben, in der zur rekursiven Erstellung des neuen Subsets klassische Splitting-Methoden verwendet werden können. Daher wird im Folgenden von der zweiten Variante, also der mehrfachen Verwendung der PCA, als Grundlage der auf der Hauptkomponentenanalyse basierenden Bounding Volume Hierarchy ausgegangen, welche im restlichen Teil der Arbeit vereinfachend als PCA-BVH bezeichnet wird.

4.1.2. Verwendung der Hauptkomponentenanalyse

Die PCA wird gemäß des vorherigen Unterabschnitts mehrmals verwendet, indem vor jedem Split die Hauptkomponenten des aktuellen Subsets ermittelt werden und gemäß dieser die Transformation durchgeführt wird. Als Eingabedaten für die PCA werden hierbei nicht die Vertices der Primitive genutzt, sondern die Mittelpunkte der Bounding Boxes der Primitive. Auch sich vom Mittelpunkt unterscheidende Sample-Punkte werden nicht genutzt, da das Splitting sich am Mittelpunkt der Bounding Box des Primitivs ausrichtet. Die Nutzung von derartigen Sample-Punkten wäre nur dann zu erwägen, wenn gleichzeitig die Split-BVH von Stich u. a. (2009) genutzt wird, welche dafür sorgt, dass große Primitive in kleinere Teil-Bounding Boxes zerteilt wird, wodurch die Aufteilung mit minimierten Überlappungen geschehen kann. Allerdings müsste auch hier eine enge Abstimmung zwischen den gesetzten Sample-Punkten sowie den entstehenden Bounding Boxes getroffen werden.

Die Mittelpunkte müssen vor der Durchführung der PCA vorverarbeitet werden. Dies geschieht für die Primitive, indem das arithmetische Mittel aus den kumulierten Mittelpunkten als neuer Ursprung für die zentralisierten Mittelpunkte genutzt wird. Dementsprechend müssen die Zentroide mittels des dadurch entstehenden Vektors translatiert werden. Die Normalisierung der Daten kann im vorliegenden Anwendungsfall entfallen, da die Daten in der selben Skala, also dem karthesischen Koordinatensystem, vorliegen.

Nach der Vorverarbeitung der Primitive kann die Hauptkomponentenanalyse angewandt werden. Hierzu wird aus den x-, y- und z-Werten der Mittelpunkte die Kovarianzmatrix ermittelt. Diese Kovarianzmatrix wird innerhalb der Eigenvector Decomposition dazu genutzt, um die Eigenvektoren sowie Eigenwerte zu berechnen. Im vorliegenden Kontext werden die Eigenwerte zur Bewertung der Aussagekraft der Hauptkomponente nicht benötigt, da alle drei Dimensionen erhalten bleiben sollen und lediglich die Transformation in eine neue Basis erfolgen soll. Die Eigenvektoren dienen als neue Basis für die ausgedrückten Mittelpunkte, sodass sie als Reihenvektoren der Transformationsmatrix genutzt werden, um die Transformation zu gewährleisten. Die Matrixmultiplikation der Transformationsmatrix mit der Matrix, bestehend aus den Ursprungsdaten, führt zur Ergebnismatrix, welche die transformierten Punkte in der neuen Basis enthält.

Anschließend können, wie im folgenden Unterabschnitt beschrieben, diese transformierten Punkte mittels klassischer Splitting-Methoden aufgeteilt werden.

4.1.3. Verwendung verschiedener Splitting-Methoden

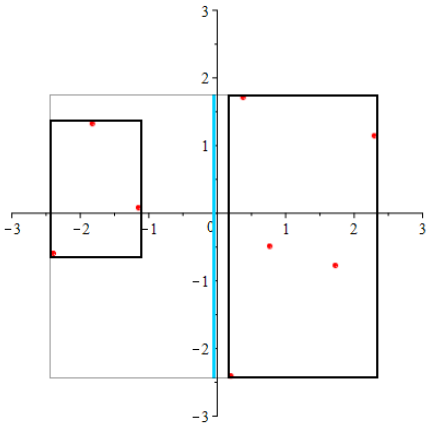


Abbildung 4.2.: Beispielhafte Darstellung für Middle-Splitting bei durch PCA-transformierten Daten

Die Middle-Methode (Abbildung 4.2) eignet sich als relativ simple Splitting-Methode für schnelle Erstellungszeiten. Die zugrundeliegende Qualität der BVH ist allerdings niedriger als die Qualität der SAH-Methode, was sich unter anderem auch in einer etwas geringeren Rendergeschwindigkeit und somit höheren Laufzeit ausdrückt (Sauer, 2019). Allerdings kann die Methode trotzdem im Rahmen der PCA-BVH genutzt werden und produziert partiell einen ähnlichen Ansatz wie der aus Sauer (2019) bekannte, distanzbasierte Ansatz, da die neu ausgerichteten Mittelpunkte entlang der längsten Ausdehnung des Knotens mittig aufgeteilt werden, entsteht so näherungsweise eine Zuordnung, welche die entferntesten Primitive in unterschiedliche Subsets aufteilt und die näheren Primitive jeweils in das entsprechende Subset eingliedert.

Des Weiteren kann auch die SAH-Methode implementiert werden, da diese bei längeren Erstellungszeiten qualitativ höherwertige BVHs erzeugen kann sowie im Allgemeinen die beste Partitionierungstechnik darstellt (Pharr u. a., 2016, S. 263 - 264). Auch die EqualCounts-Methode wäre eine valide Möglichkeit, die sich aufgrund ihrer zeitlichen Nachteile in der in Sauer (2019) durchgeführten Evaluation als nicht relevant erwiesen hat.

Die Verwendung eines HLBVH-Ansatzes bietet sich nicht an, da der hier verwendete Ansatz nicht rekursiv anwendbar ist, wie beispielsweise die Middle-Methode oder die SAH-Methode, wodurch auch das Ziel der HLBVH, die Erstellung des Binärbaums möglichst stark zu parallelisieren, durch den bei Verwendung der PCA entstehenden Zwang zur Top-Down-Verfahrensweise nicht gewahrt werden kann.

Als Erweiterung für die dargestellten Splitting-Methoden kann auch eine kombinierte Verwendung der SAH-Methode in Betracht gezogen werden. Dies bedeutet, dass für die Splitting-Methode nicht nur die durch die PCA produzierten Achsen benutzt werden, sondern auch die ursprünglichen Koordinatenachsen miteinbezogen werden. Dies soll

in der Theorie sicherstellen, dass jeweils die Partitionierung mit den günstigsten Kosten ausgewählt wird, um zur klassischen SAH-Methode ein mindestens gleichwertiges Ergebnis zu erhalten.

Als Fazit lässt sich somit festhalten, dass neben der Middle-Methode für eine schnelle Erstellungsdauer, die SAH-Methode für eine qualitativ hochwertige BVH sowie die kombinierte SAH-Methode für die Implementierung innerhalb dieser Arbeit verwendet wird.

4.2. Implementierung der Bounding Volume Hierarchy

Innerhalb dieses Abschnitts wird die Implementierung der Bounding Volume Hierarchy erläutert. Dies beinhaltet neben einer Einführung zum verwendeten Raytracer und der Implementierung der Hauptkomponentenanalyse auch die Implementierung der verschiedenen Splitting-Methoden.

4.2.1. Verwendung des PBRT

Beim innerhalb dieser Arbeit verwendeten Raytracer handelt es sich um den Physically Based Raytracer (PBRT) aus dem Referenzwerk von Pharr u. a. (2016). Die für die Programmierung mit dem Raytracer zu verwendende Sprache ist C++. Vorteilhaft bei der Verwendung des Raytracers ist neben der ausführlichen Dokumentation und Modularität auch die Kontrolle der zu verwendenden Mechanismen über die Randparameter des speziell für diesen Raytracer entwickelten pbrt-Fileformats. Durch Austausch des innerhalb des Listing 4.1 zu sehenden Accelerator-Keywords kann die innerhalb dieser Arbeit vorgestellte PCA-BVH als Beschleunigungsdatenstruktur für das Rendering festgelegt werden. Für die Verwendung der PCA-basierten Bounding Volume Hierarchy wird das Schlüsselwort `pcabvh` genutzt. Der Parameter `splitmethod` definiert die zu verwendende Splitting-Methode. Im Zusammenhang mit der PCA-BVH sind die Werte `middle` für das Middle-Splitting, `sah` für die SAH-basierte Splitting-Methode mit den Hauptkomponenten als Achsen sowie `sahcomb` für die kombinierte Methode aus der klassischen SAH- sowie der PCA-basierten SAH-Methode verfügbar. Andere Splitting-Methoden sind, wie in Unterabschnitt 4.1.3 angeführt, nicht verfügbar.

```
Camera "perspective" "float fov" 45

Sampler "halton" "integer pixelsamples" 128
Accelerator "pcabvh" "string splitmethod" "sah"

Integrator "path"
Film "image" "string filename" "simple.png"
     "integer xresolution" [400] "integer yresolution" [400]
```

Listing 4.1: Ausschnitt aus einem pbrt-File

Als Vorbereitung für die Implementierung muss zunächst in der `api.cpp`-Datei die `MakeAccelerator`-Funktion um den „pcabvh“-Parameter als `if`-Bedingung erweitert werden (Listing B.1).

Die Erstellungsfunktion `CreatePCAAccelerator` (Listing B.2) für die Klasse `PCAAccel` wird hier analog zu den anderen Funktionen mit der Übergabe der Primitive sowie dem geparsten Parameterset übergeben. Dieses Parameterset enthält unter anderem die zu verwendende Splitting-Methode als `string` und wird in der Methode auf die für der BVH zur Verfügung stehenden Splitting-Methoden geprüft.

Die `recursiveBuild`-Funktion wird im Folgenden dazu genutzt, die der BVH zugrundeliegende Baumstruktur rekursiv zu erstellen. Hierbei wird hauptsächlich zwischen der Erstellung von inneren Knoten und von Blattknoten unterschieden. Die Blattknoten werden hierbei gebildet, wenn eine ausreichend granulare Anzahl von standardmäßig einem bis vier Primitiven innerhalb der aktuellen Verteilung vorliegt. Für die Erstellung von inneren Knoten werden die Splitting-Methoden, welche unter anderem in Unterabschnitt 4.1.3 vorgestellt wurden, genutzt, sodass die Primitive entsprechend der verwendeten Methode zugeordnet werden und diese dann als zwei Kindknoten des inneren Knotens hinzugefügt werden. Eine entsprechende Rekursion führt für die Kindknoten zu einer weiteren Ausführung, sodass diese weiterhin entweder zu inneren oder zu Blattknoten werden.

4.2.2. Implementierung der Hauptkomponentenanalyse

Für die Fehlervermeidung bei der Implementierung der PCA wird die sogenannte „Eigen“-Bibliothek genutzt, welche Hilfsstrukturen wie mehrdimensionale Matrizen bietet und auch elegante Implementierungen für die Ermittlung der Kovarianzmatrix, Eigenwerte sowie Eigenvektoren und dementsprechend auch für die PCA beinhaltet (Guennebaud u. a., 2010). Da es sich bei dieser Library um eine reine Header-Bibliothek handelt, wurden die entsprechenden Header-Dateien in das Projekt importiert. Als Alternative wäre auch die Verwendung der CMake-Einbindung möglich gewesen, allerdings hätte dies zur Folge gehabt, die Eigen-Library notwendigerweise auf dem Nutzergerät zu installieren.

Bevor die Bounding Volumes innerhalb der `recursiveBuild`-Funktion in innere Knoten gesplittet werden, wird auf das derzeitige Subset der Primitive die PCA ausgeführt. Hierzu wird die bereits im PBRT implementierte Struktur `BVHPrimitiveInfo` um eine Variable `pcacentroid` des Typs `Point3f` erweitert, welche genutzt wird, um den Mittelpunkt der Bounding Box nach der Transformation mit den Eigenvektoren der PCA zu speichern. Eine eigene Struktur für diese Informationen mit der Referenz zum entsprechenden Primitiv zu speichern ist zwar möglich, allerdings führt dies zu gewisser Redundanz und Mehraufwand bei der späteren Erzeugung des Binärbaums innerhalb der `recursiveBuild`-Funktion, da der Funktion die komplette Liste der dadurch neu entstehenden Struktur übergeben werden muss und nach der Neuordnung dieser Strukturen muss auch die ursprüngliche Liste der Primitive angepasst werden.

Zunächst wird, wie in Listing B.3 zu sehen ist, innerhalb der `recursiveBuild`-Funktion die Zentralisierung der Daten vorgenommen, indem innerhalb der Variable `middle` die kumulierten Mittelpunkte durch die Anzahl der Primitive geteilt werden. Der Vector `mean` wird dazu genutzt, um die Variable `pcacentroid` zu zentrieren.

Die Matrix `centeredPrimMidpoints` soll die zentrierten, dreidimensionalen Mittelpunkte später mittels einer Matrixmultiplikation mit der entstehenden Transformationsmatrix in die neue Basis transformieren.

Mittels der `for`-Schleife wird die Matrix mit den Mittelpunkten der Bounding Boxes der Primitive gefüllt. Hierbei wird die aktuelle Iterationsvariable durch die Subtraktion der Variable `start` für die Matrix so manipuliert, dass die Schleife immer an Position 0 der Matrix beginnt und nicht an der Position des betrachteten Primitivs im `primitiveInfo`-Array.

Die Kovarianzmatrix der Variable `covarianceMatrix` kann durch die Multiplikation der Transponierten der `centeredPrimMidpoints`-Matrix mit der `centeredPrimMidpoints`-Matrix selbst erzeugt werden. Die hierbei entstehende Matrix wird dann durch die um eins verminderte Anzahl der Reihen gemäß der in Shlens (2014) dargestellten Gleichung für die Kovarianzmatrix geteilt.

Mittels der Eigenvector Decomposition kann der `SelfAdjointEigenSolver` für die Variable `eigen` genutzt werden, um aus der Kovarianzmatrix Eigenwerte und Eigenvektoren zu ermitteln (Guennebaud u. a., 2010). Über die Variable `eigen` können die ermittelten Eigenwerte und Eigenvektoren abgerufen werden. Im vorliegenden Beispiel werden allerdings nur die Eigenvektoren mittels der `eigenvectors`-Funktion abgerufen. Klassischerweise würden dann mittels der `rightCols`-Funktion respektive `leftCols`-Funktion die rechten bzw. linken Spalten der Matrix extrahiert werden, um so nur die aussagekräftigsten Parameter beizubehalten und damit die Dimensionsreduktion herbeizuführen (Guennebaud u. a., 2010). Im vorliegenden Fall soll allerdings keine Dimensionsreduktion durchgeführt werden, sondern lediglich die Transformation der Mittelpunkte in eine neue Basis erfolgen, welche sich durch die Multiplikation von den `centeredPrimMidpoints` und der `pcaTransformationMatrix` ergibt. In der abschließenden `for`-Schleife werden die Werte aus der Matrix dem jeweiligen Primitiv innerhalb des `primitiveInfo`-Arrays zugeordnet.

Die derzeitige Implementierung bietet hierbei noch Optimierungspotenzial. So könnte die `centeredPrimMidpoints`-Matrix mit den einzelnen Primitiven in `primitiveInfo` assoziiert werden, um dadurch neben den mehrfachen `for`-Schleifen auch die wiederkehrende Erstellung neuer Matrizen zu vermeiden. Der Nachteil an dieser Methode besteht in der relativ komplexen Aktualisierung der Reihenfolge innerhalb der `centeredPrimMidpoints`-Matrix, wenn sich die Reihenfolge innerhalb des `primitiveInfo`-Arrays ändert.

4.2.3. Implementierung der zu verwendenden Splitting-Methoden

Bei den klassischen Methoden wird zunächst durch die Vereinigung aller Mittelpunkte eine Bounding Box erzeugt, anhand derer die Dimension mit der längsten Ausdehnung ermittelt wird (Listing B.4). Diese Dimension wird im Folgenden für die Partitionierungstechniken genutzt.

Im Rahmen der Middle-Methode (Listing B.5) werden im Folgenden die Bounding Boxes der Primitive mit der `partition`-Funktion geprüft, ob deren Mittelpunkte unter- oder oberhalb des Mittelwerts der zuvor ausgewählten Dimension liegt. Die `partition`-Funktion liefert einen Pointer zu der ersten Stelle des `primitiveInfo`-Array, an welcher nach der Sortierung durch die Funktion das Primitiv steht, dessen Mittelpunkt der Bounding Box oberhalb des Mittelwerts der ausgewählten Dimension liegt (Pharr u. a., 2016, S. 261).

Bei der SAH-Methode (Listing B.6) wird, wenn die Anzahl der Primitive größer als zwei ist, zunächst die Initialisierung der Buckets vorgenommen. Hierbei werden alle Primitive gemäß ihres Bounding Box-Mittelpunkts in eines von 12 Buckets eingeteilt.

Nach der Initialisierung der Buckets werden mittels einer `for`-Schleife die entsprechenden Kosten für alle Split-Positionen zwischen den Buckets berechnet. Die verschachtelten `for`-Schleifen sorgen jeweils dafür, dass zum einen die Bounding Boxes links und rechts der aktuellen Split-Position sowie die enthaltene Anzahl an Primitiven ermittelt wird. Hierfür startet eine der `for`-Schleifen zu Beginn der Buckets und die andere `for`-Schleife nach der aktuellen Split-Position. Durch die Surface Area Heuristic werden abschließend die Kosten berechnet und in einer weiteren `for`-Schleife die geringsten Kosten ermittelt.

Für die PCA-basierten Variante der Splitting-Methoden werden statt der klassischen Mittelpunkte die transformierten Mittelpunkte der Bounding Boxes des jeweiligen Primitivs in der Variable `pcacentroid` genutzt. Dies bedeutet, dass die Variable `centroid` innerhalb der `recursiveBuild`-Funktion komplett durch die Variable `pcacentroid` ausgetauscht werden kann. Somit kann gewährleistet werden, dass die Splitting-Operationen auf den transformierten Punkten stattfinden und dementsprechend auch die Bounding Box für die PCA-Mittelpunkte erstellt werden. Dies betrifft die Berechnung der Bounds der Mittelpunkte (Listing B.4), welche für die Ermittlung der Dimension mit der längsten Ausdehnung genutzt wird, entlang derer die verschiedenen Splitting-Methoden ausgeführt werden, sowie die Vergleichspunkte der verschiedenen Partitionierungsfunktionen, wie unter anderem in den `partition`- und `nth_element`-Funktionen des Listings B.5 für die Middle-Methode und des Listings B.6 für die SAH-Methode dargestellt.

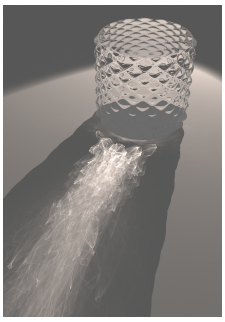
Bei der kombinierten SAH-Methode (Listing B.7) wurde die Vorgehensweise der klassischen SAH-Methode übernommen und ein zweites Mal abgebildet, sodass jeweils die Kosten für die klassische SAH-Methode sowie für die PCA-SAH-Variante berechnet werden. Für das Splitting innerhalb der `partition`-Funktion werden die beiden Werte für die niedrigsten Kosten der jeweiligen Variante verglichen, wobei über den geringeren Wert die Splitting-Position bestimmt wird.

4.3. Evaluation

Innerhalb dieses Abschnitts wird die auf der Hauptkomponentenanalyse basierende Bounding Volume Hierarchy mit den verschiedenen in Unterabschnitt 4.1.3 dargestellten Splitting-Methoden gegenüber den anderen im Grundlagenkapitel vorgestellten Splitting-Methoden mit der klassischen, bereits im PBRT implementierten BVH evaluiert. Hierzu wird es zwei Evaluationsphasen geben, welche eine umfassender angelegte, erste Evaluation sowie eine zweite, erweiternde Evaluation umfasst. Bei der ersten Evaluation werden die meisten aus Pharr u. a. (2016) bekannten Splitting-Methoden gegenüber der PCA-BVH mit dem Middle- und dem SAH-Splitting getestet. In der zweiten, erweiternden Evaluation wird die kombinierte Methode gegenüber der klassischen wie auch der PCA-basierten SAH-Methode evaluiert. Hierbei wird auch eine Variante evaluiert, die für die Kostenberechnung eine erweiterte Surface Area Heuristic nutzt, die in Unterabschnitt 4.3.3 näher erläutert wird.

4.3.1. Beschreibung der Testszenen und Metriken

(a) Krone
1000x1400
(3,5 M Prim.)



(b) Glas
1050x1500
(88 k Prim.)

(c) Buddha
750x1200
(55 k Prim.)

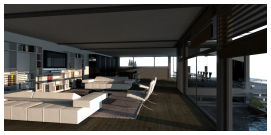
(d) San Miguel
726x1600
(2,7 M. Prim.)

(e) Badezimmer
1200x760
(449 k Prim.)

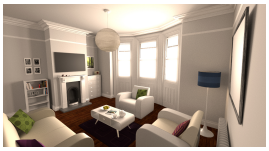
(f) Pavillion
1600x850
(5,3 M. Prim.)

(g) Buch
1280x720
(21 k Prim.)

(h) Drache
1366x1024
(7,2 M Prim.)

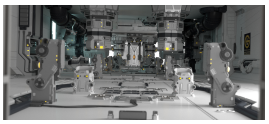


(i) Villa
1200x 580
(2,6 M Prim.)



(j) White Room
1100x600
(600 k Prim.)

(k) Audi TT
1500x833
(372 k Prim.)



(l) Measure One
1440x630
(4,5 M. Prim.)

Abbildung 4.3.: Renderbilder für die Evaluation der Bounding Volume Hierarchy (pbrt.org)

Die in Abbildung 4.3 zusammengefassten Szenen werden für die Evaluation des innerhalb dieser Arbeit entwickelten Ansatzes genutzt. Bei der Auswahl der Szenen wurde analog wie in Sauer (2019) ein Kategorisierung in Raumszenen sowie in Einzelmodelle vorgenommen. Insgesamt ergeben sich somit sechs Raumszenen sowie sechs einzelne Objekte für das Rendering. Für die verschiedenen Renderings werden jeweils 64 Pixelsamples und der Integrator „path“ für Path Tracing genutzt. Nähere Informationen zu den verschiedenen Raytracing-Techniken finden sich im Grundlagenkapitel wieder. Die Auflösung bleibt von der Vereinheitlichung unberührt. Des Weiteren wird für das Rendering ein Intel Core i7-8550U genutzt.

Als relevante Metriken werden für die erste Evaluation im Folgenden die Renderlaufzeit, die Erstellungszeitdauer der BVH, die Intersektionslaufzeit, der Speicherplatzbedarf, die Anzahl an Traversierungsschritten sowie die SAH-Kosten betrachtet. Die

Renderlaufzeit beschreibt hierbei die fürs Rendern benötigte Zeit, wobei die Zeit für die Szenenkonstruktion wie das Parsing, das Textureloading und die Baumkonstruktion exkludiert ist. Die prozentuale Angabe bezieht sich hierbei auf die Gesamtdauer. Bei der Erstellungsdauer wird die Zeit der Konstruktion des Binärbaums repräsentiert, wobei ähnlich wie bei der Renderlaufzeit der prozentuale Wert die Gesamtzeit referenziert. Die Intersektionslaufzeit teilt sich in zwei Teilzeiten für die zwei Funktionen `Intersect` und `IntersectP`. Die Methode `Intersect` ist hierbei die Funktion, die häufiger aufgerufen wird und nicht wie `IntersectP` bereits beim ersten positiven Schnitttest abbricht (Pharr u. a., 2016, S. 284). Der Speicherplatzbedarf beschreibt die Anzahl der gespeicherten Informationen innerhalb des Binärbaums. Bei den Traversierungsschritten handelt es sich um die Anzahl der durchlaufenen Knoten des Binärbaums für jeden emittierten Strahl. Die SAH beschreibt als Kostenmodell die theoretischen Kosten für die Intersektionen. Näheres hierzu findet sich unter anderem in Pharr u. a. (2016), Sauer (2019) sowie im Originalpaper von Goldsmith u. Salmon (1987).

Die zu testenden Ansätze setzen sich aus den drei im PBRT implementierten Splitting-Methoden, bestehend aus SAH- und Middle-Methode sowie HLBVH, dem distanzbasierten Ansatz aus Sauer (2019) sowie der innerhalb dieser Arbeit entwickelten PCA-BVH mit der SAH- sowie der Middle-Methode als Partitionierungstechniken zusammen. Die EqualCounts-Methode wird nicht in die Evaluation mit aufgenommen, da die Ergebnisse aus der Evaluation Sauer (2019) zu negativ ausfielen und die Werte der Evaluation somit kaum Relevanz für weitere Evaluationen hätten.

In der erweiterten Evaluation wird die Effizienz der in Unterabschnitt 4.1.3 dargelegten kombinierten SAH-Methode sowie der Variante mit der erweiterten SAH gegenüber der klassischen SAH-Methode wie auch gegenüber der PCA-basierten SAH untersucht. Als Metriken werden hier auch die Renderlaufzeit, die Erstellungsdauer, die Traversierungschritte, die SAH-Kosten sowie die Anzahl der einzelnen Splits für die jeweilig bessere SAH-Variante genutzt. Letztgenannter Parameter soll Aufschluss darüber geben, wie oft die PCA-basierte SAH-Methode im Vergleich zur klassischen SAH-Methode gewählt wurde.

4.3.2. Präsentation und Interpretation der Ergebnisse

Innerhalb dieses Unterabschnitts werden die vorher dargestellten Metriken hinsichtlich der Ergebnisse für die verschiedenen Szenen unter Verwendung der verschiedenen BVH-Ansätze mit deren Splitting-Methoden präsentiert und interpretiert.

Renderlaufzeit

Die vollständigen Ergebnisse für die Evaluation der Renderlaufzeit werden durch die Tabelle A.1 repräsentiert. Die Tabelle 4.1 beinhaltet die Unterschiede zwischen den dargestellten Zusammenhängen.

Die PCA-Middle-Methode benötigt durchschnittlich 5.90 % höhere Renderlaufzeiten als die PCA-SAH-Methode. Hierbei stellt das „Pavillion“-Modell mit 21.74 % einen Ausreißer-Wert dar, der allerdings in die Betrachtung miteinbezogen wurde. Die „Buddha“-Figur weist mit 0.61 % den geringsten prozentualen Unterschied auf. Die übrigen Werte bewegen sich zwischen 1.0 % und 8.0 %

Szenen	PCA-Middle zu PCA-SAH	PCA-SAH zu SAH	PCA-Middle zu distanzbasiert	PCA-SAH zu distanzbasiert	PCA-Middle zu Middle	PCA-SAH zu HLBVH
Krone	29.75 (4.46 %)	1:22.28 (14.35 %)	-1:57.33 (-14.62 %)	-2:27.08 (-18.32 %)	1:24.04 (13.98 %)	-1:03.47 (-8.83 %)
Glas	8.99 (6.44 %)	8.99 (6.88 %)	-14.0 (-8.61 %)	-22.99 (-14.13 %)	8.99 (6.44 %)	6.11 (4.57 %)
Buddha	2.03 (0.61 %)	2:45.86 (98.25 %)	-3:06.15 (-35.16 %)	-3:08.18 (-35.99 %)	2:44.94 (96.02 %)	2:26.09 (77.48 %)
San Miguel	20.84 (2.02 %)	4:56.83 (40.26 %)	-2:41.52 (-13.28 %)	-3:02.36 (-14.99 %)	4:59.37 (39.62 %)	Fehler
Badezimmer	4.00 (1.06 %)	24.11 (6.89 %)	-20.83 (-5.23 %)	-24.78 (-6.22 %)	23.28 (6.57 %)	32.58 (9.55 %)
Pavillion	2:08.71 (21.74 %)	3:31.67 (51.67 %)	-4:53.18 (-28.92 %)	-7:01.89 (-41.61 %)	5:19.23 (79.53 %)	3:11.78 (47.93 %)
Buch	12.02 (7.57 %)	5.74 (3.75 %)	-17.21 (-9.16 %)	-29.23 (-15.55 %)	2.77 (1.65 %)	10.05 (6.76 %)
Drache	38.81 (7.59 %)	2:01.12 (31.02 %)	-41.14 (-6.95 %)	-1:19.95 (-13.52 %)	2:34.31 (38.96 %)	-6:23.27 (-42.83 %)
Villa	8.97 (2.46 %)	1:03.20 (20.93 %)	-34.27 (-8.39 %)	-43.23 (-10.59 %)	1:04.03 (20.65 %)	35.64 (10.81 %)
White Room	14.97 (6.21 %)	18.83 (8.47 %)	-16.91 (-6.20 %)	-31.88 (-11.68 %)	28.29 (12.42 %)	22.82 (10.46 %)
Audi TT	5.90 (3.44 %)	22.83 (15.38 %)	-25.91 (-12.76 %)	-31.81 (-15.66 %)	29.19 (19.72 %)	24.54 (16.72 %)
Measure One	38.22 (7.19 %)	1:14.69 (13.86 %)	-49.09 (-7.93 %)	-1.27.31 (-14.11 %)	57.80 (11.29 %)	-32.53 (-5.77 %)

Tabelle 4.1.: Unterschiede der Renderlaufzeiten mit grün markierten, positiven Referenzwerten und rot markierten, negativen Referenzwerten

Die PCA-SAH-Methode benötigt gegenüber der klassischen SAH-Methode 18.98 % mehr Renderlaufzeit. Für die einzelnen 3D-Modelle ergibt sich daraus ein 14.28 % erhöhter Zeitaufwand, während die Landschaftsszenen 23.68 % mehr Renderlaufzeit benötigen. Das „Buddha“-Modell wurde als Ausreißer mit 98.25 % nicht miteinbezogen. Das „Pavillion“-Modell weist mit 51.67 % eine erhöhte Differenz auf. Der niedrigste Unterschied wird durch das „Buch“ mit 3.75 % abgebildet.

Die PCA-Middle-Methode benötigt gegenüber der distanzbasierten Methode 13.14 % weniger Zeit. Hierbei ist festzustellen, dass für alle Modelle eine niedrigere Renderlaufzeit benötigt wird. Die beiden Modelle mit den größten prozentualen Vorteilen sind die „Buddha“-Figur mit -35.16 % und der „Pavillion“ mit -28.92 %. Der niedrigste Vorteil ergibt sich beim „Badezimmer“ mit -5.23 %.

Die PCA-SAH-Methode bietet gegenüber der distanzbasierten Methode durchschnittlich einen zeitlichen Vorteil von 17.70 %. Auch hier kann für alle Modelle eine niedrigere Renderlaufzeit erzielt werden. Des Weiteren sind ähnlich wie bei dem Vergleich zwischen der PCA-Middle-Methode und der distanzbasierten Methode der „Buddha“ mit -35.99 % und der „Pavillion“ mit -41.61 % die prozentual vorteilhaftesten Modelle. Das Badezimmer bietet mit -6.22 % den geringsten Vorteil.

Gegenüber der klassischen Middle-Methode benötigt die PCA-Middle-Methode einen zeitlichen Mehraufwand von 28.90 %. Festzuhalten ist, dass auch hier für alle Szenen eine erhöhte Laufzeit benötigt wird. Negativ sind hier vor allem der „Pavillion“ mit 79.53 % und der „Buddha“ mit 96.02 %. Das „Buch“ benötigt mit 1.65 % den geringsten, prozentualen Mehraufwand.

Die PCA-SAH-Methode benötigt gegenüber der HLBVH durchschnittlich 11.53 % mehr Zeit. Allerdings kann die PCA-SAH-Methode vereinzelt Modelle mit teils deutlich niedrigerer Laufzeit rendern, wobei es sich lediglich um Ausreißer handeln könnte. Dies bezieht sich vor allem auf das „Drachen“-Modell mit -42.83 %. Die „Krone“ und „Measure One“ weisen zumindest geringere prozentuale Vorteile von -8.83 % und -5.77 % auf. Auch hier weisen der „Buddha“ mit 77.48 % und der „Pavillion“ mit 47.93 % die größten Erhöhungen auf.

Des Weiteren kann festgestellt werden, dass die zeitlichen Unterschiede zwischen der Middle- und der SAH-Methode sowohl für die klassische BVH wie auch für die PCA-BVH nahezu gleich sind. Dementsprechend verhalten sich auch die zeitlichen Abstände zwischen der klassischen BVH und der PCA-BVH mit den beiden Splitting-Methoden Middle- und SAH-Methode. Außerdem scheinen gerade das „Buddha“-Modell und die „Pavillion“-Szene durch die PCA-BVH wegen deren Eigenschaften nur inperformant renderbar zu sein.

Erstellungszeitdauer

Die vollständigen Ergebnisse für die Evaluation der Erstellungszeitdauer der verschiedenen Methoden befinden sich in Tabelle A.2. Innerhalb der Tabelle 4.2 sind die Unterschiede für die verschiedenen Zusammenhänge aufgeführt, wobei das „Buch“ und der „Buddha“ wegen der geringen absoluten Werte nicht in die Betrachtung miteinbezogen werden.

Szenen	PCA-SAH zu PCA-Middle	PCA-Middle zu Middle	PCA-SAH zu SAH	PCA-SAH zu distanzbasiert	PCA-Middle zu distanzbasiert
Krone	0.08 (1.92 %)	2.48 (146.75 %)	1.25 (41.67 %)	1.77 (71.37 %)	1.69 (72.31 %)
Glas	0.02 (40.00 %)	0.03 (150.00 %)	0.02 (40.00 %)	0.04 (133.33 %)	0.02 (66.67 %)
San Miguel	0.39 (15.92 %)	1.33 (118.75 %)	0.81 (39.90 %)	1.08 (61.36 %)	0.69 (39.20 %)
Badezimmer	0.05 (14.71 %)	0.20 (142.86 %)	0.09 (30.00 %)	0.16 (69.57 %)	0.11 (47.83 %)
Pavillion	0.26 (4.79 %)	2.96 (119.84 %)	1.41 (32.94 %)	2.40 (72.95 %)	1.14 (65.05 %)
Drache	0.98 (13.21 %)	4.24 (133.33 %)	2.03 (31.87 %)	3.61 (75.37 %)	2.63 (54.91 %)
Villa	0.02 (0.76 %)	1.12 (74.17 %)	0.48 (22.12 %)	0.88 (49.72 %)	0.86 (48.55 %)
White Room	0.01 (1.82 %)	0.33 (150.00 %)	0.11 (24.44 %)	0.43 (330.77 %)	0.42 (323.08 %)
Audi TT	0.07 (23.33 %)	0.17 (76.47 %)	0.10 (37.04 %)	0.16 (76.19 %)	0.09 (42.86 %)
Measure One	0.78 (18.01 %)	3.27 (135.33 %)	1.46 (40.0 %)	2.29 (81.21 %)	1.51 (53.55 %)

Tabelle 4.2.: Unterschiede der Erstellungszeitdauer mit grün markierten, positiven Referenzwerten und rot markierten, negativen Referenzwerten

Die PCA-SAH-Methode benötigt im Vergleich zur PCA-Middle-Methode 13.45 % mehr Erstellungszeit. Der Ausreißer-Wert wird mit 40.0 % durch das „Glas“ repräsentiert, wobei dieser Wert durch den geringen absoluten Unterschied nur bedingt repräsentativ ist. Den geringsten Mehraufwand produziert die „Villa“ mit 0.76%.

Die Erstellungszeitdauer ist für die PCA-Middle-Methode gegenüber der klassischen Middle-Methode um 123.75 % erhöht. Hierbei führen bis auf die „Villa“ mit 74.17 % und den „Audi TT“ mit 76.74 % alle Szenen mindestens zu einer Verdopplung bis hin zu einer maximalen Erhöhung der Erstellungszeit um 150 %.

Die PCA-SAH-Methode benötigt durchschnittlich 34.0 % mehr Erstellungszeit als die klassische SAH-Methode. Auffällig ist hierbei der nicht vorhandene Ausreißer-Wert, sodass alle Szenen im Rahmen von ca. 20-41 % liegen.

Gegenüber dem distanzbasierten Ansatz ist die durchschnittliche Erstellungszeitdauer der PCA-SAH-Methode um 76.79 % erhöht. Mit um 330.77 % erhöhter Erstellungszeit stellt der „White Room“ den Ausreißer-Wert dar. Die einzige Szene unter einer Erhöhung um 50 % ist die „Villa“. Die restlichen Werte bewegen sich im Raum von 60 % bis 135 %.

Die Erstellungszeitdauer der PCA-Middle-Methode ist im Vergleich zur distanzbasierten Methode um 54.55 % erhöht. Ähnlich wie im vorherigen Zusammenhang ist auch hier der „White Room“ ein Ausreißer. Allerdings weist hier „San Miguel“ den geringsten Unterschied mit 39.20 % auf.

Die HLBVH wurde hierbei nicht näher evaluiert, da durch den hierfür gewählten, stark parallelisierten Ansatz entsprechend im Vergleich zu den anderen Methoden im Erstellungsprozess deutlich weniger Zeit benötigt wird.

Intersektionslaufzeit

Die vollständigen Ergebnisse für die Intersektionslaufzeiten finden sich in der Tabelle A.4 wieder. Die Tabelle 4.3 enthält die Unterschiede zwischen den Intersektionslaufzeiten aus der vorher erwähnten Tabelle.

Szenen	PCA-SAH zu SAH	PCA-Middle zu PCA-SAH	PCA-Middle zu Middle	PCA-Middle zu distanzbasiert	PCA-SAH zu distanzbasiert
Krone	0:59.12 (51.89 %) 1.23 (1.48 %)	15.32 (8.85 %) 10.03 (11.5 %)	55.31 (41.56 %) 1.07 (1.11 %)	-1:21.65 (-30.24 %) -14.59 (-13.05 %)	-1:36.97 (-35.91 %) -24.46 (-22.02 %)
Glas	6.45 (24.29 %) 0.32 (8.7 %)	7.78 (23.61 %) 1.88 (47.00 %)	6.97 (20.65 %) 0.75 (14.62 %)	-10.90 (-21.11 %) -0.78 (-11.71 %)	-18.68 (-36.18 %) -2.66 (39.94 %)
Buddha	2:06.07 (144.15 %) 0.24 (1.5 %)	6.98 (3.27 %) -1.36 (-8.38 %)	2:09.38 (141.71 %) -1.47 (-9.00 %)	-2:24.52 (-39.59 %) -3.03 (-16.93 %)	-2:31.50 (-41.5 %) -1.67 (-16.93 %)
San Miguel	3:52.11 (62.72 %) -7.90 (-12.59 %)	23.41 (3.89 %) 18.56 (33.83 %)	3:50.72 (58.43 %) 7.13 (10.75 %)	-2:21.38 (-18.43 %) 29.1 (65.64 %)	-2:44.79 (-21.49 %) 10.54 (23.78 %)
Badezimmer	14.89 (23.79 %) 2.62 (7.31 %)	5.27 (6.8 %) 0.29 (0.75 %)	14.36 (20.99 %) 0.03 (0.08 %)	-13.93 (-14.41 %) -3.07 (-7.34 %)	-19.2 (-19.86 %) -3.36 (-8.03 %)
Pavillion	2:31.46 (212.88 %) 7.65 (16.87 %)	2:44.88 (36.72 %) 10.63 (21.76 %)	5:03.57 (165.8 %) 13.17 (28.44 %)	-3:52.29 (-37.3 %) -9.45 (-13.71 %)	-5:37.17 (-54.14 %) -20.08 (-29.14 %)
Buch	4.02 (8.77 %) 1.33 (11.85 %)	10.03 (20.11 %) 1.91 (15.22 %)	-0.09 (-0.15 %) -0.54 (-3.6 %)	-11.87 (-16.54 %) -3.13 (-17.79 %)	-21.90 (-30.51 %) -5.04 (-28.65 %)
Drache	1:45.09 (50.58 %) 0.2 (0.74 %)	35.46 (11.33 %) 2.05 (7.49 %)	2:12.72 (61.56 %) 1.6 (5.75 %)	-35.42 (-9.23 %) -0.24 (-0.81 %)	-1:10.88 (-18.47 %) -2.29 (-7.72 %)
Villa	49.44 (50.87 %) -0.99 (-4.48 %)	10.06 (6.86 %) 3.19 (15.12 %)	45.46 (41.44 %) -1.86 (-7.11 %)	-26.29 (-14.37 %) -1.90 (-7.25 %)	-36.35 (-19.87 %) -5.09 (-19.43 %)
White Room	12.7 (23.36 %) 3.56 (8.24 %)	6.83 (10.18 %) 7.01 (14.99 %)	16.20 (28.08 %) 3.75 (7.5 %)	-15.11 (-16.98 %) 1.11 (2.11 %)	-21.94 (-24.65 %) -5.9 (-11.21 %)
Audi TT	18.56 (53.47 %) 0.5 (7.28 %)	5.41 (10.16 %) -0.67 (-9.09 %)	21.03 (55.86 %) -0.62 (-8.47 %)	-17.26 (-22.73 %) -0.78 (-10.43 %)	-22.67 (-29.85 %) -0.11 (-1.47 %)
Measure One	49.63 (39.57 %) 2.65 (3.64 %)	30.52 (17.44 %) 11.56 (15.31 %)	50.58 (32.64 %) -3.33 (-3.68 %)	-30.72 (-13.0 %) -8.77 (-21.21 %)	-1:01.24 (-25.92 %) -20.33 (-21.21 %)

Tabelle 4.3.: Unterschiede der Intersektionslaufzeiten mit grün markierten, positiven Referenzwerten und rot markierten, negativen Referenzwerten

Für die PCA-basierte SAH-Methode im Vergleich zu klassischen SAH-Methode ergibt sich ein Nachteil von 62.20 % für die `Intersect`-Methode sowie ein Nachteil von 4.21 % für die `IntersectP`-Methode. Hierbei sind der „Buddha“ sowie der „Pavillion“ mit 144.15 % und 212.88 % an erhöhter Zeit für die `Intersect`-Methode als Ausreißer anzusehen. Das „Buch“ hingegen weist mit 8.77 % den geringsten prozentualen Mehraufwand für die `Intersect`-Methode auf.

Die PCA-Middle-Methode benötigt gegenüber der PCA-SAH-Methode 13.27 % mehr Zeit für die `Intersect`-Methode und für die `IntersectP`-Methode wird 13.79 % höhere Laufzeit benötigt. Auch in diesem Zusammenhang erbringt der „Pavillion“ 36.72% an Mehraufwand für die `Intersect`-Methode. Allerdings kann entgegen dem vorherigen Zusammenhang der „Buddha“ den mit 3.27 % niedrigsten Mehraufwand aufweisen.

Die PCA-Middle-Methode benötigt um 58.23 % erhöhte Laufzeit für die `Intersect`-Methode und eine um 2.63 % erhöhte Laufzeit für die `IntersectP`-Methode gegenüber der klassischen Middle-Methode. Wie beim ersten Zusammenhang sind hier auch wieder der „Buddha“ und der „Pavillion“ die Ausreißer. Das „Buch“ kann hier im Gegensatz zu den restlichen Szenen sogar einen Vorteil von -0.15 % für die `Intersect`-Methode erbringen.

Die PCA-Middle-Methode benötigt im Vergleich zur distanzbasierten Methode 21.16 % weniger Zeit für die `Intersect`-Methode und 4.73 % weniger Zeit für die `IntersectP`-Methode. Hierbei sind für alle Szenen durchweg Verringerungen der benötigten Zeit zu beobachten, wobei der „Buddha“ mit -39.59 % die größte Verringerung für die „Intersect“-Methode bewirkt. Der „Drache“ hingegen bewirkt die niedrigste Verbesserung in Höhe von -9.23 %.

Die PCA-SAH-Methode benötigt gegenüber der distanzbasierten Methode 29.86 % weniger Zeit für die `Intersect`-Methode und für die `IntersectP`-Methode 7.67 % weniger Zeit. Innerhalb dieses Zusammenhangs kann der „Pavillion“ mit -54.14 % den größten Zeitvorteil für die `Intersect`-Methode erzielen, während der „Drache“ mit -18.47 % den niedrigsten Vorteil erwirkt.

Speicherplatzbedarf

Die Werte für den Speicherplatzbedarf der verschiedenen Methoden werden innerhalb der Tabelle A.3 dargestellt.

Bei der Betrachtung der Werte ergibt sich die Erkenntnis, dass die Middle-, die distanzbasierte sowie die PCA-Middle-Methode nahezu identische Werte hinsichtlich des Speicherplatzbedarfs haben. Ähnlich wie bereits in Sauer (2019) festgestellt wurde, ergibt sich auch hier für die HLBVH der geringste Speicherplatzbedarf, da auch die Anzahl der Knoten nicht derart hoch ist, wie bei den anderen Top-Down-Verfahren, dementsprechend auch mehr Ray-Triangle-Intersection-Tests direkt durchgeführt werden müssen. Die beiden SAH-Varianten liefern im Vergleich zu den drei nahezu identischen Methoden jeweils weniger speicherplatzbedürftige BVHs.

Die klassische SAH-Methode ist hierbei allerdings noch effizienter als die PCA-Variante. So ergibt sich für die „Krone“ 2.28 MB, für das „Glas“ 0.01 MB, für den „Buddha“ 0.03 MB, für „San Miguel“ 1.6 MB, für das „Badezimmer“ 0.26 MB, für den „Pavillion“ 2.05 MB, für das „Buch“ 3.62 kB, für den „Drachen“ -0.32 MB als Ausreißerwert, für die „Villa“ 1.65 MB, „White Room“ 0.61 MB, „Audi TT“ 0.07 MB und

für „Measure One“ 3.72 MB an Speicherplatz, welchen die PCA-Variante gegenüber der klassischen SAH-Methode mehr benötigt.

Traversierungsschritte

Die vollständigen Ergebnisse für die Anzahlen der Traversierungsschritte in den verschiedenen Szenen befinden sich in der Tabelle A.5. Die aus diesen Ergebnissen abgeleiteten Unterschiede finden sich in der Tabelle 4.4.

Szenen	PCA-SAH zu SAH	PCA-Middle zu Middle	PCA-Middle zu PCA-SAH	PCA-Middle zu distanzbasiert	PCA-SAH zu distanzbasiert
Krone	5.5 Mrd. (17.67 %)	3.5 Mrd. (8.75 %)	6.2 Mrd. (16.62 %)	-12.9 Mrd. (-22.88 %)	-19.1 Mrd. (-33.87 %)
Glas	1.0 Mrd. (20.41 %)	1.8 Mrd. (26.09 %)	2.8 Mrd. (47.46 %)	-1.5 Mrd. (-14.71 %)	-4.3 Mrd. (-42.16 %)
Buddha	12.0 Mrd. (100 %)	12.7 Mrd. (96.21 %)	1.9 Mrd. (7.92 %)	-13.6 Mrd. (-34.44 %)	-15.5 Mrd. (-39.25 %)
San Miguel	20.2 Mrd. (41.8 %)	24.3 Mrd. (44.34 %)	11.0 Mrd. (16.15 %)	-1.5 Mrd. (-1.87 %)	-11.0 Mrd. (-15.51 %)
Badezimmer	2.5 Mrd. (14.61 %)	2.1 Mrd. (10.34 %)	2.0 Mrd. (9.80 %)	-2.0 Mrd. (-8.20 %)	-4.0 Mrd. (-16.40 %)
Pavillion	18.0 Mrd. (70.87 %)	29.0 Mrd. (97.64 %)	15.3 Mrd. (35.25 %)	-19.8 Mrd. (-25.23 %)	-35.1 Mrd. (-44.72 %)
Buch	1.5 Mrd. (16.42 %)	0.8 Mrd. (5.67 %)	3.8 Mrd. (34.23 %)	-2.1 Mrd. (-12.36 %)	-5.9 Mrd. (-34.71 %)
Drache	13.0 Mrd. (46.76 %)	16.1 Mrd. (53.85 %)	5.2 Mrd. (12.75 %)	-2.5 Mrd. (-5.16 %)	-7.7 Mrd. (-15.88 %)
Villa	6.2 Mrd. (39.13 %)	5.0 Mrd. (25.38 %)	2.3 Mrd. (10.27 %)	-2.6 Mrd. (-9.53 %)	-4.9 Mrd. (-17.95 %)
White Room	2.5 Mrd. (16.0 %)	2.7 Mrd. (12.92 %)	3.3 Mrd. (16.26 %)	-0.5 Mrd. (-2.08 %)	-3.8 Mrd. (-15.77 %)
Audi TT	2.6 Mrd. (41.27 %)	2.7 Mrd. (38.57 %)	0.8 Mrd. (8.99 %)	-2.3 Mrd. (-19.17 %)	-3.1 Mrd. (-25.84 %)
Measure One	6.9 Mrd. (21.21 %)	6.8 Mrd. (16.59 %)	7.8 Mrd. (19.50 %)	-3.7 Mrd. (-7.19 %)	-11.5 Mrd. (-22.34 %)

Tabelle 4.4.: Unterschiede der Traversierungsschritte mit rot markierten, negativen Referenzwerten und grün markierten, positiven Referenzwerten

Die Anzahl der Traversierungsschritte erhöht sich für die PCA-SAH-Methode gegenüber der klassischen SAH-Methode um 37.18 %. Auch hier fallen der „Buddha“ mit einer fast exakten Verdopplung der Traversierungschritte sowie der „Pavillion“ mit einer Erhöhung um 70.87 % auf. Die „Badezimmer“-Szene weist hierbei die mit 14.61 % niedrigste Erhöhung auf.

Für die PCA-Middle Methode erhöht sich die Schrittanzahl gegenüber der klassischen Middle-Methode um durchschnittlich 36.36 %. Ähnlich wie bei der vorherigen Betrachtung sind auch hier der „Buddha“ mit 96.21 % und der „Pavillion“ mit 97.64 % die beiden Szenen mit der größten Erhöhung. Die „Krone“ mit 8.75 % sowie das „Buch“ mit 5.67 % stellen hierbei die Modelle mit dem niedrigsten Mehraufwand dar.

Beim Vergleich der PCA-Middle-Methode mit der PCA-SAH-Methode fällt ein Anstieg der Traversierungsschritte um 19.58 % auf. Hier ist das „Glas“ mit 47.46 % die ungünstigste Szene, während der Buddha mit 7.92 % die günstige Szene darstellt.

Für die PCA-Middle-Methode ergibt sich gegenüber dem distanzbasierten Ansatz eine 13.57 % niedrigere Anzahl an Traversierungsschritten. Hierbei haben alle Szenen niedriger Schrittzahlen, wobei der Vorteil für „San Miguel“ mit -1.87 % am niedrigsten ausfällt. Der „Buddha“ hingegen weist mit -34.44 % den höchsten Vorteil auf.

Die PCA-SAH-Methode benötigt im Vergleich zum distanzbasierten Ansatz 27.03 % weniger Schritte. Die günstigste Szene mit -44.72 % stellt hierbei im Vergleich der „Pavillion“ dar. Den wenigsten Vorteil bringen mit ähnlichen Werten „San Miguel“, der „Drache“ sowie der „White Room“ mit 15.51 bis 15.88 %.

SAH-Kosten

Die vollständigen Ergebnisse für die Kosten der verschiedenen Splitting-Methoden werden durch die Tabelle A.6 repräsentiert. Innerhalb der Tabelle 4.5 befinden sich die errechneten Unterschiede.

Die kumulierten Kosten nach der SAH sind für die PCA-SAH-Methode im Vergleich zur klassischen SAH-Methode um 1.08 % erhöht, während die Durchschnittskosten um 0.015 % erhöht sind. Die „Buddha“-Figur stellt mit um ca. 125 % erhöhten kumulierten und durchschnittlichen Kosten den negativen Ausreißer dar, welcher in der zuvor berechneten durchschnittlichen Erhöhung nicht mit inbegriffen ist. Eine Verringerung der Kosten kann nur für „San Miguel“ mit -5.02 % und -6.57 % erreicht werden, wobei sich dieser Vorteil praktisch nicht widerspiegelt.

Die PCA-Middle-Methode weist gegenüber der PCA-SAH-Methode um 13.75 % erhöhte kumulierte Kosten auf und um 2.78 % verringerte Durchschnittskosten. Lediglich für das „Glas“ verringern sich die kumulierten Kosten um -2.39 %. Die „Villa“ hingegen stellt mit 28.36 % die größte prozentuale Erhöhung dar.

Die PCA-Middle-Methode hat um 3.63 % niedrigere kumulierte Kosten und um 3.60 % niedrigere Durchschnittskosten gegenüber der Middle-Methode. Der „Buddha“ stellt auch hier wieder mit jeweils ca. 125 % einen Ausreißer dar. Für die „Krone“ sind die prozentualen Vorteil mit 10.59 % am höchsten.

Gegenüber dem distanzbasierten Ansatz weist die PCA-Middle-Methode 3.67 % geringere kumulierte Kosten und 3.71% geringere Durchschnittskosten auf. Wie auch schon bei den Zusammenhängen zuvor stellt der „Buddha“ mit ähnlich hohen Werten den Ausreißer dar. Für diesen Zusammenhang kann das „Buch“ mit -9.15 % die größten Vorteile aufweisen.

Bei der PCA-SAH-Methode ergeben sich gegenüber des distanzbasierten Ansatzes 15.31 % niedrigere Gesamtkosten und 0.68 % niedrigere Durchschnittskosten. Ähnlich wie bei den sonstigen Zusammenhängen zuvor stellt der „Buddha“ auch hier den Ausreißer dar. Die vorteilhaftesten Szenen hinsichtlich der kumulierten Kosten, bestehend aus der „Krone“, „San Miguel“, der „Villa“ sowie dem „Audi TT“, bewegen sich im Rahmen von -22.15 % bis zu -24.41 %.

Interpretation

Aus den vorherigen Erkenntnissen lässt sich schließen, dass sowohl die PCA-BVH mit Middle-Methode wie auch mit der SAH-Methode in der wichtigste Kategorie, der Renderlaufzeit, um 13.14% respektive 17.70% besser sind als die klassische BVH mit dem

Szenen	PCA-SAH zu SAH	PCA-Middle zu PCA-SAH	PCA-Middle zu Middle	PCA-Middle zu distanzbasiert	PCA-SAH zu distanzbasiert
Krone	1.9 Mio. (3.30%) 0.34 (1.94%)	11.2 Mio. (20.74%) 0.13 (0.73%)	-7.4 Mrd. (-10.59%) -2.13 (-10.58%)	-3.9 Mio. (-5.85%) -1.11 (-6.49%)	-14.7 Mio. (-22.02%) -1.24 (-6.49%)
Glas	4 Tsd. (0.53%) 0.04 (0.37%)	-23 Tsd. (-2.39%) -0.48 (-3.65%)	-21 Tsd. (-2.24%) -0.24 (-2.24%)	4 Tsd. (0.44%) 0.05 (0.48%)	26 Tsd. (2.89%) 0.53 (5.07%)
Buddha	301 Tsd. (125.17%) 5.91 (123.13%)	28 Tsd. (5.14%) -0.46 (-3.37%)	315 Tsd. (123.97%) 5.73 (124.03%)	215 Tsd. (123.78%) 5.72 (123.54%)	287 Tsd. (112.83%) 6.08 (131.32%)
San Miguel	-1.8 Mio. (-5.02%) -1.15 (-6.57%)	8.1 Mio. (26.50%) -0.84 (-5.14%)	-2.6 Mio. (-6.24%) -1.03 (-6.22%)	-1.8 Mio. (-4.38%) -0.71 (-4.38%)	-10.3 Mio. (-24.41%) 0.13 (0.80%)
Badezimmer	90 Tsd. (1.60%) 0.04 (0.23%)	0.9 Mio. (14.65%) -0.81 (-4.66%)	-280 Tsd. (-4.21%) -0.69 (-4.56%)	-260 Tsd. (-3.92%) -0.65 (-3.78%)	-1.1 Mio. (-16.22%) 0.16 (0.93%)
Pavillion	915 Tsd. (1.27%) 0.08 (0.47%)	7.2 Mio. (9.95%) -1.96 (-11.43%)	-2.4 Mio. (-2.94%) -0.46 (-2.94%)	-3.0 Mio. (-3.61%) -0.55 (-3.62%)	-10.3 Mio. (-12.34%) 1.39 (8.81%)
Buch	3 Tsd. (1.86%) 0.19 (1.27%)	1.3 Tsd. (0.79%) -0.36 (-2.39%)	-1.7 Tsd. (-0.74%) -0.11 (-0.74%)	-16 Tsd. (-9.15%) -1.49 (-9.17%)	-18 Tsd. (-9.86%) -1.13 (-6.96%)
Drache	2.2 Mio. (2.18%) 0.33 (2.31%)	9.4 Mio. (9.02%) 1.13 (7.73%)	-4.2 Mio. (-3.55%) -0.58 (-3.56%)	-2.0 Mio. (-1.75%) -0.28 (-1.75%)	-11.4 Mio. (-9.88%) -1.41 (-8.80%)
Villa	300 Tsd. (0.84%) -0.12 (-0.61%)	10.3 Mio (28.36%) -0.66 (-3.40%)	-600 Tsd. (-1.29%) -0.25 (-1.32%)	-800 Tsd. (-1.60%) -0.31 (-1.63%)	-11.1 Mio. (-23.34%) 0.35 (1.83%)
White Room	270 Tsd. (4.65%) 0.41 (2.27%)	1.4 Mio. (17.79%) -0.06 (-0.33%)	-430 Tsd. (-4.32%) -0.83 (-4.32%)	-300 Tsd. (-3.01%) -0.57 (-3.01%)	-1.7 Mio. (-17.66%) -0.51 (-2.69%)
Audi TT	2 Tsd. (0.07%) -0.05 (-0.34%)	0.9 Mio. (20.71%) -0.35 (-2.37%)	-100 Tsd. (-1.84%) -0.27 (-1.32%)	-340 Tsd. (-6.01%) -0.92 (-5.99%)	-1.2 Mio. (-22.15%) -0.57 (-3.72%)
Measure One	400 Tsd. (0.61%) -0.24 (-1.18%)	8.8 Mio. (13.77%) -0.93 (-5.02%)	-1.4 Mio. (-1.92%) -0.33 (-1.85%)	-1.2 Mio. (-1.52%) -0.26 (-1.46%)	-10.0 Mio. (-13.44%) 0.67 (3.75%)

Tabelle 4.5.: Unterschiede der SAH-Kosten mit rot markierten, negativen Referenzwerten und grün markierten, positiven Referenzwerten

distanzbasierten Ansatz. Erwartungsgemäß ist auch die PCA-BVH mit der Middle-Methode um 5.90% langsamer als der SAH-Ansatz im Rahmen der PCA-BVH. Für manche Modelle ist die PCA-BVH mit SAH-Methode besser als die HLBVH, wobei es sich bei den Werten auch um Ausreißer handeln könnte.

Auffällig waren vor allem die größeren Ausreißer gerade bei dem „Buddha“-Modell sowie bei der „Pavillion“-Szene. Die Eigenschaften der beiden Objekte führte bei nahezu allen Auswertungen zu starken prozentualen Unterschieden. So war nicht nur der Unterschied zwischen der klassischen SAH-Methode zu den PCA-basierten Methoden relativ groß, sondern auch der Unterschied zwischen den PCA-basierten Methoden zu dem distanzbasierten Ansatz für diese beiden Modelle relativ hoch.

Es ergab sich für die PCA-SAH-Methode gegenüber der klassischen SAH-Methode, dass Landschaftsszenen durchschnittlich 23.68 % mehr Renderzeit benötigen, während für einzelne 3D-Modelle lediglich ein Mehraufwand von 14.28 % entstand, wonach die gewählte Methode für 3D-Modelle geeigneter sein müsste als für Landschaftsszenen. Ein möglicher Grund hierfür könnte die bei 3D-Objekten stärker ausgeprägte Symmetrie sein, in deren Konsequenz bei der Anwendung der PCA Hauptkomponenten ermittelt werden, die sich stärker an den Achsen ausrichten. Daraus resultieren auch entsprechende Split-Ebenen, was sich bei der hier, nach den Traversierungsschritten zu urteilen, in schwächerer Form auftretenden Überlappungsproblematik, welche bereits in Sauer (2019) für den distanzbasierten Ansatz identifziert wurde, positiv auswirkt. Durch diese Problematik kann auch keine Verbesserung im Vergleich zur klassischen SAH-Methode erzielt werden. Für zukünftige Forschung hingegen kann im vorliegenden Kontext der Einsatz von Oriented Bounding Boxes hinsichtlich des entstehenden Mehraufwands im Gegensatz zu den daraus verminderten Überlappungen untersucht werden. Des Weiteren scheint auch eine Evaluation der möglichen Verbesserungen der PCA-BVH durch die Implementierung der Split-BVH (Stich u. a., 2009) sinnvoll.

4.3.3. Präsentation und Interpretation der erweiterten Evaluation

Die vollständigen Ergebnisse der erweiterten Evaluation für die kombinierte Methode sind in Tabelle A.7 dargestellt. Da die kombinierte Methode allerdings, entgegen dem in Unterabschnitt 4.1.3 theoretisch dargelegten Vorteil, keinerlei praktische Verbesserung der Renderlaufzeit gegenüber der SAH-Methode erbrachte, sondern nur die theoretischen Kosten nach der Surface Area Heuristic optimiert, werden im Folgenden mittels einer Erweiterung der SAH neue Evaluationsergebnisse erhoben, um festzustellen, ob hierdurch ein Performancegewinn erzielt werden kann. Grund für die vorher genannte, ausbleibende Verbesserung könnte die ungenügende Bewertung von überlappenden Bounding Boxes innerhalb der SAH sein (Aila u. a., 2013). Dementsprechend muss eine Metrik verwendet werden, die auch die Überlappungen entsprechend in die Kalkulation miteinbezieht. Eine solche Metrik kann in Anlehnung an Aila u. a. (2013), welcher die unzureichende Aussagekraft der SAH thematisiert und mit dem Endpoint Overlap auch eine ähnliche Heuristik für Überlappungen darlegt, in etwa wie folgt aussehen:

$$c(A,B) = t_{\text{trav}} + p_A * \sum_{i=1}^{N_\text{A}} t_{\text{isect}}(a_\text{i}) + p_B * \sum_{i=1}^{N_\text{B}} t_{\text{isect}}(b_\text{i}) + p_C * \left(\sum_{i=1}^{N_\text{A}} t_{\text{isect}}(a_\text{i}) + \sum_{i=1}^{N_\text{B}} t_{\text{isect}}(b_\text{i}) \right)$$

Hierbei ist die klassische SAH-Methode, um die durch Überlappung entstehende Bounding Box C (Abbildung 4.4) erweitert worden. Ähnlich wie die Wahrscheinlichkeiten p_A für die Bounding Box A und p_B für die Bounding Box B stellt die Wahrscheinlichkeit p_C die Trefferwarscheinlichkeit für die überlappende Bounding Box C dar. Für die Berechnung der Kosten der Bounding Box C werden im Folgenden die kumulierten Summen der Kosten von A und B genutzt, da ein Treffer in diesem Bereich zu der Traversierung beider Knoten führen würde und somit die Kosten für beide genutzt werden müssen. Das Listing B.8 zeigt die ursprüngliche sowie die erweiterte Heuristik als Code umgesetzt.

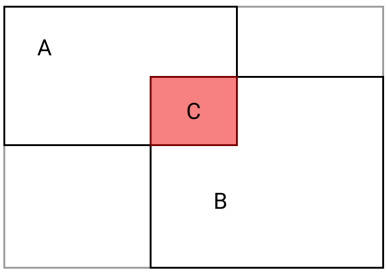


Abbildung 4.4.: Beispiel von überlappenden Bounding Boxes

Im Folgenden werden die ermittelten Unterschiede aus den Ergebnissen (Tabelle A.7) der kombinierten PCA-SAH-Methode mit und ohne die Verwendung der zuvor vorgestellten erweiterten Metrik präsentiert. Dies umfasst die Renderlaufzeit (Tabelle 4.6), die Erstellungszeitdauer (Tabelle 4.7), die Anzahl an Traversierungsschritten (Tabelle 4.8) sowie die SAH-Kosten (Tabelle 4.9). Die Anzahl der Aufrufe der klassischen sowie der PCA-basierten Surface Area Heuristic ist enthalten allerdings gesondert zu betrachten. Diese Metrik soll einen Vergleich über die Häufigkeit des Einsatzes im direkten Vergleich ermöglichen. Es ist des Weiteren anzumerken, dass die kumulierte Anzahl der beiden Aufrufzahlen nicht der Anzahl an Splits entspricht, da für entsprechend kleinere Primitivanzahl das SAH-Splitting nicht mehr durchgeführt wird, sondern stattdessen weniger komplexe Methoden wie die EqualCounts-Methode genutzt werden.

Ergebnisauswertung

Die folgenden Tabellen enthalten die ausgewerteten Unterschiede aus der Tabelle A.7. Hierbei stehen die Abkürzungen „Comb“ und „Comb + M“ für die kombinierte Methode mit der klassischen Surface Area Heuristic sowie mit der erweiterten Surface Area Heuristic.

Szenen	Comb zu SAH	Comb zu PCA	Comb + M zu SAH	Comb + M zu PCA	Comb + M zu Comb
Krone	1:24.64 (14.90 %)	-2.98 (-0.45 %)	1:35.06 (16.74 %)	7.44 (1.13 %)	10.42 (1.60 %)
Glas	5.13 (3.39 %)	-3.86 (-2.76 %)	7.16 (5.48 %)	-1.83 (-1.31 %)	2.03 (1.49 %)
Buddha	1:40.09 (59.29 %)	-1:05.77 (-19.65 %)	2:23.07 (84.75 %)	-22.79 (-6.81 %)	42.98 (15.98 %)
San Miguel	3:19.90 (26.98 %)	-1:37.12 (-9.39 %)	5:50.70 (47.52 %)	54.48 (5.27 %)	2:31.60 (16.18 %)
Badezimmer	15.49 (4.42 %)	-7.72 (-2.06 %)	1:46.79 (30.45 %)	1:23.58 (22.36 %)	1:31.30 (24.93 %)
Pavillion	1:50.43 (28.22 %)	-1:30.23 (-15.24 %)	4:49.77 (74.06 %)	1:29.11 (15.05 %)	2:59.34 (35.75 %)
Buch	1.96 (1.28 %)	-3.78 (-2.38 %)	6.01 (3.93 %)	0.27 (0.17 %)	4.05 (2.61 %)
Drache	44.87 (11.49 %)	-1:16.25 (-14.99 %)	1:28.20 (22.59 %)	-32.92 (-6.43 %)	43.33 (9.95 %)
Villa	47.16 (15.88 %)	-21.02 (-5.61 %)	Fehler	Fehler	Fehler
White Room	13.60 (6.27 %)	-7.03 (-2.92 %)	23.62 (10.72 %)	2.99 (1.24 %)	10.02 (4.28 %)
Audi TT	17.47 (11.77 %)	-5.36 (-3.13 %)	18.47 (12.44 %)	-4.36 (-2.55 %)	1.00 (0.60 %)
Measure One	1:08.52 (15.00 %)	-6.17 (-1.16 %)	Fehler	Fehler	Fehler

Tabelle 4.6.: Unterschiede der Renderlaufzeiten der erweiterten Evaluation mit grün markierten Minimal- und rot markierten Maximalwerten

Szenen	Comb zu SAH	Comb zu PCA	Comb + M zu SAH	Comb + M zu PCA	Comb + M zu Comb
Krone	3.11 (102.98 %)	1.88 (44.24 %)	3.24 (107.28 %)	2.01 (47.29 %)	0.13 (2.12 %)
Glas	0.07 (140.00 %)	0.05 (71.43 %)	0.07 (140.00 %)	0.05 (71.43 %)	0.00 (0 %)
San Miguel	2.67 (114.19 %)	1.98 (69.72 %)	3.59 (166.98 %)	2.90 (102.11 %)	0.92 (19.09 %)
Badezimmer	0.34 (106.25 %)	0.27 (69.23 %)	1.37 (428.13 %)	1.30 (333.33 %)	1.03 (156.06 %)
Pavillion	5.40 (115.63 %)	4.38 (76.98 %)	16.16 (346.04 %)	15.14 (266.08 %)	10.76 (106.85 %)
Drache	7.09 (111.3 %)	5.06 (60.24 %)	6.87 (107.85 %)	4.84 (57.62 %)	-0.22 (-1.63 %)
Villa	2.31 (106.94 %)	1.82 (68.68 %)	Fehler	Fehler	Fehler
White Room	0.48 (104.35 %)	0.38 (67.86 %)	0.52 (113.04 %)	0.42 (75.00 %)	0.04 (4.26 %)
Audi TT	0.29 (107.41 %)	0.19 (51.35 %)	0.27 (100.00 %)	0.17 (45.95 %)	-0.02 (-3.57 %)
Measure One	4.16 (113.32 %)	2.70 (52.84 %)	Fehler	Fehler	Fehler

Tabelle 4.7.: Unterschiede der Erstellungszeitdauer der erweiterten Evaluation mit grün markierten Minimal- und rot markierten Maximalwerten

Szenen	Comb zu SAH	Comb zu PCA	Comb + M zu SAH	Comb + M zu PCA	Comb + M zu Comb
Krone	2.6 Mrd. (8.20 %)	-3.0 Mrd. (-8.05 %)	2.8 Mrd. (8.83 %)	-2.8 Mrd. (-7.51 %)	0.2 Mrd. (0.58 %)
Glas	0.4 Mrd. (8.16 %)	-0.6 Mrd. (-10.17 %)	0.1 Mrd. (2.04 %)	-0.9 Mrd. (-15.26 %)	-0.3 Mrd. (-5.67 %)
Buddha	6.4 Mrd. (53.33 %)	-5.6 Mrd. (-23.34 %)	7.1 Mrd. (59.17 %)	-4.9 Mrd. (-20.42 %)	0.7 Mrd. (3.80 %)
San Miguel	9.3 Mrd. (19.38 %)	-10.8 Mrd. (-15.86 %)	22.4 Mrd. (46.67 %)	2.4 Mrd. (3.38 %)	13.1 Mrd. (22.86 %)
Badezimmer	1.0 Mrd. (5.62 %)	-1.6 Mrd. (-7.85 %)	10.2 Mrd. (57.30 %)	7.6 Mrd. (37.25 %)	9.2 Mrd. (48.94 %)
Pavillion	7.3 Mrd. (28.74 %)	-10.7 Mrd. (-24.66 %)	10.8 Mrd. (42.52 %)	-7.2 Mrd. (-16.60 %)	3.5 Mrd. (10.70 %)
Buch	0.1 Mrd. (1.05 %)	-1.5 Mrd. (-13.52 %)	1.0 Mrd. (10.53 %)	-0.6 Mrd. (-5.41 %)	0.9 Mrd. (9.38 %)
Drache	4.5 Mrd. (16.19 %)	-8.5 Mrd. (-20.84 %)	5.3 Mrd. (19.06 %)	-7.7 Mrd. (-19.88 %)	0.8 Mrd. (2.48 %)
Villa	2.0 Mrd. (12.42 %)	-4.3 Mrd. (-19.20 %)	Fehler	Fehler	Fehler
White Room	-0.5 Mrd. (-2.86 %)	-3.3 Mrd. (-16.26 %)	2.0 Mrd. (11.43 %)	-0.8 Mrd. (-3.95 %)	2.5 Mrd. (14.71 %)
Audi TT	1.5 Mrd. (23.81 %)	-1.1 Mrd. (-12.36 %)	1.7 Mrd. (26.98 %)	-0.9 Mrd. (-10.12 %)	0.2 Mrd. (2.56 %)
Measure One	3.9 Mrd. (11.82 %)	-3.1 Mrd. (-7.75 %)	Fehler	Fehler	Fehler

Tabelle 4.8.: Unterschiede der Traversierungsschritte der erweiterten Evaluation mit grün markierten Minimal- und rot markierten Maximalwerten

Szenen	Comb zu SAH	Comb zu PCA	Comb + M zu SAH	Comb + M zu PCA	Comb + M zu Comb
Krone	-892.0 Tsd. (-1.78 %) -0.31 (-1.77 %)	-2.6 Mio. (-4.92 %) -0.65 (-3.64 %)	7.4 Mio. (14.73 %) 9.46 (53.93 %)	5.7 Mio. (11.06 %) 9.12 (51.01 %)	8.3 Mio. (16.80 %) 9.77 (56.70 %)
Glas	-15.0 Tsd. (-1.60 %) -0.13 (-1.19 %)	-19.8 Tsd. (-2.1 %) -0.17 (-1.55 %)	-20.5 Tsd. (-2.18 %) 2.77 (25.32 %)	-25.3 Tsd. (-2.68 %) 2.73 (24.86 %)	-5.5 Tsd. (-0.59 %) 2.90 (26.83 %)
Buddha	284.2 Tsd. (118.40 %) 5.72 (119.17 %)	-16.3 Tsd. (-3.01 %) -0.19 (-1.78 %)	326.7 Tsd. (135.91 %) 14.86 (309.58 %)	25.8 Tsd. (4.77 %) 8.95 (83.57 %)	42.1 Tsd. (8.02 %) 9.14 (86.88 %)
San Miguel	-1.2 Mio. (-3.68 %) -0.38 (-2.17 %)	0.5 Mio. (1.71 %) 0.77 (4.7 %)	28.5 Mio. (84.74 %) 22.34 (127.51 %)	30.3 Mio. (95.07 %) 23.49 (143.49 %)	29.8 Mio. (91.79 %) 22.72 (132.56 %)
Badezimmer	-18 Tsd. (-0.33 %) 0 (0 %)	-0.11 Mio. (-1.91 %) -0.04 (-0.23 %)	4.5 Mio. (81.54 %) 23.40 (134.95 %)	4.4 Mio. (78.67 %) 23.36 (134.41 %)	4.5 Mio. (82.13 %) 23.40 (134.95 %)
Pavillion	-0.7 Mio. (-1.04 %) -0.15 (-0.88 %)	-1.7 Mio. (-2.28 %) -0.23 (-1.35 %)	11.7 Mio. (16.14 %) 6.01 (35.19 %)	10.7 Mio. (14.69 %) 5.93 (34.56 %)	12.4 Mio. (17.35 %) 6.16 (36.39 %)
Buch	-848 (-0.53 %) -0.11 (-0.74 %)	-3.9 Tsd. (-2.35 %) -0.30 (-1.98 %)	16.3 Tsd. (10.06 %) 1.71 (11.45 %)	13.3 Tsd. (8.05 %) 1.52 (10.05 %)	17.1 Tsd. (10.64 %) 1.82 (12.28 %)
Drache	-183 Tsd. (-0.18 %) 0.00 (0 %)	-2.4 Mio. (-2.32 %) -0.33 (-2.26 %)	3.8 Mio. (3.71 %) 5.78 (40.45 %)	1.6 Mio. (1.50 %) 5.45 (37.28 %)	4.0 Mio. (3.90 %) 5.78 (40.45 %)
Villa	-780 Tsd. (-2.17 %) -0.40 (-2.05 %)	-1.1 Mio. (-2.98 %) -0.28 (-1.44 %)	Fehler	Fehler	Fehler
White Room	-171 Tsd. (-2.22 %) -0.44 (-2.44 %)	-531 Tsd. (-6.57 %) -0.85 (-4.61 %)	947 Tsd. (12.24 %) 6.08 (33.70 %)	587 Tsd. (7.25 %) 5.67 (30.73 %)	1.1 Mio. (14.78 %) 6.52 (37.05 %)
Audi TT	-28 Tsd. (-0.65 %) -0.02 (-0.14 %)	-30 Tsd. (-0.69 %) 0.03 (0.20 %)	35 Tsd. (0.78 %) 5.66 (38.14 %)	33 Tsd. (0.73 %) 5.71 (38.61 %)	63 Tsd. (1.43 %) 5.68 (38.33 %)
Measure One	-1.2 Mio. (-1.84 %) -0.27 (-1.44 %)	-1.6 Mio. (-2.43 %) -0.05 (-0.27 %)	Fehler	Fehler	Fehler

Tabelle 4.9.: Unterschiede der SAH-Kosten der erweiterten Evaluation mit grün markierten Minimal- und rot markierten Maximalwerten

Zusammenfassung

Wie bereits zu Beginn beschrieben, stellt die kombinierte Methode aus der klassischen SAH-Methode und der SAH-Methode auf Basis der PCA-BVH im Vergleich zur reinen SAH-Methode keine Verbesserung dar. So war die Laufzeit durchschnittlich um 16.57 % höher und die Erstellungszeitdauer um 112.24 % erhöht. Die Anzahl der Traversierungsschritte erhöhte sich um 15.49 %. Lediglich die kumulierten und durchschnittlichen SAH-Kosten konnten um 1.46 % respektive 1.17 % gesenkt werden.

Im Gegensatz dazu kann die kombinierte Methode allerdings eine Verbesserung zur reinen Verwendung der PCA-SAH-Methode bieten. Diese Methode verringert die Renderlaufzeit um 6.65 %, reduziert die Anzahl an Traversierungsschritten um 14.99 % und verringert die theoretischen SAH-Kosten in den kumulierten Kosten um 2.49 %

sowie in den durchschnittlichen Kosten um 1.18 %. Die Erstellungszeitdauer erhöht sich allerdings durch die wiederholte Vorausberechnung der Kosten für die Splits um 63.26 %.

Außerdem lässt sich im Allgemeinen festhalten, dass die klassische SAH-Methode durchschnittlich 7.99-mal so häufig wie die PCA-basierte SAH-Methode benutzt wurde. Der Ergebniswert des „Buch“-Modells wurde hierbei als Ausreißer deklariert und nicht in den Durchschnittswert miteinbezogen.

Der experimentelle Ansatz, eine Erweiterung der Surface Area Heuristic zu nutzen, ergab kaum Verbesserung. Die Renderlaufzeit war im Vergleich zu SAH um 30.87 % erhöht. Die Erstellungszeitdauer war um 188.67 % erhöht. Die Traversierungsschritte waren um 28.45 % erhöht. Die SAH-Kosten waren kumuliert um 35.77 % und im Durchschnitt um 81.01 % höher.

Ebenso war die kombinierte Methode mit der neuen Metrik im Durchschnitt auch gegen die PCA nicht besser, sondern nur auf vereinzelte Szenen. Die Renderlaufzeit war im Durchschnitt um 2.81 % höher, während die Erstellungszeitdauer 124.85 % erhöht war. Die Traversierungschritte konnten allerdings um 5.86 % reduziert werden. Die kumulierten Kosten waren um 21.91 % höher und die durchschnittlichen Kosten waren um 58.86 % erhöht.

Gegen die reine kombinierte Methode mit der klassischen SAH-Metrik benötigt die kombinierte Methode mit der erweiterten Metrik 11.34 % mehr Renderlaufzeit. Die Erstellungslaufzeit erhöht sich um 35.40 % und die Anzahl der Traversierungsschritte steigt um 11.03 %. Die kumulierten Kosten stiegen um 24.63 %, während die Durchschnittskosten um 60.24 % stiegen.

Bei der Ausführungsanzahl ergab sich ein 7.62-mal häufigerer Aufruf der klassischen SAH im Gegensatz zur PCA-basierten SAH-Methode. Der Wert für das „Buch“ wurde als Ausreißer identifziert und nicht in die Betrachtung miteinbezogen.

Interpretation

Hieraus lässt sich als Fazit ziehen, dass die erweiterte Metrik zwar die Überlappungen berücksichtigt, diesen aber eine zu starke Gewichtung beimisst. Dies ist unter anderem an den hohen Kosten nach der ursprünglichen Surface Area Heuristic erkenntlich, die bei gleicher Primitivanzahl vor allem ausdrücken, dass die Oberflächen der Bounding Boxes deutlich größer waren, um Überlappungen möglichst zu vermeiden. Daraus ergibt sich auch das weitere Forschungspotenzial, diese erweiterte Metrik entsprechend weiterzuentwickeln, sodass Überlappungen kostentechnisch genauer abgebildet werden.

5. Schluss

Gegenstand dieser Arbeit war die Untersuchung über die Einsatzmöglichkeit der Hauptkomponentenanalyse im Zusammenhang mit der Erstellung einer Bounding Volume Hierarchy für das Raytracing sowie die Effizienz dieses Ansatzes im Vergleich zur aus Pharr u. a. (2016) bekannten BVH und deren klassischen Splitting-Methode sowie des distanzbasierten Ansatzes aus Sauer (2019).

Hierzu wurden im Grundlagenkapitel zunächst verschiedene Techniken im Kontext des Raytracings dargestellt. Daraufhin wurde näher auf die Bounding Volume Hierarchy eingegangen, hierbei vor allem auf die mathematisch zugrundeliegenden Intersektionsberechnungen. Zum Abschluss des Grundlagenkapitels wurde die Hauptkomponentenanalyse näher betrachtet. Dabei wurde zunächst ein graphischer und anwendungsbezogener Ansatz dargestellt und darauffolgend mittels des mathematischen Hintergrunds die zugrundeliegenden Annahmen dargelegt.

Im Folgenden wurde die theoretische Entwicklung, die praktische Implementierung sowie die abschließende Evaluation entsprechend aufgeteilt. Für die Einsatzmöglichkeit der Hauptkomponentenanalyse wurden die einmalige Nutzung sowie die mehrmalige Nutzung identifiziert. Die einmalige Nutzung beschreibt hierbei die einmalige Transformation der Mittelpunkte der Primitive bevor der Binärbaum der BVH mittels Splitting erstellt wird, wodurch diese Variante eine große Ähnlichkeit zu der in Sauer (2019) vorgestellten Methode aufweist. Daher wurde im weiteren Verlauf der Arbeit die mehrmalige Verwendung betrachtet, um so eine neue Perspektive zu schaffen.

Die Implementierung wurde analog zu Sauer (2019) im PBRT vorgenommen. Hierzu wurde die neue Klasse `PCAAccel` angelegt und deren Details bezüglich der Umsetzung der verschiedenen Funktionen innerhalb dieser Arbeit festgehalten. Dies umfasst unter anderem die Implementierung der PCA ,die zu verwendenden Splitting-Methoden wie die Middle-Methode, die SAH-Methode sowie die aus der klassischen und der PCA-basierten SAH-Methode bestehenden kombinierten Methode.

In der Evaluation wurden die SAH-, die Middle-Methode, die HLBVH und das distanzbasierte Splitting sowie die zwei innerhalb dieser Arbeit entwickelten Ansätze der SAH- und der Middle-Methode unter Verwendung der PCA evaluiert. Die Ergebnisse der Evaluation führten unter anderem dazu, dass die PCA-SAH-Methode wie auch die PCA-Middle-Methode gegenüber der distanzbasierten Methode eine um 17.70 % sowie 13.14 % geringere Laufzeit. Gegenüber der SAH-Methode konnten allerdings keine Vorteile erzielt werden. Dem gegenüber steht die kombinierte Methode, welche theoretisch mindestens gleichwertig mit der klassischen SAH-Methode sein müsste und für kostengünstigere Partitionierungen sowohl die klassische SAH wie auch die PCA-SAH verwendet. Bei der praktischen Evaluation ergab sich allerdings, dass dieser theoretische Vorteil praktisch nicht umsetzbar ist, da die SAH die bei der PCA-basierten SAH-Methode entstehenden Überlappungen nicht kalkulatorisch abbilden kann. Daher wurde die klassische SAH-Metrik, um die Berechnung der Bounding Box der Überlappung

erweitert, sodass auch die Überlappungen die Kosten erhöhen. Daraus ergab sich im Wesentlichen, dass die neue Metrik keinen Vorteil erbringt. Es kann lediglich die PCA-SAH-Methode durch die Verwendung der kombinierten Methode aus der klassischen SAH-Methode und der PCA-SAH-Methode unter anderem eine um 6.65 % geringere Renderlaufzeit erreichen.

Für zukünftige Forschung könnte daher in Frage kommen, wie effizient sich die Nutzung von Oriented Bounding Boxes im Kontext der in dieser Arbeit vorgestellten Ansätze erweist sowie auch die in Stich u. a. (2009) vorgestellten Split-BVHs mögliche Nachteile der vorgestellten Ansätze eliminieren. Auch die genauere Untersuchung der erweiterten Metrik hinsichtlich einer Überarbeitung der durch die Überlappungen tatsächlich verursachten Mehrkosten konnte als Forschungspotenzial identifiziert werden.

6. Reflektion

Nach dem Abschluss der Bearbeitung der Bachelorarbeit soll innerhalb dieses Kapitels noch eine persönliche Reflektion über den Ablauf und die Methodik erfolgen.

Im Vergleich zum Praxisprojektarbeit (Sauer, 2019) werden verschiedene Möglichkeiten weniger stark abgewogen, wodurch vermehrt deskriptiv auf die zugrundeliegenden Vorgänge eingegangen wird. Ein Grund hierfür liegt in der weniger explizit formulierten Fragenstellung der Praxisprojektarbeit und der daraus resultierenden Offenheit in der Lösungsweise, sodass verschiedene Ansätze gegeneinander abgewogen werden mussten, wohingegen innerhalb dieser Arbeit der Ansatz mit der Hauptkomponentenanalyse explizit formuliert war. Daher wurden nur für kleinere Teilaspekte mögliche Optionen abgewogen, wie beispielsweise die Frage welche Splitting-Methode im Zusammenhang mit der Hauptkomponentenanalyse sinnvoll erscheint.

Allerdings konnte innerhalb dieser Arbeit der Evaluationsteil deutlich zahlenbasierter ausgearbeitet werden. Nichtsdestotrotz besteht hinsichtlich der daraus resultierenden Verknüpfung und Wertschöpfung noch Optimierunspotenzial, sodass weitere Korrelationen hätten ausgearbeitet werden können, was auch weitere Möglichkeiten für zukünftige Forschung skizzierbar gemacht hätte. Zudem hätten die beiden Evaluation theoretisch innerhalb einer Evaluation verbunden werden können, was sich aus subjektiver Sicht als nicht sinnvoll erwies, da die beiden Evaluationen unterschiedliche Ziele verfolgen.

Auch Teilaspekte wie die Ermittlung und Auswertung möglicher Umschlagspunkte der Nutzung von PCA-SAH gegenüber klassischer SAH im Rahmen der Verwendung der kombinierten Methode bei der Erstellung der BVH konnten wegen Zeitmangels nicht mehr qualitativ eingearbeitet werden.

Abkürzungsverzeichis

BVH Bounding Volume Hierarchy. 1, 2, 4, 5, 9–11, 16, 25, 27–31, 33–35, 39, 41, 42, 49

HLBVH Hierarchical Bounding Volume Hierarchy. 13, 15, 29, 35, 38, 59–64

PBRT Physically Based Raytracer. 2, 5, 12, 13, 30, 31, 33, 35, 49

PCA Hauptkomponentenanalyse. 1, 4, 5, 18, 19, 21, 22, 25, 27–31, 33, 35, 43, 49, 53

SAH Surface Area Heuristic. 1, 4, 13, 14, 29, 30, 33–35, 37, 39, 41, 43, 44, 48, 49, 54, 64, 65

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Aila u. a. 2013] Aila, Timo ; Karras, Tero ; Laine, Samuli: On quality metrics of bounding volume hierarchies. In: *Proceedings of the 5th High-Performance Graphics Conference* ACM, 2013, S. 101–107

[Arvo u. a. 1986] Arvo, James u. a.: Backward ray tracing. In: *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*, 1986, S. 259–263

[Barré-Brisebois u. a. 2019] Barré-Brisebois, Colin ; Halén, Henrik ; Wihlidal, Graham ; Lauritzen, Andrew ; Bekkers, Jasper ; Stachowiak, Tomasz ; Andersson, Johan: Hybrid Rendering for Real-Time Ray Tracing. In: *Ray Tracing Gems*. Springer, 2019, S. 437–473

[Boksansky u. a. 2019] Boksansky, Jakub ; Wimmer, Michael ; Bittner, Jiri: Ray Traced Shadows: Maintaining Real-Time Frame Rates. In: *Ray Tracing Gems*. Springer, 2019, S. 159–182

[Christensen u. a. 2005] Christensen, Jan H. ; Tomasi, Giorgio ; Hansen, Asger B.: Chemical fingerprinting of petroleum biomarkers using time warping and PCA. In: *Environmental science & technology* 39 (2005), Nr. 1, S. 255–260

[Ernst u. Greiner 2007] Ernst, Manfred ; Greiner, Gunther: Early split clipping for bounding volume hierarchies. In: *2007 IEEE Symposium on Interactive Ray Tracing* IEEE, 2007, S. 73–78

[Garanzha u. a. 2011] Garanzha, Kirill ; Premože, Simon ; Bely, Alexander ; Galaktionov, Vladimir: Grid-based SAH BVH construction on a GPU. In: *The Visual Computer* 27 (2011), Nr. 6-8, S. 697–706

[Glassner 1989] Glassner, Andrew S.: *An introduction to ray tracing*. Elsevier, 1989

[Goldsmith u. Salmon 1987] Goldsmith, Jeffrey ; Salmon, John: Automatic creation of object hierarchies for ray tracing. In: *IEEE Computer Graphics and Applications* 7 (1987), Nr. 5, S. 14–20

[Gottumukkal u. Asari 2004] Gottumukkal, Rajkiran ; Asari, Vijayan K.: An improved face recognition technique based on modular PCA approach. In: *Pattern Recognition Letters* 25 (2004), Nr. 4, S. 429–436

[Gourmel u. a. 2010] Gourmel, Olivier ; Pajot, Anthony ; Paulin, Mathias ; Barthe, Loïc ; Poulin, Pierre: Fitted BVH for fast raytracing of metaballs. In: *Computer Graphics Forum* Bd. 29 Wiley Online Library, 2010, S. 281–288

[Gu u. a. 2013] Gu, Yan ; He, Yong ; Fatahalian, Kayvon ; Blelloch, Guy: Efficient BVH construction via approximate agglomerative clustering. In: *Proceedings of the 5th High-Performance Graphics Conference* ACM, 2013, S. 81–88

[Guennebaud u. a. 2010] Guennebaud, Gaël ; Jacob, Benoît u. a.: *Eigen v3*. http://eigen.tuxfamily.org, 2010

[Jensen u. Christensen 2007] Jensen, Henrik W. ; Christensen, Per: High quality rendering using ray tracing and photon mapping. In: *ACM SIGGRAPH 2007 courses* ACM, 2007, S. 1

[Jolliffe 2011] Jolliffe, Ian: *Principal component analysis*. Springer, 2011

[Lafortune u. Willems 1993] Lafortune, Eric P. ; Willems, Yves D.: Bi-directional path tracing. (1993)

[Liu u. a. 2019] Liu, Edward ; Llamas, Ignacio ; Kelly, Patrick u. a.: Cinematic Rendering in UE4 with Real-Time Ray Tracing and Denoising. In: *Ray Tracing Gems*. Springer, 2019, S. 289–319

[Marinetti u. a. 2004] Marinetti, Sergio ; Grinzato, Ermanno ; Bison, Paolo G. ; Bozzi, Edoardo ; Chimenti, Massimo ; Pieri, Gabriele ; Salvetti, Ovidio: Statistical analysis of IR thermographic sequences by PCA. In: *Infrared Physics & Technology* 46 (2004), Nr. 1-2, S. 85–91

[Novembre u. Stephens 2008] Novembre, John ; Stephens, Matthew: Interpreting principal component analyses of spatial population genetic variation. In: *Nature genetics* 40 (2008), Nr. 5, S. 646

[pbrt.org] pbrt.org: *PBRT-Renderszenen*. `https://pbrt.org/scenes-v3.html`, . – Zugriff am: 15.06.2019

[Pharr u. a. 2016] Pharr, Matt ; Jakob, Wenzel ; Humphreys, Greg: *Physically Based Rendering: From Theory to Implementation (3rd ed.)*. 3rd. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2016. – 1266 S. – ISBN 9780128006450

[Sauer 2019] Sauer, Matthias: *Entwicklung und Evaluation einer distanzbasierten Splitting-Methode für Bounding Volume Hierarchies*. 2019. – Praxisprojektarbeit an der TH Köln, Betreuer: Dr.-Ing. Martin Eisemann

[Shlens 2014] Shlens, Jonathon: A tutorial on principal component analysis. In: *arXiv preprint arXiv:1404.1100* (2014)

[Smith 2002] Smith, Lindsay I.: A tutorial on principal components analysis. 2002. – Forschungsbericht

[Stich u. a. 2009] Stich, Martin ; Friedrich, Heiko ; Dietrich, Andreas: Spatial splits in bounding volume hierarchies. In: *Proceedings of the Conference on High Performance Graphics 2009* ACM, 2009, S. 7–13

[Vinkler u. a. 2012] Vinkler, Marek ; Havran, Vlastimil ; Sochor, Jiří: Visibility driven BVH build up algorithm for ray tracing. In: *Computers & Graphics* 36 (2012), Nr. 4, S. 283–296

[Whitted 1979] Whitted, Turner: An improved illumination model for shaded display. In: *ACM SIGGRAPH Computer Graphics* Bd. 13 ACM, 1979, S. 14

[Wold u. a. 1987] Wold, Svante ; Esbensen, Kim ; Geladi, Paul: Principal component analysis. In: *Chemometrics and intelligent laboratory systems* 2 (1987), Nr. 1-3, S. 37–52

[Yang u. a. 2004] Yang, Jian ; Zhang, David D. ; Frangi, Alejandro F. ; Yang, Jing-yu: Two-dimensional PCA: a new approach to appearance-based face representation and recognition. In: *IEEE transactions on pattern analysis and machine intelligence* (2004)

A. Evaluationsergebnisse

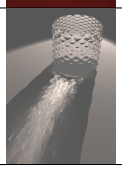
Szene	SAH	Middle	HLBVH	Distanzbasiert	PCA (Middle)	PCA (SAH)
	9:33.29 (99.12%)	10:01.28 (99.35%)	11:59.04 (99.77%)	13:22.65 (99.43%)	11:25.32 (99.16%)	10:55.57 (99.17%)
	2:10.70 (99.92%)	2:19.69 (99.94%)	2:13.58 (99.95%)	2:42.68 (99.93%)	2:28.68 (99.93%)	2:19.69 (99.91%)
	2:48.82 (99.94%)	2:51.77 (99.94%)	3:08.59 (99.97%)	8:42.86 (99.98%)	5:36.71 (99.98%)	5:34.68 (99.97%)
	12:17.35 (98.30%)	12:35.65 (98.54%)	Fehler	20:16.54 (99.03%)	17:35.02 (98.89%)	17:14.18 (98.89%)
	5:49.69 (99.52%)	5:54:47 (99.57%)	5:41.22 (99.59%)	6:38.58 (99.60%)	6:17.75 (99.55%)	6:13.80 (99.53%)
	6:30.27 (94.79%)	6:41.42 (95.58%)	6:40.16 (96.01%)	16:53.83 (98.14%)	12:00.65 (97.10%)	9:51.94 (96.43%)
	2:33.00 (99.28%)	2:47.99 (99.38%)	2:28.69 (99.29%)	3:07.97 (99.42%)	2:50.76 (99.39%)	2:38.74 (99.34%)
	6:30.49 (97.89%)	6:36.11 (98.88%)	14:54.88 (99.80%)	9:51.56 (98.97%)	9:10.42 (98.44%)	8:31.61 (98.13%)
	5:01.99 (95.22%)	5:10.13 (95.94%)	5:29.55 (96.53%)	6:48.43 (96.77%)	6:14.16 (96.28%)	6:05.19 (96.19%)
	3:42.21 (98.93%)	3:47.72 (99.16%)	3:38.22 (99.20%)	4:32.92 (99.25%)	4:16.01 (99.12%)	4:01.04 (99.07%)
	2:28.48 (99.35%)	2:28.02 (99.66%)	2:26.77 (99.69%)	3:23.12 (99.71%)	2:57.21 (99.58%)	2:51.31 (99.53%)
	7:36.71 (97.47%)	8:31.82 (97.83%)	9:23.93 (98.49%)	10:18.71 (98.13%)	9:29.62 (97.78%)	8:51.40 (97.59%)

Tabelle A.1.: Renderlaufzeit der verschiedenen Szenen in Minuten

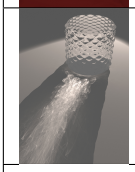
Szene	SAH	Middle	HLBVH	Distanz-basiert	PCA (Middle)	PCA (SAH)
	3.00 (0.52%)	1.69 (0.28%)	0.17 (0.02%)	2.48 (0.31%)	4.17 (0.60%)	4.25 (0.64%)
	0.05 (0.05%)	0.02 (0.02%)	0.01 (0.01%)	0.03 (0.02%)	0.05 (0.04%)	0.07 (0.05%)
	0.03 (0.02%)	0.01 (0.01%)	0.00 (0.00%)	0.02 (0.01%)	0.03 (0.01%)	0.04 (0.01%)
	2.03 (0.27%)	1.12 (0.15%)	Fehler	1.76 (0.14%)	2.45 (0.23%)	2.84 (0.27%)
	0.30 (0.09%)	0.14 (0.04%)	0.02 (0.01%)	0.23 (0.06%)	0.34 (0.09%)	0.39 (0.11%)
	4.28 (1.04%)	2.47 (0.59%)	0.46 (0.11%)	3.29 (0.32%)	5.43 (0.73%)	5.69 (0.93%)
	0.01 (0.01%)	0.00 (0.00%)	0.00 (0.00%)	0.00 (0.00%)	0.01 (0.01%)	0.01 (0.01%)
	6.37 (1.60%)	3.18 (0.80%)	0.32 (0.04%)	4.79 (0.80%)	7.42 (1.33%)	8.40 (1.61%)
	2.17 (0.69%)	1.51 (0.47%)	0.16 (0.05%)	1.77 (0.42%)	2.63 (0.68%)	2.65 (0.70%)
	0.45 (0.20%)	0.22 (0.10%)	0.04 (0.02%)	0.13 (0.35%)	0.55 (0.22%)	0.56 (0.23%)
	0.27 (0.18%)	0.13 (0.09%)	0.03 (0.03%)	0.21 (0.11%)	0.30 (0.17%)	0.37 (0.22%)
	3.65 (0.78%)	1.84 (0.35%)	0.29 (0.05%)	2.82 (0.45%)	4.33 (0.74%)	5.11 (0.94%)

Tabelle A.2.: Erstellungszeitdauer der Bounding Volume Hierarchy für die verschiedenen Szenen in Sekunden

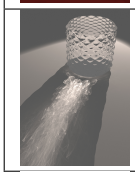
Szene	SAH	Middle	HLBVH	Distanzbasiert	PCA (Middle)	PCA (SAH)
	229.15 MiB	266.74 MiB	67.65 MiB	266.74 MiB	266.74 MiB	231.43 MiB
	6.59 MiB	6.71 MiB	3.87 MiB	6.71 MiB	6.71 MiB	6.60 MiB
	3.90 MiB	4.20 MiB	2.42 MiB	4.20 MiB	4.20 MiB	3.93 MiB
	157.88 MiB	199.11 MiB	Fehler	199.11 MiB	199.09 MiB	159.47 MiB
	26.41 MiB	30.74 MiB	14.17 MiB	30.73 MiB	30.69 MiB	26.67 MiB
	339.28 MiB	404.02 MiB	180.77 MiB	404.02 MiB	404.02 MiB	341.33 MiB
	1009.08 kB	1.01 MiB	968.08 kB	1.01 MiB	1.01 MiB	1012.70 kB
	545.98 MiB	550.65 MiB	124.56 MiB	550.65 MiB	550.65 MiB	545.66 MiB
	152.79 MiB	192.04 MiB	54.45 MiB	192.04 MiB	192.03 MiB	154.48 MiB
	35.33 MiB	40.80 MiB	16.10 MiB	40.80 MiB	40.80 MiB	35.94 MiB
	23.75 MiB	28.11 MiB	12.94 MiB	28.11 MiB	28.11 MiB	23.82 MiB
	277.47 MiB	323.29 MiB	104.66 MiB	323.29 MiB	323.10 MiB	281.19 MiB

Tabelle A.3.: Speicherplatzbedarf der verschiedenen Szenen

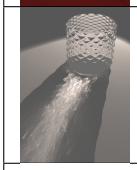
Szene	SAH	Middle	HLBVH	Distanzbasiert	PCA (Middle)	PCA (SAH)
	1:53.94 (19.70%) & 1:25.98 (14.87%)	2:13.07 (21.99%) & 1:36.17 (15.89%)	2:21.15 (19.58%) & 1:32.67 (12.86%)	4:30.03 (33.45%) & 1:51.83 (13.85%)	3:08.38 (27.27%) & 1:37.24 (14.07%)	2:53.06 (26.18%) & 1:27.21 (13.19%)
	0:26.51 (20.27%) & 0:03.68 (2.82%)	0:33.76 (24.16%) & 0:05.13 (3.67%)	0:27.12 (20.30%) & 0:03.67 (2.75%)	0:51.63 (31.72%) & 0:06.66 (4.09%)	0:40.73 (27.38%) & 0:05.88 (3.96%)	0:32.95 (23.57%) & 0:04.00 (2.86%)
	1:27.46 (51.78%) & 0:15.99 (9.47%)	1:31:23 (53.08%) & 0:16.34 (12.77%)	1:38.66 (52.30%) & 0:17.07 (9.05%)	6:05.03 (69.80%) & 0:17.90 (3.42%)	3:40.51 (65.47%) & 0:14.87 (4.42%)	3:33.53 (63.78%) & 0:16.23 (4.85%)
	6:10.07 (49.34%) & 1:02.77 (8.37%)	6:34.87 (51.50%) & 1:06.30 (8.65%)	Fehler	12:46.97 (62.43%) & 0:44.33 (3.61%)	10:25.59 (58.70%) & 1:13.43 (6.89%)	10:02.18 (57.58%) & 0:54.87 (5.25%)
	1:02.60 (17.82%) & 0:35.84 (10.20%)	1:08.40 (19.22%) & 0:38.72 (10.88%)	0:59.78 (17.45%) & 0:30.49 (8.90%)	1:36.69 (24.16%) & 0:41.82 (10.45%)	1:22.76 (21.81%) & 0:38.75 (10.21%)	1:17.49 (20.63%) & 0:38.46 (10.24%)
	2:14.16 (32.59%) & 0:41.79 (10.15%)	2:26.91 (34.98%) & 0:46.30 (11.02%)	2:17.98 (33.10%) & 0:42.83 (10.28%)	10:22.77 (60.29%) & 1:08.92 (6.67%)	6:30.48 (52.61%) & 0:59.47 (8.01%)	4:45.60 (46.53%) & 0:48.84 (7.96%)
	0:45.85 (29.75%) & 0:11.22 (7.29%)	0:59.99 (35.49%) & 0:15.00 (8.88%)	0:43.74 (29.21%) & 0:11.05 (7.38%)	1:11.77 (37.96%) & 0:17.59 (9.31%)	0:59.90 (34.87%) & 0:14.46 (8.42%)	0:49.87 (31.21%) & 0:12.55 (7.85%)
	3:27.77 (52.09%) & 0:27.16 (6.81%)	3:35.60 (53.82%) & 0:27.81 (6.94%)	5:55.37 (39.63%) & 0:38.96 (4.35%)	6:23.74 (64.20%) & 0:29.65 (4.96%)	5:48.32 (62.29%) & 0:29.41 (5.26%)	5:12.86 (60.01%) & 0:27.36 (5.25%)
	1:37.18 (30.65%) & 0:22.09 (6.97%)	1:51.22 (34.41%) & 0:26.15 (8.09%)	1:45.03 (30.77%) & 0:24.82 (7.27%)	3:02.97 (43.35%) & 0:26.19 (6.21%)	2:36.68 (40.32%) & 0:24.29 (6.25%)	2:26.62 (38.62%) & 0:21.10 (5.56%)
	0:54.37 (24.21%) & 0:43.19 (19.23%)	0:57.70 (25.13%) & 0:50.01 (21.78%)	0:53.26 (24.21%) & 0:43.49 (19.77%)	1:29.01 (32.37%) & 0:52.65 (19.15%)	1:13.90 (28.61%) & 0:53.76 (20.82%)	1:07.07 (27.57%) & 0:46.75 (19.22%)
	0:34.71 (23.23%) & 0:06.87 (4.60%)	0:37.65 (25.35%) & 0:07.32 (4.93%)	0:35.75 (24.28%) & 0:07.04 (4.78%)	1:15.94 (37.28%) & 0:07.48 (3.67%)	0:58.68 (32.98%) & 0:06.70 (3.77%)	0:53.27 (30.95%) & 0:07.37 (4.29%)
	2:05.41 (26.77%) & 1:12.86 (15.55%)	2:34.98 (29.62%) & 1:30.40 (17.28%)	2:53.30 (30.27%) & 1:40.40 (17.54%)	3:56.28 (37.47%) & 1:35.84 (15.20%)	3:25.56 (35.29%) & 1:27.07 (14.95%)	2:55.04 (32.15%) & 1:15.51 (13.87%)

Tabelle A.4.: Intersektionslaufzeit für die `Intersect`- und `IntersectP`-Methode in Minuten

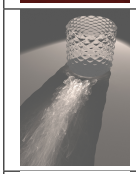
Szene	SAH	Middle	HLBVH	Distanzbasiert	PCA (Middle)	PCA (SAH)
	~31,7 Mrd.	~40,0 Mrd.	~30,2 Mrd.	~56,4 Mrd.	~43,5 Mrd.	~37,3 Mrd.
	~4,9 Mrd.	~6,9 Mrd.	~4,7 Mrd.	~10,2 Mrd.	~8,7 Mrd.	~5,9 Mrd.
	~12,0 Mrd.	~13,2 Mrd.	~11,9 Mrd.	~39,5 Mrd.	~25,9 Mrd.	~24,0 Mrd.
	~48,0 Mrd.	~54,8 Mrd.	Fehler	~80,6 Mrd.	~79,1 Mrd.	~68,1 Mrd.
	~17,8 Mrd.	~20,3 Mrd.	~15,7 Mrd.	~24,4 Mrd.	~22,4 Mrd.	~20,4 Mrd.
	~25,4 Mrd.	~29,7 Mrd.	~26,7 Mrd.	~78,5 Mrd.	~58,7 Mrd.	~43,4 Mrd.
	~9,5 Mrd.	~14,1 Mrd.	~9,5 Mrd.	~17,0 Mrd.	~14,9 Mrd.	~11,1 Mrd.
	~27,8 Mrd.	~29,9 Mrd.	~24,6 Mrd.	~48,5 Mrd.	~46,0 Mrd.	~40,8 Mrd.
	~16,1 Mrd.	~19,7 Mrd.	~16,5 Mrd.	~27,3 Mrd.	~24,7 Mrd.	~22,4 Mrd.
	~17,5 Mrd.	~20,9 Mrd.	~17,5 Mrd.	~24,1 Mrd.	~23,6 Mrd.	~20,3 Mrd.
	~6,3 Mrd.	~7,0 Mrd.	~6,1 Mrd.	~12,0 Mrd.	~9,7 Mrd.	~8,9 Mrd.
	~33,0 Mrd.	~41,0 Mrd.	~43,0 Mrd.	~51,5 Mrd.	~47,8 Mrd.	~40,0 Mrd.

Tabelle A.5.: Traversierungsschritte für das Durchlaufen der Bounding Volume Hierarchy

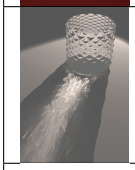
Szene	SAH	Middle	HLBVH	Distanzbasiert	PCA (Middle)	PCA (SAH)
	50.315.728 / 2.869.338 (17.54x)	70.188.608 / 3.485.179 (20.14x)	51.353.238 / 223.294 (229.98x)	66.653.000 / 3.485.179 (19.12x)	62.757.945 / 3.485.153 (18.01x)	51.978.349 / 2.906.771 (17.88x)
	940.132 / 85.932 (10.94x)	943.555 / 87.888 (10.74x)	881.102 / 41.366 (21.30x)	918.415 / 87.888 (10.45x)	922.418 / 87.888 (10.50x)	944.946 / 86.066 (10.98x)
	240.376 / 50.130 (4.80x)	254.090 / 54.963 (4.62x)	516.748 / 25.888 (19.96x)	254.311 / 54.963 (4.63x)	569.096 / 54.963 (10.35x)	541.262 / 50.544 (10.71x)
	33.648.956 / 1.920.355 (17.52x)	42.992.675 / 2.595.878 (16.56x)	Fehler	42.157.037 / 2.595.878 (16.24x)	40.311.796 / 2.595.665 (15.53x)	31.868.006 / 1.946.412 (16.37x)
	5.554.478 / 320.415 (17.34x)	6.755.724 / 391.327 (17.26x)	5.776.035 / 119.915 (48.17x)	6.735.641 / 391.249 (17.22x)	6.471.341 / 390.546 (16.57x)	5.643.725 / 324.766 (17.38x)
	72.254.591 / 4.230.809 (17.08x)	82.885.093 / 5.291.585 (15.66x)	75.861.465 / 1.633.859 (46.43x)	83.461.213 / 5.291.585 (15.77x)	80.449.378 / 5.291.565 (15.20x)	73.169.656 / 4.264.454 (17.16x)
	161.703 / 10.832 (14.93x)	167.242 / 11.246 (14.87x)	170.143 / 10.176 (16.72x)	182.722 / 11.246 (16.25x)	166.011 / 11.246 (14.76x)	164.711 / 10.890 (15.12x)
	102.062.208 / 7.140.454 (14.29x)	117.871.840 / 7.216.953 (16.33x)	99.269.428 / 235.885 (420.84x)	115.723.672 / 7.216.953 (16.03x)	113.698.253 / 7.216.953 (15.75x)	104.291.407 / 7.135.280 (14.62x)
	36.170.495 / 1.847.003 (19.58x)	47.429.621 / 2.490.112 (19.05x)	39.849.562 / 235.842 (168.97x)	47.578.756 / 2.490.110 (19.11x)	46.819.340 / 2.489.960 (18.80x)	36.474.332 / 1.874.632 (19.46x)
	7.735.148 / 428.799 (18.04x)	9.965.295 / 518.476 (19.22x)	8.535.160 / 113.747 (75.04x)	9.830.591 / 518.476 (18.96x)	9.535.608 / 518.476 (18.39x)	8.095.335 / 438.724 (18.45x)
	4.395.883 / 296.200 (14.84x)	5.408.119 / 367.748 (14.71x)	5.066.878 / 119.178 (42.52x)	5.648.324 / 367.748 (15.36x)	5.309.242 / 367.748 (14.44x)	4.397.974 / 297.347 (14.79x)
	63.981.352 / 3.412.834 (18.75x)	74.657.402 / 4.163.410 (17.93x)	67.367.670 / 581.415 (115.87x)	74.359.851 / 4.163.407 (17.86x)	73.231.377 / 4.160.301 (17.60x)	64.370.659 / 3.473.721 (18.53x)

Tabelle A.6.: Kumulierte Kosten, Anzahl der Splits sowie durchschnittliche Kosten der Surface Area Heuristic

Szenen	Methode	RL	EZ	TS	SAH-K.	Anzahl Aufrufe (SAH / PCA-SAH)
Krone	SAH	9:27.95	3.02	∼31.7 Mrd.	50.315.727 / 2.869.338 (17.54x)	1.504.733 / 243.969 (6.17x) (Comb)
	PCA	10:55.57	4.25	∼37.3 Mrd.	51.978.349 / 2.906.771 (17.88x)	
	Comb	10:52.59	6.13	∼34.3 Mrd.	49.423.245 / 2.868.217 (17.23x)	1.216.733 / 228.405 (5.33x) (Comb + M)
	Comb + M	11:03.01	6.26	∼34.5 Mrd.	57.726.112 / 2.138.254 (27.00x)	
Glas	SAH	2:10.70	0.05	∼4.9 Mrd.	940.132 / 85.932 (10.94x)	42.001 / 5.296 (7.93x) (Comb)
	PCA	2:19.69	0.07	∼5.9 Mrd.	944.946 / 86.066 (10.98x)	
	Comb	2:15.83	0.12	∼5.3 Mrd.	925.113 / 85.582 (10.81x)	36.016 / 3.184 (11.31x) (Comb + M)
	Comb + M	2:17.86	0.12	∼5.0 Mrd.	919.671 / 67.056 (13.71x)	
Buddha	SAH	2:48.82	0.03	∼12.0 Mrd.	240.376 / 50.130 (4.80x)	25.167 / 4.371 (5.76x) (Comb)
	PCA	5:34.68	0.08	∼24.Mrd.	541.262 / 50.544 (10.71x)	
	Comb	4:28.91	0.08	∼18.4 Mrd.	524.983 / 49.914 (10.52x)	18.012 / 3.512 (5.13x) (Comb + M)
	Comb + M	5:11.89	0.07	∼19.1 Mrd.	567.078 / 28.844 (19.66x)	
San Miguel	SAH	12:17.96	2.15	∼48.0 Mrd.	33.648.956 / 1.920.355 (17.52x)	1.039.125 / 204.177 (5.09x) (Comb)
	PCA	17:14.18	2.84	∼68.1 Mrd.	31.868.006 / 1.946.412 (16.37x)	
	Comb	15:37.06	4.82	∼57.3 Mrd.	32.412.826 / 1.891.276 (17.14x)	919.092 / 243.736 (3.77x) (Comb + M)
	Comb + M	18:08.66	5.74	∼70.4 Mrd.	62.164.212 / 1.559.426 (39.86x)	
Badezimmer	SAH	5:50.67	0.32	∼17.8 Mrd.	5.554.478 / 320.415 (17.34x)	189.619 / 15.897 (11.93x) (Comb)
	PCA	6:13.88	0.39	∼20.4 Mrd.	5.643.725 / 324.766 (17.38x)	
	Comb	6:06.16	0.66	∼18.8 Mrd.	5.536.436 / 319.320 (17.34x)	160.617 / 15.321 (10.48x) (Comb + M)
	Comb + M	7:37.46	1.69	∼28.0 Mrd.	10.083.643 / 247.491 (40.74x)	
Pavilion	SAH	6:31.28	4.67	∼25.4 Mrd.	72.254.591 / 4.230.809 (17.08x)	2.226.634 / 373.667 (5.96x) (Comb)
	PCA	9:51.94	5.69	∼43.4 Mrd.	73.169.656 / 4.264.454 (17.16x)	
	Comb	8:21.71	10.07	∼32.7 Mrd.	71.507.739 / 4.223.795 (16.93x)	2.046.161 / 446.575 (4.58x)(Comb + M)
	Comb + M	11:21.05	20.83	∼36.2 Mrd.	83.917.675 / 3.633.982 (23.09x)	
Buch	SAH	2:33.00	0.01	∼9.5 Mrd.	161.703 / 10.832 (14.93x)	10.373 / 167 (62.11x) (Comb)
	PCA	2:38.74	0.01	∼11.1 Mrd.	164.711 / 10.890 (15.12x)	
	Comb	2:34.96	0.02	∼9.6 Mrd.	160.855 / 10.851 (14.82x)	10.334 / 152 (67.99x) (Comb + M)
	Comb + M	2:39.01	0.02	∼10.5 Mrd.	177.963 / 10.696 (16.64x)	
Drache	SAH	6:30.49	6.37	∼27.8 Mrd.	102.062.208 / 7.140.454 (14.29x)	3.556.805 / 560.074 (6.35x) (Comb)
	PCA	8:31.61	8.40	∼40.8 Mrd.	104.291.407 / 7.135.280 (14.62x)	
	Comb	7:15.36	13.46	∼32.3 Mrd.	101.879.371 / 7.128.710 (14.29x)	2.795.354 / 480.855 (5.81x) (Comb + M)
	Comb + M	7:58.69	13.24	∼33.1 Mrd.	105.850.914 / 5.272.941 (20.07x)	
Villa	SAH	4:57.01	2.16	∼16.1 Mrd.	36.170.495 / 1.847.003 (19.58x)	1.038.226 / 130.740 (7.94x) (Comb)
	PCA	6:05.19	2.65	∼22.4 Mrd.	36.474.332 / 1.874.632 (19.46x)	
	Comb	5:44.17	4.47	∼18.1 Mrd.	35.387.039 / 1.844.585 (19.18x)	Fehler (Comb + M)
	Comb + M	Fehler				
White Room	SAH	3:40.41	0.46	∼17.5 Mrd.	7.735.148 / 428.799 (18.04x)	257.684 / 20.811 (12.38x) (Comb)
	PCA	4:01.04	0.56	∼20.3 Mrd.	8.095.335 / 438.724 (18.45x)	
	Comb	3:54.01	0.94	∼17.0 Mrd.	7.564.126 / 429.773 (17.60x)	230.546 / 19.105 (12.07x) (Comb + M)
	Comb + M	4:04.03	0.98	∼19.5 Mrd.	8.682.097 / 360.015 (24.12x)	
Audi TT	SAH	2:28.48	0.27	∼6.3 Mrd.	4.395.883 / 296.200 (14.84x)	161.952 / 13.769 (11.76x) (Comb)
	PCA	2:51.31	0.37	∼8.9 Mrd.	4.397.974 / 297.347 (14.79x)	
	Comb	2:45.95	0.56	∼7.8 Mrd.	4.367.572 / 294.624 (14.82x)	131.209 / 13.020 (10.08x) (Comb + M)
	Comb + M	2:46.95	0.54	∼8.0 Mrd.	4.430.049 / 216.133 (20.50x)	
Measure One	SAH	7:36.71	3.65	∼33.0 Mrd.	63.981.352 / 3.412.834 (18.75x)	1.895.817 / 285.169 (6.65x) (Comb)
	PCA	8:51.40	5.11	∼40.0 Mrd.	64.370.659 / 3.473.721 (18.53x)	
	Comb	8:45.23	7.81	∼36.9 Mrd.	62.808.472 / 3.398.984 (18.48x)	Fehler (Comb + M)
	Comb + M	Fehler				

Tabelle A.7.: Evaluation der kombinierten SAH-Methode mit Renderlaufzeit (RL), Erstellungszeitdauer (EZ), Traversierungsschritten (TS), Surface Area Heuristic-Kosten (SAH-K.) und der Anzahl der Aufrufe der klassischen im Vergleich zur PCA-basierten SAH-Methode

B. Codeausschnitte

Der komplette Quellcode kann im Repository unter der URL `https://github.com/MSauerM/pbrt-v3_PCA-BVH` im Verzeichnis `src/accelerators` im File `bvh_pca.cpp` eingesehen werden.

```
std::shared_ptr<Primitive> MakeAccelerator(
    const std::string &name,
    std::vector<std::shared_ptr<Primitive>> prims,
    const ParamSet &paramSet) {
    std::shared_ptr<Primitive> accel;
    if (name == "bvh")
        accel = CreateBVHAccelerator(std::move(prims), paramSet);
    else if (name == "kdtree")
        accel = CreateKdTreeAccelerator(std::move(prims), paramSet);
    else if (name == "pcabvh")
        accel = CreatePCAAccelerator(std::move(prims), paramSet);
    else
        Warning("Accelerator \"%s\" unknown.", name.c_str());
    paramSet.ReportUnused();
    return accel;
}
```

Listing B.1: Erweiterte MakeAccelerator-Funktion (Pharr u. a., 2016)

```
std::shared_ptr<PCAAccel> CreatePCAAccelerator( std::vector<std::
    shared_ptr<Primitive>> prims, const ParamSet &ps)
    {
        std::string splitMethodName = ps.FindOneString("splitmethod",
            "sah");
        PCAAccel::SplitMethod  splitMethod;
        if(splitMethodName == "sah")
            splitMethod = PCAAccel::SplitMethod::SAH;
        else if(splitMethodName == "middle")
            splitMethod = PCAAccel::SplitMethod::Middle;
        else{
            Warning("BVH split method \"%s\" unknown. Using \"sah\".",
                    splitMethodName.c_str());
            splitMethod = PCAAccel::SplitMethod::SAH;
        }

        int maxPrimsInNode = ps.FindOneInt("maxnodeprims",4);
        return std::make_shared<PCAAccel>(std::move(prims),
            maxPrimsInNode, splitMethod);
    }
```

Listing B.2: Erzeugungsfunktion für die PCA-BVH (Pharr u. a., 2016)

```
Point3f middle = Point3f();
for (int k = start; k < end; ++k) {
    middle += primitiveInfo[k].pcacentroid;
}
middle = middle / (end-start);
Vector3f mean = Vector3f(middle);

for (int j = start; j < end ; ++j) {
    primitiveInfo[j].pcacentroid-= mean;
}

Eigen::MatrixXf centeredPrimMidpoints = Eigen::MatrixXf(end-start,3);

for (int l = start; l < end ; ++l) {
    centeredPrimMidpoints(l-start,0) = primitiveInfo[l].pcacentroid.x;
    centeredPrimMidpoints(l-start,1) = primitiveInfo[l].pcacentroid.y;
    centeredPrimMidpoints(l-start,2) = primitiveInfo[l].pcacentroid.z;
}

Eigen::MatrixXf covarianceMatrix =
            (centeredPrimMidpoints.adjoint() * centeredPrimMidpoints)
                        / (centeredPrimMidpoints.rows()-1);

Eigen::SelfAdjointEigenSolver<Eigen::MatrixXf>
                                    eigen(covarianceMatrix);

Eigen::MatrixXf eigenVectors = eigen.eigenvectors();
Eigen::MatrixXf pcaTransformationMatrix = eigenVectors.rightCols(3);

centeredPrimMidpoints =
            centeredPrimMidpoints * pcaTransformationMatrix;

for (int m = start; m < end; ++m) {
    primitiveInfo[m].pcacentroid =
        Point3f(centeredPrimMidpoints(m-start,0),
                centeredPrimMidpoints(m-start,1),
                centeredPrimMidpoints(m-start,2));
}
```

Listing B.3: Implementierung der PCA in der `recursiveBuild`-Funktion

```
//...
Bounds3f centroidBounds;
for (int i = start; i < end; ++i)
    centroidBounds = Union(centroidBounds, primitiveInfo[i].
       pcacentroid);
int dim = centroidBounds.MaximumExtent();
//...
```

Listing B.4: Anpassungen innerhalb der `recursiveBuild`-Funktion für die PCA-BVH (Pharr u. a., 2016)

```
case SplitMethod::Middle: {
// Partition primitives through node's midpoint
Float pmid =
(centroidBounds.pMin[dim] + centroidBounds.pMax[dim]) / 2;
BVHPrimitiveInfo *midPtr = std::partition(
                &primitiveInfo[start], &primitiveInfo[end - 1] + 1,
                    [dim, pmid](const BVHPrimitiveInfo &pi) {
                           return pi.pcacentroid[dim] < pmid;
                      });
mid = midPtr - &primitiveInfo[0];

// For lots of prims with large overlapping bounding boxes, this
// may fail to partition; in that case don't break and fall
// through
// to EqualCounts.
if (mid != start && mid != end){
    break;
}
else{
mid = (start + end) / 2;
std::nth_element(&primitiveInfo[start],
                &primitiveInfo[mid],
                &primitiveInfo[end - 1] + 1,
        [dim](const BVHPrimitiveInfo &a, const BVHPrimitiveInfo &b) {
                    return a.pcacentroid[dim] < b.pcacentroid[dim];
            });
    break;
    }
}
```

Listing B.5: PCA-Middle-Splitting (Pharr u. a., 2016)

```
if (nPrimitives <= 2) {
    // Partition primitives into equally-sized subsets
    mid = (start + end) / 2;
    std::nth_element(&primitiveInfo[start], &primitiveInfo[mid],
                     &primitiveInfo[end - 1] + 1,
                  [dim](const BVHPrimitiveInfo &a, const
                      BVHPrimitiveInfo &b) {
                     return a.pcacentroid[dim] < b.pcacentroid[dim];
                     });
} else {
     // Allocate _BucketInfo_ for SAH partition buckets
     PBRT_CONSTEXPR int nBuckets = 12;
     BucketInfo buckets[nBuckets];

     // Initialize _BucketInfo_ for SAH partition buckets
     for (int i = start; i < end; ++i) {
        int b = nBuckets *  centroidBounds.Offset(primitiveInfo[i].
            pcacentroid)[dim];
        if (b == nBuckets) b = nBuckets - 1;
        CHECK_GE(b, 0);
        CHECK_LT(b, nBuckets);
        buckets[b].count++;
```

```
        buckets[b].bounds =
            Union(buckets[b].bounds, primitiveInfo[i].bounds);
    }

    // Compute costs for splitting after each bucket
    Float cost[nBuckets - 1];
    for (int i = 0; i < nBuckets - 1; ++i) {
        Bounds3f b0, b1;
        int count0 = 0, count1 = 0;
        for (int j = 0; j <= i; ++j) {
            b0 = Union(b0, buckets[j].bounds);
            count0 += buckets[j].count;
        }
        for (int j = i + 1; j < nBuckets; ++j) {
            b1 = Union(b1, buckets[j].bounds);
            count1 += buckets[j].count;
        }
        cost[i] = 1 +
          (count0 * b0.SurfaceArea() +
          count1 * b1.SurfaceArea()) /
         bounds.SurfaceArea();
    }

    // Find bucket to split at that minimizes SAH metric
    Float minCost = cost[0];
    int minCostSplitBucket = 0;
    for (int i = 1; i < nBuckets - 1; ++i) {
        if (cost[i] < minCost) {
            minCost = cost[i];
            minCostSplitBucket = i;
        }
    }
    // Either create leaf or split primitives at selected SAH
    // bucket
    Float leafCost = nPrimitives;
    if (nPrimitives > maxPrimsInNode || minCost < leafCost) {
        BVHPrimitiveInfo *pmid = std::partition(
        &primitiveInfo[start], &primitiveInfo[end - 1] + 1,
            [=](const BVHPrimitiveInfo &pi) {
                    int b = nBuckets *
                        centroidBounds.Offset(pi.pcacentroid)[dim];
                    if (b == nBuckets) b = nBuckets - 1;
                    CHECK_GE(b, 0);
                    CHECK_LT(b, nBuckets);
                    return b <= minCostSplitBucket;
                });
    mid = pmid - &primitiveInfo[0];
    } else {
    // Create leaf _BVHBuildNode_
    int firstPrimOffset = orderedPrims.size();
    for (int i = start; i < end; ++i) {
        int primNum = primitiveInfo[i].primitiveNumber;
        orderedPrims.push_back(primitives[primNum]);
    }
    node->InitLeaf(firstPrimOffset, nPrimitives, bounds);
    return node;
```

```
    }
 }
break;
}
```

Listing B.6: PCA-SAH-Splitting für die PCA-BVH (Pharr u. a., 2016)

```
// ...
// Allocate _BucketInfo_ for SAH partition buckets
PBRT_CONSTEXPR int nBuckets = 12;
BucketInfo buckets[nBuckets];

Bounds3f classicCentroidBounds;
for (int i = start; i < end; ++i)
    classicCentroidBounds = Union(classicCentroidBounds, primitiveInfo
        [i].centroid);
int classicdim = classicCentroidBounds.MaximumExtent();


// Initialize _BucketInfo_ for SAH partition buckets
for (int i = start; i < end; ++i) {
    int b = nBuckets *
                classicCentroidBounds.Offset( primitiveInfo[i].
                    centroid)[classicdim];
    if (b == nBuckets) b = nBuckets - 1;
    CHECK_GE(b, 0);
    CHECK_LT(b, nBuckets);
    buckets[b].count++;
    buckets[b].bounds =
                Union(buckets[b].bounds, primitiveInfo[i].bounds);
}

// Compute costs for splitting after each bucket
Float classicCost[nBuckets - 1];
for (int i = 0; i < nBuckets - 1; ++i) {
    Bounds3f b0, b1;
    int count0 = 0, count1 = 0;
    for (int j = 0; j <= i; ++j) {
        b0 = Union(b0, buckets[j].bounds);
        count0 += buckets[j].count;
    }
    for (int j = i + 1; j < nBuckets; ++j) {
        b1 = Union(b1, buckets[j].bounds);
        count1 += buckets[j].count;
    }
    classicCost[i] = 1 +
      (count0 * b0.SurfaceArea() +
       count1 * b1.SurfaceArea() )
        / bounds.SurfaceArea();
}

// Find bucket to split at that minimizes SAH metric
Float minCost = classicCost[0];
int minCostSplitBucket = 0;
for (int i = 1; i < nBuckets - 1; ++i) {
    if (classicCost[i] < minCost) {
```

```
        minCost = classicCost[i];
        minCostSplitBucket = i;
    }
}

BucketInfo pcabuckets[nBuckets];

// Initialize _BucketInfo_ for SAH partition buckets
for (int i = start; i < end; ++i) {
    int b = nBuckets *
            centroidBounds.Offset(
                primitiveInfo[i].pcacentroid)[dim];
    if (b == nBuckets) b = nBuckets - 1;
    CHECK_GE(b, 0);
    CHECK_LT(b, nBuckets);
    pcabuckets[b].count++;
    pcabuckets[b].bounds =
        Union(pcabuckets[b].bounds, primitiveInfo[i].bounds);
}

// Compute costs for splitting after each bucket
Float pcacost[nBuckets - 1];
for (int i = 0; i < nBuckets - 1; ++i) {
    Bounds3f b0, b1;
    int count0 = 0, count1 = 0;
    for (int j = 0; j <= i; ++j) {
        b0 = Union(b0, pcabuckets[j].bounds);
        count0 += pcabuckets[j].count;
    }
    for (int j = i + 1; j < nBuckets; ++j) {
        b1 = Union(b1, pcabuckets[j].bounds);
        count1 += pcabuckets[j].count;
    }
    pcacost[i] = 1 +
        (count0 * b0.SurfaceArea() +
         count1 * b1.SurfaceArea())
             /  bounds.SurfaceArea() ;
}

// Find bucket to split at that minimizes SAH metric
Float minCostPCA = pcacost[0];
int minCostSplitBucketPCA = 0;
for (int i = 1; i < nBuckets - 1; ++i) {
    if (pcacost[i] < minCostPCA) {
        minCostPCA = pcacost[i];
        minCostSplitBucketPCA = i;
    }
}

// Either create leaf or split primitives at selected SAH
// bucket
Float leafCost = nPrimitives;
if (nPrimitives > maxPrimsInNode || minCost < leafCost || minCostPCA <
     leafCost) {
    if(minCostPCA<minCost){
        pcasahCalls++;
```

```
    } else {
        sahCalls++;
    }
    BVHPrimitiveInfo *pmid = std::partition(
        &primitiveInfo[start], &primitiveInfo[end - 1] + 1,
        [=](const BVHPrimitiveInfo &pi) {
            if(minCostPCA<minCost){
                int b = nBuckets *
                        centroidBounds.Offset(pi.pcacentroid)[dim];
                if (b == nBuckets) b = nBuckets - 1;
                CHECK_GE(b, 0);
                CHECK_LT(b, nBuckets);
                return b <= minCostSplitBucketPCA;
            } else{
                int b = nBuckets *
                        classicCentroidBounds.Offset(pi.centroid)[
                            classicdim];
                if (b == nBuckets) b = nBuckets - 1;
                CHECK_GE(b, 0);
                CHECK_LT(b, nBuckets);
                return b <= minCostSplitBucket;
            }
        });
    mid = pmid - &primitiveInfo[0];
}

// ...
```

Listing B.7: Kombinierte SAH-Funktion für die PCA-BVH

```

// Klassisch

cost = 1 + (count0 * b0.SurfaceArea() + count1 * b1.SurfaceArea())
                                      / bounds.SurfaceArea();

// Erweitert

Bounds3f overlap = pbrt::Intersect(b0, b1);

cost = 1 + (count0 * b0.SurfaceArea() +
            count1 * b1.SurfaceArea() +
           (count0 + count1) * overlap.SurfaceArea())
            / bounds.SurfaceArea();
```

Listing B.8: Klassisches SAH-Kostenmodell als Code und erweitertes SAH-Kostenmodell für die PCA-BVH als Code

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 29. Juli 2019

Matthias Sauer