

Hybride kd-bomen voor photonmapping en het versnellen van ray tracing

Matthias Moulin

Thesis voorge dragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen,
hoofdspecialisatie Mens-machine
communicatie

Promotor:
Prof. dr. ir. Philip Dutré

Academiejaar 2017 – 2018

Hybride kd-bomen voor photonmapping en het versnellen van ray tracing

Matthias Moulin

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen,
hoofdspecialisatie Mens-machine
communicatie

Promotor:
Prof. dr. ir. Philip Dutré

Assessoren:
Ir. Jeroen Baert
Dr. Aram Hovsepyan

Begeleider:
Ir. Niels Billen

© Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail info@cs.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Voorwoord

Een thesis maak je noch schrijf je alleen. Ik zou daarom in eerste instantie mijn ouders en familie willen bedanken voor hun steun en aanpassingen om het maken van deze thesis doorheen het volledige academiejaar zo veel mogelijk te vereenvoudigen. Daarnaast zou ik ook mijn begeleider Niels Billen in het bijzonder willen bedanken voor alle hulp, inzichten en feedback niet alleen in het onderwerp van deze thesis maar in alles wat erbij het maken van een thesis komt kijken. Ik zou hem ook willen bedanken voor de flexibele wekelijkse meetingregeling, voor de enorme bijdragen bij het schrijven van de voorlopige en finale gepubliceerde paper, voor het maken van de fantastische diagrammen van de paper en voor de verschillende reviewiteraties van deze thesistekst. Ik zou ook mijn promotor Philip Dutré willen bedanken voor de mogelijkheid om aan dit onderwerp en het genomen zijpad te werken, voor de volledige vrijheid om het onderwerp aan te pakken, voor de feedback tijdens thesisssessies en voor de feedback op de paper.

Mijn assessoren wil ik bedanken voor de tijd genomen om dit werk te lezen. Verder zou ik ook Roald Frederickx willen bedanken voor de feedback op de paper.

Daarnaast zou ik ook al de bedrijven, organisaties en individuen willen bedanken die hun software gratis ter beschikking stellen (voor studenten): Enthought Inc., GitHub Inc., Inkscape, Irfanview, Matt Pharr & Greg Humphreys, Microsoft Inc., TeamViewer, T_EX community, Texmaker.

Matthias Moulin

Inhoudsopgave

Voorwoord	i
Samenvatting	v
Lijst van figuren	vii
Lijst van tabellen	viii
Lijst van afkortingen en symbolen	ix
1 Inleiding	1
1.1 Fysisch gebaseerd renderen	1
1.2 Doelstellingen	2
1.3 Methodologie	3
1.4 Contributie	4
1.5 Overzicht	5
2 Kd-bomen	7
2.1 Acceleratiestructuren	7
2.2 Motivatie voor de kd-boom als hybride datastructuur	9
2.3 Heuristieken voor het bouwen van kd-bomen	11
2.4 Doorloopvolgorde voor schaduwstralen	13
3 Photonmapping	15
3.1 Renderingvergelijking	15
3.2 Lichttransportalgoritmen	16
3.3 Photonmapping lichttransportalgoritme	17
3.4 Photonmap	20
4 Hybride photonmappen	23
4.1 Hybrid Additive Photon Map	23
4.2 Hybrid Swapping Photon Map	26
4.3 Hybrid Connected Photon Map	27
5 Mathematisch framework	29
5.1 Notatie	30
5.2 Surface Area Heuristic	30
5.2.1 Assumpties	31
5.2.2 Kostfunctie	31
5.3 Ray Termination Surface Area Heuristic	33

5.3.1	Kostfunctie	34
5.4	Voxel Volume Heuristic	38
5.4.1	Assumpties	38
5.4.2	Kostfunctie	39
5.4.3	Voxel Volume Heuristic 1	42
5.5	Hybrid Heuristic	43
5.5.1	Assumpties	43
5.5.2	Kostfuncties	43
6	Praktische bouwalgoritmen	51
6.1	Praktisch bouwalgoritme RTSAH	51
6.1.1	All Points One Direction RTSAH	52
6.1.2	Average Projected Surface Area RTSAH	54
6.1.3	Validatie	54
6.1.4	Splitskandidaten	56
6.1.5	Bouwalgoritme	56
6.2	Praktisch bouwalgoritme HH	57
6.2.1	Componenten	57
6.2.2	Validatie	60
6.2.3	Splitskandidaten	60
6.3	Optimalisaties HH	61
6.3.1	Gedeelde componenten	61
6.3.2	Offsetpassing	62
6.3.3	Automatische maximale diepte	62
6.3.4	Aantal geforceerde splitsbeslissingen	62
6.3.5	Lookahead	64
6.3.6	Photonjumps	64
6.3.7	Vroege terminatie	65
6.3.8	Photons op het splitsingsvlak	66
7	Resultaten	67
7.1	Praktische aspecten	67
7.2	Resultaten RTSAH	68
7.2.1	Structuur van de kd-bomen	68
7.2.2	Performantie	69
7.3	Resultaten hybride photonmappen	71
7.3.1	Structuur van de HCPM kd-bomen	81
7.3.2	Structuur van de HSPM kd-bomen	82
7.3.3	Performantie	84
8	Besluit	103
8.1	Conclusies RTSAH	103
8.2	Conclusies hybride photonmappen	103
8.3	Toekomstig onderzoek	104
A	Poster	107
B	Artikel	113

INHOUDSOPGAVE

Bibliografie

125

Samenvatting

Acceleratiestructuren zoals kd-bomen waarin geometrische primitieven worden ondergebracht, hebben tot doel de kost voor het volgen van stralen te verminderen, wat cruciaal is voor de renderingperformantie. Photonmappen zoals kd-bomen waarin photons worden ondergebracht, hebben tot doel de kost voor het uitvoeren van k-nearest neighbor query's te verminderen, wat cruciaal is voor de performantie van het photonmapping lichttransportalgoritme ([Jen96], [Jen01]) en zijn varianten. We introduceren drie hybride kd-bomen waarin zowel de geometrische primitieven als photons worden ondergebracht met als doel de gecombineerde kost van ray tracing en k-nearest neighbor query's fundamenteel te verminderen:

- De Hybrid Additive Photon Map (HAPM) voegt de photons toe aan de kd-boom acceleratiestructuur en gebruikt hiervoor de opdeling van de geometrische primitieven om de photons op te delen.
- De Hybrid Swapping Photon Map (HSPM) veralgemeent het idee achter de HAPM door een hybride kd-boom te bouwen op basis van de geometrische primitieven én photons met behulp van de door ons geïntroduceerde Hybrid Heuristic (HH) bouwheuristiek die zowel de kost van de ray tracing als k-nearest neighbor query's in rekening neemt en aan elkaar koppelt. Een k-nearest neighbor query kan namelijk enkel starten vanuit een hitpunt van een straal met een geometrisch primitief.
- De Hybrid Connected Photon Map (HCPM) behoudt de aparte kd-boom acceleratiestructuur en kd-boom photonmap, elk bekomen met een optimale bouwheuristiek voor hun specifieke data en verbindt deze kd-bomen door middel van ropes.

Onze HH kapselt zowel de Ray Termination Surface Area Heuristic (RTSAH) voor het bouwen van kd-boom acceleratiestructuren als de Voxel Volume Heuristic (VVH) [WGS04] voor het bouwen van kd-boom photonmappen in. De de-facto standaard voor het bouwen van kd-boom acceleratiestructuren, de Surface Area Heuristic (SAH) [MB90], neemt geen straalterminatie in rekening, maar veronderstelt dat stralen nooit een geometrische primitief raken en is bijgevolg onbruikbaar in de HH.

De RTSAH [IH11] is een kostmetriek origineel gebruikt voor het bepalen van de doorloopvolgorde van de voxels voor occlusiestralen door straalterminatie in rekening te nemen. We passen deze RTSAH aan om kd-boom acceleratiestructuren te bouwen. Onze bouwprocedures, op basis van de orthogonale geprojecteerde en gemiddelde geprojecteerde oppervlakten van de geometrische primitieven op het splitsingsvlak, hebben dezelfde totale computationele complexiteit en beschouwen dezelfde eindige verzameling van splitsingsvlakken als de SAH. Door straalterminatie in rekening te nemen, moedigen we het splitsen van kindvoxels aan die niet of nauwelijks zichtbaar zijn ten opzichte van elkaar. Dit resulteert in fundamenteel verschillende en meer kwalitatieve kd-bomen in vergelijking met de SAH. We behalen reducties in intersectietesten tot 47% voor primaire stralen en tot 41% voor schaduwstralen (wanneer de voxels van voor naar achter doorlopen worden) in vergelijking met de SAH.

De HAPM en HSPM resulteren algemeen in aanzienlijke toenamen van de totale rendertijd ten opzichte van een niet-hybride photonmap. In eerste instantie werken de geometrische primitieven (niet-pundata) en photon (pundata) elkaar tegen wat de mate van verfijning van de kd-boom betreft. In tweede instantie zijn het volgen van stralen en het uitvoeren van k-nearest neighbor query's fundamenteel verschillende operaties. Dit heeft zowel gevolgen voor de manier waarop de hybride kd-boom doorlopen wordt als voor de informatie die nodig is om de kostfuncties te evalueren. Enkel de HCPM resulteert gemiddeld in reducties tot minder dan 1% in totale rendertijd.

Uit onze resultaten en argumentatie concluderen we dat geometrische primitieven en photons niet verzoend kunnen worden noch in kd-bomen noch in de veel gebruikte huidige acceleratiestructuren (reguliere roosters, BVHs, BSPs, octrees) opdat de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel verminderd wordt ten opzichte van twee aparte datastructuren.

Lijst van figuren

2.1	FCIs voor verschillende acceleratiestructuren	10
3.1	Renderingvergelijking	16
4.1	Hybride kd-bomen	24
4.2	Photondensiteitshistogram van de bladvoxels	25
5.1	Opsplitsing van een voxel met geometrische primitieven	32
5.2	Types van stralen beschouwd door de RTSAH	35
5.3	Verband tussen de probabiliteiten van de RTSAH en de kindvoxelvlakken	36
5.4	Opsplitsing van een voxel met photons	40
5.5	Splitskandidaten van de VVH	42
5.6	Opsplitsing van een voxel met geometrische primitieven en photons	46
6.1	Benaderingen van de visibiliteitsprobabiliteiten	53
6.2	Validatie van APOD en APSA benadering van de visibiliteitsprobabiliteiten	55
6.3	Splitskandidaten van de SAH, APOD en APSA RTSAH	56
6.4	Validatie van APOD en APSA benadering van de hitprobabiliteit	60
6.5	Renderingvergelijking	64
7.1	Testscenes voor de RTSAH	69
7.2	FCIs van intersectietesten voor primaire stralen	73
7.3	FCIs van intersectietesten voor primaire stralen, sibenik	74
7.4	FCIs van intersectietesten voor schaduwstralen	75
7.5	FCIs van intersectietesten voor schaduwstralen, sibenik	76
7.6	FCIs van doorlopen voxels voor primaire stralen	77
7.7	FCIs van doorlopen voxels voor primaire stralen, sibenik	78
7.8	FCIs van doorlopen voxels voor schaduwstralen	79
7.9	FCIs van doorlopen voxels voor schaduwstralen, sibenik	80
7.10	Testscenes voor de hybride photonmappen	81
7.11	Dieptehistogrammen van de ropes in HCPM, cornell	83
7.12	Dieptehistogrammen van de ropes in HCPM, room	84
7.13	Dieptehistogrammen van de ropes in HCPM, sibenik	86
7.14	Dieptehistogrammen van de ropes in HCPM, sponza	87

Lijst van tabellen

7.1	Specificaties van het testsysteem	68
7.2	Structuur van kd-bomen bekomen met SAH, APOD en APSA RTSAH .	70
7.3	Verschil in intersectietesten voor APOD en APSA RTSAH t.o.v. SAH .	71
7.4	Verschil in doorlopen voxels voor APOD en APSA RTSAH t.o.v. SAH .	72
7.5	Verschil in totale rendertijd voor APOD en APSA RTSAH t.o.v. SAH .	72
7.6	Structuur van de HSPM kd-bomen	85
7.7	Rendertijd hybride photonmappen, cornell	91
7.8	Rendertijd hybride photonmappen, room	92
7.9	Rendertijd hybride photonmappen, sibenik1	93
7.10	Rendertijd hybride photonmappen, sibenik2	94
7.11	Rendertijd hybride photonmappen, sponza1	95
7.12	Rendertijd hybride photonmappen, sponza2	96
7.13	Rendertijd hybride photonmappen, sponza3	97
7.14	Rendertijd hybride photonmappen, sponza4	98
7.15	Bouwtijd hybride photonmappen, cornell	99
7.16	Bouwtijd hybride photonmappen, room	100
7.17	Bouwtijd hybride photonmappen, sibenik	101
7.18	Bouwtijd hybride photonmappen, sponza	102

Lijst van afkortingen en symbolen

Afkortingen

APAD	All Points All Directions
APOD	All Points One Direction
APSA	Average Projected Surface Area
BH	Balanced Heuristic
BSDF	Bidirectional Scattering Distribution Function
BSP	Binary Space Partitioning
BVH	Bounding Volume Hierarchy
ext, int	exterior, interior
FCI	False Color Image
fL, fR	first Left, first Right
HAPM	Hybrid Additive Photon Map
HCPM	Hybrid Connected Photon Map
HH	Hybrid Heuristic
HRTSAH	Hybrid Ray Termination Surface Area Heuristic
HSPM	Hybrid Swapping Photon Map
jL, jR	just left, just Right
Kd-boom	k-dimensionale boom
kNN	k-Nearest Neighbor
PM	Photon Map
RT	Ray Tracing
RTSAH	Ray Termination Surface Area Heuristic
SAH	Surface Area Heuristic
spp	samples per pixel
VVH	Voxel Volume Heuristic

Symbolen

α	Lege-ruimte-bonus
$C_{\text{check}}^{\text{kNN}}$	Kost voor het controleren van één photon in een kNN query
$C_{\text{ext}}^{\text{kNN}}(V)$	Externe kost voor het bezoeken van een voxel in de hybride context
$\mathcal{C}_{\mathcal{H}}(S : V \rightarrow \{V_L, V_R\})$	Kostfunctie voor het splitsen van een voxel in S volgens heuristiek \mathcal{H}
$C_{\text{int}}(V)$	Interne kost voor het bezoeken van een voxel in de hybride context
$C_{\text{isect}}^{\text{RT}}$	Kost om één geometrisch primitief op intersectie te testen
$C_{\text{isect}}^{\text{RT}}(G_{V,i})$	Intersectiekostfunctie
$C_{\text{trav}}^{\text{RT}}$	Kost voor het doorlopen van een intermediaire voxel
$C_{\text{visit}}(V)$	Kost voor het bezoeken van een voxel in de hybride context
$C_{\text{visit}}^{\text{kNN}}(V)$	Kost voor het bezoeken van een voxel in kNN context
$C_{\text{visit}}^{\text{RT}}(V)$	Kost voor het bezoeken van een voxel in RT context
G_V	Verzameling van geometrische primitieven in voxel V
\mathcal{H}^2	Verzameling van richtingen over de eenheidsbol
k	Maximaal aantal photons voor een kNN query
$L_e(p, \vec{\omega})$	Zelf-uitgestraalde radiantiefunctie
$L_i(p, \vec{\omega}), L_o(p, \vec{\omega})$	Invallende/uitgestraalde radiantiefunctie
\mathcal{N}_V	Het naburige gebied van de voxel V van waaruit onvoltooide kNN query's deze voxel kunnen bezoeken.
$p_{\text{ext}}^{\text{kNN}}$	Kans om de oudervoxel te moeten controleren in een onvoltooide kNN query
$p_{\text{extL}}^{\text{kNN}}, p_{\text{extR}}^{\text{kNN}}$	Kans om de linkse/rechtse kindvoxel te moeten controleren in een onvoltooide kNN query
$p_{\text{fL}}^{\text{RT}}, p_{\text{fR}}^{\text{RT}}$	Kans om de linkse/rechtse kindvoxel eerst te doorkruisen, gegeven dat beide kindvoxels doorkruist worden
$p_{\text{hit}}^{\text{RT}}$	Kans op een hit in de oudervoxel, gegeven dat de oudervoxel doorkruist wordt
$p_{\text{hitL}}^{\text{RT}}, p_{\text{hitR}}^{\text{RT}}$	Kans op een hit in de linkse/rechtse kindvoxel, gegeven dat de oudervoxel doorkruist wordt
$p_{\text{jL}}^{\text{RT}}, p_{\text{jR}}^{\text{RT}}$	Kans om enkel de linkse/rechtse kindvoxel te doorkruisen, gegeven dat de oudervoxel doorkruist wordt
$p_L^{\text{kNN}}, p_R^{\text{kNN}}$	Kans om de linkse/rechtse kindvoxel te moeten controleren, gegeven de oudervoxel te moeten controleren
$p_L^{\text{RT}}, p_R^{\text{RT}}$	Kans om de linkse/rechtse kindvoxel te doorkruisen, gegeven dat de oudervoxel doorkruist wordt
$p_{\text{LR}}^{\text{RT}}$	Kans om beide kindvoxels te doorkruisen, gegeven dat de oudervoxel doorkruist wordt

P_V	Verzameling van photons horende bij voxel V
R_{\max}	Maximale kNN querystraal
S	Positie van het splitsingsvlak
SA_{G_V}	Totale oppervlakte van alle geometrisch primitieven in de voxel V
SA_V	Oppervlakte van de voxel V
V	Voxel
V_L, V_R	Linkse/rechtse kindvoxel van de voxel V
$\mathcal{V}_{LR}^{\text{RT}}, \mathcal{V}_{RL}^{\text{RT}}$	Visibiliteitsprobabiliteiten $V_L \rightarrow V_R / V_R \rightarrow V_L$
$\lambda(V_L, V_R)$	Lege-ruimte-bonusfunctie
$\rho(p, \vec{\omega}, \vec{\omega}')$	BSDF (Bidirectional Scattering Distribution Function)
$\vec{\omega}$	Genormaliseerde vector over de eenheidsbol
\emptyset	Lege verzameling

Hoofdstuk 1

Inleiding

“The decoupling of the photon map(s) from the geometry is a significant advantage that not only simplifies the representation but also makes it possible to use the structure to represent lighting in very complex models.”

— Henrik Wann Jensen, *Realistic image synthesis using photon maps*

In sectie 1.1 geven we een overzicht van wat fysisch gebaseerd renderen is, welke technieken de efficiëntie van dit fysisch gebaseerd renderen kunnen verbeteren en waar het onderwerp van deze thesis zich situeert. De doelstellingen van deze thesis worden besproken in sectie 1.2. Onze methodologie om deze doelstellingen aan te pakken wordt besproken in sectie 1.3. In sectie 1.4 geven we een overzicht van onze contributies. Sectie 1.5 geeft een overzicht van de structuur van deze thesis.

1.1 Fysisch gebaseerd renderen

Fysisch gebaseerd renderen is één van de grote en fundamentele onderwerpen in het veld van de computer graphics. Het is het proces van het converteren van een beschrijving van een virtuele 3D wereld naar fotorealistische 2D afbeeldingen. In het ideale geval wekken deze afbeeldingen dezelfde respons voor een menselijke waarnemer als wanneer deze waarnemer naar de echte scène zou kijken, indien deze scène in realiteit zou bestaan. Het modelleren van de 3D wereld houdt het beschrijven van de geometrie, het voorkomen van de vormen en objecten, de eigenschappen en positionering van de camera en het voorkomen van lichtbronnen in.

De meest geavanceerde fysisch gebaseerde renderingalgoritmen zijn gebaseerd op het path tracing paradigma: paden worden gevuld doorheen de scène om het lichttransport te simuleren tussen de lichtbronnen en de virtuele camera. De paden worden gegenereerd door gebruik te maken van een random bemonsteringsschema dat de onderliggende lichttransportintegralen numeriek evalueert met Monte Carlo integratietechnieken. Om de efficiëntie van de fysisch gebaseerde renderingalgoritmen te verbeteren, zijn er drie grote aanpakken, die niet exclusief zijn ten opzichte van elkaar en meestal gecombineerd kunnen worden:

1. INLEIDING

- De eerste aanpak tracht de geometrische berekeningen te versnellen. Stralen moeten geïntersecteerd worden met de geometrische vormen in de scène om te bepalen welke oppervlakken zichtbaar zijn en om occlusie op te lossen. Het aantal intersectietesten kan binnen bepaalde theoretische grenzen gehouden worden door gebruik te maken van geschikte acceleratiestructuren. Het (herop)bouwen en/of partieel updaten van deze acceleratiestructuren wordt meestal uitgevoerd tijdens een pre-rendering fase.
- De tweede aanpak tracht de lichttransportberekeningen efficiënter te maken in het capteren van de verschillende belichtingscomponenten. Dit wordt verwezenlijkt door middel van uitgebreide bemonsteringsprocedures tijdens de Monte Carlo evaluatie van de lichttransportvergelijkingen. Deze optimalisaties vinden plaats tijdens de rendering fase.
- De finale aanpak past adaptieve bemonstering- en filtertechnieken toe die gebaseerd zijn op een mathematische analyse van het onderliggende signaal dat de lichtdistributie in de scène beschrijft. Deze methodes worden gebruikt tijdens de rendering fase en/of tijdens een post-rendering fase.

Het onderwerp van deze thesis heeft betrekking op de eerste twee aanpakken. We proberen meer bepaald de acceleratiedatastructuur waarin de geometrische primitieven zijn ondergebracht, te combineren met de photonmap, een datastructuur gebruikt door het photonmapping lichttransportalgoritme ([Jen96], [Jen01]).

1.2 Doelstellingen

Photonmapping is een algoritme, geïntroduceerd door Jensen ([Jen96], [Jen01]), voor het berekenen van de verschillende lichttransportpaden waarbij de invallende radiantie bemonsterd wordt op een aantal locaties in de scène en opgeslagen in zogenoemde photons (puntdata). Deze photons worden ondergebracht in een spatiale datastructuur, de photonmap. Deze photonmap wordt gebruikt om k-nearest neighbor query's uit te voeren om de dichtstbijzijnde photons ten opzichte van een hitpunt van een straal met een geometrisch primitief te vinden. Deze dichtstbijzijnde photons worden vervolgens gebruikt om de invallende radiantie in dit hitpunt te benaderen.

Het is deze photonmap die we proberen te verzoenen met een acceleratiestructuur waarin geometrische primitieven zijn ondergebracht. We onderzoeken meer bepaald of het mogelijk is de geometrische primitieven en photons in eenzelfde datastructuur onder te brengen met als doel de gecombineerde kost van ray tracing en k-nearest neighbor query's fundamenteel te reduceren. De bijdrage van de ray tracing en k-nearest neighbor query's staan niet los van elkaar. K-nearest neighbor query's kunnen enkel starten vanuit een hitpunt van een straal met een geometrisch primitief.

Een mogelijke koppeling tussen een acceleratiestructuur en een photonmap is echter nog niet onderzocht. Beide datastructuren worden onafhankelijk van elkaar

gebouwd en gebruikt. Daarnaast stelt Henrik Wann Jensen [Jen01] zelf (zie inleidend citaat) dat het loskoppelen van de geometrie van de photonmap significante voordelen heeft waaronder een vereenvoudigende representatie en de mogelijkheid om de belichting in complexe scenes voor te stellen. Hierbij verwijst Jensen wat de koppeling van de geometrie en de photons betreft, impliciet naar belichtingsmappen [AK87] waarbij een textuurmap met belichtingswaarden per geometrisch primitief gebruikt wordt. Belichtingsmappen zijn niet de enige mogelijke koppeling. Geometrische primitieven en photons kunnen eveneens gekoppeld worden door deze onder te brengen in eenzelfde datastructuur.

1.3 Methodologie

We zijn van mening (zoals we later zullen argumenteren) dat enkel kd-bomen (van de veelgebruikte acceleratiestructuren) in aanmerking komen om de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel te doen dalen ten opzichte van twee aparte datastructuren die optimaal gebouwd zijn voor hun specifieke data. We introduceren drie verschillende hybride kd-bomen om deze gecombineerde kost van ray tracing en k-nearest neighbor query's te verminderen:

- De Hybrid Additive Photon Map (HAPM) voegt de photons toe aan de kd-boom acceleratiestructuur en gebruikt hiervoor de opdeling van de geometrische primitieven om de photons op te delen.
- De Hybrid Swapping Photon Map (HSPM) veralgemeent het idee achter de HAPM door een hybride kd-boom te bouwen op basis van de geometrische primitieven én photons met behulp van de door ons geïntroduceerde Hybrid Heuristic (HH) bouwheuristiek die zowel de kost van de ray tracing als k-nearest neighbor query's in rekening neemt.
- De Hybrid Connected Photon Map (HCPM) behoudt de aparte kd-boom acceleratiestructuur en kd-boom photonmap, elk bekomen met een optimale heuristiek voor hun specifieke data en verbindt deze kd-bomen door middel van ropes.

Onze HH kapselt zowel een bouwheuristiek voor kd-boom acceleratiestructuren als voor kd-boom photonmappen in en koppelt de kost van de ray tracing en k-nearest neighbor query's. Een k-nearest neighbor query kan namelijk enkel starten vanuit een hitpunt van een straal met een geometrisch primitief.

De de-facto standaard voor het bouwen van kd-boom acceleratiestructuren is de Surface Area Heuristic (SAH) [MB90], die aan elke kandidaatvoxel een kost toekent gelijk aan het product van de kost om deze voxel te doorlopen en de kans om deze voxel effectief te moeten doorlopen. Een gulzige top-down constructie minimaliseert deze kost. Om de probabiliteiten voor elke kandidaatvoxel te berekenen, veronderstelt

1. INLEIDING

de SAH dat stralen oneindig lang zijn en de oorsprong en richting van deze stralen uniform verdeeld zijn buiten de omhullende box van de scene. Door te veronderstellen dat stralen oneindig lang zijn en dus nooit een geometrisch primitief raken, is er geen koppeling mogelijk tussen stralen en het voorkomen van k-nearest neighbor query's (vanuit een hitpunt).

We proberen deze assumptie te herzien door de Ray Termination Surface Area Heuristic (RTSAH) [IH11] te herbekijken. De RTSAH is een kostmetriek origineel gebruikt voor het bepalen van de doorloopvolgorde van de voxels voor occlusiestralen door straalterminatie in rekening te nemen. We passen deze RTSAH aan om kijkpuntonafhankelijke kd-boom acceleratiestructuren te bouwen.

Voor het bouwen van kd-boom photonmappen gebruiken we de Voxel Volume Heuristic (VVH) [WGS04].

1.4 Contributie

De contributies van deze thesis met betrekking tot acceleratiestructuren voor geometrische primitieven zijn de volgende:

- We stellen een nieuwe manier voor om de RTSAH te gebruiken om kd-bomen te bouwen.
- We stellen twee snelle bouwalgoritmen voor om kwalitatieve kd-bomen te bouwen op basis van de orthogonale geprojecteerde en gemiddelde geprojecteerde oppervlakten van de geometrische primitieven op het splitsingsvlak. Deze bouwalgoritmen hebben dezelfde totale computationale complexiteit en moeten dezelfde eindige verzameling van splitsingsvlakken beschouwen als de SAH.
- We behalen reducties in intersectietesten tot 47% voor primaire stralen en tot 41% voor schaduwstralen (wanneer de voxels van voor naar achter doorlopen worden) in vergelijking met de SAH.

De contributies van deze thesis met betrekking tot hybride datastructuren voor geometrische primitieven en photons zijn de volgende:

- We stellen het eerste onderzoek naar hybride datastructuren voor waarin zowel geometrische primitieven als photons worden opgeslagen om de gecombineerde kost van de ray tracing en k-nearest neighbor query's (fundamenteel) te doen dalen.
- We stellen drie hybride kd-bomen (HAPM, HSPM, HCPM) voor waarin zowel geometrische primitieven als photons worden opgeslagen.

- We stellen een nieuwe gulzige bouwheuristiek (HH) voor die zowel de ray tracing kost als k-nearest neighbor querykost in rekening neemt.
- We stellen een nieuw hybride kostmodel voor om de kost van het volgen van stralen te koppelen aan het voorkomen van (interne) k-nearest neighbor query's.
- We behalen voor de HCPM gemiddeld reducties tot minder dan 1% voor de totale rendertijd.

1.5 Overzicht

In hoofdstuk 2 geven we een overzicht van veel gebruikte acceleratiestructuren. We motiveren waarom enkel kd-bomen in aanmerking komen om de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel te doen dalen ten opzichte van twee aparte datastructuren die optimaal gebouwd zijn voor hun specifieke data. Daarnaast geven we een overzicht van hoe kd-boom acceleratiestructuren gebouwd en doorlopen worden. In hoofdstuk 3 leggen we de renderingvergelijking uit en geven een overzicht van veelgebruikte lichttransportalgoritmen. Vervolgens leggen we het photonmapping lichttransportalgoritme uit en geven we een overzicht van veelgebruikte photonmap datastructuren. In hoofdstuk 4 introduceren we onze nieuwe hybride kd-bomen: HAPM, HSPM en HCPM. In hoofdstuk 5 introduceren we onze aanpassing van de originele RTSAH om deze vervolgens te gebruiken in onze nieuwe HH bouwheuristiek die zowel de ray tracing kost als k-nearest neighbor querykost in rekening neemt. In hoofdstuk 6 introduceren we praktische bouwalgoritmen, benaderingen en optimalisaties voor de RTSAH en HH. In hoofdstuk 7 testen we de bekomen RTSAH bouwprocedures en onze hybride photonmappen.

Hoofdstuk 2

Kd-bomen

Kwalitatieve acceleratiestructuren voor ray tracing zoals kd-bomen hebben als doel de kost van het volgen van een straal te verminderen. Zonder deze structuren zou elke straal op intersectie getest moeten worden met alle geometrische primitieven in de scene. In sectie 2.1 geven we een overzicht van enkele veelgebruikte acceleratiestructuren.

In deze thesis onderzoeken we of we naast geometrische primitieven (niet-puntdaten) ook photons (puntdaten) kunnen onderbrengen in deze acceleratiestructuren. Niet elk van deze acceleratiestructuren is geschikt om photons op te slaan met als doel de kost van de k-nearest neighbor query's te verminderen. Daarnaast zijn de meeste van deze acceleratiestructuren niet geschikt om zowel geometrische primitieven als photons op te slaan met als doel de gecombineerde kost van de ray tracing en k-nearest neighbor query's te verminderen. We zijn van mening dat enkel kd-bomen in aanmerking komen om deze gecombineerde kost fundamenteel te doen dalen ten opzichte van twee aparte datastructuren die optimaal gebouwd zijn voor hun specifieke data. Deze keuze voor kd-bomen motiveren we in sectie 2.2.

De de-facto standaard voor het construeren van kd-bomen is de Surface Area Heuristic (SAH) [MB90]. In sectie 2.3 gaan we dieper in op deze SAH en alternatieve heuristieken die gebruikt worden voor het bouwen van kd-bomen. Ondanks de belangrijkheid van een goede bouwheuristiek, zijn de assumenties onderliggende aan de SAH onrealistisch. In sectie 2.4 komen heuristieken aan bod voor het bepalen van de doorloopvolgorde van de voxels in een kd-boom voor oclusiestralen.

2.1 Acceleratiestructuren

Stralen worden geïntersecteerd met de geometrische primitieven in de scene om te bepalen welke oppervlakken zichtbaar zijn en om oclusie te bepalen. Het aantal intersectietesten kan binnen theoretische grenzen gehouden worden door het gebruik van geschikte acceleratiestructuren. Voor een recent overzicht verwijzen we de

2. KD-BOMEN

geïnteresseerde lezer naar [WMG*09]. Deze acceleratiestructuren zijn op te delen in adaptieve en niet-adaptieve acceleratiestructuren.

Een *regulier rooster* ([FTI86], [LD08b]) deelt de spatiale ruimte op in gelijke roostercellen. Voor een straal moet eerst bepaald worden welke roostercel ze eerst doorkruist (externe straal) of van waaruit ze vertrekt (interne straal). Vervolgens wordt deze straal gevuld doorheen de roostercellen tot deze een geometrisch primitief raakt of het rooster verlaat. De vrijheidsgraad bij het bouwen van een regulier rooster is de celgrootte of alternatief het totaal aantal cellen. Ize et al. [ISP07] introduceerden een strategie voor het bepalen van de roosterresolutie en voor het beslissen om bepaalde roostercellen te verfijnen met een subrooster.

Veelgebruikte adaptieve acceleratiestructuren zijn *Bounding Volume Hierarchies* (BVH) [Whi80] en *Binary Space Partitioning* (BSP) bomen. BVHs en BSP bomen worden meestal geconstrueerd door een gulzige verdeel-en-heers strategie voor het partitioneren van respectievelijk de geometrische primitieven en driedimensionale ruimte. *kd-bomen* [Kap85] ook wel rectilineaire BSP bomen genoemd [dBCvKO08], zijn BSP bomen waarbij er steeds in één van de drie primaire richtingen gesplitst wordt. Door slechts drie vooraf bepaalde richtingen in rekening te nemen, is het eenvoudiger kd-bomen dan de meer algemene BSP bomen te construeren. Ize et al. [IWP08] introduceerden een manier om BSP bomen te bouwen met een kleinere kost per straal maar hogere constructietijd dan kd-bomen.

Naast de keuze van de bouwheuristiek zijn er verschillende strategieën mogelijk voor het doorlopen van deze binaire bomen met een straal (voor BVHs [HDW*11], voor BSP bomen en kd-bomen [HH11]). Een straal kan beide kindvoxels op een intermediaire knoop bezoeken. Slechts één van beide kindvoxels kunnen we onmiddellijk bezoeken en kan op zijn beurt resulteren in meerdere te bezoeken voxels. De nog te bezoeken voxels kunnen bijgehouden worden op een stack. Voor BSP bomen is het echter mogelijk deze stack te elimineren door gebruik te maken van *ropes* ([HHS05], [PGSS07]). Elke bladvoxel heeft ten hoogste zes ropes. De ropes zijn de aangrenzende voxels die een volledige zijvlak van deze bladvoxel zo nauw mogelijk omspannen. Eenmaal de bladvoxel gekend is van waaruit een straal vertrekt, kan de straal gevuld worden via de ropes van deze bladvoxel. Dit is analoog aan het doorlopen van een regulier rooster en vereist geen stack. Een rope kan zowel horen bij een bladknoop als intermediaire knoop. Bij deze laatste moeten we in een driedimensionale subboom afdalen om de volgende bladknoop te kennen die een straal bezoekt terwijl het zijvlak slechts tweedimensionaal is. *Rope bomen* [HHS05] zijn analoog aan ropes, maar in het geval van een intermediaire knoop bouwen we een tweedimensionale kd-boom op het zijvlak van de kindvoxel om de doorloop te versnellen.

Om een gedetailleerd overzicht van het aantal intersectesten van stralen met geometrische primitieven per pixel te bekomen, kunnen *false color afbeeldingen*

gebruikt worden. In Figuur 2.1 vergelijken we voor een eenvoudige scène bestaande uit een konijn en een vloerplaat het aantal intersectietesten voor een kd-boom (SAH bouwheuristiek), een BVH (SAH bouwheuristiek) en een regulier rooster. Hierbij beschouwen we zowel de intersectietesten met geometrische primitieven als met omhullende boxen [WBKS03]. Deze false color afbeeldingen geven een beeld van de opbouw van deze acceleratiestructuren. Bij een regulier rooster nemen we de roosterstructuur waar. Bij een BVH nemen we de omhullende boxen waaruit de boom bestaat waar. Bij de kd-boom nemen we het grootste aantal intersectietesten net naast de randen van het konijn waar. In deze gebieden moeten we helemaal afdalen in de boom en vervolgens alle geometrische primitieven in de bladknopen op intersectie testen. De false color afbeeldingen geven aan dat een regulier rooster minder geschikt is als acceleratiestructuur indien de distributie van geometrische primitieven in de scène niet uniform is. Door de concentratieverschillen varieert de bezetting van de roostercellen sterk. Adaptieve structuren zoals kd-bomen en BVHs zijn hiervoor beter geschikt.

Kd-bomen en BVHs behoren tot de huidige state-of-the-art in acceleratiestructuren. Een hybride combinatie van deze bomen, genaamd spatiale kd-bomen (skd-bomen) ([OMSD87], [WK06]), kan bekomen worden door voor elke kindvoxel een apart splitsingsvlak te gebruiken. Hierbij is het mogelijk om zowel objecten op te splitsen (BVH) als een efficiëntie doorloopstrategie (kd-boom) te bekomen.

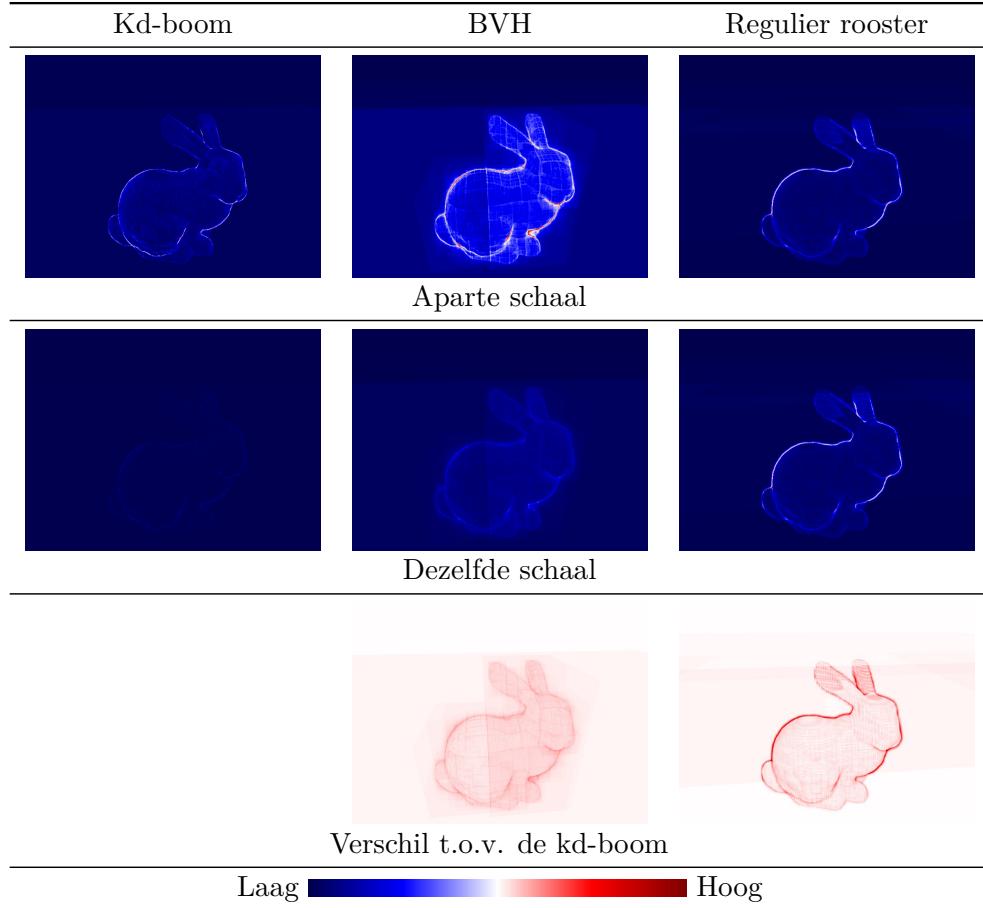
Verder worden ook octrees [Gla84] als acceleratiestructuren gebruikt. Hierbij wordt de driedimensionale ruimte steeds door midden gesplitst in elk van de drie primaire richtingen. Dit resulteert in acht gelijke kindvoxels.

Naast deze ‘basis’ acceleratiestructuren zijn er een aantal innovatieve aanpakken voorgesteld. Arvo en Kirk [AK87] introduceerden een vijfdimensionale datastructuur waarbij zowel de driedimensionale spatiale ruimte als tweedimensionale richtingsruimte van de stralen wordt opgedeeld. Lagae en Dutré [LD08a] introduceerden een nieuwe acceleratiestructuur voor scenes bestaande uit driehoeken waarbij een constrained tetrahedralisatie berekend wordt. Hierbij behoort elke driehoek tot de tetrahedralisatie. Stralen worden nu gevolgd door de tetraëders in plaats van roostercellen zoals bij een regulier rooster.

2.2 Motivatie voor de kd-boom als hybride datastructuur

In deze sectie motiveren we de keuze voor kd-bomen alsook waarom andere veel gebruikte huidige acceleratiestructuren minder tot niet geschikt zijn om zowel photons (puntdata) als geometrische primitieven (niet-puntdata) op te slaan. Merk op dat we hier focussen op bestaande acceleratiestructuren. Het zou eveneens mogelijk zijn fundamenteel nieuwe datastructuren of combinaties van deze datastructuren te ontwerpen om zowel photons als geometrische primitieven op te slaan. Deze laatste

2. KD-BOMEN



FIGUUR 2.1: False color afbeeldingen van het aantal **intersectietesten voor primaire stralen** voor een kd-boom, BVH en regulier rooster en het verschil (blauw (rood) komt overeen met winst (verlies)) ten opzichte van de kd-boom. De gebruikte schaal is lineair. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 samples per pixel (spp).

structuren behoren niet tot het doel van deze thesis.

Regulier rooster. Ten eerste is een regulier rooster geen hiërarchische datastructuur. Hierdoor zal het doorlopen van een hybride rooster voor stralen en voor k-nearest neighbor query's nagenoeg onafhankelijk en op dezelfde manier verlopen als wanneer er twee aparte roosters gebruikt worden. Ten tweede moet het hybride rooster zowel puntdata als niet-puntdata opslaan. Wanneer een fijne roosterresolutie wordt gekozen, zullen grote geometrische primitieven een aanzienlijk aantal roostercellen overlappen met een tragere acceleratiestructuur als gevolg. Daarnaast zal het aantal te doorlopen roostercellen per te volgen straal ook toenemen. Wanneer een ruwe roosterresolutie wordt gekozen, zullen grote hoeveelheden photons gegroepeerd worden (zonder verdere verfijning) per roostercel met tragere k-nearest neighbor

query's tot gevolg. Voor een regulier rooster is het daarom beter om twee aparte roosters elk met een aan de data aangepaste roosterresolutie te gebruiken.

BVH. Een BVH wordt niet gebruikt om photons op te slaan. Doordat een BVH een niet-spatiale datastructuur is, bestaat de noodzaak om photons meerdere malen op te slaan. Hierdoor zal voor de correcte uitvoering van een k-nearest neighbor query het meermaals in rekening nemen van dezelfde photons explicet vermeden moeten worden. Aangezien we vooral de k-nearest neighbor query's willen versnellen lijkt een BVH datastructuur daarom niet geschikt.

kd-boom. De kd-boom is een hiërarchische spatiale datastructuur. Dit is ideaal om puntdata op te slaan in voxels met een aangepaste grootte. Door zowel de geometrische primitieven als photons op te delen in eenzelfde hybride kd-boom, verwachten we dat k-nearest neighbor query's voornamelijk dichtstbijzijnde photons moeten controleren en slechts een klein deel overige photons (overhead). Bij een fijne tesselatie van de geometrie, verwachten we dat de opdeling van de geometrische primitieven de opdeling van de photons meestuurt. Verder maakt een hybride kd-boom zowel een top-down (vanuit de wortelknoop) als bottom-up (vanuit de knoop met het hitpunt) doorloopstrategie van de voxels voor k-nearest neighbor query's mogelijk. Eenmaal de voxel met het hitpunt gevonden is via het top-down doorlopen van de kd-boom voor een straal, kan de k-nearest neighbor query starten vanuit deze voxel met een bottom-up doorloopstrategie. Een deel van de k-nearest neighbor query's kunnen nu vroegtijdig beëindigd worden en voor het bereiken van de wortelknoop de kd-boom verlaten. Hierdoor neemt de gemiddelde doorlooptijd in de kd-boom af ten opzichte van de doorlooptijd bij een top-down strategie voor dezelfde kd-boom. De hybride kd-boom zal echter ook algemeen dieper zijn doordat naast de photons nu ook geometrische primitieven moeten worden opgeslagen.

Octree. Voor een octree kan een gelijkaardige redenering worden toegepast als voor een kd-boom. Het adaptieve karakter van een octree is echter minder agressief dan bij een kd-boom. Dit betekent dat we gemiddeld meer niveaus en dus een diepere boom nodig hebben om eenzelfde mate van verfijning van de voxels te bekomen in vergelijking met een kd-boom. Verder moet een voxel telkens opgedeeld worden in acht kindvoxels wat leidt tot een hoog geheugengebruik. Deze problematiek wordt bovendien erger doordat we puntdata (photons) willen opslagen.

2.3 Heuristieken voor het bouwen van kd-bomen

Heuristieken worden gebruikt om de driedimensionale ruimte op te delen in een kd-boom. De best gekende en meest gebruikte heuristiek voor het genereren van kwalitatieve kd-bomen is de SAH. De SAH is oorspronkelijk geïntroduceerd door Goldsmith en Salmon [GS87] voor het bouwen van BVHs en is aangepast door MacDonald en Booth [MB90] voor het bouwen van kd-bomen.

2. KD-BOMEN

De SAH kent een kost toe aan elke kandidaatvoxel die gelijk is aan het product van de kost en de probabilitet om deze voxel te doorlopen met een straal. Om deze probabiliteten te berekenen voor elke kandidaatvoxel, veronderstelt de SAH:

1. de oorsprong van de stralen is uniform verdeeld in de ruimte buiten de omhullende box van de scene;
2. de richting van de stralen is uniform verdeeld over de eenheidsbol;
3. de stralen zijn oneindig lang.

Verschillende verbeteringen zijn voorgesteld voor de constructie van de SAH. Havran et al. [HB02] introduceerden een efficiënt automatisch terminatiecriterium en clipping algoritme. Wald en Havran [WH06] moedigen het afsplitsen van grote lege kindvoxels aan door de verwachte kost van deze splitsingsvlakken te reduceren met een constante factor. Deze verbetering resulteert gewoonlijk in meer kwalitatieve kd-bomen. Hunt [Hum08] voegde mail-boxing toe, een ray tracing optimalisatie die redundante intersectietesten (door geometrische primitieven die meerdere voxels overlappen) verwijderd. Wald en Havran [WH06] introduceerden een robuust $\mathcal{O}(|G| \log |G|)$ bouwgoritme voor kd-bomen met $|G|$ het totaal aantal geometrische primitieven. Voor praktische scenes resulteert dit in een twee tot driemaal snellere constructie in vergelijking met de eenvoudigere $\mathcal{O}(|G| \log^2 |G|)$ variant.

Daarnaast zijn veel fundamenteel verschillende heuristieken geïntroduceerd om een meer accurate kostfunctie te bekomen aangezien de veronderstellingen van de SAH niet in de praktijk gelden. Fabianowski et al. [FFD09] stelden de Scene Interior Ray Origins Heuristic voor die veronderstelt dat de oorsprong van stralen uniform verdeeld is binnen de omhullende box van de scene. Havran en Bittner stelden bouwmethoden voor die rekening houden met de actuele distributie van stralen voor een vaste oorsprong of richting [HB99] en door bemonstering [BH09]. Choi et al. [CCI12] stelden de Voxel Visibility Heuristic voor die de niet-uniforme distributie van de stralen beschouwd door de oclusie van stralen door geometrische primitieven in rekening te nemen. Reinhard et al. ([RKJ96], [RKC98]) en Havran [Hav00] beschouwden straalterminatie binnen een voxel door een blocking factor toe te voegen.

Reinhard et al. ([RKJ96], [RKC98]) introduceerde een kostmodel (zonder bijhorende bouwprocedure) voor het volgen van een straal waarbij per kindvoxel een *blocking factor* wordt berekend door de omhullende boxen van de geometrische primitieven te projecteren op de vlakken van de voxel en rekening te houden met overlappingen. Het berekenen van de overlapping tussen elk paar van geometrische primitieven in een kindvoxel resulteert in een combinatorische explosie en is daarom niet bruikbaar als praktisch bouwgoritme voor kd-bomen. Al deze blocking factoren worden vervolgens gebruikt om één gemiddelde blocking factor te berekenen. Op

basis van deze gemiddelde blocking factor en de gemiddelde diepte van de kd-boom wordt de doorloopkost van een straal berekend. Havran [Hav00] gebruikt dit idee van een blocking factor in zijn algemene kostmodel om kd-bomen te bouwen. Dit model maakt een onderscheid tussen stralen die enkel de linkse, rechtse, eerst de linkse en dan de rechtse of eerst de rechtse en dan de linkse kindvoxel doorkruisen. De blocking factor tussen twee kindvoxels wordt bemonsterd. Alvorens de monsters te volgen door de scène moet eerst een tijdelijke kd-boom gebouwd worden. De bemonstering en bouw van de tijdelijke kd-boom resulteren in niet-aanvaardbare bouwtijden.

Al deze bouwheuristieken gebruiken een gulzige verdeel-en-heers strategie. Wanneer deze lokale heuristieken uitgebreid worden naar een globale heuristiek die de positionering van de splitsingsvlakken op elk niveau van de boom simultaan in rekening neemt, wordt het probleem om de optimale kd-boom te bouwen NP-hard. Voor dynamische scenes worden deze kd-bomen partieel aangepast en/of heropgebouwd omdat het volledig opnieuw bouwen van de kd-boom per frame duur is ($\mathcal{O}(|G| \log |G|)$). Voor een overzicht van deze technieken verwijzen we naar [WMG*09].

2.4 Doorloopvolgorde voor schaduwstralen

Schaduwstralen moeten niet gevolgd worden van voor naar achter om de eerste intersectie te vinden, omdat hun enige functie bestaat uit het rapporteren van occlusie. Ize en Hansen [IH11] introduceerden de RTSAH voor het bepalen van de doorloopvolgorde voor occlusiestralen in BVHs en BSPs. Ze kennen aan elke voxel de kans dat een straal eindigt tijdens de doorloop door deze voxel toe. Om een schaduwstraal te volgen, worden voxels met een hogere terminatieprobabiliteit eerst doorlopen. Voor de eenvoud, worden niet-lege bladvoxels volledig opaak verondersteld. De visibiliteit van de intermediaire knopen, wordt recursief bepaald op basis van hun kindknopen. Deze RTSAH is een reactie op het algemene kostmodel van Havran [Hav00] waarbij de kans om slechts één kindvoxel te doorkruisen foutief wordt berekend zoals Ize en Hansen [IH11] aantoonden. Nah en Manocha [NM14] introduceerden de Surface Area Traversal Order metriek die een hogere doorloopprioriteit geeft aan de kindvoxel met de grootste oppervlakte.

Hoofdstuk 3

Photonmapping

De renderingvergelijking definieert de lichtenergie-evenwichtstoestand in de scene. We introduceren deze vergelijking in sectie 3.1. In sectie 3.2 geven we een overzicht van de lichttransportalgoritmen zoals photonmapping gebaseerd op Monte Carlo integratietechnieken om het lichttransport numeriek te berekenen. In sectie 3.3 gaan we dieper in op het photonmapping lichttransportalgoritme. Photonmapping maakt gebruik van zogenoemde photons om de belichting in de scène te bemonsteren. Deze photons worden opgeslagen in een spatiale datastructuur, de photonmap. In sectie 3.4 geven we een overzicht van verschillende mogelijke datastructuren die dienst doen als photonmap.

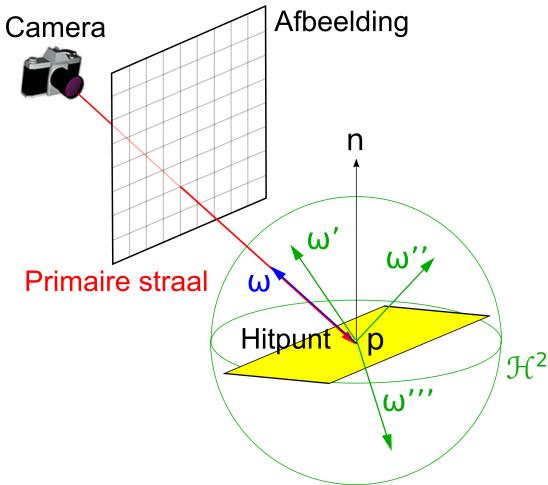
Photonmapping houdt zowel het *photonmapping lichttransportalgoritme* op basis van photons als de *photonmap datastructuur* om deze photons op te slaan in. Het doel van dit hoofdstuk is daarom tweeledig. In eerste instantie willen we de lezer de nodige achtergrond over en inzichten in het photonmapping algoritme meegeven. Vervolgens richten we ons op de photonmap datastructuur die we in de volgende hoofdstukken zullen proberen te verzoenen met een acceleratiestructuur.

3.1 Renderingvergelijking

Bouwend op het werk van Whitted [Whi80] en Cook et al. [CPC84] introduceerden Kajiya [Kaj86] en Immel et al. [ICG86] onafhankelijk van elkaar de *renderingvergelijking* die de lichtenergie-evenwichtstoestand in de scène definieert:

$$L_o(p, \vec{\omega}) = L_e(p, \vec{\omega}) + \int_{\mathcal{H}^2} L_i(p, \vec{\omega}') \cdot \rho(p, \vec{\omega}, \vec{\omega}') \cdot (\vec{\omega}' \cdot \vec{n}) d\vec{\omega}' \quad (3.1.1)$$

Hierbij is de uitgestraalde radiantie $L_o(p, \vec{\omega})$ in een punt p in de richting $\vec{\omega}$ gelijk aan de uitgestraalde radiantie in de richting $\vec{\omega}$ die het punt p zelf uitstraalt, $L_e(p, \vec{\omega})$, en de uitgestraalde radiantie die van elders dit punt p bereikt en uitgezonden wordt in de richting $\vec{\omega}$. Deze laatste bijdrage bekomen we door de invallende radiantie $L_i(p, \vec{\omega}')$ vermenigvuldigd met de Bidirectionele Scattering Reflectance Distribution Function



FIGUUR 3.1: De variabelen van de renderingvergelijking.

(BSDF), $\rho(p, \vec{\omega}, \vec{\omega}')$ te integreren over de eenheidsbol \mathcal{H}^2 van inkomende richtingen $\vec{\omega}'$. De BSDF geeft de hoeveelheid inkomende radiantie vanuit een richting $\vec{\omega}'$ in een punt p weer die gereflecteerd of gebroken wordt in een richting $\vec{\omega}$. Figuur 3.1 illustreert deze renderingvergelijking.

3.2 Lichttransportalgoritmen

Doordat de renderingvergelijking een complexe recursieve integraalvergelijking is, is Monte Carlo integratie ([Caf98], [Sam15]) voorgesteld als een unbiased techniek om het lichttransport numeriek te berekenen voor elke pixel in de finale afbeelding. Vandaag zijn vele van de beste lichttransportalgoritmen nog steeds gebaseerd op Monte Carlo integratietechnieken [DBBS03]. In essentie worden willekeurige paden geconstrueerd tussen de lichtbronnen en de camera. Verschillende manieren om deze paden te genereren liggen aan de basis van verschillende algoritmen voor het berekenen van het lichttransport.

Path tracing [Kaj86] volgt paden startende van de virtuele camera. Bidirectionele path tracing ([LW93], [VG95]) construeert paden startende van zowel de virtuele camera als van de lichtbronnen. Metropolis lichttransport ([VG97], [Vea98]) is een variant van bidirectionele path tracing die een sequentie van lichttransportpaden genereert door het willekeurig muteren van het huidige pad. Virtuele puntlichtbronnen [Kel97] berekenen op voorhand een puntbemonsterde lichtdistributie in de 3D scene die vervolgens wordt opgeslagen. Hierna wordt een finale afbeelding samengesteld door middel van een klassieke ray tracing stap. Virtuele sferische lichtbronnen [HKWB09], virtuele ray lichtbronnen [NNDJ12b] en virtuele beam lichtbronnen [NNDJ12a] breiden dit idee uit en zijn in staat om numerieke singulariteiten te vermijden, en kunnen eveneens additionele lichttransportfenomenen zoals participerende media

afhandelen. Photonmapping ([Jen96], [Jen01]), één van de meest populaire en flexibele algoritmen tot op heden, is een twee-staps-methode. De methode is geschikt voor een groot aantal verschillende belichtingscondities en is in staat om complexe belichtingsfenomenen zoals caustics te produceren. Volumetric photonmapping [JC98] is een uitbreiding van photonmapping in de aanwezigheid van participerende media.

Merk op dat al deze lichttransportalgoritmen op grotendeels dezelfde manier de directe belichtingsbijdrage in een hitpunt berekenen, maar verschillen in de berekening van de indirecte belichtingsbijdrage.

3.3 Photonmapping lichttransportalgoritme

Het photonmapping lichttransportalgoritme ([Jen96], [Jen01]) is een twee-staps-methode. Tijdens de eerste stap (photon tracing) wordt de invallende radiantie bemonsterd op een aantal locaties in de scène. Hiervoor worden paden vanuit de lichtbronnen gevolgd. Bij elke niet-speculaire spong wordt er een zogenoemde *photon*¹ gecreëerd. Al deze photons worden vervolgens opgeslagen in een spatiale datastructuur, de *photonmap*. Het is ook mogelijk meerdere photonsoorten voor verschillende lichtpaden, elk met hun eigen photonmap, te gebruiken. In een tweede stap (radiance estimate) worden de dichtstbijzijnde photons van een hitpunt van een straal met een geometrisch primitief bekomen via een k-nearest neighbor query in de photonmap (photon lookup). Deze photons worden vervolgens gebruikt om de invallende radiantie te benaderen in dit hitpunt via density estimation technieken (zoals histogram gebaseerde density estimation, kernel density estimation, etc.) ([Sil86], [Sim96]). Vele implementaties gebruiken een vaste maximale querystraal, R_{\max} , alsook een vast maximaal aantal photons, k , voor de k-nearest neighbor query's. We verwijzen de geïnteresseerde lezer naar [PH10] voor een tot op codeniveau beschrijving van het photonmapping lichttransportalgoritme.

Om het verband tussen de verschillende soorten lichtpaden, die het photonmapping algoritme apart beschouwd, en de renderingvergelijking (3.1.1) duidelijk te maken, kunnen we gebruik maken van integralen analoog aan deze renderingvergelijking. We gebruiken echter een eenvoudigere en overzichtelijker reguliere expressienotatie: - is de concatenatie, | is de alternatie, * is de Kleene star operatie en $S^+ = S - S^*$. Elk lichtpad start op een lichtbron L. Indien we dit lichtpad willen waarnemen, moet het invallen op de camera C. Tussen de lichtbron en de camera kan het lichtpad botsen op een oppervlak waardoor het lichtpad gereflecteerd wordt volgens één van de componenten van de BSDF horende bij dit oppervlak. Dit kan volgens een diffuse D of speculaire S component.

De lichtpaden horende bij de verschillende photonsoorten kunnen geschreven worden als:

¹Deze photons hebben niets met fotonen te maken.

3. PHOTONMAPPING

L-D	directe photon
L-S ⁺ -D	caustic photon
L-S [*] -D-(S D) [*] -D	indirecte photon

Elk van deze photons bevindt zich op een oppervlak waarbij de BSDF een diffuse component heeft. Het lichtpad van directe photons is rechtstreeks afkomstig van een lichtbron. Het lichtpad van caustic photons is minstens eenmaal en enkel gereflecteerd volgens speculaire BSDF componenten. Het lichtpad van indirecte photons is minstens eenmaal gereflecteerd volgens een diffuse BSDF component. Bij het construeren van hybride kd-bomen voor geometrische primitieven en photons, nemen we enkel deze indirecte photons in rekening.²

Stralen vertrekken vanuit de camera. In het hitpunt van een straal met een oppervlak willen we de uitgestraalde radiantie $L_o(p, \vec{\omega})$ bepalen. Dit in de omgekeerde richting van de straal, dus naar de camera toe. Het oppervlak kan zelf radiantie uitstralen (zelf-emissie). Daarnaast moeten we de invallende directe en indirecte belichting in het hitpunt bepalen. De directe belichting kan bepaald worden door een directe photon lookup in de buurt van het hitpunt of door rechtstreeks de lichtbronnen te bemonsteren. De indirecte belichting is iets complexer. We moeten hiervoor zowel een caustic als indirecte photon lookup in de buurt van het hitpunt uitvoeren. Daarnaast moeten we vanuit het hitpunt opnieuw een straal volgen in de perfect speculaire richting indien de BSDF van het oppervlak zo'n component heeft. Deze laatste bijdrage belicht het hitpunt enkel vanuit één specifieke richting uit een oneindige verzameling van richtingen. Een photonmapping algoritme beschouwt dus volgende lichtpaden:

1. L-C zelf-emissie
2. L-D-C directe photon lookup
of rechtstreeks bemonsteren van directe belichting
3. L-S⁺-D-C caustic photon lookup
4. L-(S|D)^{*}-S-C recursieve ray tracing
5. L-S^{*}-D-(S|D)^{*}-D-C indirecte photon lookup
of final gathering ray tracing stap

De bijdragen van alle photon lookups kunnen we samennemen:

1. L-C zelf-emissie
2. L-(S|D)^{*}-D-C directe + caustic + indirecte photon lookup
4. L-(S|D)^{*}-S-C recursieve ray tracing

²Het is uiteraard mogelijk caustic photons of zowel caustic als indirecte photons in rekening te nemen zonder al te veel aanpassingen. Dit hebben we niet verder onderzocht.

Deze overige drie bijdragen kunnen we samennemen:

$$1. L - (S|D)^* - C \quad \text{globale belichting}$$

Deze laatste reguliere expressie drukt de verzameling van alle mogelijke lichtpaden tussen de lichtbronnen en de camera uit. Al deze reguliere expressies tonen aan dat het photonmapping lichttransportalgoritme alle mogelijke lichtpaden in rekening neemt en slechts eenmaal in rekening neemt.

Photonmapping is consistent (de numerieke benadering convergeert naar de exacte oplossing als het aantal Monte Carlo monsters naar oneindig gaat), maar is biased door het interpoleren van naburige photons. Het gebruik van de photonmap kan resulteren in verschillende soorten artefacten [Wal99]:

- Lage frequentie ruis door te weinig photons en een te kleine k voor kNN query's;
- Blurring door te weinig photons en een te grote k voor kNN query's;
- Energy bleeding door de geometrie niet in rekening te nemen;
- Overmodulatie door een te kleine R_{\max} ;
- Verkeerde oppervlakteschatting;
- Lage photondensiteiten in belangrijke gebieden.

Verschillende verbeteringen zijn voorgesteld om dit lichttransportalgoritme efficiënter te maken in het berekenen van de verschillende belichtingsbijdragen. Lichtbronnen worden rechtstreeks bemonsterd voor het berekenen van de directe belichtingscomponent. Een extra final gather ray tracing stap vanuit een hitpunt elimineert verschillende artefacten. De integraal is namelijk een smoothing operator. Suykens en Willems [SW00] stellen een methode voor om de densiteit van de photonmaps te controleren door de photons selectief te selecteren op basis van een lokaal densiteitscriterium. Hierdoor is er een asymptotische convergentie naar de correcte oplossing.

Bij het gebruik van het photonmapping algoritme is het belangrijk aandacht te besteden aan de geheugenvereisten nodig om de photons op te slaan alsook aan de hoeveelheid photons nodig om op accurate wijze de belichting voor te stellen. Reverse photonmapping [HHS05] is een twee-staps-methode die zoals de naam het zegt een omgekeerde strategie hanteert. In een ray tracing stap worden de hitpunten opgeslagen in een kd-boom. In de daaropvolgende photon tracing stap wordt deze datastructuur gebruikt om de dichtstbijzijnde hitpunten te vinden waaraan een photon bijdraagt. Deze methode is zeer geschikt wanneer het aantal te volgen stralen groot

3. PHOTONMAPPING

is (bv.: final gathering stap). De unbiased varianten van photonmapping verhelpen de geheugenbottleneck door niet de volledige photonmap op te slaan. Progressieve photonmapping [HOJ08] is een meer-staps-methode bestaande uit een ray tracing stap gevolgd door meerdere photon tracing stappen. De recentere stochastic progressive photonmapping [HJ09] is in staat om de correcte radiantiewaarde voor een gebied in plaats van een punt te berekenen om distributie ray tracing effecten effectiever af te handelen. De memoryless progressive photonmapping [KZ11] elimineert de nood om lokale statistieken bij te houden ten koste van het volgen van meer paden.

3.4 Photonmap

Eenmaal alle photons bekomen zijn, worden deze photons opgeslagen in een spatiale datastructuur met als doel de per-query kost te verminderen. Deze datastructuur wordt de *photonmap* genoemd. Door slechts een beperkt aantal photons te genereren en door dezelfde photons te gebruiken voor de belichting van meerdere oppervlakken, worden photons niet opgeslagen op de geometrische primitieven. Een *belichtingsmap* [Arv86] daarentegen gebruikt een textuurmap met belichtingswaarden per geometrisch primitief. Het gebruik van belichtingsmappen heeft echter veel nadelen:

- Het op voorhand bepalen van de nodige resolutie van deze belichtingsmappen is moeilijk;
- Een belichtingsmap per geometrisch primitief is zowel wat de berekening als het geheugengebruik betreft duur in scenes met veel geometrische primitieven;
- Het parametriseren van de belichtingsmap voor veel soorten geometrische primitieven is niet evident.

De eenvoudigste photonmap is een *regulier rooster* [Jen01]. Voor een k-nearest neighbor query moet eerst de roostercel waarin het hitpunt gelegen is bepaald worden. Vervolgens moet elke photon gelegen in deze roostercel op nabijheid getest worden. Afhankelijk van de ligging van het hitpunt binnen de roostercel, moeten de photons in naburige roostercellen ook gecontroleerd worden. Een regulier rooster is ideaal wanneer de photons uniform verdeeld zijn in de scene.

Photons liggen steeds op een geometrisch primitief. Door de distributie van de geometrie, zijn de photons in de praktijk niet uniform verdeeld. Daarom verdienen *kd-bomen* ([BF79], [Jen01]) omwille van hun adaptieve karakter de voorkeur. Doordat photons puntdata zijn, wordt er zowel een photon op de intermediaire knopen als bladknopen opgeslagen. Het vinden van de dichtstbijzijnde photon heeft een gemiddelde $\mathcal{O}(\log |P|)$ en worst-case $\mathcal{O}(|P|)$ tijdscomplexiteit met $|P|$ het totaal aantal photons. Het balanceren van de boom verlaagt deze worst-case tijdscomplexiteit tot $\mathcal{O}(\log |P|)$. De bouwheuristiek die de kd-boom steeds expliciet balanceert, noemen we de Balanced Heuristic (BH).

Het balanceren van de kd-boom is enkel optimaal indien elke photon een gelijke kans heeft om gecontroleerd te worden in een k-nearest neighbor query of indien we enkel de dichtstbijzijnde photon gebruiken (binair zoeken). De Voxel Volume Heuristic (VVH) [WGS04] probeert de per-query kost te reduceren door de kd-boom niet te balanceren. Aan elke kandidaatvoxel wordt een kost toegekend gelijk aan de kost om deze voxel te bezoeken vermenigvuldigd met de kans om deze voxel effectief te bezoeken in de context van een k-nearest neighbor query. De VVH kan gebruikt worden voor zowel indirecte als caustic photons. De performantiewinst is het grootste voor caustic photons.

Een alternatieve aanpak [Fle09] gebaseerd op spatiale hashing, organiseert de photons in de photonmap. Deze strategie reduceert de onregelmatige geheugentoegang van de kd-boom tijdens de constructie en lookup en laat toe de constructie en lookup eenvoudig te paralleliseren op de GPU.

Het *Voronoi diagram* [Aur91] maakt een verbinding tussen elke knoop (photon) en zijn dichtste buren. Startende vanuit een willekeurige knoop kunnen we het pad volgen naar de dichtste photon. Eenmaal deze photon gevonden is, moeten de dichtste buren recursief gecontroleerd worden om de k-nearest neighbor query te voltooien. Het nadeel van dergelijke structuur is het hoge geheugengebruik waardoor deze minder bruikbaar is als photonmap.

Hoofdstuk 4

Hybride photonmappen

In dit hoofdstuk introduceren we drie verschillende hybride kd-bomen die de gecombineerde kost van ray tracing en k-nearest neighbor query's proberen te verminderen. In sectie 4.1 voegen we de photons toe aan de kd-boom acceleratiestructuur met geometrische primitieven. In sectie 4.2 veralgemenen we dit idee door een hybride kd-boom te bouwen op basis van de geometrische primitieven én photons. Beide strategieën resulteren in één hybride kd-boom die zowel de geometrische primitieven als photons bevat. In sectie 4.3 behouden we de aparte kd-boom acceleratiestructuur en kd-boom photonmap, elk bekomen met een optimale heuristiek voor hun specifieke data en verbinden we deze kd-bomen door middel van ropes.

De niet-hybride setting is weergegeven in Figuur 4.1a.

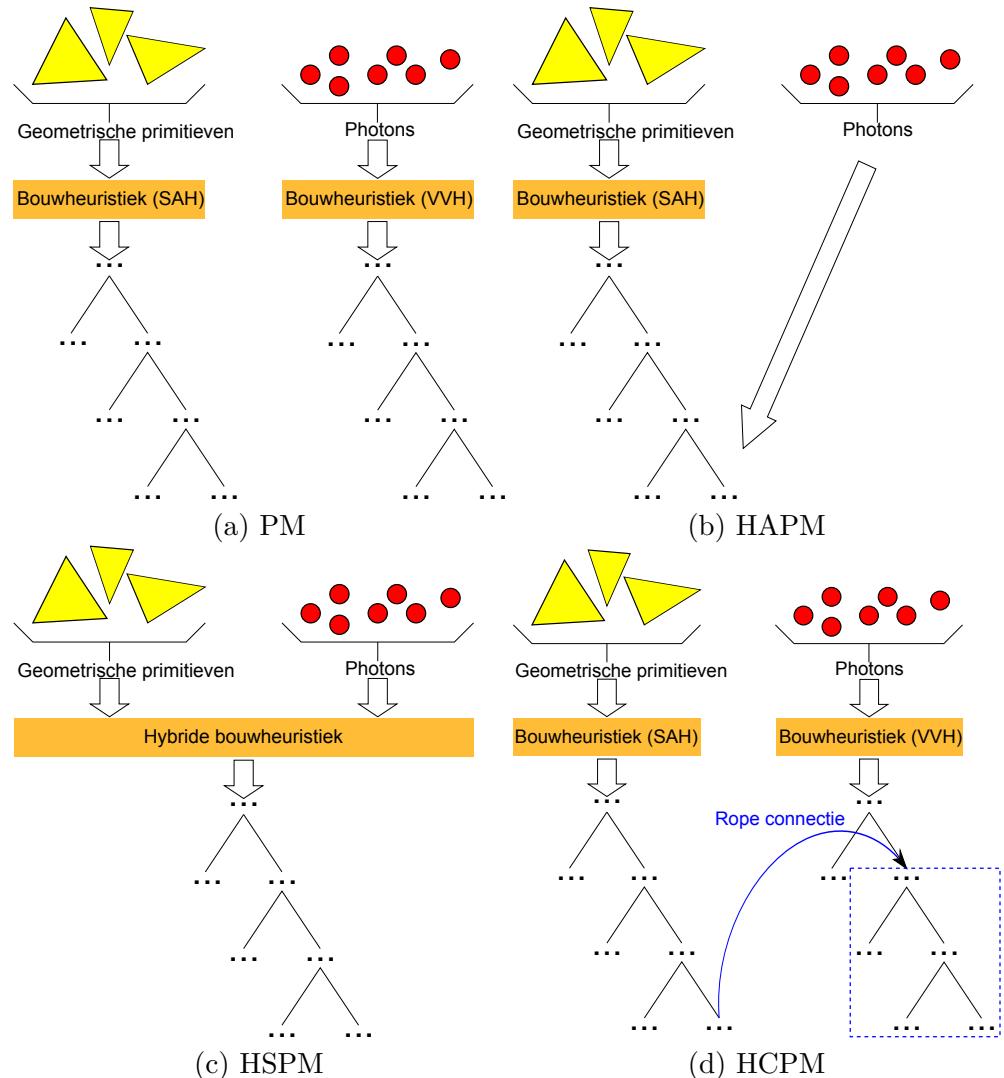
4.1 Hybrid Additive Photon Map

Het efficiënt distribueren van de photons in de scène tijdens de photon tracing fase (zie sectie 3.3) vereist reeds een acceleratiestructuur waarin de geometrische primitieven van de scène zijn ondergebracht. In het geval van een spatiale acceleratiestructuur zoals een regulier rooster of kd-boom, kunnen we de photons eenmaal deze gegenereerd zijn, onderbrengen in de bestaande acceleratiestructuur. Het onderbrengen van de photons kan dan snel, eenvoudig en zonder veel geheugengebruik uitgevoerd worden.

De Hybrid Additive Photon Map (HAPM) bestaat uit een kd-boom voor geometrische primitieven waaraan vervolgens photons worden toegevoegd. De verzameling van photons wordt opgedeeld volgens de splitsingsvlakken van de kd-boom, opdat elke photon in zijn omsluitende bladvoxel terecht komt. Photons die juist op een splitsingsvlak gelegen zijn, worden slechts eenmaal opgesplitst. De resulterende kd-boom bestaat uit geometrische primitieven en photons die beide enkel voorkomen op de bladknopen. Dit idee is weergegeven in Figuur 4.1b.

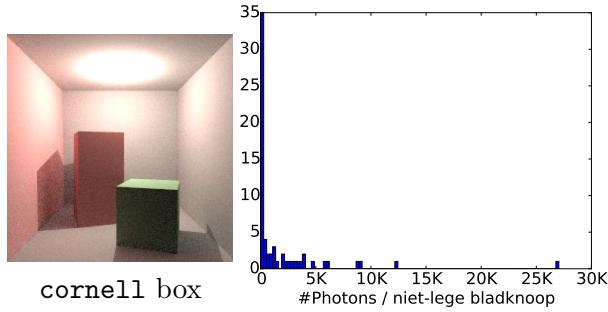
Door zowel de geometrische primitieven als photons op te delen in eenzelfde

4. HYBRIDE PHOTONMAPPEN



FIGUUR 4.1: (a) Niet-hybride kd-bomen (b) Hybrid Photon Map (c) Hybrid Swapping Photon Map (d) Hybrid Connected Photon Map.

4.1. Hybrid Additive Photon Map



FIGUUR 4.2: Histogram (100 equidistante bins) van het aantal photons per niet-lege bladknoop van een **cornell** box voor $100k$ photons en maximale querystraal $R_{\max} = 1$.

hybride kd-boom, verwachten we dat k-nearest neighbor query's voornamelijk dichtstbijzijnde photons moeten controleren en slechts een klein deel overige photons (overhead). Bij een fijne tesselatie van de geometrie, verwachten we dat de opdeling van de geometrische primitieven de opdeling van de photons meestuurt.

Deze eenvoudige strategie leidt echter niet tot kwalitatieve kd-bomen die de gecombineerde kost van de ray tracing en k-nearest neighbor query's verminderen. Voor een eenvoudige **cornell** box, nemen we enorme hoeveelheden photons per geometrisch primitief waar (zie Figuur 4.2). Photons bevinden zich steeds op een geometrisch primitief. Indien er weinig of veel grote geometrische primitieven (versus de photon puntdata) aanwezig zijn in de scene, bezetten veel photons dezelfde geometrische primitieven en bijgevolg dezelfde bladvoxels.

Om grote photonclusters te elimineren, construeren we een kd-boom photonmap in elke bladvoxel met alle photons van die bladvoxel. Deze kd-boom photonmappen zorgen voor een verdere verfijning van de kd-boom acceleratiestructuur. De geometrische primitieven (niet-puntdata) worden niet verder opgedeeld aangezien bij de bouw van de acceleratiestructuur reeds de optimale (volgens de gebruikte bouwheuristiek) kd-boom geconstrueerd is. De resulterende datastructuur is nog steeds een kd-boom waarbij de geometrische primitieven zich nu zowel op intermediaire knopen als bladknopen kunnen bevinden.

De verschillende photonmappen kunnen gebouwd worden met de BH of VVH. In de praktijk zal de bouwkost voor het geheel van de verschillende kleinere photonmappen lager zijn dan de bouwkost voor de volledige photonmap.

De HAPM kan zowel top-down (vanuit de wortelknoop) als bottom-up (vanuit de knoop met het hitpunt) doorlopen worden voor een k-nearest neighbor query. Om een bottom-up doorloopstrategie mogelijk te maken, is het voor een implementatie nodig informatie over de voxel waarin een hit plaatsvindt, terug te geven. Op deze manier kan de k-nearest neighbor query rechtstreeks starten vanuit de voxel waarin

het hitpunt alsook een deel van de dichtst gelegen photons zich bevinden. Bij een bottom-up doorloopstrategie is het mogelijk query's vroegtijdig te beëindigen voor het bereiken van de wortelknoop (zie sectie 6.3.7).

We hebben zowel geprobeerd een efficiënte bottom-up doorloopstrategie voor k-nearest neighbor query's te bekomen via een stack met de nog te controleren voxels als via ropes ([HHS05], [PGSS07]). De doorloopstrategie van een query is fundamenteel verschillend van de doorloopstrategie van een straal. Voor een straal moeten we een rechte lijn in de driedimensionale ruimte volgen. Voor een query moeten we de photons in een bolvormig gebied eenmalig controleren. Indien we gebruik wensen te maken van ropes zijn we daarom genoodzaakt bij te houden welke voxels reeds gecontroleerd zijn. Dit kan zowel op de knopen als stralen bijgehouden worden (schaalbaar naar meerdere threads). Dit resulteert in beide gevallen in meer overhead in vergelijking met een stack. Onze HAPM implementatie maakt gebruik van een bottom-up doorloopstrategie en een stack.

Opmerking 4.1 (Photonsoorten). Indien er meerdere photonsoorten (caustic, indirecte, etc.) toegevoegd moeten worden aan de kd-boom, stellen we voor een photonmap per photonsoort in elke bladvoxel te construeren. K-nearest neighbor query's moeten op deze manier geen onderscheid maken tussen de photons horende bij verschillende photonsoorten en kunnen efficiënter uitgevoerd worden. De resulterende datastructuur voor alle photonsoorten samen is dan wel geen kd-boom meer.

Opmerking 4.2 (Bouwheuristiek). De bekomen hybride kd-bomen zijn kd-boom acceleratiestructuren waaraan photons worden toegevoegd. Deze kd-boom acceleratiestructuren construeren we met (niet-hybride) bouwheuristieken zoals de SAH. Het is ook mogelijk een heuristiek te definiëren die de kost van k-nearest neighbor query's in rekening neemt. De moeilijkheid bestaat er nu in de verwachte hoeveelheid en distributie van de photons te voorspellen aangezien de photons pas later gegenereerd worden. Dit is het kip-en-eiprobleem: enerzijds willen we tijdens de constructie van de acceleratiestructuur de photons in rekening nemen en anderzijds hebben we de acceleratiestructuur nodig om de photons te genereren. De oplossing bestaat erin de (volledige) acceleratiestructuur opnieuw op te bouwen op basis van de geometrische primitieven én de photons (zie sectie 4.2).

4.2 Hybrid Swapping Photon Map

Om te onderzoeken of we kwalitatieve hybride kd-bomen voor geometrische primitieven en photons kunnen construeren, moeten we zowel de geometrische primitieven als photons in rekening nemen tijdens de constructiefase. In een photonmapping algoritme wordt er zowel een aanzienlijke deel van de totale rendertijd gespendeerd aan het volgen van stralen als aan het uitvoeren van k-nearest neighbor query's. Daarnaast is het volgen van een straal ook fundamenteel anders dan het uitvoeren van een query. Enerzijds moeten we zowel de ray tracing kost als k-nearest neighbor

querykost in rekening nemen en anderzijds mogen we de bijdrage van het één niet representatief laten zijn voor de bijdrage van het ander en visa versa.

De Hybrid Swapping Photon Map (HSPM) is een hybride kd-boom geconstrueerd op basis van de geometrische primitieven en photons. De bouwheuristiek die aan de basis ligt van deze kd-bomen, wordt geïntroduceerd en uitgewerkt in de volgende twee hoofdstukken. De naam van de photonmap duidt op de noodzaak om van acceleratiestructuur te wisselen tijdens de pre-rendering fase. Dit idee is weergegeven in Figuur 4.1c.

Opmerking 4.3 (HAPM versus HSPM). De HAPM is een speciaal geval van de meer algemene HSPM. Bij de HAPM zijn de photons steeds op lagere knopen dan de geometrische primitieven gelegen. Verder deelt enkel de photon op de wortelknoop van een photonmap een knoop met geometrische primitieven. De HSPM daarentegen laat alle mogelijke knoopbezettingen van photons en geometrische primitieven toe.

4.3 Hybrid Connected Photon Map

De Hybrid Connected Photon Map (HCPM) bestaat uit twee aparte kd-bomen die optimaal gebouwd zijn voor hun specifieke data. De ene kd-boom bevat de geometrische primitieven en de andere kd-boom bevat de photons. Tot zover is de setting identiek aan de huidige state-of-the-art. In plaats van ropes te laten mappen op voxels binnen dezelfde kd-boom, kunnen deze ropes eveneens overeenkomen met voxels van een andere kd-boom.

We maken nu gebruik van een vaste maximale querystraal R_{\max} . Elke kindvoxel V met geometrische primitieven krijgt nu één rope. Deze rope komt overeen met de kleinste voxel in de photonmap die de voxel V uitgebreid met R_{\max} in alle richtingen volledig omsluit. Indien er een hit van een straal met een geometrisch primitief in V plaatsvindt, moet een k-nearest neighbor query vanuit dit hitpunt enkel het volume van de bijhorende rope in rekening nemen. Dit vermindert de doorlooptijd (afdalen en controleren van photons tijdens de afdaling) in vergelijking met een query die steeds vanuit de wortel van de photonmap moet starten. Dit idee is weergegeven in Figuur 4.1d.

Voor de eenvoud breiden we de zijden van een niet-lege kindvoxel uit met R_{\max} aan weerszijden in de drie primaire richtingen. Dit resulteert opnieuw in een balkvormig gebied dat het gebied van de voxel V uitgebreid met R_{\max} in alle richtingen volledig omsluit. Deze voxel noemen we de *minimaal omsluitende photonvoxel*. De minimaal omsluitende photonvoxel van dit gebied kan als volgt berekend worden: gegeven een oudervoxel die het gebied volledig omsluit, controleren we of de twee kindvoxels het gebied overlappen. Indien beide kindvoxels het gebied overlappen, is de oudervoxel de minimaal omsluitende voxel. Indien slechts één van beide kindvoxels het gebied overlapt en dus omsluit, herhalen we dit proces recursief voor deze kindvoxel.

4. HYBRIDE PHOTONMAPPEN

Deze strategie kunnen we telkens opnieuw en onafhankelijk toepassen voor elke niet-lege kindvoxel. Bij veel kleine, niet-lege kindvoxels en diepe photonmappen is dit omslachtig doordat we steeds opnieuw afdaalen uit de wortel van de photonmap. Door de rope voor elke voxel (zowel intermediair als kind) te berekenen, kunnen we op basis van de rope van de oudervoxel de rope van de twee kindvoxels bepalen. Dit vermindert redundant rekenwerk.

Opmerking 4.4 (Voxels buiten de photonmap). Voxels met geometrische primitieven die volledig buiten het volume van de photonmap gelegen zijn, kunnen een speciale rope krijgen om te vermijden de photonmap te controleren.

Opmerking 4.5 (Variabele querystraal). De implementatie van het photonmapping algoritme in `pbrt` [PH10] maakt ondanks een vaste maximale querystraal gebruik van multiple importance sampling voor het bemonsteren van de richting van de final gather stralen, op basis van de N dichtste photons ten opzichte van het hitpunt. De maximale querystraal R_{\max} wordt hiervoor geleidelijk aan opgehoogd tot het gewenste aantal photons bekomen wordt. Voor deze query's is het belangrijk ook de originele lookup methode in de photonmap te behouden. De rope van een voxel omsluit enkel het volume van deze voxel uitgebreid met R_{\max} in alle richtingen. Merk op dat de querystraal geleidelijk aan opgehoogd moet worden in plaats van gelijk te stellen aan oneindig. In dit laatste geval is het mogelijk dat de query elke photon in de photonmap moet controleren.

Hoofdstuk 5

Mathematisch framework

Het uiteindelijke doel van dit hoofdstuk is het bekomen van een hybride bouwheuristiek om *hybride kd-bomen* te construeren die zowel geometrische primitieven (niet-puntdata) als photons (puntdata) stockeren zodanig dat de gecombineerde kost van ray tracing en k-nearest neighbor query's verminderd wordt. De ray tracing kost bestaat uit het volgen van stralen doorheen de hybride kd-boom totdat deze de hybride kd-boom verlaten of totdat het dichtste hitpunt van deze stralen met een geometrisch primitief gevonden wordt. Vanuit een hitpunt kan een k-nearest neighbor query starten waarbij de dichtstbijzijnde photons ten opzichte van dit hitpunt gevonden moeten worden. Doordat het volgen van een straal en het uitvoeren van een k-nearest neighbor query fundamenteel verschillend zijn, is het belangrijk om zowel de bijdrage van de ray tracing als k-nearest neighbor query's in rekening te nemen bij het bouwen van de hybride kd-bomen. Dit verklaart de zoektocht naar een *hybride bouwheuristiek* die we gemakshalve Hybrid Heuristic (HH) noemen. Deze HH kan vervolgens gebruikt worden om onze HSPMs (zie sectie 4.2) te bouwen.

In deze HH kapselen we een bouwheuristiek voor kd-boom acceleratiestructuren en kd-boom photonmappen in. Om kd-boom acceleratiestructuren te construeren, moeten splitsingsvlakken die de per-straal-kost minimaliseren, gekozen worden op elk niveau van de boom. Om kd-boom photonmappen te construeren, moeten splitsingsvlakken die de per-query-kost minimaliseren, gekozen worden op elk niveau van de boom. Voor computationele efficiëntie, worden hiervoor meestal geschikte (gulzige) heuristieken gebruikt. De kostmodellen van beide onderliggende bouwheuristieken combineren we tot een hybride kostmodel waarbij de ray tracing kost en k-nearest neighbor querykost aan elkaar gekoppeld worden. Een k-nearest neighbor query kan namelijk enkel starten vanuit een hitpunt.

De de-facto standaard voor het bouwen van kd-boom acceleratiestructuren is de SAH [MB90]. Deze SAH veronderstelt echter dat stralen oneindig lang zijn en nooit een geometrisch primitief raken waardoor er geen koppeling mogelijk is met het voorkomen van k-nearest neighbor query's (vanuit een hitpunt). Daarom breiden we

deze SAH uit door straalterminatie in rekening te nemen. We passen meer bepaald de RTSAH [IH11] aan om kd-bomen te bouwen die de kost voor het volgen van primaire stralen reduceren.

Voor het bouwen van kd-boom photonmappen gebruiken we de VVH [WGS04]. De BH kunnen we niet gebruiken aangezien deze geen kostmodel heeft; de photonmap wordt steeds explicet gebalanceerd.

Alvorens de verschillende heuristieken te bespreken, geven we in sectie 5.1 de intuïtie achter de gebruikte notatie van de componenten van de verschillende kostfuncties. In sectie 5.2 herhalen we kort de SAH en in sectie 5.3 introduceren we onze aanpassing van de originele RTSAH om kd-bomen te bouwen. In sectie 5.4 herhalen we kort de VVH. Uiteindelijk introduceren we in sectie 5.5 onze HH.

5.1 Notatie

Elk van de heuristieken die in dit hoofdstuk aan bod komen is opgebouwd uit zijn eigen specifieke componenten. Deze componenten worden besproken daar waar ze geïntroduceerd worden. Toch willen we reeds nu de intuïtie achter de notatie van deze componenten duidelijk maken, aangezien we deze consistent toepassen voor elk van de heuristieken.

p duidt op een probabilitéit en \mathcal{C} duidt op een kost. Een subscript geeft extra informatie over de component. Een superscript geeft informatie over de context. Doordat we opbouwen naar een bouwheuristic die zowel de ray tracing als k-nearest neighbor query's in rekening neemt, willen we explicet duidelijk maken waarop de probabilitéit- en kostencomponenten betrekking hebben. De ray tracing context stellen we voor als RT en de k-nearest neighbor query context als kNN. Een probabilitéit- of kostencomponent zonder expliciete contextnotatie heeft betrekking op beide contexten en dient nog verder opgesplitst te worden.

Verder gebruiken we consistent dezelfde notatie voor componenten die gedeeld worden door verschillende heuristieken.

5.2 Surface Area Heuristic

De SAH [MB90] is een bouwheuristic voor het bouwen van kd-boom acceleratiestructuren met als doel de kost voor het volgen van primaire stralen te verminderen. De SAH kent aan elke kandidaatvoxel een kost toe die gelijk is aan de kost om deze voxel te bezoeken vermenigvuldigd met de kans om deze voxel effectief te bezoeken in de context van het volgen van een primaire straal (RT). Om deze kansen te kunnen berekenen voor elke kandidaatvoxel, maakt de SAH een aantal assumpties over de stralen. Deze assumpties worden besproken in sectie 5.2.1. De SAH kostfunctie wordt besproken in sectie 5.2.2.

5.2.1 Assumpties

De SAH maakt de volgende drie assumpties over de stralen:

Assumptie 5.2.1 (Oorsprong van de stralen). *De oorsprong van de stralen is uniform verdeeld in de ruimte buiten de omhullende box van de scene.*

Assumptie 5.2.2 (Richting van de stralen). *De richting van de stralen is uniform verdeeld.*

Assumptie 5.2.3 (Terminatie van de stralen). *De stralen zijn oneindig lang.*

Ondanks de belangrijkheid van de SAH, gelden de assumpties onderliggende aan deze SAH niet in de praktijk.

Secundaire stralen zoals schaduw-, reflectie- en refractiestralen vertrekken altijd vanop een geometrisch primitief. Indien de camera gepositioneerd is binnen de omhullende box van de scene, vertrekken al de primaire stralen (stralen die vertrekken vanuit de camera) van binnen de scene. De oorsprong van stralen is dus vaak gelegen in de ruimte binnen de omhullende box van de scene.

De uniforme verdeling van de stralen in [assumptie 5.2.2](#) beoogt twee vereenvoudigingen. Ten eerste willen we een camera-onafhankelijke acceleratiestructuur bekomen. In animaties van statische scenes, kan hierdoor de positionering van de camera gewijzigd worden zonder de acceleratiestructuur opnieuw te moeten opbouwen. Ten tweede is het computationeel duur om de niet-uniforme verdeling van de stralen te bemonsteren door de geometrische distributie.

De laatste [assumptie 5.2.3](#) is wellicht de meest contra-intuïtieve. Stralen die oneindig lang zijn, raken geen geometrische primitieven. In de meeste praktische scenes, raken de stralen vroeg of laat een oppervlak waarna ze eindigen. Deze laatste [assumptie relaxeren we in sectie 5.3](#) bij de introductie van de RTSAH.

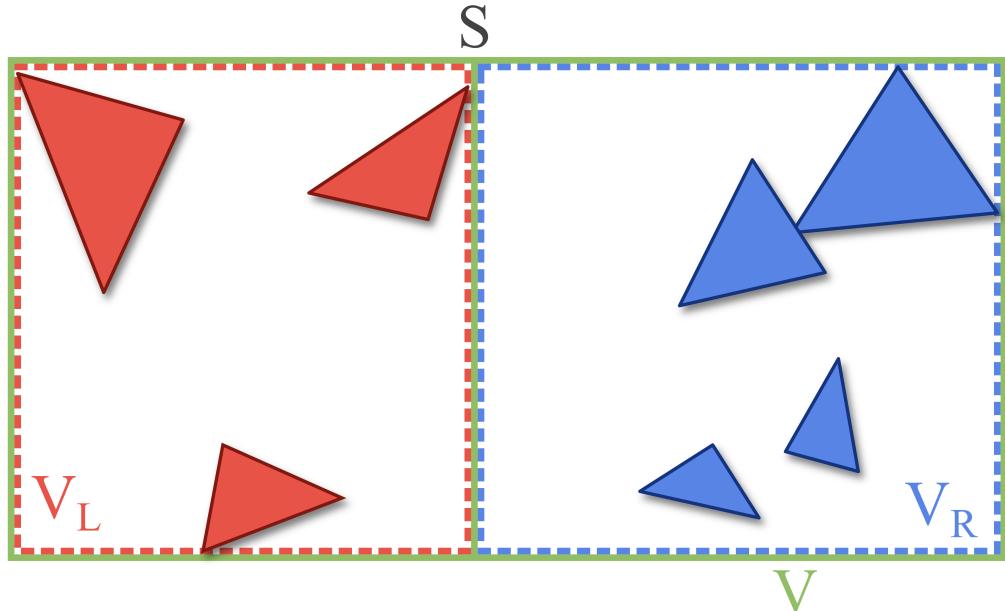
5.2.2 Kostfunctie

De SAH kostfunctie geeft een schatting van de kost van het splitsen van een voxel V met een splitsingsvlak gepositioneerd in S in een linkse V_L en rechtse V_R kindvoxel als:

$$\mathcal{C}_{\text{SAH}}(S : V \rightarrow \{V_L, V_R\}) = \mathcal{C}_{\text{trav}}^{\text{RT}} + p_L^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_L) + p_R^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_R) \quad (5.2.1)$$

Figuur [5.1](#) illustreert de opsplitsing van een voxel V en de overlappende geometrische primitieven. De heuristiek bestaat uit de volgende componenten:

- $\mathcal{C}_{\text{trav}}^{\text{RT}}$, de kost om de oudervoxel V te doorlopen. Wanneer een straal de oudervoxel doorkruist, moet het verloop van deze straal doorheen de kindvoxels



FIGUUR 5.1: De opsplitsing van een voxel V en de overlappende geometrische primitieven met een splitsingsvlak gepositioneerd in S in een linkse V_L en rechtse V_R kindvoxel.

bepaald worden. De straal kan één of beide kindvoxels doorkruisen. Indien beide kindvoxels doorkruist worden, moet bepaald worden welke kindvoxel eerst doorkruist wordt.

- $\mathcal{C}_{\text{visit}}^{\text{RT}}(V_L)$ ($\mathcal{C}_{\text{visit}}^{\text{RT}}(V_R)$), de kost om de linkse (rechtse) kindvoxel te bezoeken.
- p_L^{RT} (p_R^{RT}), de kans om de linkse (rechtse) kindvoxel te doorkruisen gegeven dat de oudervoxel V doorkruist wordt.

$\mathcal{C}_{\text{visit}}^{\text{RT}}(V)$ is gedefinieerd als de kost om een straal op intersectie te testen met alle geometrische primitieven $G_{V,i}$ in de voxel V :

$$\mathcal{C}_{\text{visit}}^{\text{RT}}(V) = \sum_{i=1}^{|G_V|} \mathcal{C}_i^{\text{RT}}(G_{V,i}) \approx \mathcal{C}_{\text{isect}}^{\text{RT}} \cdot |G_V| \quad (5.2.2)$$

Hierbij stelt G_V de verzameling van geometrische primitieven die V overlappen voor, $|\cdot|$ de kardinaliteitsoperator die het aantal elementen van een verzameling uitdrukt en $\mathcal{C}_i^{\text{RT}}(\cdot)$ de intersectiekost van het geometrische primitief (bv.: driehoek, bol, etc.). Indien er weinig variatie is in de intersectiekost of in de soorten geometrische primitieven die in de scène voorkomen, wordt de intersectiekostfunctie meestal vervangen door een constante intersectiekost, $\mathcal{C}_{\text{isect}}^{\text{RT}}$, omwille van computationele efficiëntie.

p_L^{RT} (p_R^{RT}) is de conditionele probabiliteit om de linkse (rechtse) kindvoxel te doorkruisen gegeven dat de oudervoxel doorkruist wordt. Door de assumpties over de externe ligging en uniforme verdeling van de oorsprong 5.2.1 en richting 5.2.2 van de stralen zijn deze probabiliteiten exact gegeven als:

$$p_L^{RT} = p(V_L \text{ doorkruist} \mid V \text{ doorkruist}) = \frac{SA_{V_L}}{SA_V} \quad (5.2.3)$$

$$p_R^{RT} = p(V_R \text{ doorkruist} \mid V \text{ doorkruist}) = \frac{SA_{V_R}}{SA_V} \quad (5.2.4)$$

Hierbij stelt SA_{V_k} de oppervlakte van de voxel V_k voor. Cauchy toonde aan dat de gemiddelde geprojecteerde oppervlakte van een driedimensionaal convex veelvlak gelijk is aan één vierde van zijn oppervlakte.

De kost van het niet splitsen van een voxel V is gedefinieerd als de kost om alle geometrische primitieven in deze voxel op intersectie te testen (5.2.2):

$$\mathcal{C}_{\text{no split}}^{RT}(V) = \mathcal{C}_{\text{visit}}^{RT}(V). \quad (5.2.5)$$

Een voxel zal enkel gesplitst worden in twee kindvoxels indien de kost van het splitsen kleiner is dan de kost van het niet splitsen.

Lege-ruimte-bonus. Wald en Havran [WH06] stelden voor om het afsplitsen van lege kindvoxels te bevoordelen door de verwachte kost voor het doorlopen van de kindvoxels te vermenigvuldigen met

$$\lambda(V_L, V_R) = \begin{cases} 1 - \alpha, & \text{als } G_{V_L} = \emptyset \vee G_{V_R} = \emptyset \\ 1, & \text{anders} \end{cases} \quad (5.2.6)$$

Hierbij is de *lege-ruimte-bonus* α typisch 0,2. De aangepaste SAH kostfunctie wordt dan

$$\begin{aligned} \mathcal{C}'_{\text{SAH}}(S : V \rightarrow \{V_L, V_R\}) \\ = \mathcal{C}_{\text{trav}}^{RT} + \lambda(V_L, V_R) (p_L^{RT} \cdot \mathcal{C}_{\text{visit}}^{RT}(V_L) + p_R^{RT} \cdot \mathcal{C}_{\text{visit}}^{RT}(V_R)) \end{aligned} \quad (5.2.7)$$

Deze optimalisatie resulteert meestal in kwalitatief betere kd-bomen.

5.3 Ray Termination Surface Area Heuristic

De RTSAH breidt de SAH uit door assumptie 5.2.3 die veronderstelt dat stralen oneindig lang zijn, te elimineren door de mogelijke terminatie van stralen in rekening te nemen. De overblijvende assumpties 5.2.1 en 5.2.2 liggen zowel bij de SAH als RTSAH aan de basis.

Door straalterminatie in rekening te nemen in de kostmetriek, bevoordelen we het splitsen van voxels die niet of nauwelijks zichtbaar zijn voor elkaar. Dit leidt

5. MATHEMATISCH FRAMEWORK

tot een verschillende en correctere rangorde van de splitsingsvlakken wat resulteert in fundamenteel verschillende, meer verfijnde en zoals zal blijken meer kwalitatieve kd-bomen in vergelijking met deze bekomen met de standaard SAH.

De RTSAH [IH11] is origineel geïntroduceerd om de doorloopvolgorde van de voxels voor schaduwstralen te bepalen in BVHs en BSPs. In sectie 5.3.1, passen we deze RTSAH aan om kd-boom acceleratiestructuren te bouwen die de kost voor het volgen van primaire stralen verminderen.

5.3.1 Kostfunctie

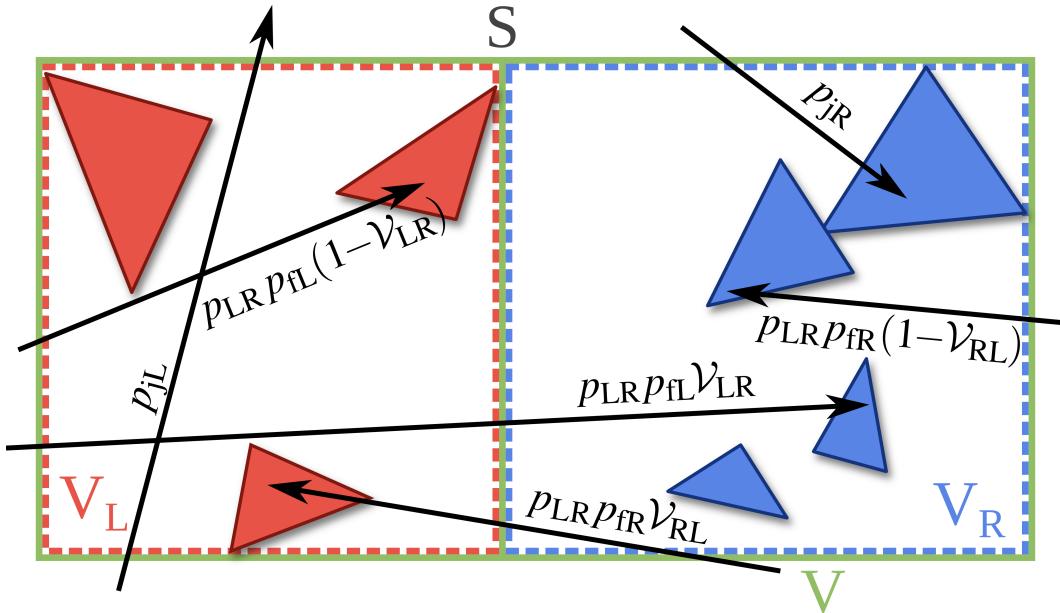
De SAH neemt de aanwezigheid van de geometrische primitieven niet in rekening in de probabiliteiten p_L^{RT} (5.2.3) en p_R^{RT} (5.2.4). Hierdoor maakt de SAH geen onderscheid tussen stralen die beide kindvoxels doorkruisen met het wel of niet raken van een geometrische primitief binnen de kindvoxel die eerst doorkruist wordt. De RTSAH breidt de SAH uit door de mogelijke terminatie van stralen in rekening te nemen en maakt meer bepaald een onderscheid tussen de volgende stralen die de oudervoxel doorkruisen:

1. stralen die enkel de linkse kindvoxel doorkruisen
2. stralen die eerst de linkse en dan de rechtse kindvoxel doorkruisen maar reeds beëindigd worden in de linkse kindvoxel
3. stralen die eerst de linkse en dan de rechtse kindvoxel doorkruisen zonder te eindigen in de linkse kindvoxel
4. stralen die eerst de rechtse en dan de linkse kindvoxel doorkruisen maar reeds beëindigd worden in de rechtse kindvoxel
5. stralen die eerst de rechtse en dan de linkse kindvoxel doorkruisen zonder te eindigen in de rechtse kindvoxel
6. stralen die enkel de rechtse kindvoxel doorkruisen

De RTSAH kostfunctie is gedefinieerd als:

$$\begin{aligned}
 \mathcal{C}_{RTSAH}(S : V \rightarrow \{V_L, V_R\}) &= \mathcal{C}_{trav}^{RT} + p_{jL}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_L) + p_{jR}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_R) + p_{LR}^{RT} \cdot \\
 &\left(p_{fL}^{RT} \left(\mathcal{C}_{visit}^{RT}(V_L) + \mathcal{V}_{LR}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_R) \right) + p_{fR}^{RT} \left(\mathcal{C}_{visit}^{RT}(V_R) + \mathcal{V}_{RL}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_L) \right) \right)
 \end{aligned} \tag{5.3.1}$$

In vergelijking met de SAH, zijn de kostcomponenten hetzelfde gebleven en zijn er meerdere probabiliteitcomponenten bijgekomen:

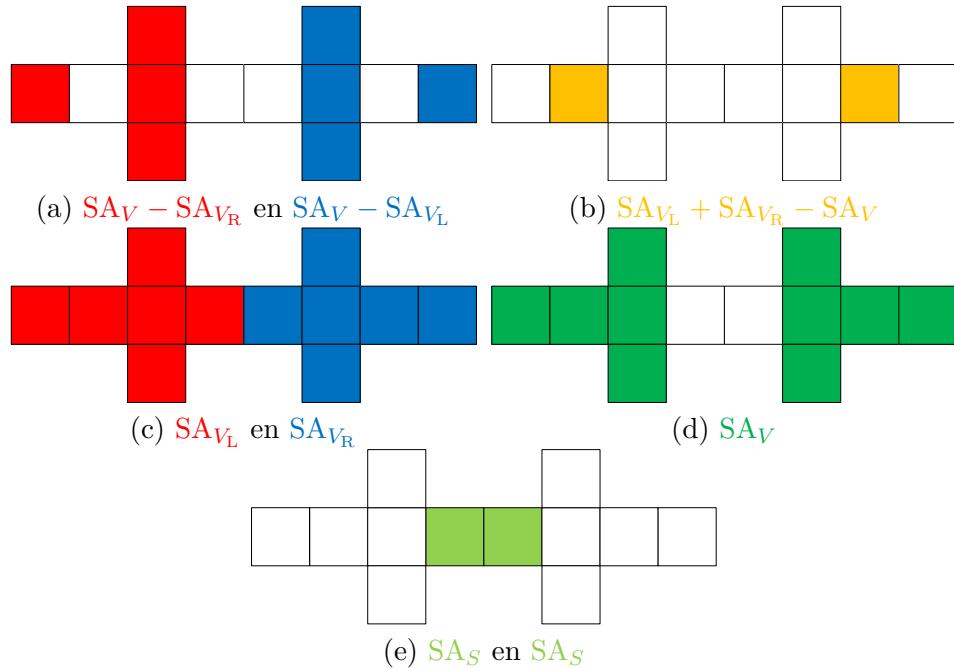


FIGUUR 5.2: De types van stralen beschouwd door de RTSAH: stralen die slechts één kindvoxel doorkruisen (met probabiliteiten p_{jL}^{RT} en p_{jR}^{RT}), stralen die beide kindvoxels doorkruisen met het wel (met probabiliteiten $p_{LR}^{RT} \cdot p_{fL}^{RT} \cdot (1 - V_{LR}^{RT})$ en $p_{LR}^{RT} \cdot p_{fR}^{RT} \cdot (1 - V_{RL}^{RT})$) of niet raken (met probabiliteiten $p_{LR}^{RT} \cdot p_{fL}^{RT} \cdot V_{LR}^{RT}$ en $p_{LR}^{RT} \cdot p_{fR}^{RT} \cdot V_{RL}^{RT}$) van een geometrisch primitief binnen de kindvoxel die eerst doorkruist wordt. De RTSAH reduceert tot de SAH indien $V_{LR}^{RT} = V_{RL}^{RT} = 1$.

- p_{jL}^{RT} (p_{jR}^{RT}) drukt de kans uit om enkel de linkse (rechtse) kindvoxel te doorkruisen (gegeven dat de oudervoxel doorkruist wordt).
- p_{LR}^{RT} drukt de kans uit om beide kindvoxels te doorkruisen (gegeven dat de oudervoxel doorkruist wordt).
- p_{fL}^{RT} (p_{fR}^{RT}) drukt de kans uit om eerst de linkse (rechtse) kindvoxel te doorkruisen gegeven dat beide kindvoxels doorkruist worden.
- V_{LR}^{RT} (V_{RL}^{RT}) drukt de fractie van stralen uit die eerst de linkse (rechtse) kindvoxel en vervolgens de rechtse (linkse) kindvoxel doorkruisen en die het splitsingsvlak bereiken zonder beëindigd te worden door een geometrisch primitief in de linkse (rechtse) kindvoxel.

Figuur 5.2 illustreert deze verschillende types van stralen met hun bijhorende probabiliteiten.

5. MATHEMATISCH FRAMEWORK



FIGUUR 5.3: Het verband tussen de probabiliteiten en de vlakken van de kindvoxels. De tellers van (a) p_{jL}^{RT} en p_{jR}^{RT} (b) p_{LR}^{RT} (c) p_L^{RT} en p_R^{RT} (d) De bijhorende noemer (e) Het splitsingsvlak (aan weerszijden).

We bekomen uitdrukkingen voor de probabiliteiten p_{jL}^{RT} , p_{jR}^{RT} en p_{LR}^{RT} door gebruik te maken van p_L^{RT} (5.2.3) en p_R^{RT} (5.2.4):

$$\begin{aligned} p_{jL}^{RT} &= p(\text{alleen } V_L \text{ doorkruist} \mid V \text{ doorkruist}) \\ &= 1 - p_R^{RT} \end{aligned} \quad (5.3.2)$$

$$\begin{aligned} p_{LR}^{RT} &= p(V_L \text{ en } V_R \text{ doorkruist} \mid V \text{ doorkruist}) \\ &= 1 - p_{jL}^{RT} - p_{jR}^{RT} \\ &= p_L^{RT} + p_R^{RT} - 1 \end{aligned} \quad (5.3.3)$$

$$\begin{aligned} p_{fL}^{RT} &= p(V_L \text{ eerst doorkruist} \mid (V_L \text{ en } V_R \text{ doorkruist} \mid V \text{ doorkruist})) \\ &= p(V_L \text{ eerst doorkruist, dan } V_R \mid V \text{ doorkruist}) \\ &= \frac{1}{2} \end{aligned} \quad (5.3.4)$$

Dit is analoog voor p_{jR}^{RT} en p_{fR}^{RT} . Figuur 5.3 geeft het verband tussen deze probabiliteiten en de vlakken van de kindvoxels.

De laatste gelijkheid (5.3.4) is eenvoudig aan te tonen. Gegeven SA_S de oppervlakte van het splitsingsvlak gepositioneerd in S :

$$SA_S = \frac{1}{2}(SA_{V_L} + SA_{V_R} - SA_V) \quad (5.3.5)$$

5.3. Ray Termination Surface Area Heuristic

dan kan nu de conditionele p_{fL}^{RT} uitgedrukt worden in functie van zijn niet-conditionele variant:

$$p(V_L \text{ eerst doorkruist, dan } V_R) = \frac{SA_{V_L} - SA_S}{SA_V} \quad (5.3.6)$$

als:

$$\begin{aligned} p(V_L \text{ eerst doorkruist, dan } V_R) &= \sum_n p(V_L \text{ eerst doorkruist, dan } V_R | X_n) \cdot p(X_n) \\ &= p_{jL}^{RT} + p_{fL}^{RT} \cdot p_{LR}^{RT} \\ &= \left(1 - \frac{SA_{V_R}}{SA_V}\right) + p_{fL}^{RT} \cdot \left(\frac{SA_{V_L} + SA_{V_R}}{SA_V} - 1\right) \\ &= \left(\frac{SA_{V_L} - 2 \cdot SA_S}{SA_V}\right) + p_{fL}^{RT} \cdot \left(\frac{2 \cdot SA_S}{SA_V}\right) \\ &= \left(\frac{SA_{V_L} + 2(p_{fL}^{RT} - 1)SA_S}{SA_V}\right) \end{aligned}$$

Uit (5.3.6) volgt nu dat $p_{fL}^{RT} = p_{fR}^{RT} = 1/2$. Dit is ook in te zien op basis van de assumpties 5.2.1 en 5.2.2. Voor elke straal die van buiten V vertrekt, V_L eerst doorkruist en dan V_R , bestaat er een tweede straal die van buiten V vertrekt, V_R eerst doorkruist en dan V_L . Dit door de richting van deze eerste straal te inverteren.

De visibiliteitsprobabiliteiten tussen de kindvoxels zijn gedefinieerd als:

$$\mathcal{V}_{LR}^{RT} = p(\text{geen hit in } V_L | (V_L \text{ eerst doorkruist, dan } V_R | V \text{ doorkruist})) \quad (5.3.7)$$

$$\mathcal{V}_{RL}^{RT} = p(\text{geen hit in } V_R | (V_R \text{ eerst doorkruist, dan } V_L | V \text{ doorkruist})) \quad (5.3.8)$$

Merk op dat de RTSAH kost gelijk is aan de SAH kost indien $\mathcal{V}_{LR}^{RT} = \mathcal{V}_{RL}^{RT} = 1$:

$$\begin{aligned} \mathcal{C}_{RTSAH}(S : V \rightarrow \{V_L, V_R\}) &= \mathcal{C}_{trav}^{RT} + p_{jL}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_L) + p_{jR}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_R) + \\ &\quad p_{LR}^{RT} \left(p_{fL}^{RT} \left(\mathcal{C}_{visit}^{RT}(V_L) + \mathcal{C}_{visit}^{RT}(V_R) \right) + p_{fR}^{RT} \left(\mathcal{C}_{visit}^{RT}(V_R) + \mathcal{C}_{visit}^{RT}(V_L) \right) \right) \\ &= \mathcal{C}_{trav}^{RT} + p_{jL}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_L) + p_{jR}^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_R) + p_{LR}^{RT} \left(\mathcal{C}_{visit}^{RT}(V_L) + \mathcal{C}_{visit}^{RT}(V_R) \right) \\ &= \mathcal{C}_{trav}^{RT} + p_L^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_L) + p_R^{RT} \cdot \mathcal{C}_{visit}^{RT}(V_R) \\ &= \mathcal{C}_{SAH}(S : V \rightarrow \{V_L, V_R\}) \end{aligned}$$

De kost van het niet splitsen van een voxel is hetzelfde als voor de SAH (5.2.5).

Lege-ruimte-bonus. De lege-ruimte-bonus (5.2.6) en zijn onderliggende motivatie kan eveneens toegepast worden op de RTSAH. De aangepaste RTSAH wordt dan

$$\begin{aligned} \mathcal{C}'_{\text{RTSAH}}(S : V \rightarrow \{V_L, V_R\}) \\ = \mathcal{C}_{\text{trav}}^{\text{RT}} + \lambda(V_L, V_R) \cdot \left(p_{jL}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_L) + p_{jR}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_R) + p_{LR}^{\text{RT}} \cdot \right. \\ \left. \left(p_{fL}^{\text{RT}} \left(\mathcal{C}_{\text{visit}}^{\text{RT}}(V_L) + \mathcal{V}_{LR}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_R) \right) + p_{fR}^{\text{RT}} \left(\mathcal{C}_{\text{visit}}^{\text{RT}}(V_R) + \mathcal{V}_{RL}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{RT}}(V_L) \right) \right) \right) \end{aligned} \quad (5.3.9)$$

5.4 Voxel Volume Heuristic

De VVH [WGS04] is een bouwheuristiek voor het bouwen van kd-boom photon-mappen met als doel de kost voor het uitvoeren van k-nearest neighbor query's te verminderen. De VVH kent aan elke kandidaatvoxel een kost toe die gelijk is aan de kost om deze voxel te bezoeken vermenigvuldigd met de kans om deze voxel effectief te bezoeken in de context van een k-nearest neighbor query (kNN). Om deze kansen te kunnen berekenen voor elke kandidaatvoxel, maakt de VVH een aantal assumpties over de query's. Deze assumpties worden besproken in sectie 5.4.1. De VVH kostfunctie wordt besproken in sectie 5.4.2. In sectie 5.4.3 introduceren we de computationeel goedkopere VVH1 variant van de VVH heuristiek.

5.4.1 Assumpties

De VVH maakt de volgende drie assumpties over de query's:

Assumptie 5.4.1 (Straal van de query's). *De straal van de k-nearest neighbor query's is een vooraf bepaalde constante R_{\max} . (Deze maximale querystraal kan enkel afnemen tijdens een photon lookup.)*

Assumptie 5.4.2 (Locatie van de query's op oppervlakken). *De querylocaties zijn uniform verdeeld over de oppervlakken van de geometrische primitieven binnen het volume van de voxel uitgebreid met R_{\max} in alle richtingen.*

Assumptie 5.4.3 (Locaties van de query's in het volume). *De querylocaties zijn uniform verdeeld binnen het volume van de voxel uitgebreid met R_{\max} in alle richtingen.*

In het photonmapping algoritme wordt voor de berekening van de uitgestraalde radiantie in een hitpunt gelegen op een oppervlak met een BSDF met enkel perfect speculaire componenten, de ray tracing verdergezet in de perfect speculaire richting. Door te veronderstellen in assumptie 5.4.2 dat de locatie van de k-nearest neighbor query's uniform verdeeld is over deze oppervlakken, behandelen we geometrische primitieven met een BSDF met enkel perfect speculaire componenten (reflexief of

transmissief) niet apart. Dit is geen probleem indien alle materialen die voorkomen in de scene diffuus zijn. In de voxels waarin oppervlakken met BSDFs met enkel perfect speculaire componenten aanwezig zijn, wordt de kost per k-nearest neighbor query overschat.

De laatste assumptie 5.4.3 is ingevoerd om een snel bouwalgoritme te bekomen. Er is geen verbinding tussen de photons en de geometrische primitieven omdat beiden in een aparte datastructuur worden opgeslagen. Daarom is het computationeel duur om de aanwezigheid van de geometrische primitieven in rekening te nemen bij het bouwen van de photonmap. De resultaten van Wald et al. [WGS04] tonen aan dat deze strenge assumptie in de praktijk goed werkt.

5.4.2 Kostfunctie

De VVH kostfunctie geeft een schatting van de kost van het splitsen van een voxel V met een splitsingsvlak geplaatst in S in een linkse V_L en rechtse V_R kindvoxel als:

$$\mathcal{C}_{\text{VVH}}(S : V \rightarrow \{V_L, V_R\}) = \mathcal{C}_{\text{check}}^{\text{kNN}} + p_L^{\text{kNN}} \cdot \mathcal{C}_{\text{visit}}^{\text{kNN}}(V_L) + p_R^{\text{kNN}} \cdot \mathcal{C}_{\text{visit}}^{\text{kNN}}(V_R) \quad (5.4.1)$$

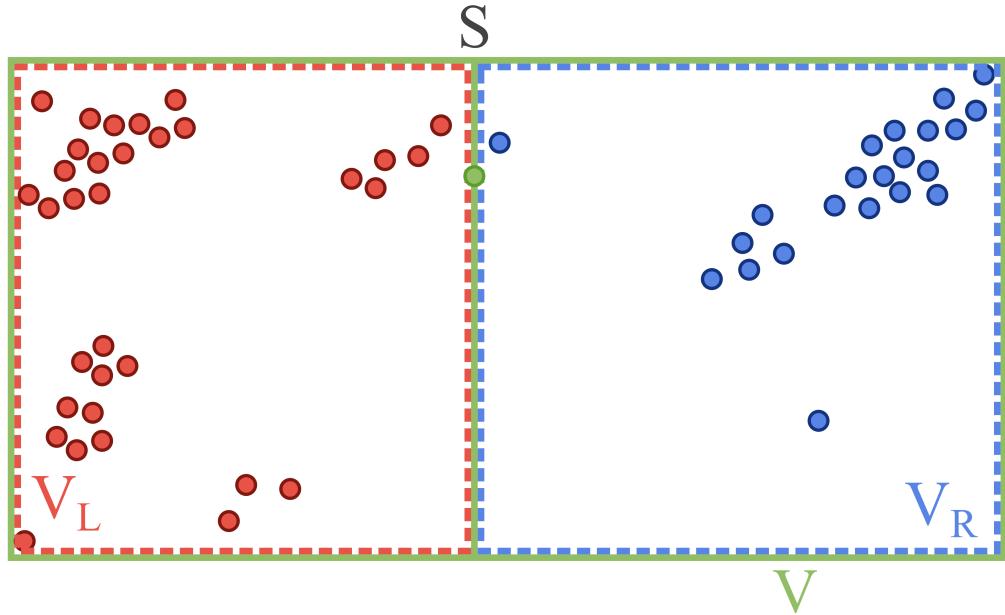
Figuur 5.4 illustreert de opsplitsing van een voxel V en de overlappende photons. De heuristiek bestaat uit de volgende componenten:

- $\mathcal{C}_{\text{check}}^{\text{kNN}}$, de kost om de oudervoxel V te controleren in een kNN query. Omdat photons puntdata zijn en er steeds gesplitst moet worden op een photon (voor deze kostfunctie), bevat elke oudervoxel (inwendige knoop) één photon.
- $\mathcal{C}_{\text{visit}}^{\text{kNN}}(V_L)$ ($\mathcal{C}_{\text{visit}}^{\text{kNN}}(V_R)$), de kost om de linkse (rechtse) kindvoxel te controleren in een kNN query.
- p_L^{kNN} (p_R^{kNN}), de kans om de linkse (rechtse) kindvoxel te moeten controleren gegeven de oudervoxel V te moeten controleren in een kNN query.

$\mathcal{C}_{\text{visit}}^{\text{kNN}}(V)$ is gedefinieerd als de kost om elke photon in de verzameling photons P_V in de voxel V te controleren op nabijheid in een kNN query:

$$\mathcal{C}_{\text{visit}}^{\text{kNN}}(V) = \mathcal{C}_{\text{check}}^{\text{kNN}} \cdot |P_V| \quad (5.4.2)$$

p_L^{kNN} (p_R^{kNN}) is de conditionele probabiliteit om de linkse (rechtse) kindvoxel te controleren gegeven de oudervoxel te moeten controleren in een kNN query. Door de



FIGUUR 5.4: De opsplitsing van een voxel V en de overlappende photons met een splitsingsvlak positioneerd in S in een linkse V_L en rechtse V_R kindvoxel.

assumpties 5.4.1 en 5.4.2 zijn deze probabiliteiten exact gegeven als:

$$\begin{aligned} p_L^{\text{kNN}} &= p(V_L \text{ te controleren} \mid V \text{ te controleren}) \\ &= \frac{\text{SA}_{G_{V_L} \pm R_{\max}}}{\text{SA}_{G_V \pm R_{\max}}} \end{aligned} \quad (5.4.3)$$

$$\begin{aligned} p_R^{\text{kNN}} &= p(V_R \text{ te controleren} \mid V \text{ te controleren}) \\ &= \frac{\text{SA}_{G_{V_R} \pm R_{\max}}}{\text{SA}_{G_V \pm R_{\max}}} \end{aligned} \quad (5.4.4)$$

Hierbij is $\text{SA}_{G_{V_k} \pm R_{\max}}$ de som van de oppervlakten van de geometrische primitieven in de voxel V_k uitgebreid met de querystraal R_{\max} in alle richtingen. Assumptie 5.4.3 vereenvoudigt deze probabiliteiten tot:

$$p_L^{\text{kNN}} = \frac{\text{VOL}_{G_{V_L} \pm R_{\max}}}{\text{VOL}_{G_V \pm R_{\max}}} \quad (5.4.5)$$

$$\approx \frac{\prod_{i=x,y,z} (V_{L,i,\max} - V_{L,i,\min} + 2 \cdot R_{\max})}{\prod_{i=x,y,z} (V_{i,\max} - V_{i,\min} + 2 \cdot R_{\max})} \quad (5.4.6)$$

$$p_R^{\text{kNN}} = \frac{\text{VOL}_{G_{V_R} \pm R_{\max}}}{\text{VOL}_{G_V \pm R_{\max}}} \quad (5.4.7)$$

$$\approx \frac{\prod_{i=x,y,z} (V_{R,i,\max} - V_{R,i,\min} + 2 \cdot R_{\max})}{\prod_{i=x,y,z} (V_{i,\max} - V_{i,\min} + 2 \cdot R_{\max})} \quad (5.4.8)$$

Hierbij is $\text{VOL}_{G_{V_k} \pm R_{\max}}$ het volume van de voxel V_k uitgebreid met de query straal R_{\max} in alle richtingen. Indien V_k slechts een punt omspant, is $\text{VOL}_{G_{V_k} \pm R_{\max}}$ het volume van de bol met V_k als centrum en straal R_{\max} . Het volume $\text{VOL}_{G_{V_k} \pm R_{\max}}$ wordt omwille van computationele redenen benaderd door het volume van de voxel V_k waarbij elke zijde (met lengte $V_{k,i,\max} - V_{k,i,\min}$ in de primaire richting i) is verlengd met tweemaal de querystraal R_{\max} . Indien V_k slechts een punt omspant, is dit het volume van de kubus met V_k als zwaartepunt en zijde $2 \cdot R_{\max}$.

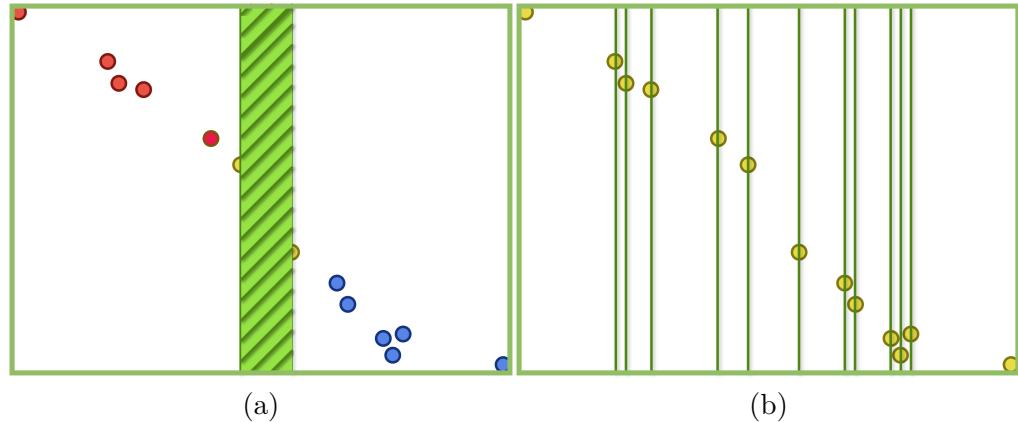
Deze laatste benadering heeft zoals Wald et al. [WGS04] aanhalen, het bijkomende voordeel het balanceren van de kd-boom af te wegen (in plaats van altijd te balanceren zoals bij de BH). Indien de $2 \cdot R_{\max}$ term domineert, zal het balanceren van de boom de voorkeur verdienen. Voxel V is dan zodanig klein dat de gemiddelde k-nearest neighbor query elke photon in de voxel zal controleren. Dit pleit voor een balansering van de kd-boom. Indien de $2 \cdot R_{\max}$ term verwaarloosbaar is, zal een evenwichtige opdeling van de photondensiteit de voorkeur verdienen.

Opmerking 5.1 (Variabele maximale querystraal). Zoals Wald et al. [WGS04] aanhalen is het eveneens mogelijk de maximale straal van een k-nearest neighbor query te schatten op basis van het volume van de voxel en het aantal aanwezige photons in de voxel. Op deze manier varieert de maximale querystraal over de verschillende gebieden van de scène. Voor voxels met een hoge densiteit aan photons, zal de kans kleiner zijn om naburige voxels te moeten controleren voor de gemiddelde query. Om dit in de VVH kostfunctie (5.4.1) uit te drukken, kan een kindvoxel met een kleinere maximale querystraal $R'_{\max} \leq R_{\max}$ uitgebreid worden. Op basis van hun resultaten besluiten Wald et al. [WGS04] dat een vaste maximale querystraal volstaat.

De VVH heuristiek zal een voxel steeds opsplitsen opdat elke knoop van de kd-boom ten hoogste één photon bevat. De VVH kostfunctie stijgt of daalt monotoon in het gebied tussen twee opeenvolgende photons langs dezelfde as (geïllustreerd in Figuur 5.5a). Binnen zo'n gebied, zijn de probabiliteiten p_L^{kNN} en p_R^{kNN} beiden lineaire functies van de positie van het splitsingsvlak S . Dit betekent dat we enkel de splitsingsvlakken door de photons in rekening moeten nemen om het beste splitsingsvlak volgens onze kostmetriek (5.4.1) te bekomen (zie Figuur 5.5b).

Opmerking 5.2 (Checkkost). De kost om een photon op nabijheid te testen in een k-nearest neighbor query, $C_{\text{check}}^{\text{kNN}}$, is overbodig en kan weggedeeld worden in de kostfunctie (5.4.1). Wanneer de kostfunctie (5.4.1) in een groter geheel gebruikt wordt (zie sectie 5.5), mag deze component echter niet zomaar geëlimineerd worden wanneer kostfuncties onderling vergelijkbaar moeten zijn.

Opmerking 5.3 (Photon invloedssfeer). We zouden photons ook kunnen voorstellen door hun invloedssfeer: een bol met de photon in het centrum en R_{\max} als straal. Op deze manier kunnen we op de minimale en maximale grenzen van deze bol splitsen analoog aan de SAH en RTSAH voor geometrische primitieven. Dit betekent dat



FIGUUR 5.5: Splitskandidaten van de VVH. (a) Toont een gebied (groen) waar de VVH kostfunctie monotoon stijgend of dalend is. (b) De eindige verzameling van splitsingsvlakken (groen) in één richting die beschouwd moeten worden om het optimale splitsingsvlak te bekomen volgens de VVH kostfunctie.

photons in beide kindvoxels opgeslagen moeten worden indien het splitsingsvlak binnen hun bijhorende bol gelegen is. Voor de correcte uitvoering van een k-nearest neighbor query zal daarom het meerraads in rekening nemen van dezelfde photons explicet vermeden moeten worden. Deze performantiekost willen we niet betalen, daarom is de VVH zoals ze is.

5.4.3 Voxel Volume Heuristic 1

De Voxel Volume Heuristic 1 (VVH1) is identiek aan de VVH maar splitst enkel in de primaire richting waarin de voxel zijn grootste omvang heeft. De BH die de photonmap perfect balanceert, splitst ook steeds in de primaire richting waarin de voxel zijn grootste omvang heeft. Zoals in de praktijk zal blijken moet deze BH niet veel onderdoen voor de VVH voor indirecte photons wat de totale rendertijd betreft. Daarom lijkt het ons voordelig om de bouwtijd van de VVH te verminderen door slechts één primaire richting (in plaats van drie) te beschouwen en nog steeds kwalitatieve photonmappen te bekomen.

In de praktijk (zie bijvoorbeeld [PH10]) wordt het aantal richtingen die de SAH moet beschouwen eveneens verminderd. Indien de kost van het niet-splitsen groter is dan de beste kost voor het wel splitsen voor één richting, worden de overige primaire richtingen niet meer onderzocht. Eenzelfde strategie kan eveneens voor de RTSAH gebruikt worden.

Opmerking 5.4 (Implementatie). Door slechts één primaire richting in rekening te nemen en steeds te splitsen tot elke knoop ten hoogste één photon bevat, kan de kostfunctie van de VVH1 (5.4.1) zeer efficiënt geïmplementeerd worden. Een aantal termen van de kostfunctie blijven namelijk steeds hetzelfde voor elk van de mogelijke

splitsingsvlakken en kunnen dus weggelaten worden in de implementatie. Voor de VVH is dit niet mogelijk aangezien de kosten voor verschillende splitsrichtingen onderling vergelijkbaar moeten blijven.

5.5 Hybrid Heuristic

De HH kent aan elke kandidaatvoxel een kost toe die gelijk is aan de kost om deze voxel te bezoeken vermenigvuldigd met de kans om deze voxel effectief te bezoeken in de gecombineerde context van ray tracing (RT) en k-nearest neighbor query's (kNN). Doordat k-nearest neighbor query's vertrekken vanuit een hitpunt van een straal met een geometrisch primitief, is het nu belangrijk om straalterminatie in rekening te nemen. Dit verklaart ook onze omweg via de RTSAH (zie sectie 5.3).

Om de kansen te kunnen berekenen voor elke kandidaatvoxel, maakt de HH een aantal assumpties over de stralen en query's. Deze assumpties worden besproken in sectie 5.5.1. De HH kostfuncties worden besproken in sectie 5.5.2.

5.5.1 Assumpties

Aangezien de RTSAH en VVH heuristiek deel uit maken van de HH, gebruiken we dezelfde onderliggende assumpties. Als een gevolg van deze assumpties zal de ray tracing altijd gevolgd worden door een k-nearest neighbor query indien de straal een geometrisch primitief raakt.

Verder zijn indirecte photons de enige soort van photons die we onderbrengen in de hybride kd-boom. De kostfuncties die in sectie 5.5.2 aan bod zullen komen, kunnen eenvoudig worden aangepast om k-nearest neighbor query's voor verschillende photonsoorten in rekening te nemen. Dit betekent dat ofwel in de datastructuur voor de kd-boom knopen ofwel in de datastructuur voor de photons een onderscheid gemaakt moet worden tussen de verschillende photonsoorten. De eerste oplossing heeft een nadelig effect op het doorlopen van de kd-boom voor zowel de ray tracing als de k-nearest neighbor query's ten gevolge van een slechtere cache coherentie. De tweede oplossing zal de kost om een photon te controleren, $\mathcal{C}_{\text{check}}^{\text{kNN}}$, verhogen.

5.5.2 Kostfuncties

Kostmodel. Alvorens de kostfuncties te introduceren, moeten we eerst het kostmodel herdefiniëren voor de gecombineerde context van ray tracing en k-nearest neighbor query's. De kost van het bezoeken van een voxel, $\mathcal{C}_{\text{visit}}(V)$, bestaat uit een (gewogen) bijdrage voor de ray tracing en een (gewogen) bijdrage voor de k-nearest neighbor query's die niet los van elkaar staan. De bijdrage voor de ray tracing is gelijk aan de kost om een straal op intersectie te testen met alle geometrische primitieven in de voxel, $\mathcal{C}_{\text{visit}}^{\text{RT}}(V)$ (5.2.2). We beschouwen twee soorten k-nearest neighbor query's: *interne query's* vertrekken vanuit de voxel zelf en *externe query's* zijn niet-voltooide query's afkomstig van de naburige voxels. De bijdrage voor deze twee soorten query's

5. MATHEMATISCH FRAMEWORK

is steeds gelijk aan de kans dat het soort query effectief voorkomt, vermenigvuldigd met de kost om een query uit te voeren. De kans dat een interne query voorkomt, is gelijk aan de kans op een hit van een straal met één van de geometrische primitieven in de voxel, $p_{\text{hit}}^{\text{RT}}$. De kans dat een externe query voorkomt, is gelijk aan de kans dat een niet-voltooide query vanuit een naburige gebied de voxel bezoekt, $p_{\text{ext}}^{\text{kNN}}$. De kost om een interne of externe query uit te voeren is de kost om alle photons in de voxel te controleren, $\mathcal{C}_{\text{visit}}^{\text{kNN}}(V)$ (5.4.2).

De kostmodellen voor de verschillende contexten zijn gedefinieerd als:

$$\mathcal{C}_{\text{visit}}^{\text{RT}}(V) = \sum_{i=1}^{|\mathcal{G}_V|} \mathcal{C}_{\text{isect}}^{\text{RT}}(\mathcal{G}_V, i) \quad (5.5.1)$$

$$\mathcal{C}_{\text{visit}}^{\text{kNN}}(V) = \mathcal{C}_{\text{check}}^{\text{kNN}} \cdot |\mathcal{P}_V| \quad (5.5.2)$$

$$\mathcal{C}_{\text{int}}(V) = \mathcal{C}_{\text{visit}}^{\text{RT}}(V) + p_{\text{hit}}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{kNN}}(V) \quad (5.5.3)$$

$$\mathcal{C}_{\text{ext}}^{\text{kNN}}(V) = p_{\text{ext}}^{\text{kNN}} \cdot \mathcal{C}_{\text{visit}}^{\text{kNN}}(V) \quad (5.5.4)$$

$$\mathcal{C}_{\text{visit}}(V) = w_{\text{int}} \cdot \mathcal{C}_{\text{int}}(V) + w_{\text{ext}} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V) \quad (5.5.5)$$

$$1 = w_{\text{int}} + w_{\text{ext}} \text{ met } w_{\text{int}}, w_{\text{ext}} \geq 0 \quad (5.5.6)$$

$\mathcal{C}_{\text{int}}(\cdot)$ (5.5.3) is de kost om met een straal een voxel te bezoeken samen met de kost om een interne query uit te voeren indien deze straal een geometrisch primitief raakt binnen de voxel. $\mathcal{C}_{\text{ext}}^{\text{kNN}}(\cdot)$ (5.5.4) is de kost om een externe query verder te zetten in een voxel. $\mathcal{C}_{\text{visit}}(\cdot)$ (5.5.5) is de gewogen kost van deze twee kosten waarbij w_{int} en w_{ext} (5.5.6) de gewichten zijn om de bijdrage van interne en externe query's ten opzichte van elkaar af te wegen. De probabiliteiten zijn gedefinieerd als:

$$p_{\text{hit}}^{\text{RT}} = p(\text{kNN query start in } V) \quad (5.5.7)$$

$$= p(\text{hit in } V \mid V \text{ doorkruist}) \quad (5.5.8)$$

$$p_{\text{hitL}}^{\text{RT}} = p(\text{kNN query start in } V_L) \quad (5.5.9)$$

$$= p(\text{hit in } V_L \mid V \text{ doorkruist}) \quad (5.5.10)$$

$$p_{\text{hitR}}^{\text{RT}} = p(\text{kNN query start in } V_R) \quad (5.5.11)$$

$$= p(\text{hit in } V_R \mid V \text{ doorkruist}) \quad (5.5.12)$$

$$p_{\text{ext}}^{\text{kNN}} = p(\text{kNN query niet voltooid in } \mathcal{N}_V) \quad (5.5.13)$$

$$p_{\text{extL}}^{\text{kNN}} = p(\text{kNN query niet voltooid in } \mathcal{N}_{V_L}) \quad (5.5.14)$$

$$p_{\text{extR}}^{\text{kNN}} = p(\text{kNN query niet voltooid in } \mathcal{N}_{V_R}) \quad (5.5.15)$$

Hierbij stelt \mathcal{N}_{V_k} het naburige gebied van de voxel V_k voor van waaruit onvoltooide query's V_k kunnen bezoeken.

Opmerking 5.5 (Interne en externe notatie). Zowel interne als externe query's kunnen enkel starten vanuit een hitpunt van een straal met een geometrisch primitief. Toch gebruiken we een verschillende notatie voor het voorkomen van beide query's: $p_{\text{hit}}^{\text{RT}}$ en $p_{\text{ext}}^{\text{kNN}}$ in plaats van $p_{\text{int}}^{\text{kNN}}$ en $p_{\text{ext}}^{\text{kNN}}$ of $p_{\text{hit,int}}^{\text{RT}}$ en $p_{\text{hit,ext}}^{\text{RT}}$. We willen door middel

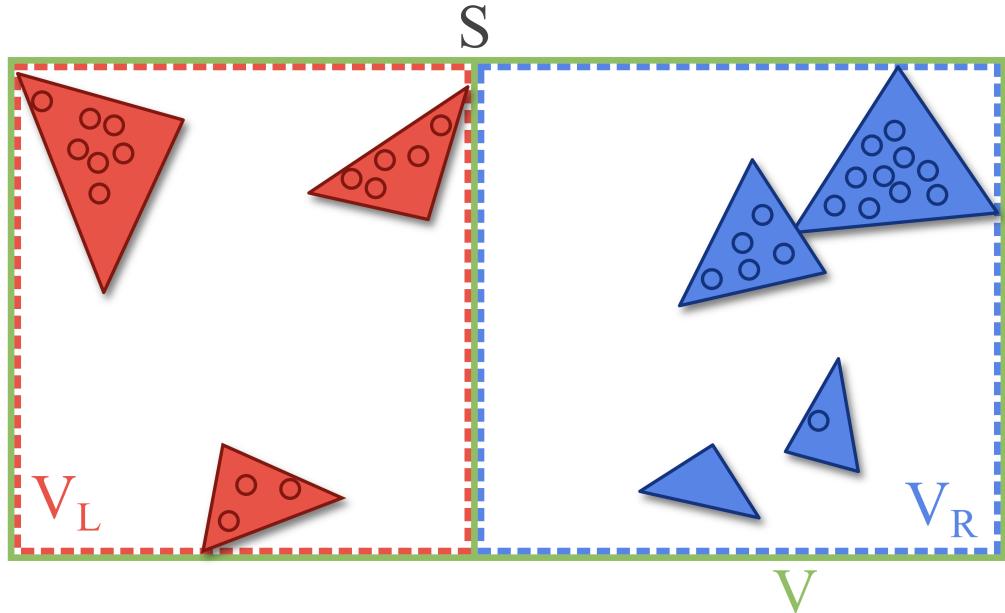
van het hit subscript explicet het interne verloop benadrukken om vervolgens een kwalitatieve benadering te zoeken. Een kwalitatieve benadering behoort tot de mogelijkheden doordat we de interne geometrische distributie van de voxel onder beschouwing kennen. In deze voxel weten we welke geometrische primitieven aanwezig zijn en hoe deze gepositioneerd zijn. Externe query's daarentegen zijn afkomstig van buiten de voxel. De geometrische distributies van de omliggende voxels zijn in het algemeen niet gekend tijdens de constructiefase van de kd-boom. Onze benadering voor $p_{\text{ext}}^{\text{kNN}}$ zal daarom onafhankelijk van deze geometrische distributies gekozen worden en zal dus minder kwalitatief (worst-case) zijn.

Splitsbeslissingen. Bij de SAH en RTSAH moet er bij elke splitsbeslissing bepaald worden of de huidige voxel wel of niet gesplitst moet worden. Bij de VVH en VVH1 wordt de voxel steeds gesplitst totdat elke knoop van de kd-boom juist één photon bevat. Voor de HH is dit iets complexer. We onderscheiden **drie situaties** met elk een aantal mogelijkheden waaruit gekozen moet worden:

1. $G_V \neq \emptyset \wedge P_V \neq \emptyset^1$
 - 1.1. Deel G_V en P_V verder op bij het opslitsen ([5.5.16](#))
 - 1.2. Drop P_V en deel G_V verder op bij het opslitsen ([5.5.17](#))
 - 1.3. Drop G_V en deel P_V verder op bij het opslitsen ([5.5.18](#))
 - 1.4. Drop G_V en P_V en stop het opslitsen ([5.5.19](#))
2. $G_V \neq \emptyset \wedge P_V = \emptyset$
 - 2.1. Deel G_V verder op bij het opslitsen ([5.3.1](#))
 - 2.2. Drop G_V en stop het opslitsen ([5.2.5](#))
3. $G_V = \emptyset \wedge P_V \neq \emptyset$
 - 3.1. Deel P_V verder op bij het opslitsen ([5.4.1](#))

Voor de *eerste situatie* zijn er vier mogelijke beslissingen. De geometrische primitieven en photons kunnen beide verder opgesplitst worden (zie [Figuur 5.6](#)). Eén van de verzamelingen kan gedropt worden op de intermediaire knoop. Het is mogelijk dat een klein aantal photons representatief is voor een groot aantal geometrische primitieven. Indien al de k-nearest neighbor query's startende vanop deze geometrische primitieven deze photons nodig hebben, heeft het geen nut deze photons verder op te delen met de geometrische primitieven. Daarnaast is het mogelijk dat op een groot geometrisch primitief zeer veel photons gelegen zijn. In dit geval is het opportuun dit

¹ P_V verwijst hier naar de verzameling photons in V die nog in rekening gebracht moeten worden bij een splitsbeslissing. Indien de photons reeds op een vroeger niveau van de kd-boom gedropt zijn, moeten er geen photons meer in rekening gebracht worden voor deze splitsbeslissing ($P_V = \emptyset$) ook al zijn de gedropte photons binnen de ruimte van de voxel V gelegen. Dit is analoog voor G_V .



FIGUUR 5.6: De opsplitsing van een voxel V en de overlappende geometrische primitieven en photons met een splitsingsvlak geplaatst in S in een linkse V_L en rechtse V_R kindvoxel.

geometrisch primitief niet verder op te delen met de photons om de duplicatie van dit geometrisch primitief (indien beide kindvoxels overlappen) in te perken. Verder kunnen zowel de geometrische primitieven als photons gedropt worden. De *tweede situatie* komt overeen met het gebruik van een bouwheuristiek voor een kd-boom acceleratiedatastructuur (bv.: SAH, RTSAH, etc.). De *derde situatie* komt overeen met het gebruik van een bouwheuristiek voor een kd-boom photonmap (bv.: VVH, VHH1).

Kostfuncties. Gegeven welke van de drie situaties van toepassing is voor de huidige voxel, moet de kost van elk van de bijhorende beslissingen berekend worden. De beslissing die resulteert in de kleinste kost wordt vervolgens toegepast.

Om de juiste keuze te maken tussen deze vier mogelijkheden is het belangrijk dat de overeenkomstige kosten onderling vergelijkbaar zijn. De kostfuncties voor de

eerste situatie zijn gedefinieerd als:

$$\begin{aligned} \mathcal{C}_{\text{HRTSAH}}(S : V \rightarrow \{V_L, V_R\}) \\ = w_{\text{int}} \cdot \left(\mathcal{C}_{\text{trav}}^{\text{RT}} + p_{jL}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_L) + p_{jR}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_R) + p_{LR}^{\text{RT}} \right. \\ \left. \left(p_{fL}^{\text{RT}} \left(\mathcal{C}_{\text{int}}(V_L) + \mathcal{V}_{LR}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_R) \right) + p_{fR}^{\text{RT}} \left(\mathcal{C}_{\text{int}}(V_R) + \mathcal{V}_{RL}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_L) \right) \right) \right) \\ + w_{\text{ext}} \cdot \left(\frac{\mathcal{N}_{V_L}}{\mathcal{N}_V} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V_L) + \frac{\mathcal{N}_{V_R}}{\mathcal{N}_V} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V_R) \right) \end{aligned} \quad (5.5.16)$$

$$\begin{aligned} \mathcal{C}_{P_V \text{ drop}}(S : V \rightarrow \{V_L, V_R\}) \\ = w_{\text{int}} \cdot \left(\mathcal{C}_{\text{RTSAH}}(S : V \rightarrow \{V_L, V_R\}) + p_{\text{hit}}^{\text{RT}} \cdot \mathcal{C}_{\text{visit}}^{\text{kNN}}(V) \right) \\ + w_{\text{ext}} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V) \end{aligned} \quad (5.5.17)$$

$$\begin{aligned} \mathcal{C}_{G_V \text{ drop}}(S : V \rightarrow \{V_L, V_R\}) \\ = w_{\text{int}} \cdot \left(\mathcal{C}_{\text{visit}}^{\text{RT}}(V) + p_{\text{hit}}^{\text{RT}} \cdot \mathcal{C}_{\text{VVH,int}}(S : V \rightarrow \{V_L, V_R\}) \right) \\ + w_{\text{ext}} \cdot p_{\text{ext}}^{\text{kNN}} \cdot \mathcal{C}_{\text{VVH,ext}}(S : V \rightarrow \{V_L, V_R\}) \end{aligned} \quad (5.5.18)$$

$$\begin{aligned} \mathcal{C}_{\text{no split}}(S : V \rightarrow \{V_L, V_R\}) \\ = \mathcal{C}_{\text{visit}}(V) \end{aligned} \quad (5.5.19)$$

Hierbij is de Hybride RTSAH (HRTSAH), $\mathcal{C}_{\text{HRTSAH}}(\cdot)$ (5.5.16), de hybride variant van de RTSAH, $\mathcal{C}_{\text{RTSAH}}(\cdot)$ (5.3.1). Deze HRTSAH maakt gebruik van $\mathcal{C}_{\text{int}}(\cdot)$ (5.5.3) in plaats van $\mathcal{C}_{\text{visit}}^{\text{RT}}(\cdot)$ (5.2.2) als kost om een voxel te bezoeken met een straal. De ray tracing kost is namelijk uitgebreid met de bijdrage van interne query's. Het opsplitsen van de voxel beïnvloedt niet alleen de kost van de interne query's maar ook van de externe query's. De kost voor externe query's wordt in het laatste gedeelte van (5.5.16) in rekening genomen. Hierbij stellen $\frac{\mathcal{N}_{V_L}}{\mathcal{N}_V}$ en $\frac{\mathcal{N}_{V_R}}{\mathcal{N}_V}$ de kansen voor dat een externe query verdergezet wordt in de respectievelijk linkse en rechtse kindvoxel.

$\mathcal{C}_{P_V \text{ drop}}(\cdot)$ (5.5.17) is $\mathcal{C}_{\text{RTSAH}}(\cdot)$ (5.3.1) gecombineerd met de kost om k-nearest query's uit te voeren gegeven dat al de photons P_V op de inwendige knoop horende bij V worden opgeslagen. Vandaar dat zowel de interne als externe querykost betrekking hebben op de voxel horende bij de inwendige knoop. Hierbij zorgt $p_{\text{hit}}^{\text{RT}}$ (5.5.7) voor de koppeling tussen de bijdrage van de ray tracing en de bijdrage van interne query's. Merk op dat we de hitprobabiliteit voor de voxel V gebruiken en niet de aparte hitprobabiliteiten van de kindvoxels, aangezien de photons niet verder opgesplitst worden.

$\mathcal{C}_{G_V \text{ drop}}(\cdot)$ (5.5.18) is $\mathcal{C}_{\text{VVH}}(\cdot)$ (5.4.1) gecombineerd met de kost om een straal op intersectie te testen gegeven dat al de geometrische primitieven G_V op de inwendige knoop horende bij V worden opgeslagen. Hierbij dienen we de VVH op te splitsen in de bijdrage voor interne, $\mathcal{C}_{\text{VVH,int}}(\cdot)$, en externe query's, $\mathcal{C}_{\text{VVH,ext}}(\cdot)$. Hierbij zorgt

5. MATHEMATISCH FRAMEWORK

$p_{\text{hit}}^{\text{RT}}$ (5.5.7) voor de koppeling tussen de bijdrage van de ray tracing en de bijdrage van interne query's. Merk op dat we de hitprobabiliteit voor de voxel V gebruiken en niet de aparte hitprobabiliteiten van de kindvoxels, aangezien de geometrische primitieven niet verder opgesplitst worden.

Opmerking 5.6 (Interne en externe VVH). De originele VVH hoeft in zijn probabiliteiten (5.4.5) en (5.4.7) geen onderscheid te maken tussen het volume van de voxel zelf waar interne query's kunnen ontstaan en het volume buiten de voxel waar externe query's kunnen voorkomen. De HH moet de VVH echter wel opsplitsen in een kost voor de interne en externe query's omwille van de verschillende gewichten die hieraan toegekend worden. Een implementatie hoeft de originele VVH in kleine mate aan te passen om beide bijdragen tegelijkertijd in rekening te nemen.

$\mathcal{C}_{\text{no split}}(\cdot)$ (5.5.19) is de hybride variant van $\mathcal{C}_{\text{no split}}^{\text{RT}}(\cdot)$ (5.2.5) zoals ook $\mathcal{C}_{\text{visit}}(\cdot)$ (5.5.5) de hybride variant is van $\mathcal{C}_{\text{visit}}^{\text{RT}}(\cdot)$ (5.2.2). Indien de voxel V onder beschouwing niet opgesplitst wordt, worden zowel G_V als P_V gedropt.

De kostfuncties voor de tweede en derde situatie zijn zoals eerder aangehaald analoog aan de kostfuncties voor respectievelijk een kd-boom acceleratiestructuur en kd-boom photonmap. Om consequent te blijven (t.o.v. de eerste situatie) gebruiken we hiervoor respectievelijk de RTSAH (5.3.1) en VVH (5.4.1). Voor deze twee situaties is het uiteraard ook mogelijk om bijvoorbeeld de SAH en BH te gebruiken.

Lege-ruimte-bonus. De lege-ruimte-bonus (5.2.6) en zijn onderliggende motivatie kan eveneens toegepast worden op de HH. Dit zowel voor $\mathcal{C}_{\text{HRTSAH}}(\cdot)$ (5.5.16) als voor $\mathcal{C}_{\text{RTSAH}}(\cdot)$ (5.3.1) binnenin $\mathcal{C}_{P_V \text{ drop}}(\cdot)$ (5.5.17). Lege voxels bevatten geen geometrische primitieven en dus ook geen photons. Een k-nearest neighbor query zal dus niet starten in een lege voxel en zal deze ook niet moeten controleren. Zoals eerder aangehaald moeten kosten onderling vergelijkbaar zijn. Daarom mag de artificiële lege-ruimte-bonus enkel gebruikt worden voor de HRTSAH:

$$\begin{aligned} \mathcal{C}'_{\text{HRTSAH}}(S : V \rightarrow \{V_L, V_R\}) \\ = w_{\text{int}} \cdot \left(\mathcal{C}_{\text{trav}}^{\text{RT}} + \lambda(V_L, V_R) \cdot \left(p_{jL}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_L) + p_{jR}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_R) + p_{LR}^{\text{RT}} \right. \right. \\ \left. \left. \left(p_{fL}^{\text{RT}} \left(\mathcal{C}_{\text{int}}(V_L) + \mathcal{V}_{LR}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_R) \right) + p_{fR}^{\text{RT}} \left(\mathcal{C}_{\text{int}}(V_R) + \mathcal{V}_{RL}^{\text{RT}} \cdot \mathcal{C}_{\text{int}}(V_L) \right) \right) \right) \right) \\ + w_{\text{ext}} \cdot \lambda(V_L, V_R) \cdot \left(\frac{\mathcal{N}_{V_L}}{\mathcal{N}_V} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V_L) + \frac{\mathcal{N}_{V_R}}{\mathcal{N}_V} \cdot \mathcal{C}_{\text{ext}}^{\text{kNN}}(V_R) \right) \quad (5.5.20) \end{aligned}$$

Indien zonder de lege-ruimte-bonus een andere beslissing de voorkeur verdient, wordt eerst de lege ruimte afgesplitst en vervolgens wordt de andere beslissing opnieuw in rekening gebracht. Beslissingen gaan door het invoeren van de lege-ruimte-bonus niet verloren, maar kunnen wel in de wachtrij geplaatst worden.

Gevolgen. We geven nu nog enkele gevvolgen van deze kostfuncties voor de resulterende kd-bomen.

Gevolg 5.5.1 (Bezetting van een intermediaire knoop). *Elke intermediaire knoop van de kd-boom heeft één van de volgende bezettingen:*

1. *geen geometrische primitieven en geen photons;*
2. *alle geometrische primitieven in de subboom met deze knoop als wortel (en alle photons gelegen op het splitsingsvlak);*
3. *alle photons in de subboom met deze knoop als wortel en geen geometrische primitieven;*
4. *(alle photons gelegen op het splitsingsvlak).*

Gevolg 5.5.2 (Bezetting van een bladknoop). *Elke bladknoop van de kd-boom heeft één van de volgende bezettingen:*

1. *geen geometrische primitieven en geen photons;*
2. *geometrische primitieven en photons;*
3. *geometrische primitieven;*
4. *photon(s).*

Gevolg 5.5.3 (Hitpunt). *Het hitpunt van een straal met een geometrisch primitief kan op een intermediaire knoop of bladknoop gelegen zijn.*

Gevolgen 5.5.1 en 5.5.2 zijn belangrijk voor het opstellen van de datastructuur voor de knopen. Gevolg 5.5.3 is belangrijk om in rekening te nemen bij de photon lookup. Na de voxel met het hitpunt te controleren, kan het zijn dat we eerst nog moeten afdalen alvorens de hoger gelegen voxels te controleren in een k-nearest neighbor query.

Hoofdstuk 6

Praktische bouwalgoritmen

De RTSAH kostfunctie (5.3.1) vereist het kennen van de doorlaatbaarheid van stralen tussen de twee kindvoxels. Om deze visibiliteitsprobabiliteiten (5.3.7)-(5.3.8) te evalueren, introduceren we in sectie 6.1 twee benaderingen voor deze visibiliteitsprobabiliteiten die leiden naar een praktisch bouwalgoritme gebaseerd op de RTSAH.

Eenmaal we een praktisch bouwalgoritme voor de RTSAH bekomen hebben, kunnen we opzoek gaan naar een praktisch bouwalgoritme voor de HH in sectie 6.2. In sectie 6.3, vermelden we nog enkele optimalisaties en technische details voor het bouwen van hybride kd-bomen met de HH.

6.1 Praktisch bouwalgoritme RTSAH

Om een praktisch bouwalgoritme te bekomen gebaseerd op de RTSAH, moeten we de kost om te splitsen (5.3.1) evalueren voor een eindige verzameling van splitsingsvlakken.

De probabiliteiten van een straal die slechts één of beide kindvoxels doorkruist, kunnen triviaal berekend worden door gebruik te maken van de vergelijkingen (5.3.2) tot (5.3.4). De exacte berekening van de visibiliteitsprobabiliteiten V_{LR}^{RT} (5.3.7) en V_{RL}^{RT} (5.3.8) daarentegen vereist de integratie van de visibiliteit over het oppervlakte van het splitsingsvlak en de hemisfeer van inkomende richtingen. Dit is weergegeven in Figuur 6.1a. Door de afhankelijkheid van de geometrische distributies in de kindvoxels, kan er geen gesloten uitdrukking gevonden worden voor de visibiliteitsprobabiliteiten.

De visibiliteitsprobabiliteiten kunnen berekend worden via Monte Carlo integratietechnieken. Dit is echter onpraktisch door de afwezigheid van een acceleratiestructuur tijdens de constructiefase. Bovendien zou het uitvoeren van Monte Carlo bemonstering voor elk mogelijk splitsingsvlak, resulteren in te grote bouwtijden. Het aantal Monte Carlo monsters moet laag zijn wanneer de voxel die gesplitst wordt, veel

6. PRAKTISCHE BOUWALGORITMEN

geometrische primitieven bevat. Dit is het geval voor de eerste splitsbeslissingen van de kd-boom. Daartegenover is de keuze van het splitsingsvlak belangrijker en beslissender bij deze eerste splitsbeslissingen en vraagt hiervoor dus om een accurate schatter.

Om een handelbare bouwtijd te garanderen, willen we een praktisch bouwalgoritme bekomen die bij elke splitsbeslissing:

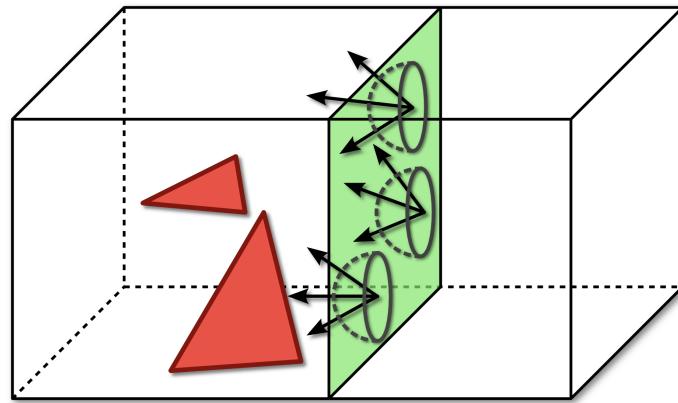
1. alleen rekening hoeft te houden met een eindige verzameling van splitsingsvlakken die het optimale splitsingsvlak bevat volgens onze benaderde RTSAH;
2. incrementeel onze benaderde visibiliteitsprobabiliteiten kan updaten tijdens het verschuiven van het splitsingsvlak in eenzelfde richting.

Beide voorwaarden zijn moeilijk te realiseren door de directionele afhankelijkheid van de visibiliteitsprobabiliteiten. Daarom stellen we voor deze afhankelijkheid te elimineren door slechts één richting in rekening te nemen. In sectie 6.1.1 gebruiken we de orthogonale projectierichting op het splitsingsvlak en in sectie 6.1.2 gebruiken we de gemiddelde geprojecteerde richting op het splitsingsvlak. In sectie 6.1.3 valideren we onze benaderingen. In de secties 6.1.4 en 6.1.5 tonen we aan dat deze benaderingen van de visibiliteitsprobabiliteiten beiden leiden tot een praktisch bouwalgoritme gebaseerd op de RTSAH waarvoor beide voorwaarden voldaan zijn.

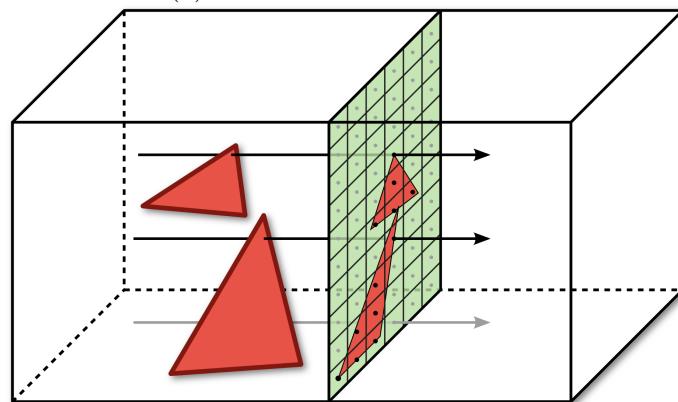
6.1.1 All Points One Direction RTSAH

De All Points One Direction (AOD) RTSAH neemt alle punten op het splitsingsvlak in rekening en negeert de directionele afhankelijkheid door enkel de richting orthogonaal op het splitsingsvlak te beschouwen. Dit komt overeen met het orthogonaal projecteren van de geometrische primitieven op het splitsingsvlak wat efficiënt door rasterizatie gedaan kan worden. Bij de projectie wordt het splitsingsvlak opgedeeld in een eindig aantal gelijke, vierkante *rasterizatiecellen*. Het centrum van deze cellen noemen we een *rasterizatiepunt*. Een rasterizatiecel is bedekt door een geometrisch primitief indien de orthogonale projectie van dit primitief op het splitsingsvlak het rasterizatiepunt van deze cel omsluit. Figuur 6.1b illustreert dit idee.

Opmerking 6.1 (Verband met blocking factoren). Het idee om de visibiliteit tussen twee naburige voxels te berekenen door een orthogonale projectie op het splitsingsvlak is gelijkaardig aan het berekenen van de blocking factor van een kindvoxel in [RKJ96] en [RKC98]. In tegenstelling tot Reinhard et al. ([RKJ96], [RKC98]) werken we niet met de omsluitende boxen van de geometrische primitieven, maar met de geometrische primitieven zelf tijdens de rasterizatie. Bovendien vermijden we door de rasterizatie impliciet overlappingen van geometrische primitieven en bekomen we zowel een kostmodel als bouwprocedure. Het rasterizeren kan gezien worden als een speciale vorm van bemonsteren met een biased schatter als gevolg. Het



(a) All Points All Directions



(b) All Points One Direction

$$\begin{array}{ccc}
 \text{SA} = 6z^2 & & \text{APSA} = \frac{3}{2}z^2 \\
 \text{SA} = 4\pi r^2 & & \text{APSA} = \pi r^2
 \end{array}$$

(c) Average Projected Surface Area

FIGUUR 6.1: De ideeën achter de verschillende benaderingen van de RTSAH. (a) Toont de ground truth, waarbij we integreren over het oppervlakte van het splitsingsvlak en de hemisfeer van inkomende richtingen. (b) Toont de APOD benadering, waarbij we integreren over het oppervlak van het splitsingsvlak maar enkel de richting orthogonaal op het splitsingsvlak in rekening nemen. (c) De gemiddelde geprojecteerde oppervlakte van een convex veelvlak.

6. PRAKTISCHE BOUWALGORITMEN

rasterizeren van geometrische primitieven is echter computationeel goedkoper dan deze op intersectie te testen met willekeurige stralen, maakt hergebruik van intersectie-informatie mogelijk bij het verplaatsen van het splitsingsvlak en vereist geen tweede kd-boom in tegenstelling tot [Hav00].

6.1.2 Average Projected Surface Area RTSAH

Cauchy toonde aan dat de gemiddelde geprojecteerde oppervlakte van een driedimensionaal convex veelvlak gelijk is aan één vierde van zijn oppervlakte (zie Figuur 6.1c).

De Average Projected Surface Area (APSA) RTSAH benadert de visibiliteitsprobabiliteiten als de fractie van het oppervlakte van het splitsingsvlak dat overblijft na het aftrekken van het gemiddelde geprojecteerde oppervlakte van de geometrie in een kindvoxel:

$$\mathcal{V}_{LR}^{RT} \approx 1 - \min \left(\frac{SA_{G_{V_L}}}{4 \cdot SA_S}, 1 \right) \quad (6.1.1)$$

$$\mathcal{V}_{RL}^{RT} \approx 1 - \min \left(\frac{SA_{G_{V_R}}}{4 \cdot SA_S}, 1 \right) \quad (6.1.2)$$

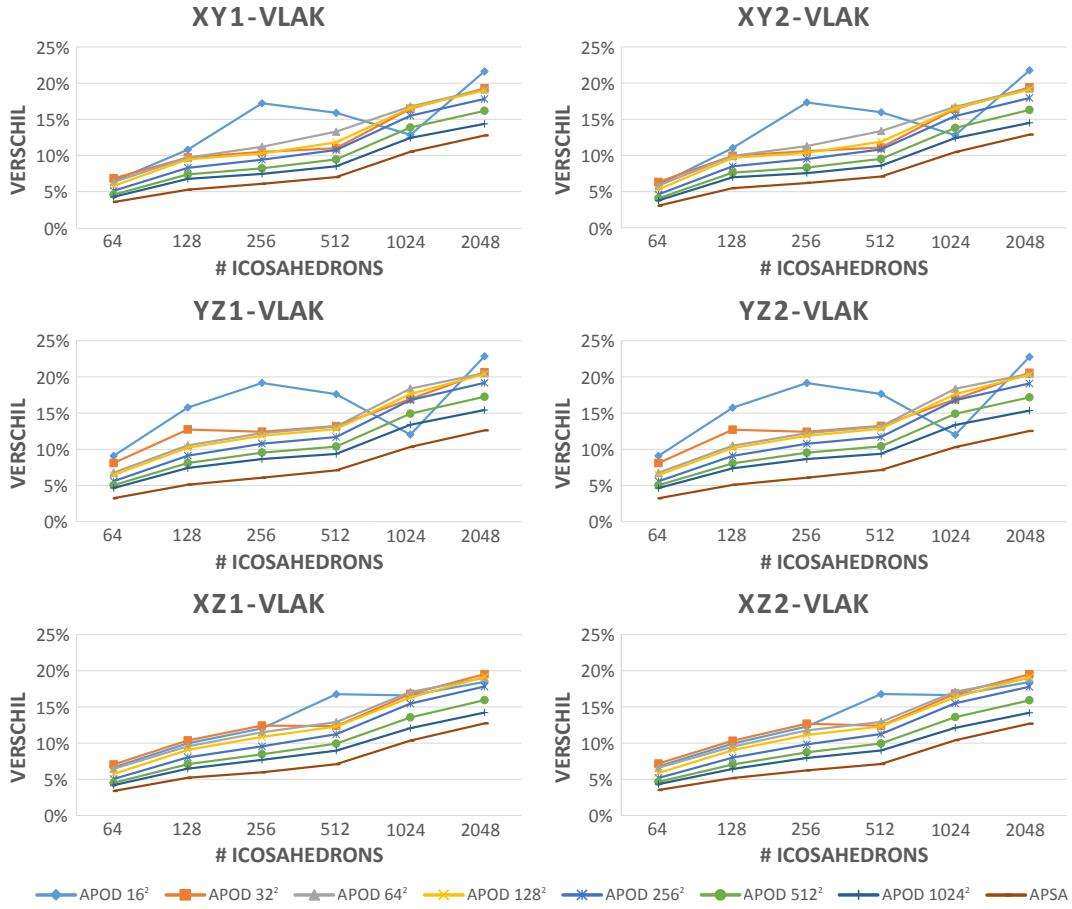
Hierbij is $SA_{G_{V_k}}$ de oppervlakte van alle geometrische primitieven in voxel V_k . Merk op dat indien de verzamelingen G_{V_L} en G_{V_R} een convex veelvlak voorstellen volledig ingesloten in respectievelijk V_L en V_R , dan zijn de bovenstaande benaderingen exact.

6.1.3 Validatie

Om onze benaderingen van de visibiliteitsprobabiliteiten (5.3.7) en (5.3.8) te valideren, construeren we een artificiële scène bestaande uit een variabel aantal willekeurig (via stratificatie) geplaatste icosahedronen. De visibiliteitsprobabiliteiten in deze context drukken de kans uit dat een straal door de verschillende vlakken van de omhullende box van de scènes geraakt zonder een geometrisch primitief te raken. Om ground truth visibiliteitsprobabiliteiten te bekomen, doorkruisen we de omhullende box van de scènes met 100M stralen die uniform verdeeld zijn over de omhullende bol van de scènes en waarvoor gegarandeerd is dat deze stralen de omhullende box van de scènes doorkruisen. Voor de APOD benadering gebruiken we een verschillend aantal (16^2 , 32^2 , 64^2 , 128^2 , 256^2 , 512^2 , 1024^2) rasterisatiepunten. De resultaten zijn weergegeven in Figuur 6.2.

De APSA benadering resulteert consistent in de kleinste verschillen tussen de benaderde en exacte visibiliteitsprobabiliteiten voor de 6 vlakken van de omhullende box van de scènes in vergelijking met de APOD benadering. Bemerk de trage lineaire toename van de onderschatting versus de exponentiële toename van het aantal icosahedronen. De APSA benadering resulteert in een gemiddelde onderschatting van 12,66% in de moeilijkste (wat de visibiliteit betreft) scène met 2048 icosahedronen wat nog altijd een redelijke benadering is gegeven de moeilijkheid om de

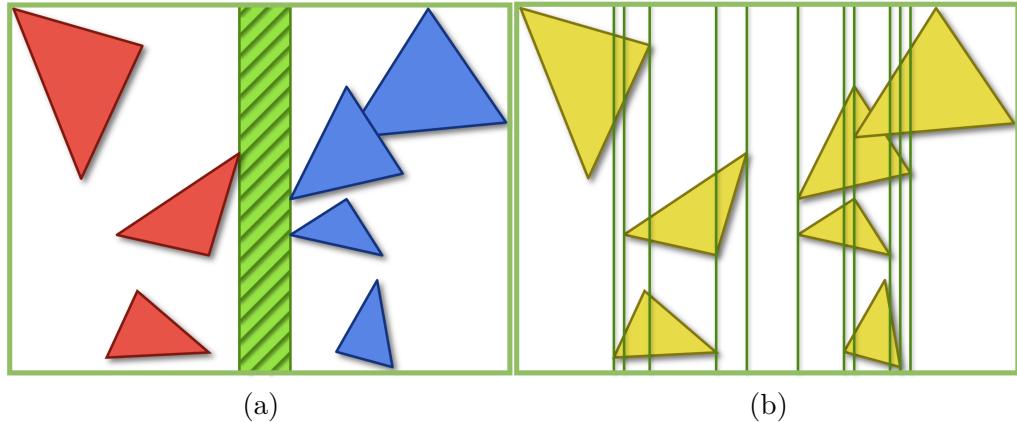
6.1. Praktisch bouwalgoritme RTSAH



FIGUUR 6.2: Het verschil tussen de benaderde en exacte (100M stralen) visibiliteitsprobabiliteiten voor de 6 vlakken van de omhullende box van de scenes. Een positief (negatief) verschil komt overeen met een onderschatting (overschatting).

visibiliteitsprobabiliteiten exact te berekenen. De APOD benadering die rasterizeert op 1024^2 rasterizatiepunten doet iets slechter. Helaas moet de APOD benadering gebruik maken van een klein aantal rasterizatiepunten om een snelle bouwprocedure te bekomen. Dit is voor deze scenes niet problematisch aangezien de andere APOD benaderingen ten hoogste 5% slechter presteren met uitzondering van de benadering met 16^2 rasterizatiepunten. Enkel 16^2 rasterizatiepunten beschouwen is niet accuraat gegeven het aantal geometrische primitieven in deze scenes. Daarnaast is het resultaat te gevoelig voor kleine verplaatsingen van de rasterizatiepunten.

We concluderen dat voor deze moeilijke (wat de visibiliteit betreft) scenes, de APSA benadering resulteert in goede benaderingen van de visibiliteitsprobabiliteiten. De APOD benadering is daarentegen minder accuraat en is afhankelijk van het aantal gebruikte rasterizatiepunten. Bij een groot aantal geometrische primitieven, is het belangrijk een groot aantal rasterizatiepunten te gebruiken. Onze implementatie van



FIGUUR 6.3: Splitskandidaten van de SAH, APOD en APSA RTSAH. (a) Toont een gebied (groen) waar de SAH, APOD RTSAH en APSA RTSAH kostfuncties monotoon stijgend of dalend zijn. (b) De eindige verzameling van splitsingsvlakken (groen) in één richting die beschouwd moet worden door de SAH, APOD RTSAH en APSA RTSAH om het optimale splitsingsvlak te bekomen volgens hun respectievelijke kostfunctie.

de APOD RTSAH maakt gebruik van minstens 16 en maximum 64 rasterizatiepunten in één dimensie afhankelijk van het aantal geometrische primitieven aanwezig in de te splitsen voxel.

6.1.4 Splitskandidaten

Voor de eenvoud, clippen we geen geometrische primitieven die meerdere voxels overlappen tijdens de rasterisatie (APOD) of de berekening van de oppervlaktes (APSA). Hierdoor stijgen of dalen de APOD en APSA kostfuncties monotoon in het gebied tussen twee opeenvolgende grenzen van de geometrische primitieven langs dezelfde as (geïllustreerd in Figuur 6.3a). Binnen zo'n gebied, blijven de benaderingen van de visibiliteitsprobabiliteiten \mathcal{V}_{LR}^{RT} en \mathcal{V}_{RL}^{RT} constant en zijn de probabiliteiten p_{jL}^{RT} en p_{jR}^{RT} beiden lineaire functies van de positie van het splitsingsvlak S . Dit betekent dat we enkel de splitsingsvlakken in rekening moeten nemen gepositioneerd op de grenzen van de geometrische primitieven om het beste splitsingsvlak volgens onze RTSAH kostfunctie (5.3.1) te bekomen (zie Figuur 6.3b). Dit is dezelfde verzameling van splitskandidaten die in rekening gebracht moet worden voor de SAH.

6.1.5 Bouwalgoritme

We moeten enkel een aantal kleine aanpassingen maken aan de bouwprocedure voor de SAH zonder de totale computationele complexiteit te veranderen. Het bekomen van p_{jL}^{RT} , p_{jR}^{RT} , p_{LR}^{RT} is vrij gelijkaardig aan het berekenen van p_L^{RT} en p_R^{RT} . De visibiliteitsprobabiliteiten worden incrementeel geüpdatet tijdens één enkele beweging doorheen de splitskandidaten voor elke primaire richting. Indien we het splitsingsvlak

opschuiven van links L naar rechts R, moeten \mathcal{V}_{LR}^{RT} en \mathcal{V}_{RL}^{RT} respectievelijk incrementeel verlaagd en opgehoogd worden. De berekening van de visibiliteitsprobabiliteiten is afhankelijk van de gebruikte benadering:

APOD RTSAH: Voor \mathcal{V}_{LR}^{RT} , moeten we bijhouden welke rasterizatiepunten niet overlapt worden door geometrische primitieven in V_L (invariant). Voor \mathcal{V}_{RL}^{RT} , moeten we bijhouden hoeveel geometrische primitieven in V_R elk rasterizatiepunt bedekken (invariant). Hiervoor moeten alle geometrische primitieven in V gerasterizeerd worden op het splitsingsvlak alvorens de kost van elke splitskandidaat te bepalen.

APSA RTSAH: Om \mathcal{V}_{LR}^{RT} en \mathcal{V}_{RL}^{RT} te berekenen, moeten we de som van de oppervlaktes van alle geometrische primitieven in de linkse en rechtse kindvoxel respectievelijk bijhouden (invarianten). Hiervoor moet de som van de oppervlaktes van alle geometrische primitieven in V berekend worden alvorens de kost van elke splitskandidaat te bepalen.

De datastructuren die deze invarianten moeten garanderen, moeten upgedated (incrementeer/decruementeer) worden wanneer een geometrische primitief de linkse kindvoxel begint te overlappen of de rechtse kindvoxel niet meer overlapt.

Opmerking 6.2 (Diëlektrische materialen). Geometrische primitieven met diëlektrische materialen moeten niet in rekening gebracht worden om deze benaderde visibiliteitsprobabiliteiten te berekenen. Diëlektrische materialen (bv.: glas) laten stralen (eventueel verzwakt of afgebogen) door.

6.2 Praktisch bouwalgoritme HH

In deze sectie bespreken we hoe de HH gebruikt kan worden in een praktisch bouwalgoritme. Gegeven de situatie die van toepassing is (zie sectie 5.5.2), moet voor elke splitsbeslissing de kost voor elk van de mogelijke beslissingen berekend worden. Hiervoor dient elke component van de bijhorende kostfuncties geëvalueerd te worden voor een eindige verzameling van splitsingsvlakken.

Voor de RTSAH en VVH hebben we al uitgebreid besproken hoe deze te gebruiken in een praktisch bouwalgoritme, zowel wat het evalueren van de kostfuncties als de keuze van de splitsingsvlakken betreft. Enkele unieke componenten van het HH kostmodel moeten nog een praktische benadering krijgen. Dit wordt besproken in sectie 6.2.1. Voor de HRTSAH moeten we nog een eindige verzameling van splitskandidaten bepalen. Dit wordt besproken in sectie 6.2.3.

6.2.1 Componenten

Verhouding interne en externe query's. De gewichten w_{int} en w_{ext} (5.5.6) om de bijdrage van interne en externe query's ten opzichte van elkaar af te wegen benaderen we op basis van volumes. Interne query's kunnen enkel binnen het volume

van de voxel ontstaan. Externe query's kunnen enkel op een afstand van ten hoogste R_{\max} buiten het volume van de voxel voorkomen. Het totale volume waarin query's kunnen ontstaan waarvoor een voxel gecontroleerd moet worden is het volume van deze voxel uitgebreid met de querystraal R_{\max} in alle richtingen. Omwille van computationele redenen benaderen we dit volume door het volume van de voxel waarbij elke zijde is verlengd met tweemaal de querystraal R_{\max} . Dit is analoog aan de benaderingen (5.4.5) en (5.4.7) van de VVH. De gewichten zijn dan als volgt gegeven:

$$w_{\text{int}} \approx \frac{\prod_{i=x,y,z} (V_{i,\max} - V_{i,\min})}{\prod_{i=x,y,z} (V_{i,\max} - V_{i,\min} + 2 \cdot R_{\max})} \quad (6.2.1)$$

$$w_{\text{ext}} = 1 - w_{\text{int}} \quad (6.2.2)$$

Indien de $2 \cdot R_{\max}$ term domineert, zal de bijdrage voor externe query's groter zijn dan deze voor interne query's en visa versa. Deze benadering veronderstelt door de koppeling tussen interne query's en stralen dat het aantal stralen (primair en secundair) in een volume rechtelevenredig is met de grootte van het volume. Hierdoor zal de ray tracing bijdrage in grote voxels groter zijn dan de bijdrage voor externe query's en visa versa.

Opmerking 6.3 (Querystraal na interne query). Indien we veronderstellen dat photons uniform verdeeld zijn binnen het volume en nabij de rand van een voxel en het maximaal aantal photons k per k-nearest neighbor query gekend is, kunnen we na het controleren van deze voxel V de nieuwe querystraal R'_{\max} berekenen:

$$k = \frac{\frac{4}{3}\pi (R'_{\max})^3}{\text{VOL}_V} \cdot |\text{P}_V| \quad (6.2.3)$$

$$R'_{\max} = \sqrt[3]{\frac{\text{VOL}_V}{\frac{4}{3}\pi} \cdot k} \quad (6.2.4)$$

We moeten uiteraard nog expliciet garanderen dat de nieuwe querystraal niet groter kan zijn dan de maximale querystraal R_{\max} . Door deze nieuwe querystraal kunnen we de impact beperken voor query's die intern zijn voor de ene kindvoxel en extern voor de andere kindvoxel. Dit doen we echter niet omwille van volgende redenen:

- Computationeel gezien is het eenvoudiger dit niet in rekening te nemen;
- De impactreductie is slechts voor een beperkt gebied van toepassing;
- De kindvoxels kunnen verder verfijnd worden waardoor de tijd om de ene oorspronkelijke kindvoxel vanuit de andere oorspronkelijke kindvoxel te bezoeken niet-verwaarloosbaar is;
- De assumptie dat de photons uniform verdeeld zijn, gaat niet op in de praktijk.

Voorkomen interne query's. Een interne query ontstaat vanuit een hitpunt van een straal met een geometrisch primitief binnen de voxel. De kans op een hit in een voxel gegeven dat deze voxel doorkruist wordt (5.5.7)-(5.5.11), is het omgekeerde van de doorlaatbaarheid/visibiliteit van een voxel (voor externe stralen). We kunnen hiervoor onze APOD en APSA benadering uit sectie 6.1.5 uitbreiden naar de 6 vlakken van de voxel in plaats van alleen naar het splitsingsvlak te kijken.

Voor de APOD benadering moeten we nu de geometrische primitieven in de richting van de drie primaire assen rasterizeren. Dit resulteert in een benadering van de hitprobabiliteit voor elk vlak van de voxel. De hitprobabiliteit voor de voxel is nu het gewogen gemiddelde van deze hitprobabiliteiten. Voor de APSA benadering moeten we nog steeds de som van de oppervlakte van alle geometrische primitieven in de voxel berekenen:

$$p_{\text{hit}}^{\text{RT}} \approx \min \left(\frac{\text{SA}_{G_V}}{\text{SA}_V}, 1 \right) \quad (6.2.5)$$

$$p_{\text{hitL}}^{\text{RT}} \approx \min \left(\frac{\text{SA}_{G_{V_L}}}{\text{SA}_{V_L}}, 1 \right) \quad (6.2.6)$$

$$p_{\text{hitR}}^{\text{RT}} \approx \min \left(\frac{\text{SA}_{G_{V_R}}}{\text{SA}_{V_R}}, 1 \right) \quad (6.2.7)$$

In de implementatie maken we geen onderscheid tussen de verschillende types van stralen en gebruiken steeds dezelfde hitprobabiliteit voor een kindvoxel. De APSA benadering resulteert namelijk, zoals zal blijken, in een zeer goede benadering.

Opmerking 6.4 (Verband met blocking factoren). Voor de APOD benadering is dit analoog aan opmerking 6.1.

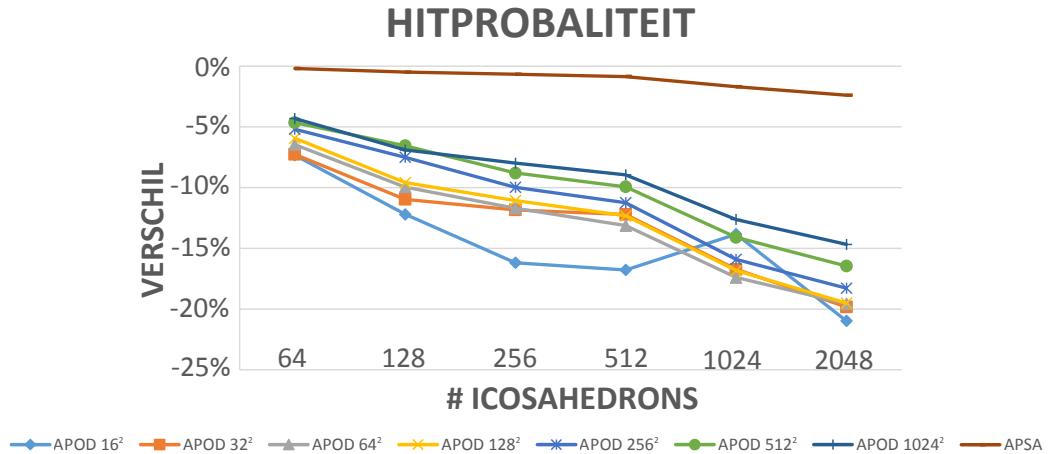
Voorkomen externe query's. Een externe query is een niet-voltooide query van buitenaf waarvoor de voxel gecontroleerd moet worden. Deze query's zijn eveneens ontstaan vanuit een hitpunt van een straal met een geometrisch primitief. Het hitpunt is echter buiten de voxel gelegen. De geometrische distributies van de omliggende voxels zijn in het algemeen niet gekend tijdens de constructiefase van de kd-boom. Daarom kiezen we onze benadering voor $p_{\text{ext}}^{\text{kNN}}$ onafhankelijk van deze geometrische distributies. We veronderstellen een worst-case gedrag:

$$p_{\text{ext}}^{\text{kNN}} \approx 1 \quad (6.2.8)$$

$$p_{\text{extL}}^{\text{kNN}} \approx 1 \quad (6.2.9)$$

$$p_{\text{extR}}^{\text{kNN}} \approx 1 \quad (6.2.10)$$

Deze benadering heeft het voordeel een hoge kost toe te kennen aan de bijdrage van externe query's. Voxels die aan elkaar grenzen in de driedimensionale ruimte kunnen ver van elkaar verwijderd zijn in de kd-boom. De tijd nodig om vanuit een voxel een buurvoxel te bezoeken is daarom niet verwaarloosbaar. Deze tijd is echter moeilijk op voorhand in te calculeren doordat de kd-boom nog niet volledig gebouwd is. Vandaar de motivatie om een worst-case gedrag te veronderstellen.



FIGUUR 6.4: Het verschil tussen de benaderde en exacte (100M stralen) hitprobabiliteit van de scenes. Een positief (negatief) verschil komt overeen met een onderschatting (overschatting).

6.2.2 Validatie

We hebben dezelfde testopstelling van sectie 6.1.3 gebruikt om onze APOD en APSA benadering van de hitprobabiliteit van de omhullende box van de scenes met icosaahedronen te valideren. De resultaten zijn weergegeven in Figuur 6.4.

De APSA benadering resulteert in zeer kleine verschillen tussen de benaderde en exacte hitprobabiliteit voor deze scenes. De APOD benadering resulteert in ondermaatse benaderingen. Dit door slechts drie richtingen uit een oneindige verzameling van richtingen in rekening te nemen.

We concluderen dat voor deze moeilijke (wat de hitprobabiliteit betreft) scenes, de APSA benadering resulteert in zéér goede benaderingen van de hitprobabiliteit. De APOD benadering is daarentegen niet accuraat.

6.2.3 Splitskandidaten

De APOD benadering is niet geschikt om de hitprobabiliteit te benaderen (zie sectie 6.2.2). Daarnaast is het incrementeel updaten van de hitprobabiliteit niet mogelijk tijdens het verschuiven van het splitsingsvlak in eenzelfde richting. We dienen hiervoor telkens opnieuw te rasterizeren in de twee overige richtingen. Voor de APSA benadering is het incrementeel updaten van de hitprobabiliteit in dezelfde richting wel mogelijk.

Nu moeten we enkel nog een eindige verzameling van splitsingsvlakken bepalen voor de APSA HRTSAH. We kiezen hiervoor zelf een eindige verzameling van ‘interessante’ splitsingsvlakken. We gebruiken de splitskandidaten van de RTSAH en

VVH: splitsingsvlakken gepositioneerd op de grenzen van de geometrische primitieven en splitsingsvlakken door de photons. Gegeven deze splitsingsvlakken kunnen we analytisch de minima van de APSA HRTSAH kostfunctie bepalen tussen twee opeenvolgende splitsingsvlakken. We bepalen deze extra splitsingsvlakken echter niet. Indien het aantal geometrische primitieven en photons hoog is, beschouwen we zodanig veel en goed verspreide splitsingsvlakken dat één van deze splitsingsvlakken wel zeer dicht tegen het optimale splitsingsvlak zal liggen. Indien het gebied tussen twee opeenvolgende splitsingsvlakken klein is, is de kans bovendien klein dat het minimum binnen dit gebied gelegen is. Verder resulteert dit in snellere bouwtijden door minder splitsingsvlakken te beschouwen en minder computationele overhead.

De APOD en APSA RTSAH bouwprocedures hebben net zoals de SAH een totale computationele complexiteit van $\mathcal{O}(|G| \log |G|)$. De VVH bouwprocedure heeft een totale computationele complexiteit van $\mathcal{O}(|P| \log |P|)$. De APSA HRTSAH heeft een totale computationele complexiteit van $\mathcal{O}((|G| + |P|) \log (|G| + |P|))$. De volledige HH bouwprocedure heeft dus een totale computationele complexiteit van $\mathcal{O}((|G| + |P|) \log (|G| + |P|))$.¹

Opmerking 6.5 (Clustering). Indien we de APOD benadering zouden willen gebruiken in een praktisch bouwalgoritme met dezelfde totale computationele complexiteit, moeten we een constant aantal splitskandidaten gebruiken. Deze splitskandidaten kunnen bijvoorbeeld bepaald worden met een $\mathcal{O}(|P|)$ clusteralgoritme voor de photons. Door de photons op te delen in clusters, verwachten we dat ook de geometrische primitieven opgedeeld worden in (dezelfde) clusters. De splitsingsvlakken kunnen nu zo gekozen worden om de clusters van elkaar te scheiden. Dit is niet verder onderzocht en zou met het oog op verder onderzoek eerst in een niet-hybride kNN context getest moeten worden.

6.3 Optimalisaties HH

In deze sectie bespreken we enkele optimalisaties en technische details voor het bouwen van hybride kd-bomen met de HH.

6.3.1 Gedeelde componenten

Indien er zowel geometrische primitieven als photons nog in rekening gebracht moeten worden bij de huidige splitsbeslissing, moeten er vier kosten (5.5.16)-(5.5.19) berekend worden. Een groot deel van de componenten kan gedeeld worden tussen deze kosten en moet dus slechts eenmaal berekend worden:

- w_{int} en w_{ext}

¹Onze eigen implementatie gebruikt voor elk van onze eigen en referentie bouwprocedures steeds de $\mathcal{O}(N \log^2 N)$ variant zoals in [PH10].

- $\mathcal{C}_{\text{visit}}^{\text{RT}}(V)$
- $p_{\text{hit}}^{\text{RT}}$
- $\mathcal{C}_{\text{visit}}^{\text{kNN}}(V)$

6.3.2 Offsetpassing

Indien er zowel geometrische primitieven als photons nog in rekening gebracht moeten worden bij de huidige splitsbeslissing, moeten er vier kosten (5.5.16)-(5.5.19) berekend worden. Voor de eerste drie kosten (5.5.16)-(5.5.18) is het opportuin telkens twee procedures te voorzien: De eerste procedure berekent de kost en geeft de splitspositie, primaire richting en de nodige offsets voor het optimale splitsingsvlak volgens de overeenkomstige kostfunctie door. De tweede procedure kan indien een beslissing effectief uitgevoerd moet worden, de splitsing doorvoeren op basis van deze parameters. We bouwen hierdoor voort op onze vorige berekeningen en moeten geen berekeningen opnieuw uitvoeren.

6.3.3 Automatische maximale diepte

Net zoals in de praktijk bij de SAH (zie [HB02], [PH10]) het geval is, kan het opportuin zijn op voorhand een maximale diepte te bepalen op basis van het aantal geometrische primitieven en photons. Op deze manier kan alle heapgealloceerde data op voorhand gealloceerd worden en kan het geheugengebruik ingeperkt worden.

Deze maximale diepte is typische van de volgende vorm:

$$\text{depth}_{\max} = k_1 + k_2 \cdot \log_2(|G|) + k_3 \cdot \log_2(|P|) \quad (6.3.1)$$

Hierbij kunnen de constanten k_1 , k_2 en k_3 optimaal gekozen worden voor een gevarieerde verzameling van testscenes.

Photons kunnen niet aan weerszijden van het splitsingsvlak gelegen zijn en kunnen daarom niet geduplicateerd worden in tegenstelling tot geometrische primitieven. Daarom kiezen we indien de maximale diepte bereikt wordt nog steeds tussen de beslissingen horende bij (5.5.18) en (5.5.19) in plaats van zowel de photons als geometrische primitieven te droppen en te stoppen. Indien we beslissen enkel de geometrische primitieven te droppen, weten we namelijk precies hoeveel knopen er bijkomen door het verder opdelen van de photons.

6.3.4 Aantal geforceerde splitsbeslissingen

Net zoals in de praktijk bij de SAH (zie [HB02], [PH10]) het geval is, kan het opportuin zijn toch te splitsen volgens de HRTSAH (5.5.16) in de eerste situatie of volgens de RTSAH (5.3.1) in de tweede situatie ook al is deze kost niet minimaal. Doordat beide heuristieken lokaal en gulzig zijn, kan de gecombineerde kost van de ray tracing en kNN query's alsnog verminderd worden bij verdere verfijning van de

kd-boom. Een gulzige heuristiek kan namelijk vast komen te zitten op een lokaal maximum.

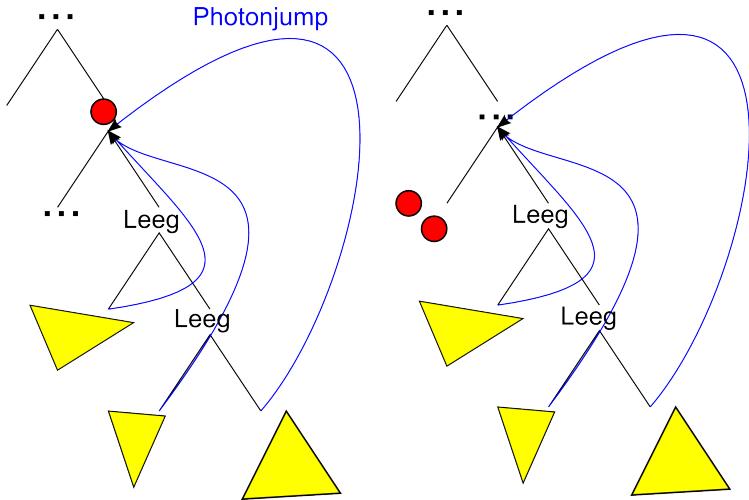
In de eerste situatie laten we maximaal 4 geforceerde splitsbeslissingen toe. Hierbij kunnen (5.5.16), (5.5.17) en (5.5.18) geforceerd bevoordeeld worden ten opzichte van (5.5.19). Daarnaast bevoordelen we de HRTSAH (5.5.16) altijd met 15%. Uit eerdere testresultaten (met $|P_V| > 500k$) is waargenomen dat het droppen van grote verzamelingen geometrische primitieven lokaal voordeliger kan zijn dan het verder opsplitsen van de geometrische primitieven. De kosten horende bij (5.5.16) en (5.5.18) verschillen in deze gevallen slechts in kleine mate van elkaar. De gevolgen van de bijkomende beslissingen zijn daarentegen zeer verschillend. Daarnaast resulteert de beslissing met de lokale minimale kost, niet noodzakelijk in de globale minimale kost. Indien we veel photons en geometrische primitieven beschouwen, zullen beide beslissingen naarmate we verder splitsen en de kd-boom verder verfijnen, goedkoper worden. Welke van de twee lokale beslissing uiteindelijk globaal de beste is, is niet op voorhand gekend.

Indien de kosten (5.5.16) en (5.5.18) groot zijn, beschouwen we een groot aantal geometrische primitieven en/of photons. De beslissingskeuze is hierdoor bepalend bij de bouw van de hybride kd-boom en cruciaal voor de performantie van de ray tracing en kNN query's. Een verdere opsplitsing van zowel de geometrische primitieven als photons verdient hierbij de voorkeur. Het droppen van de geometrische primitieven is daarentegen definitief en kan niet meer gecompenseerd worden bij de volgende splitsbeslissingen. Indien de kosten (5.5.16) en (5.5.18) klein zijn, speelt de 15% factor geen rol. Dit verantwoordt de algemene bevoordeling van de HRTSAH met 15%. Geometrische primitieven of photons zullen nu enkel gedropt worden, indien de kost significant (meer dan 15%) groter is dan de kost om beide verder op te delen.

Opmerking 6.6 (Nuancingering). Indien de minimale en maximale grens van alle geometrische primitieven buiten de voxel onder beschouwing gelegen zijn en er dus geen splitskandidaten zijn voor (5.5.17), voeren we geen bevoordeling van de HRTSAH door. Indien we het opdelen van de geometrische primitieven en photons in deze gevallen zouden forceren, neemt de duplicatie van geometrische primitieven toe.

Indien we overschakelen op de RTSAH in de tweede situatie, laten we eveneens een beperkt aantal geforceerde splitsbeslissingen toe. Er is geen koppeling tussen het aantal geforceerde splitsbeslissingen van de HRTSAH en RTSAH.

Merk op dat we geen artificiële absolute drempelwaarden gebruiken om bepaalde beslissingen uit te sluiten (bv.: het aantal photons of geometrische primitieven mag niet 'te groot' zijn om te droppen).



FIGUUR 6.5: Photonjumps.

6.3.5 Lookaheads

Bij het top-down doorlopen van de kd-boom voor een straal, kan reeds op een intermediaire knoop bijgehouden worden met twee *lookahead* bits of de linkse en rechtse subboom geometrische primitieven bevatten. Indien een subboom geen geometrische primitieven bevat en dus een lege bladknoop voorstelt, moet deze niet expliciet opgehaald worden om tot deze conclusie te komen.

De photons bevinden zich steeds op een geometrische primitief, maar niet alle geometrische primitieven bevatten photons. Daarom is het opportuun om eveneens op een intermediaire knoop twee lookahead bits te voorzien die aangeven of de linkse en rechtse subboom photons bevatten. Indien een subboom leeg is, kan op deze manier vermeden worden om deze subboom (top-down) te controleren voor een k-nearest neighbor query.

6.3.6 Photonjumps

De kostfunctie $\mathcal{C}_{PV} \text{ drop}(\cdot)$ (5.5.17) voor het droppen van photons op een intermediaire knoop houdt geen rekening met een eventuele verdere verfijning van de kindvoxels. Om geen onnodige knopen zonder photons (bottom-up) te controleren, moet het mogelijk zijn in het geval van een hit op één van de geometrische primitieven van een kindknoop om rechtstreeks naar deze intermediaire knoop met de photons te springen. Daarom voorzien we voor elke knoop een verwijzing naar de voorouderknoop waarnaar gesprongen moet worden in de context van een k-nearest neighbor query. Dit mechanisme noemen we een *photonjump* (zie Figuur 6.5).

De voorouderknoop wordt initieel gelijk gesteld aan de wortelknoop². De huidige

²Elke waarde kan hiervoor gebruikt worden zolang we maar weten wanneer de photon lookup

voorouderknoop wordt hiervoor doorgegeven naar elke nieuwe splitsbeslissing. Indien er nu zodanig gesplitst wordt dat de linkse subboom of intermediaire knoop photons bevat, moet de rechtse knoop naar de intermediaire knoop springen. Indien niet naar de doorgegeven voorouderknoop. In dit laatste geval negeren we dus de intermediaire knoop en linkse subboom. Dit is analoog voor de linkse knoop.

Zowel bij het backtracken als backjumpen naar de respectievelijk (intermediaire) ouder- en voorouderknoop, moeten we weten vanuit welke van de twee deelbomen met deze intermediaire knoop als wortel we vertrokken zijn. We moeten voor de correctheid van de photon lookup vermijden dezelfde deelboom opnieuw te controleren. Daarom houden we per knoop naast de voorouderknoop ook bij in welke van de twee subbomen met de voorouderknoop als wortel de knoop zich bevindt.

6.3.7 Vroege terminatie

Net zoals in sectie 4.3 voor de Hybrid Connected Photon Map, kunnen we gegeven de maximale querystraal, R_{\max} , de minimaal omsluitende photonvoxel bepalen om de gemiddelde k-nearest neighbor query vroegtijdig (voor het bereiken van de wortelknoop) te laten stoppen. De minimaal omsluitende photonvoxel moet bepaald worden voor elke voxel horende bij een knoop met geometrische primitieven. Dit kunnen nu zowel intermediaire knopen als bladknopen zijn.

De minimaal omsluitende photonvoxel horende bij de knoop van waaruit een k-nearest neighbor query start, moet bijgehouden worden tijdens de volledige query. Wanneer de minimaal omsluitende photonvoxel volledig gecontroleerd is, wordt de query vroegtijdig beëindigd.

Indien we gebruik maken van photonjumps (zie sectie 6.3.7) bestaat het gevaar over de minimaal omsluitende photonvoxel te springen en pas na het controleren van de wortel de query te beëindigen. Indien de subboom met de knoop horende bij de minimaal omsluitende photonvoxel als wortel, geen photons bevat³, kiezen we de voxel horende bij de photonjump van deze knoop als minimaal omsluitende photonvoxel. Deze laatste knoop doorlopen we namelijk wel.

Opmerking 6.7 (Doorloopstrategieën). We onderscheiden drie mogelijke doorloopstrategieën van de hybride kd-boom voor k-nearest neighbor query's:

1. starten vanuit de wortelvoxel en de hybride kd-boom steeds top-down doorlopen (niet-hybride setting);

beëindigd is.

³Dit kan nog verder verfijnd worden indien slechts één van de twee subbomen photons bevat. Indien de linkse subboom photons bevat, dan zullen query's vanuit de rechts subboom de intermediaire knoop wel moeten doorlopen.

2. starten vanuit de minimaal omsluitende photonvoxel en de hybride kd-boom steeds top-down doorlopen (HCPM);
3. starten vanuit de voxel met het hitpunt en de hybride kd-boom met een combinatie van bottom-up en top-down doorlopen.

In plaats van mogelijkheid 3 voor de HSPM te gebruiken, kunnen we ook mogelijkheid 2 gebruiken. Het top-down doorlopen vanuit de omsluitende voxel maakt enkel gebruik van de lookaheads voor photons (zie sectie 6.3.5). Photonjumps (zie sectie 6.3.6) en de bijhorende subtiliteiten zijn overbodig. Doorloopstrategieën 2 en 3 verlopen grotendeels gelijkaardig. Doorloopstrategie 3 bezoekt één voxel meer indien de deelboom met de voxel met het hitpunt als wortel geen photons bevat. We starten namelijk in deze voxel om er vervolgens onmiddellijk uit weg te springen via de photonjump. We hebben de (eenvoudigere) mogelijkheid 2 niet geïmplementeerd en getest. We verwachten geen significant verschil in totale rendertijd tussen doorloopstrategieën 2 en 3.

6.3.8 Photons op het splitsingsvlak

Het is mogelijk indien photons op het splitsingsvlak liggen, deze photons te droppen op de intermediaire knoop. Het is hierbij belangrijk de lookaheads (zie sectie 6.3.5) en de photonjumps (zie sectie 6.3.6) correct te bepalen.

Dit zou eveneens in rekening genomen moeten worden in de HRTSAH kostfunctie. Query's die intern zijn voor de ene kindvoxel en extern voor de andere kindvoxel, moeten deze gedeelde photons slechts eenmaal controleren.⁴

⁴Dit betreft enkel een aanpassing van de kosten tijdens de constructiefase. Het doorlopen van de kd-boom voor een kNN query blijft uiteraard correct.

Hoofdstuk 7

Resultaten

In dit hoofdstuk bespreken we de resultaten van onze kd-boom acceleratiestructuren gebouwd met onze RTSAH benaderingen en vergelijken deze met de SAH. Verder bespreken we de resultaten van onze hybride kd-bomen en vergelijken deze met de niet-hybride setting.

Alvorens de resultaten te bespreken, halen we eerst een aantal praktische aspecten omtrent onze testopstelling aan in sectie 7.1. In sectie 7.2 geven we de resultaten van onze RTSAH bouwprocedures voor het construeren van kd-boom acceleratiestructuren weer. In sectie 7.3 geven we de resultaten van onze Hybrid Photon Map, Hybrid Connected Photon Map en Hybrid Swapping Photon Map weer waarbij deze laatste gebruikmaakt van onze HH voor het bouwen van kd-bomen met geometrische primitieven en photons.

7.1 Praktische aspecten

Elk van onze eigen bouwalgoritmen en onze referenties zijn geïmplementeerd in het `pbrt`¹ framework [PH10]. Dit framework motiveert het ontwikkelen van nieuwe algoritmen, maar is minder gericht op diepe optimalisatie (geen SIMD instructies, geen GPU gebruik, etc.). Dit is ook de context waarin onze eigen bouwalgoritmen gezien moeten worden.

Al onze resultaten zijn bekomen met het computersysteem weergegeven in Tabel 7.1. Er is steeds gerenderd met 8 logische threads (4 fysieke processorkernen elk met hyper-threading technologie). De tijden zijn bekomen met de progressiereporters van `pbrt`².

Opmerking 7.1 (Radiance photons). Het photonmapping algoritme van `pbrt` maakt gebruik van zogenoemde radiance photons. Deze photons stellen de *uitgaande*

¹Versie 2

²We hebben ervoor gezorgd dat deze progressiereporters geen tussentijden weergeven.

7. RESULTATEN

Component	Specificatie
Operating system	Windows 8.1 Pro x86-64
CPU	Intel® Core™ i7-4770K @ 3,5 GHz (TB @ 3,9 GHz)
Werkgeheugen	8 GB DDR3 @ 1600 MHz

TABEL 7.1: De specificaties van het gebruikte computersysteem om alle resultaten te bekomen.

radiancie in een punt van de scene voor in de veronderstelling dat dit punt gelegen is op een oppervlak met een diffuse BSDF. We maken enkel gebruik van indirecte photons die de *invallende* radiancie in een punt van de scene voorstellen in combinatie met een final gathering ray tracing stap (die niet aanwezig was in pbrt).

7.2 Resultaten RTSAH

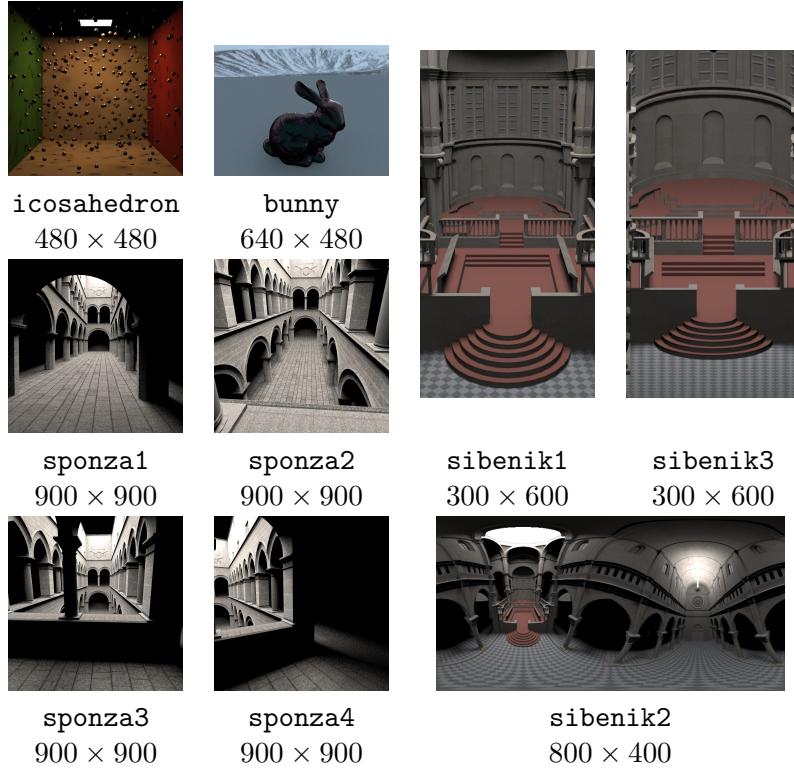
We vergelijken de bekomen kd-bomen in sectie 7.2.1 en de performantie wat het aantal intersectietesten en doorloopstappen betreft in sectie 7.2.2 ten opzichte van de SAH. Onze testscenes zijn weergegeven in Figuur 7.1.

7.2.1 Structuur van de kd-bomen

Tabel 7.2 vat de eigenschappen van de bekomen kd-bomen samen. De kd-bomen bekomen met de APSA RTSAH bevatten minder geometrische primitieven per bladvoxel ten opzichte van de SAH. Daarom worden de bladvoxels gemiddeld sneller doorlopen. Daartegenover zijn de subbomen meer verfijnd en resulteren in 30 tot 60% meer voxels en meer geduplicateerde referenties naar geometrische primitieven voor dezelfde maximale diepte. Daarom zal het gemiddeld aantal te doorlopen voxels hoger zijn, zal er meer geheugen gealloceerd moeten worden, maar kunnen ook fundamenteel betere kd-bomen bekomen worden in vergelijking met de SAH. De grotere verfijning is het gevolg van de gemiddeld lagere kosten van de splitskandidaten in vergelijking met de SAH door straalterminatie in rekening te nemen. De SAH is consequent pessimistisch en veronderstelt dat stralen die beide kindvoxels doorkruisen deze kindvoxels ook effectief bezoeken. De kosten worden hierdoor overschat maar niet steeds evenveel overschat. Dit resulteert in een suboptimale ranking van de splitskandidaten en zelfs tot de beslissing om niet meer verder te splitsen.

We zien gelijkaardige resultaten wat het aantal (lege) bladknopen en geometrische primitieven betreft voor de kd-bomen bekomen met de APOD RTSAH. Alleen de **sponza** scene is minder verfijnd in vergelijking met de SAH. Ondanks de verfijning is het aantal geometrische primitieven per bladknoop ten opzichte van de APSA RTSAH minder ideaal.

Beide RTSAH bouwtijden zijn groter dan deze van de SAH doordat de bekomen kd-bomen algemeen groter en dieper zijn. Daarom nemen we waar dat de APSA



FIGUUR 7.1: (a)-(i) Testscenes voor de RTSAH met hun resolutie. De **icosahedron** scene bestaat uit 7k, de **bunny** scene uit 69k, de **sibenik** scenes uit 80k en de **sponza** scenes uit 66k geometrische primitieven.

RTSAH in een minimale overhead resulteert in vergelijking met de SAH voor hetzelfde aantal splitsbeslissingen. De APOD RTSAH doet slechter doordat het rasterizeren op de hoogste niveaus van de kd-boom computationeel duur is. Onze APOD en APSA RTSAH (CPU) bouwprocedures zijn niet geoptimaliseerd en cachen geen oppervlaktes of rasterizatie info voor geometrische primitieven. Deze verbeteringen kunnen nog steeds resulteren in een reductie van de bouwtijden, maar zijn niet getest.

7.2.2 Performantie

Tabellen 7.3 en 7.4 tonen respectievelijk het verschil in intersectietesten met geometrische primitieven en het verschil in het aantal te doorlopen voxels voor verschillende stralen. Zoals verwacht door de structuur van de bekomen kd-bomen moeten meer voxels doorlopen worden voor de APOD en APSA RTSAH. Dit door de meer verfijnde subbomen in vergelijking met de SAH.

Belangrijker is de grote winst in intersectietesten met geometrische primitieven voor de APSA RTSAH. We bekomen verminderingen tot 47% voor primaire stralen voor de complexe **sibenik** en **sponza** scenes. In deze scenes is straalterminatie

7. RESULTATEN

scene	#bladknopen			#lege bladknopen		
	SAH	APOD	APSA	SAH	APOD	APSA
icosahedron	65 k	87 k	94 k	6 k	6 k	6 k
bunny	873 k	1,17 M	1,22 M	198 k	224 k	219 k
sibenik	607 k	829 k	888 k	127 k	142 k	156 k
sponza	502 k	452 k	823 k	130 k	109 k	176 k
scene	bouwtijd			#referenties naar geometrische primitieven		
	SAH	APOD	APSA	SAH	APOD	APSA
icosahedron	0,1s	3,3s	0,2s	170 k	224 k	239 k
bunny	0,6s	30,2s	2,1s	1,80 M	2,46 M	2,53 M
sibenik	0,7s	31,7s	2,5s	1,97 M	2,87 M	2,95 M
sponza	0,6s	16,3s	2,1s	1,64 M	1,66 M	2,59 M
scene	maximaal #geometrische primitieven/bladknoop			gemiddeld #geometrische primitieven/niet-lege bladknoop		
	SAH	APOD	APSA	SAH	APOD	APSA
icosahedron	11	8	8	2,86	2,78	2,71
bunny	12	11	11	2,67	2,59	2,52
sibenik	81	72	36	4,10	4,18	4,03
sponza	57	54	44	4,41	4,84	4,01

TABEL 7.2: Structuur van de kd-bomen bekomen met de SAH, APOD en APSA RTSAH bouwheuristieken voor de testscenes. De maximale diepte van de bekomen kd-boom is steeds 24 voor de **bunny** en 29 voor de andere testscenes.

belangrijk om in rekening te nemen in een bouwheuristiek aangezien elke straal uiteindelijk een oppervlak raakt. We bekomen een kleine toename in het aantal intersectietesten voor de **bunny** scene die uit één object en een vloerplaat bestaat. De APSA RTSAH heeft minder intersectietesten nodig voor het object, maar presteert iets slechter voor de vloerplaat. Ondanks het feit dat we enkel op externe stralen focussen nemen we zelfs reducties waar tot 41% voor schaduwstralen (wanneer de voxels in volgorde doorlopen worden). Dit komt o.a. door het grote aantal geduplicateerde referenties naar geometrische primitieven.

De APSA RTSAH resulteert algemeen in grotere winsten in vergelijking met de APOD RTSAH voor de meer complexe scenes (zoals de **sibenik** en **sponza** scenes). Door enkel de inkomende richting orthogonaal op het splitsingsvlak in rekening te nemen bij de APOD RTSAH, merken we een verschil tussen **sibenik1** (perspectief camera) en **sibenik3** (orthografische camera). De APSA RTSAH daarentegen resulteert in gelijkaardige winsten onafhankelijk van het gebruikte cameratype.

De false color afbeeldingen in Figuren 7.2-7.9 tonen een meer gedetailleerde

vergelijking tussen de verschillende bouwheuristieken. Hier zien we dat er bijvoorbeeld minder intersectietesten in de buurt van de zuilen van de **sponza** scenes uitgevoerd worden. Dit door het in rekening nemen van de straalterminatie in onze RTSAH benaderingen.

Tabel 7.5 toont het verschil in totale rendertijd. Ondanks de significante reducties in intersectietesten van onze RTSAH bouwprocedures ten opzichte van de SAH, is de impact op de totale rendertijd minder uitgesproken. Onze profiler toont dat (afhankelijk van de scene en gebruikte bouwheuristiek) minder dan 35% van de totale rendertijd gespendeerd wordt aan het op intersectie testen van geometrische primitieven en het doorlopen van de acceleratiestructuur. Daarom zijn de reducties in totale rendertijd kleiner.

scene	Δ intersectietesten voor primaire stralen		Δ intersectietesten voor schaduwstralen	
	APOD	APSA	APOD	APSA
icosahedron	-1,66%	-0,49%	-1,78%	-8,49%
bunny	-1,84%	0,92%	-1,47%	-3,65%
sibenik1	-13,46%	-18,70%	-3,88%	-13,84%
sibenik2	-10,56%	-20,20%	-9,10%	-17,68%
sibenik3	-17,92%	-17,92%	-8,03%	-9,34%
sponza1	-12,49%	-25,47%	-3,06%	-22,05%
sponza2	-38,00%	-46,74%	-5,96%	-29,24%
sponza3	-20,51%	-30,69%	-18,93%	-40,84%
sponza4	-12,92%	-27,42%	-14,78%	-33,75%

TABEL 7.3: Het verschil in het aantal intersectietesten met geometrische primitieven voor primaire stralen en schaduwstralen voor de APOD en APSA RTSAH relatief t.o.v. de SAH. Een negatief percentage komt overeen met een winst en visa versa.

Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

7.3 Resultaten hybride photonmappen

In sectie 7.3.1 bekijken we de structuur van de HCPM kd-bomen. We kijken specifiek naar de connecties (via ropes) tussen de kd-boom acceleratiestructuur en de kd-boom photonmap. In sectie 7.3.2 bekijken we de structuur van de HSPM kd-bomen. We vergelijken de performantie wat de totale rendertijd en bouwtijd betreft van de hybride photonmappen (HAPM, HSPM, HCPM) ten opzichte van de niet-hybride photonmappen (PM) in sectie 7.3.3. Onze testscenes zijn weergegeven in Figuur 7.10.

Voor de HH bouwheuristiek gebruiken we de volgende waarden voor de constante

7. RESULTATEN

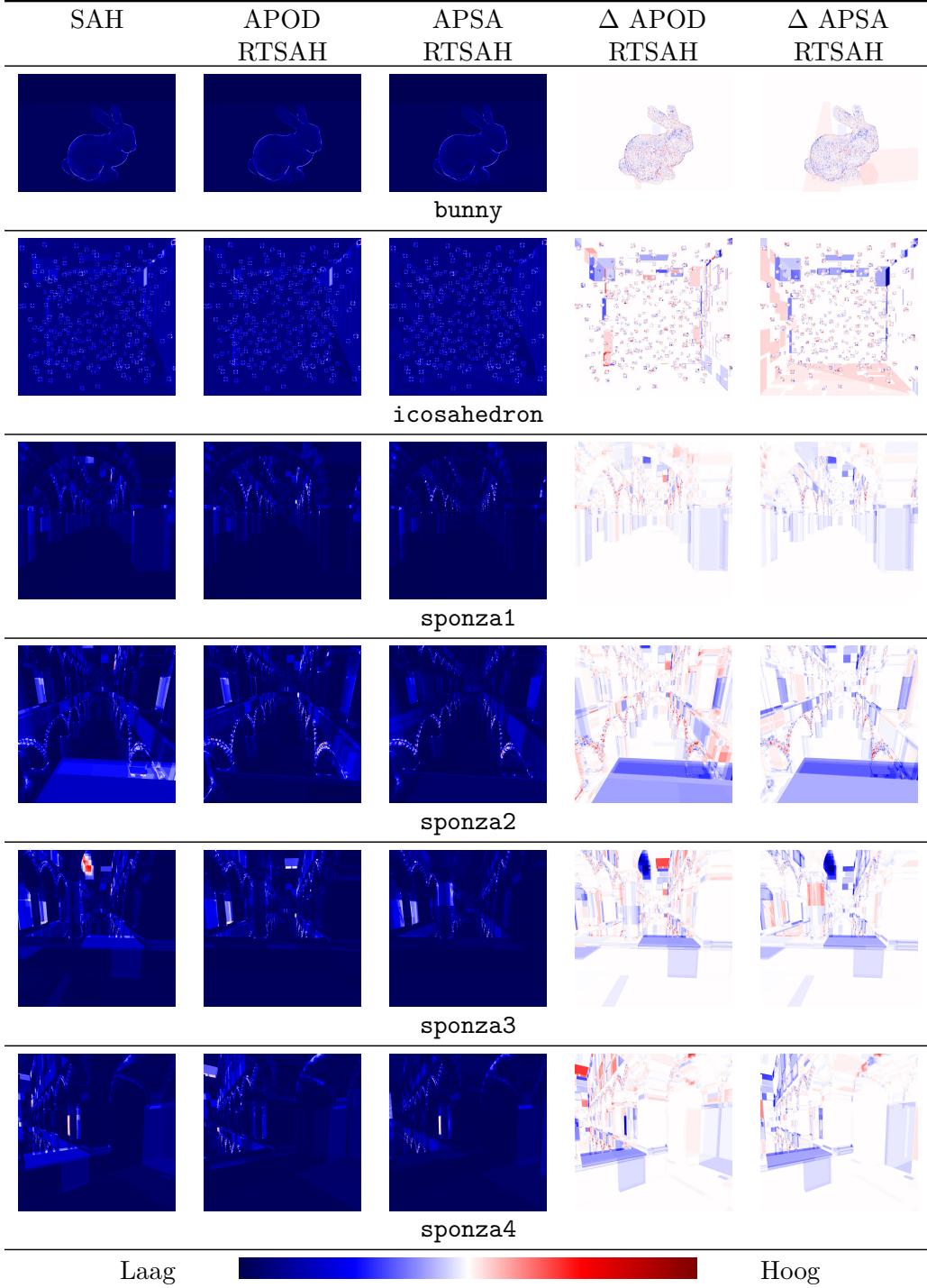
scene	Δ doorlopen voxels voor primaire stralen		Δ doorlopen voxels voor schaduwstralen	
	APOD	APSA	APOD	APSA
icosahedron	1,15%	-0,58%	2,95%	-0,79%
bunny	5,47%	2,55%	7,43%	2,73%
sibenik1	13,42%	11,61%	15,47%	15,55%
sibenik2	8,01%	6,65%	12,34%	12,01%
sibenik3	13,69%	10,74%	14,34%	15,24%
sponza1	8,17%	7,57%	-2,16%	10,56%
sponza2	2,95%	20,04%	-0,45%	14,39%
sponza3	4,55%	12,18%	1,25%	10,82%
sponza4	18,57%	7,10%	3,41%	10,44%

TABEL 7.4: Het verschil in het aantal doorlopen voxels voor primaire stralen en schaduwstralen voor de APOD en APSA RTSAH relatief t.o.v. de SAH. Een negatief percentage komt overeen met een winst en visa versa. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

scene	Totale rendertijd SAH	Δ totale rendertijd	
		APOD	APSA
icosahedron	41,2s	-3,6%	-4,4%
bunny	48,7s	1,8%	-1,0%
sibenik1	80,8s	-1,6%	-3,5%
sibenik2	129,9s	-3,2%	-4,8%
sibenik3	80,6s	0,4%	-0,1%
sponza1	261,9s	-3,9%	-4,8%
sponza2	311,5s	-7,4%	-7,6%
sponza3	281,2s	-7,5%	-8,5%
sponza4	253,0s	-0,4%	-3,1%

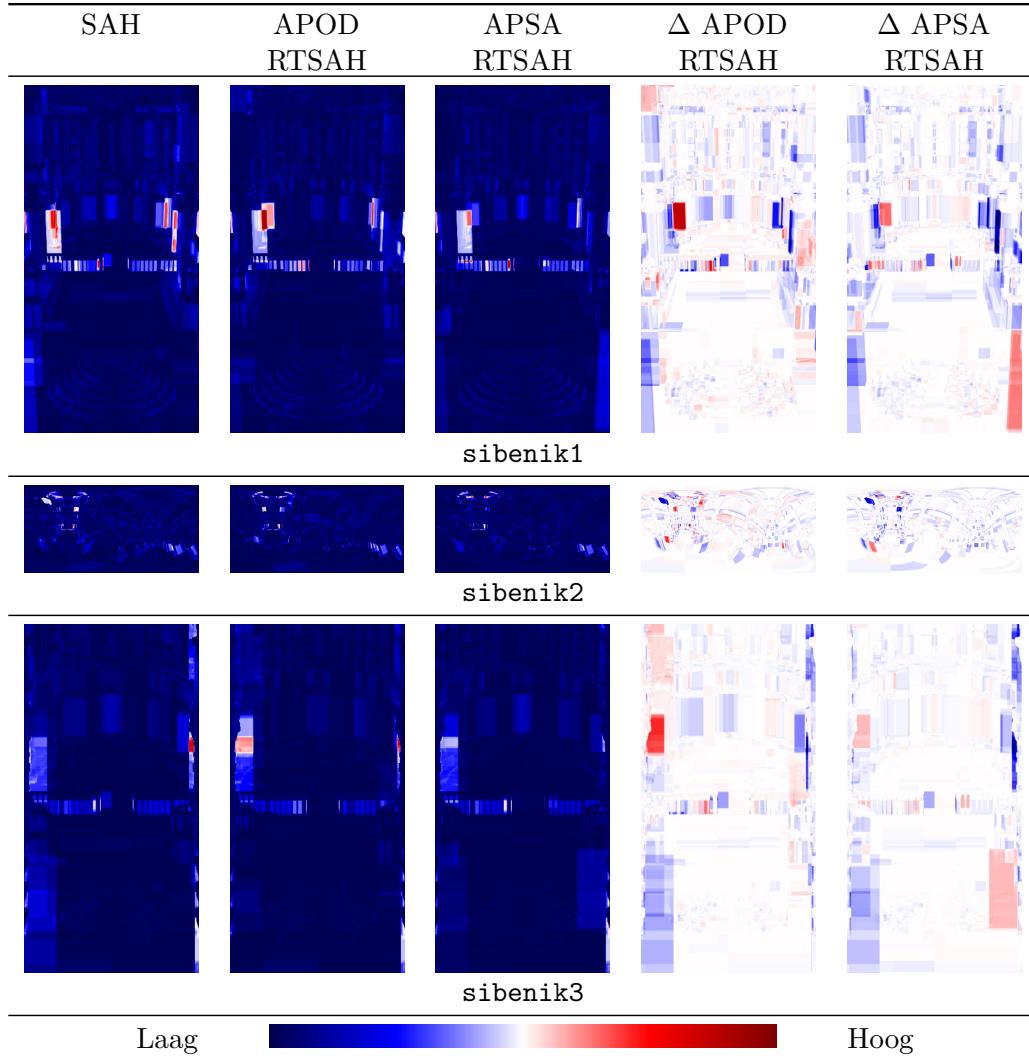
TABEL 7.5: Het verschil in de totale rendertijd voor de APOD en APSA RTSAH relatief t.o.v. de SAH. Een negatief percentage komt overeen met een winst en visa versa. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

7.3. Resultaten hybride photonmappen



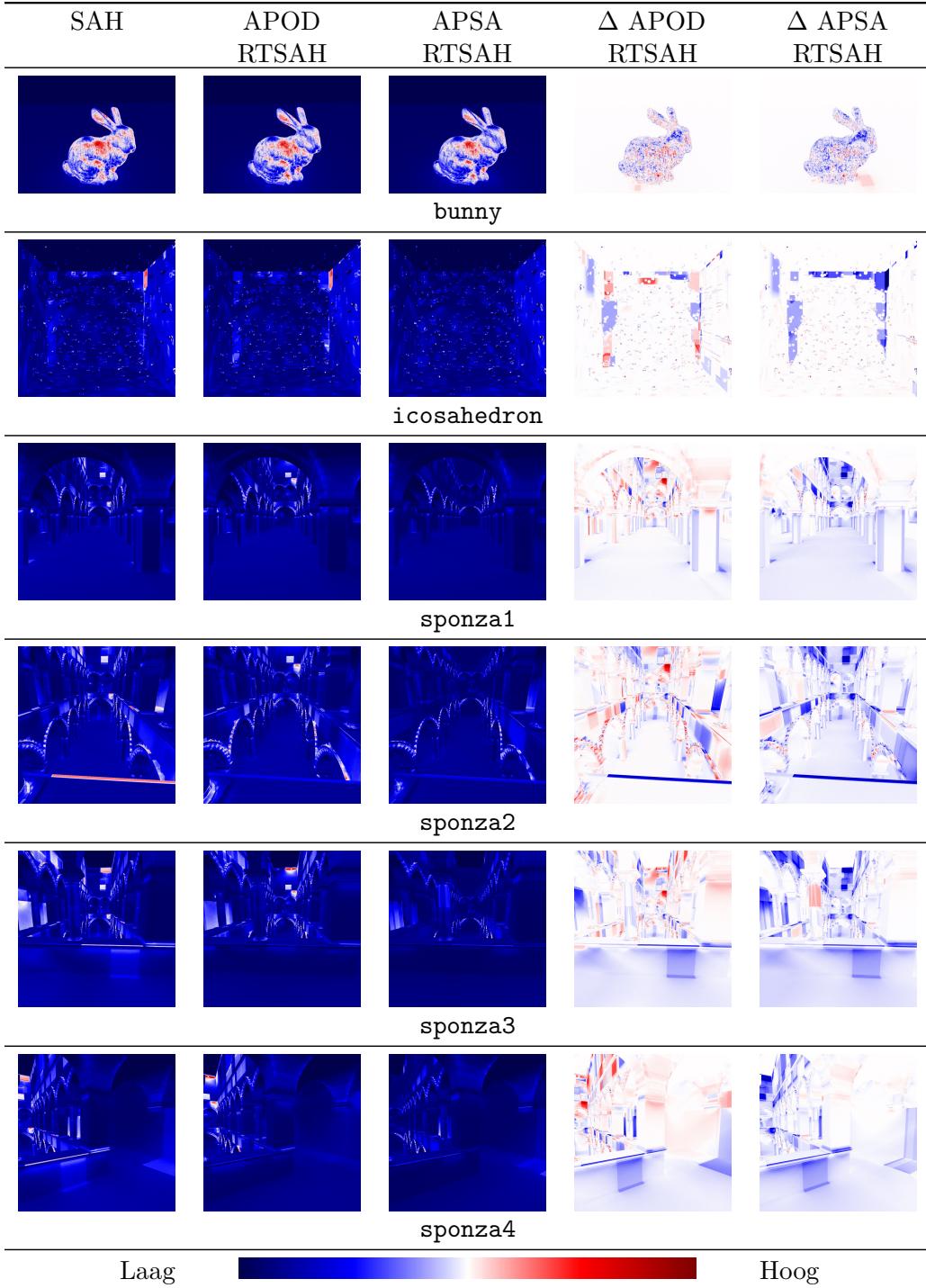
FIGUUR 7.2: False color afbeeldingen van het aantal **intersectietesten voor primaire stralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scènes zijn gerenderd met een Whitted ray tracer [Whi80] en 512 spp.

7. RESULTATEN



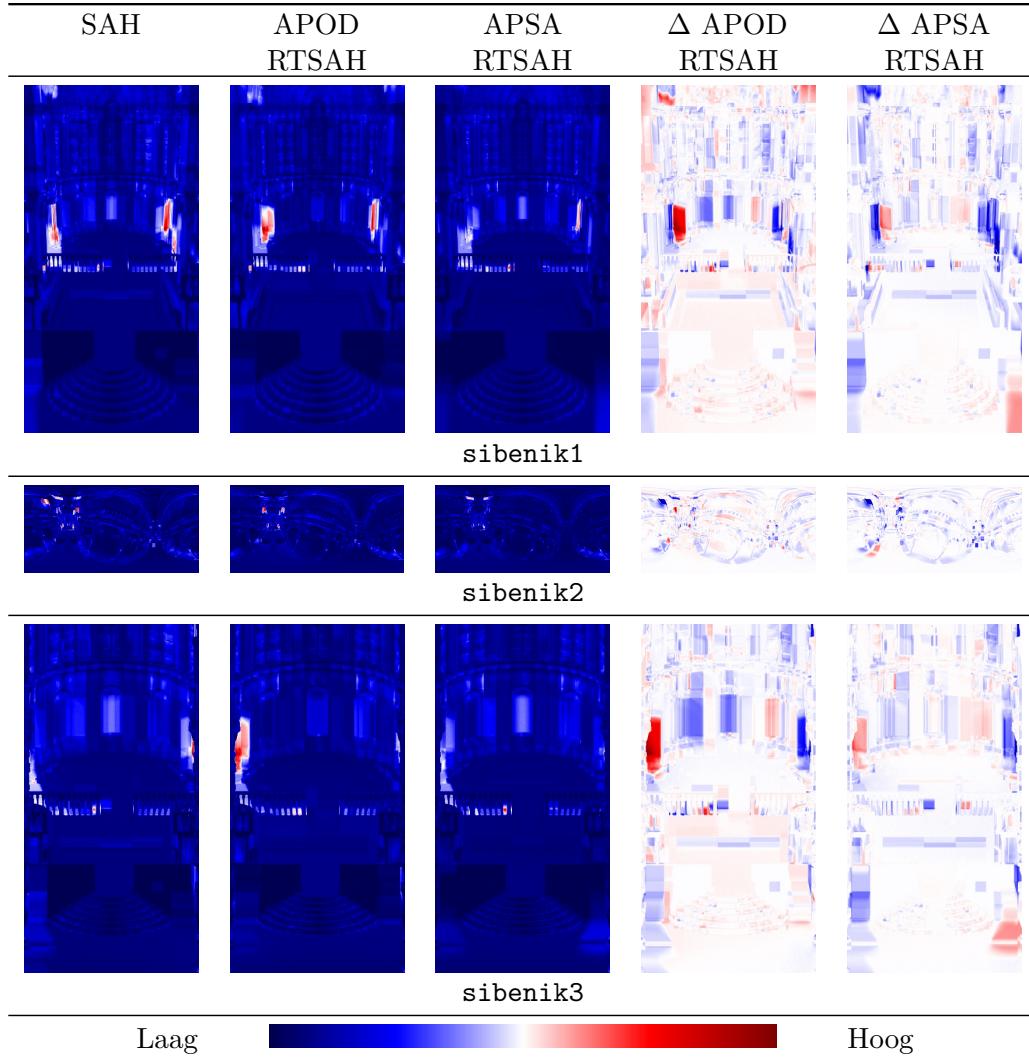
FIGUUR 7.3: False color afbeeldingen van het aantal **intersectietesten voor primaire stralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

7.3. Resultaten hybride photonmappen



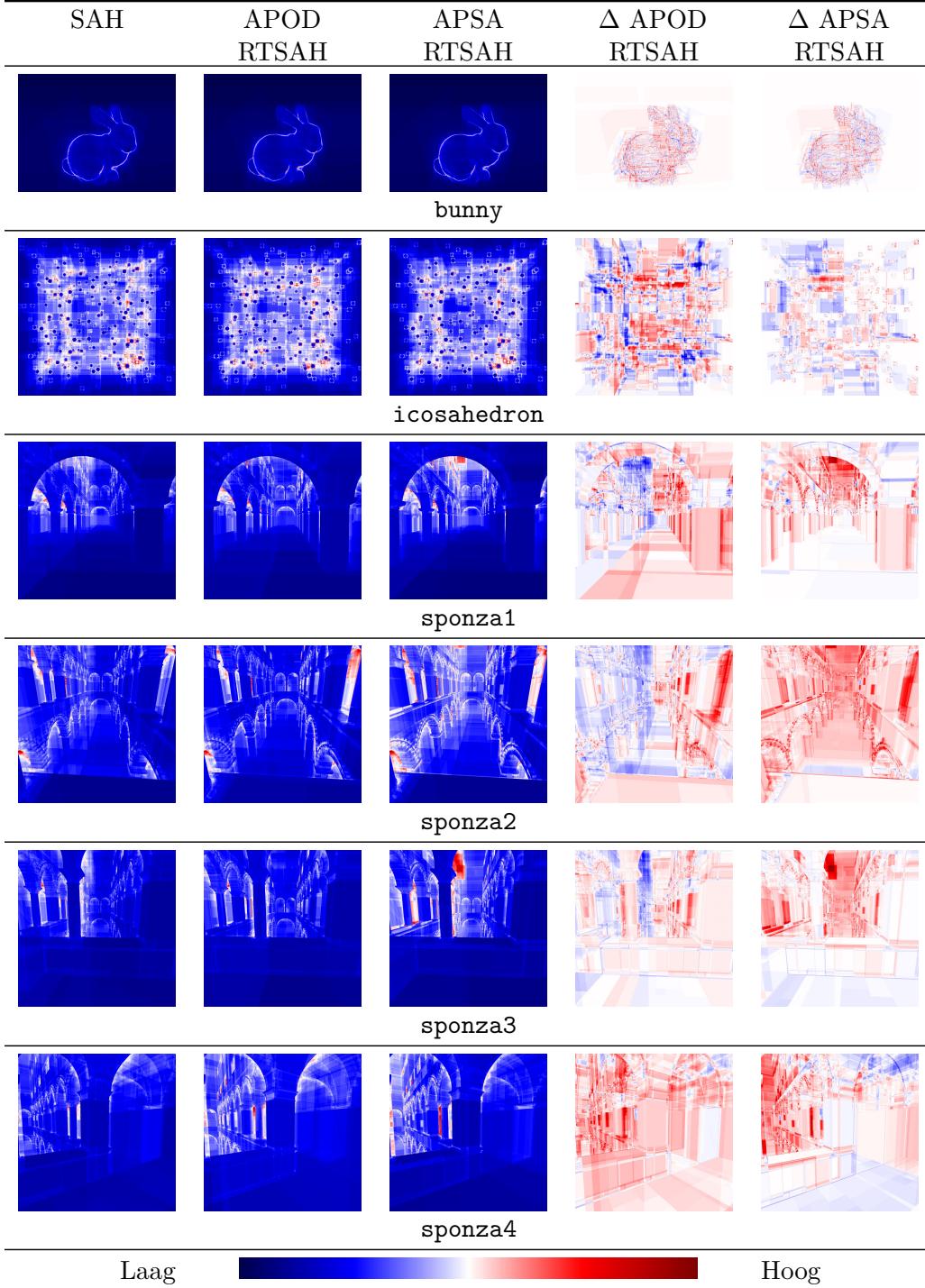
FIGUUR 7.4: False color afbeeldingen van het aantal **intersectietesten voor schaduwstralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scènes zijn gerenderd met een Whitted ray tracer [Whi80] en 512 spp.

7. RESULTATEN



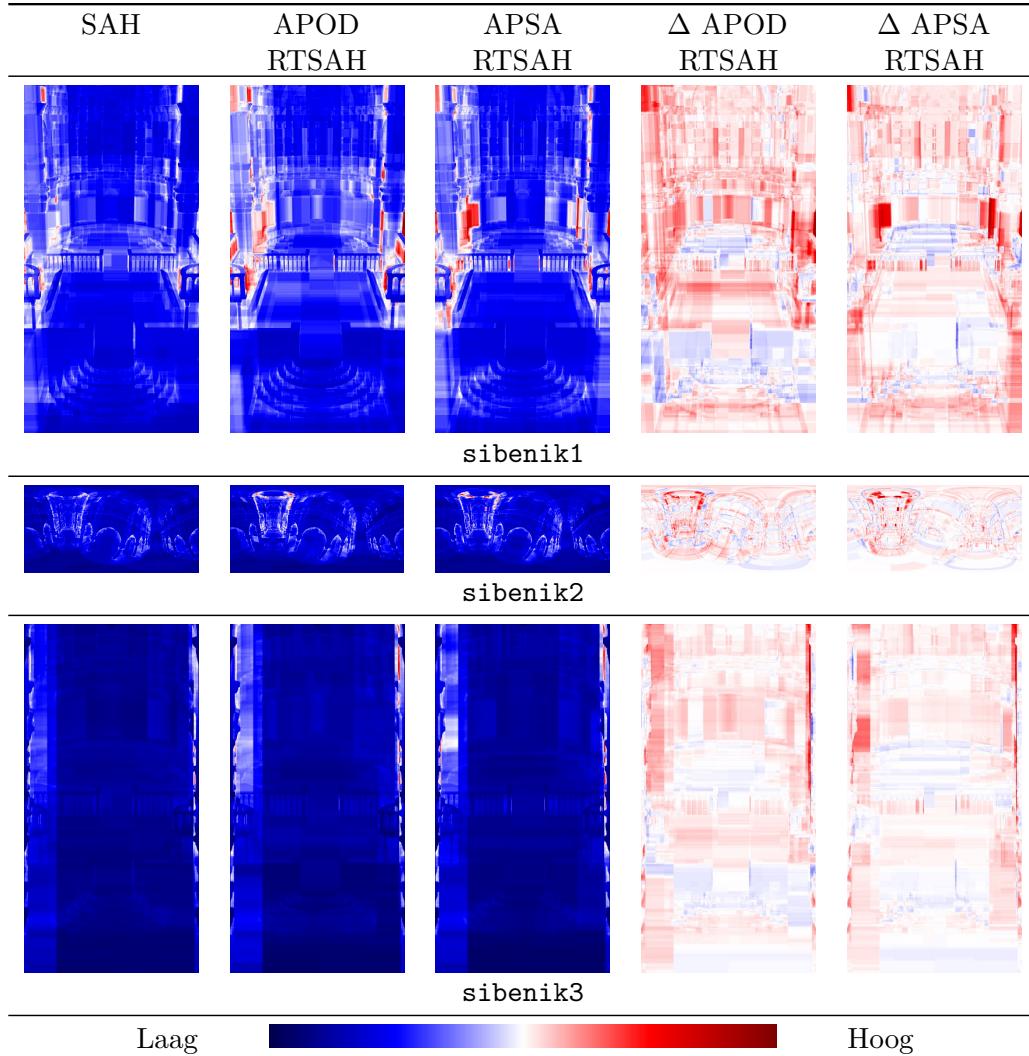
FIGUUR 7.5: False color afbeeldingen van het aantal **intersectietesten voor schaduwstralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

7.3. Resultaten hybride photonmappen



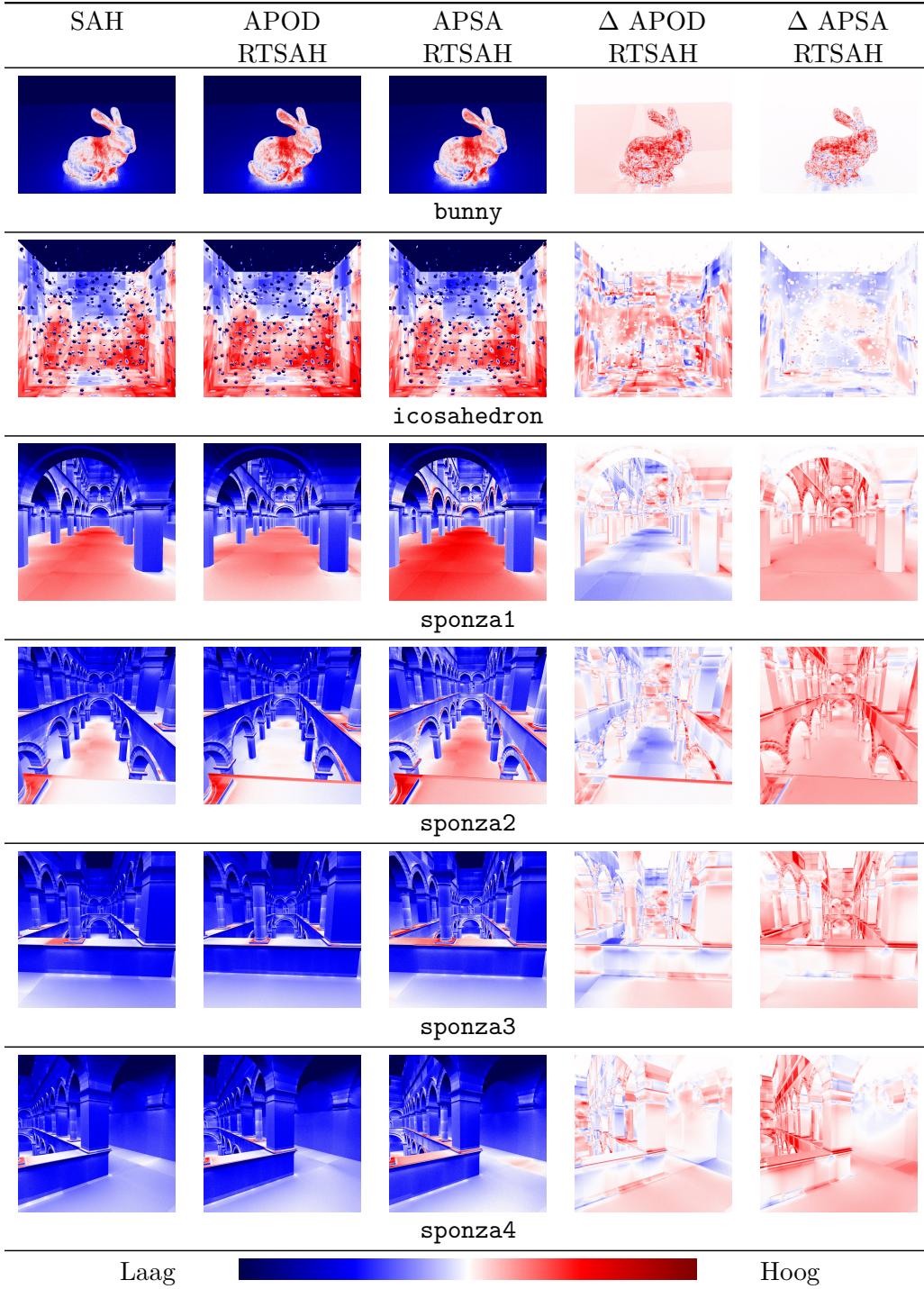
FIGUUR 7.6: False color afbeeldingen van het aantal **doorlopen voxels voor primaire stralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scènes zijn gerenderd met een Whitted ray tracer [Whi80] en 512 spp.

7. RESULTATEN



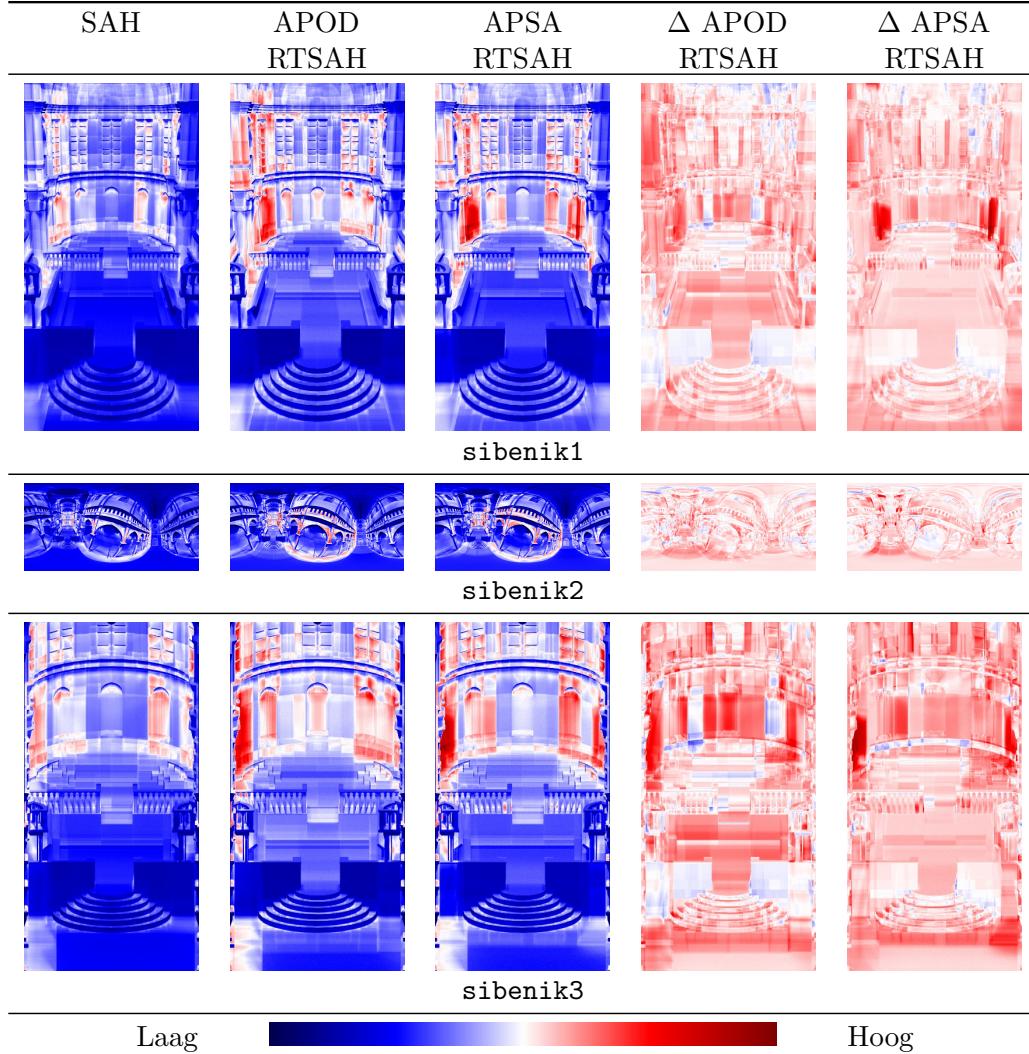
FIGUUR 7.7: False color afbeeldingen van het aantal **doorlopen voxels voor primaire stralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.

7.3. Resultaten hybride photonmappen

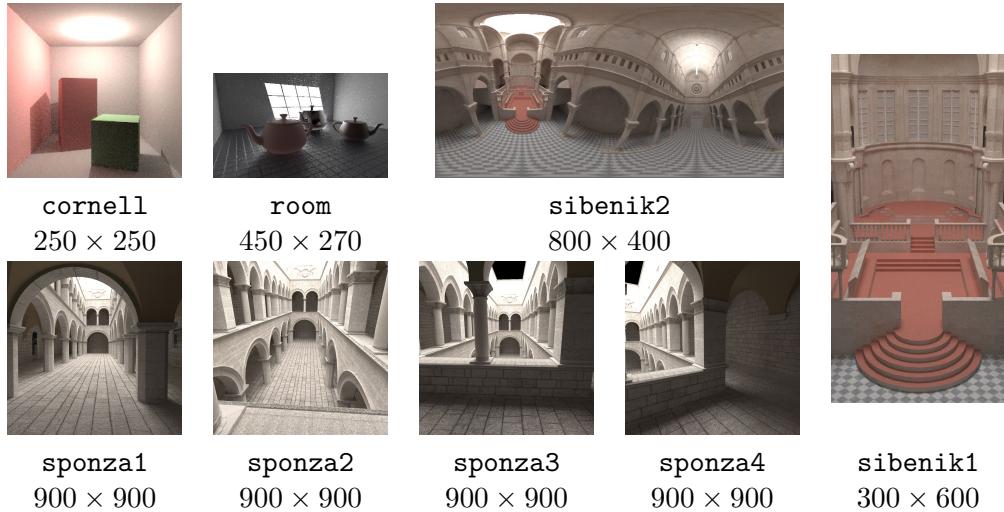


FIGUUR 7.8: False color afbeeldingen van het aantal **doorlopen voxels voor schaduwstralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scènes zijn gerenderd met een Whitted ray tracer [Whi80] en 512 spp.

7. RESULTATEN



FIGUUR 7.9: False color afbeeldingen van het aantal **doorlopen voxels voor schaduw stralen** voor de SAH, APOD en APSA RTSAH en het verschil (blauw (rood) komt overeen met winst (verlies)) tussen de SAH en RTSAH benaderingen. De gebruikte schaal is lineair. Alle scenes zijn gerendered met een Whitted ray tracer [Whi80] en 512 spp.



FIGUUR 7.10: (a)-(i) Testscenes voor de hybride photonmappen met hun resolutie. De **cornell** scene bestaat uit 30k, de **bunny** scene uit 6,9k, de **sibenik** scenes uit 80k en de **sponza** scenes uit 66k geometrische primitieven. De afbeeldingen zijn gerenderd met 100k indirecte photons, een maximale querystraal van 1,0 en maximaal 32 photons per query.

kosttermen:

$$\mathcal{C}_{\text{trav}}^{\text{RT}} = 1 \quad [\text{PH10}] \quad (7.3.1)$$

$$\mathcal{C}_{\text{isect}}^{\text{RT}} = 80 \quad [\text{PH10}] \quad (7.3.2)$$

$$\mathcal{C}_{\text{check}}^{\text{kNN}} = 2 \quad (7.3.3)$$

7.3.1 Structuur van de HCPM kd-bomen

Indien er een hit van een straal met een geometrisch primitief in een voxel plaatsvindt, moet een k-nearest neighbor query vanuit dit hitpunt enkel het volume van de bijhorende rope in rekening nemen. Dit vermindert de doorlooptijd (afdalen en controleren van photons tijdens de afdaling) in vergelijking met een query die steeds op de wortel van de photonmap moet starten. In Figuren 7.11-7.14 zijn de dieptehistogrammen van de ropes (van niet-lege bladknopen) in de photonmap weergegeven voor verschillende photonaantallen en querystralen voor de testscenes.

Naarmate de querystraal R_{\max} groter wordt, is het triviaal dat de ropes minder diep gelegen zullen zijn. We breiden elke niet-lege bladvoxel van de acceleratiestructuur uit met R_{\max} aan weerszijden in elk van de drie primaire richtingen. Het balkvormige gebied opgespannen door deze uitgebreide voxel wordt groter naarmate de querystraal toeneemt. Het gebied opgespannen door de bijhorende rope wordt eveneens groter. De bijhorende rope zal hierdoor dichter en dus minder diep tegen de wortel van de photonmap gelegen zijn.

7. RESULTATEN

Naarmate het aantal photons toeneemt, wordt de photonmap dieper. Indien de lokale photondensiteit grotendeels lineair verandert na toename van het aantal, zal de diepte van de ropes niet veel wijzigen. Dit is voor de testscenes waar te nemen vanaf 50k photons. Indien de lokale photondensiteit niet-lineair verandert na toename van het aantal en de bouwheuristiek afhankelijk is van deze photondensiteit (VVH, VVH1), zal de diepte van de ropes kunnen toenemen. Dit is voor de testscenes waar te nemen bij de overgang van 10k naar 50k photons.

De photonmappen zijn steeds gebouwd met de VVH bouwheuristiek. De bekomen resultaten zijn gelijkaardig voor de VVH1. Voor de BH worden de photonmappen expliciet gebalanceerd. De maximale diepte van de photonmappen zal hierdoor minder groot kunnen zijn ten opzichte van niet-gebalanceerde photonmappen. Bijgevolg zullen de ropes algemeen minder diep gelegen zijn en zal de performantiewinst kleiner zijn.

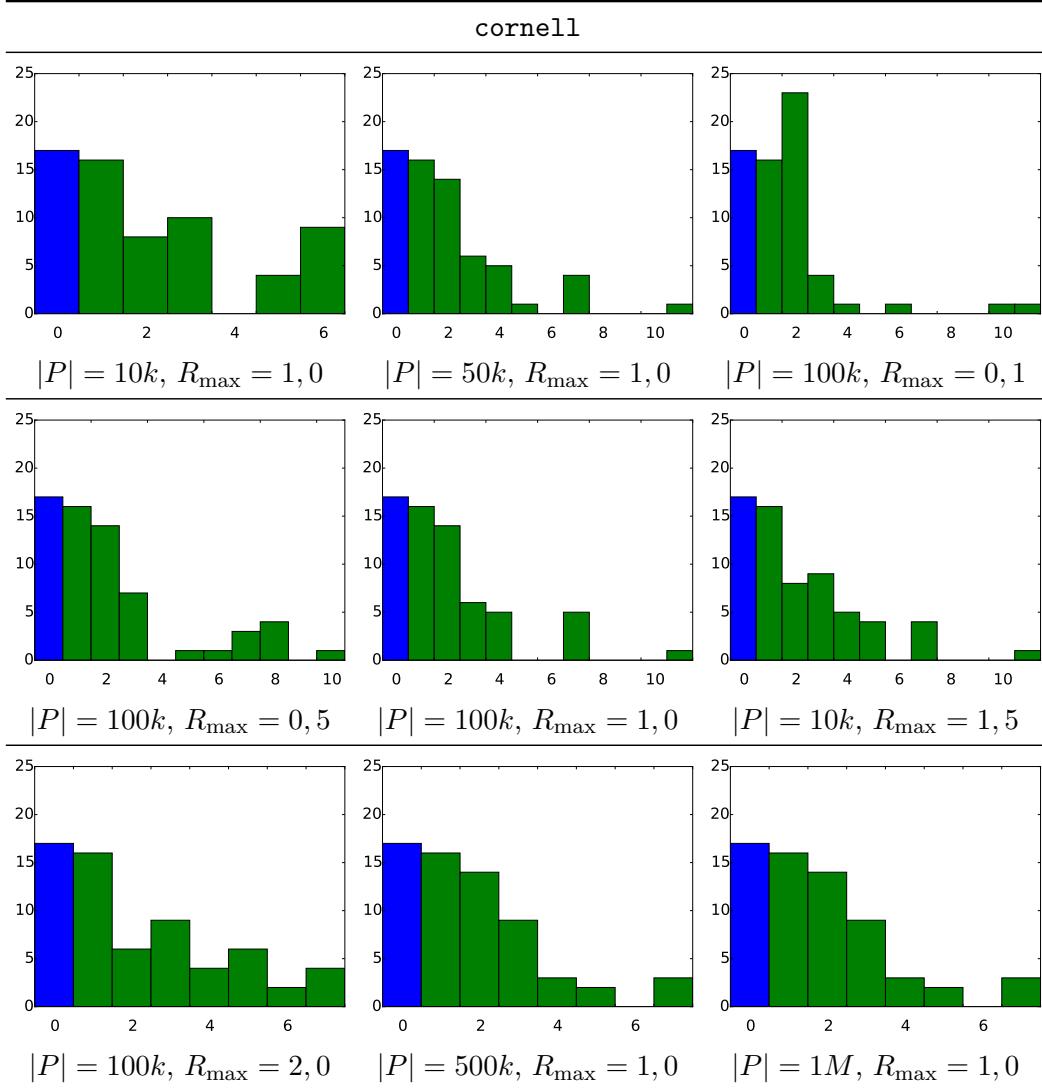
7.3.2 Structuur van de HSPM kd-bomen

De HSPM kd-bomen worden gebouwd met de HH (zie sectie 5.5). Deze bouwheuristiek kapselt op zijn beurt een bouwheuristiek voor kd-boom acceleratiestructuren en kd-boom photonmappen in. We onderscheiden de volgende twee combinaties: APSA+VVH en APSA+VVH1. APSA verwijst enerzijds naar de gebruikte benadering van de visibiliteits- en hitprobabiliteiten en anderzijds naar een praktisch bouwalgoritme voor de RTSAH. VVH en VVH1 verwijzen naar de bouwheuristiek voor een photonmap.

In Tabel 7.6 is de structuur van de HSPM kd-bomen bekomen met de HH (APSA+VVH) bouwheuristiek weergegeven voor de testscenes voor 1k en 1M photons. De bekomen hybride kd-bomen zijn veel dieper dan de overeenkomstige niet-hybride kd-boom acceleratiestructuren en photonmappen. Enerzijds moeten we meer data, zowel geometrische primitieven als photons, opslaan in eenzelfde kd-boom, anderzijds is een grote diepte noodzakelijk om grote photonclusters te vermijden. Het nastreven van een fijne spatiale opdeling van de photons, heeft ook zijn gevolgen voor de opdeling van de geometrische primitieven. Hoe ruwer de tessellatie van de geometrie, hoe meer geometrische primitieven een splitsingsvlak overlappen en hoe groter de duplicatie. De bekomen hybride kd-bomen bevatten meer dan tweemaal zoveel referenties naar geometrische primitieven in vergelijking met de niet-hybride kd-bomen bekomen met de SAH (zie Tabel 7.2).

De puntdata (photons) en niet-puntdata (geometrische primitieven) werken elkaar deels tegen. Dit is ook waar te nemen in het gemiddeld aantal geometrische primitieven voor de verschillende knoopconfiguraties. Deze zijn iets kleiner bij de SAH en APSA RTSAH (zie Tabel 7.2). Het maximaal aantal geometrische primitieven voor de verschillende knoopconfiguraties is gelijkaardig voor 1k photons. Bij 1M photons nemen we een piek waar bij de `room` en `sibenik` scenes op een intermediaire knoop. Beide knopen bevinden zich op de maximale diepte voor geometrische primitieven.

7.3. Resultaten hybride photonmappen

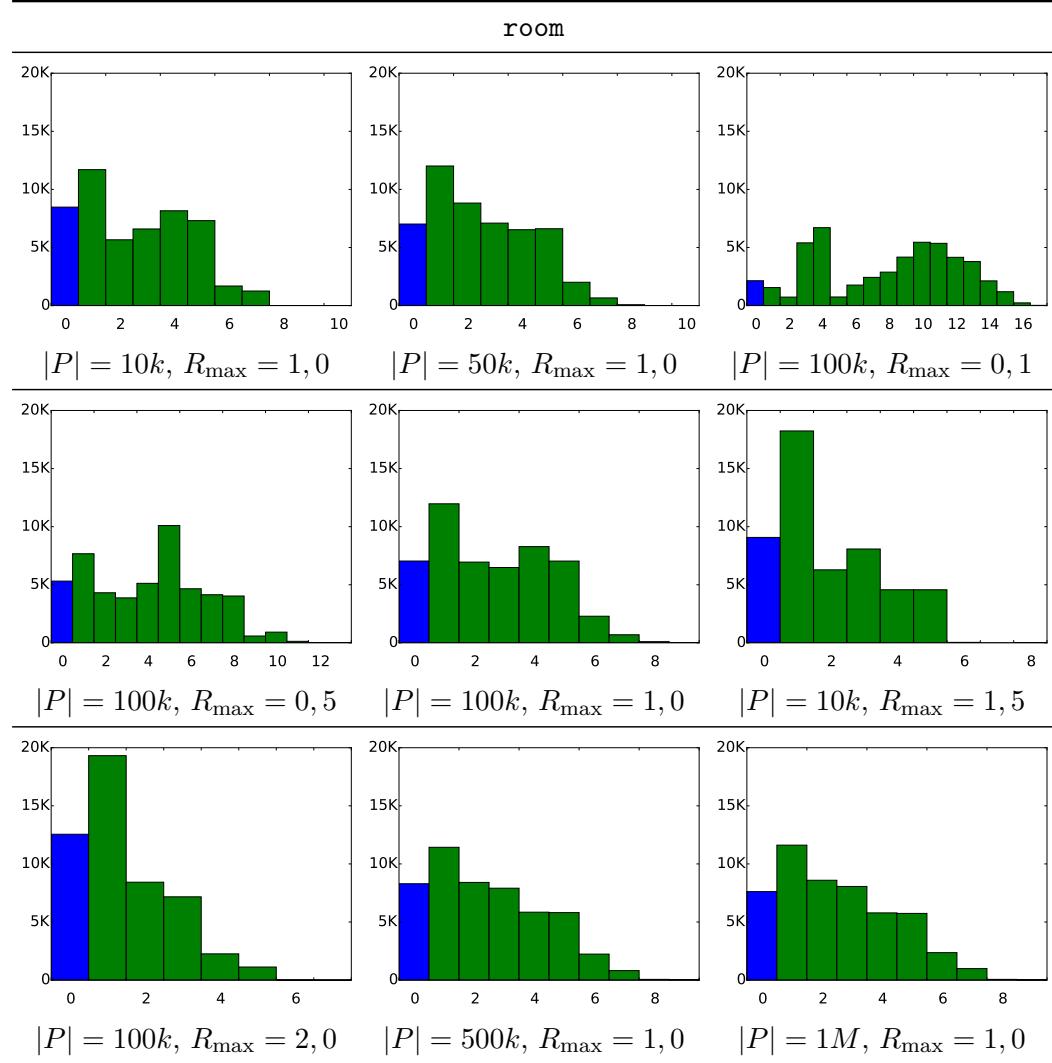


FIGUUR 7.11: Histogrammen van de diepte van de ropes (van niet-lege bladknopen) in de photonmap voor verschillende photonaantallen en querystralen voor **cornell**. Een diepte van nul (blauw) komt overeen met de wortel van de photonmap. Voor de andere dieptes (groen) doet de HCPM het beter dan de gewone photonmap. De photonmap is steeds gebouwd met de VVH.

Bij de HSPM worden de photons in rekening genomen bij de opdeling van de geometrische primitieven in tegenstelling tot de HAPM. Eenmaal deze niet-puntdaten is opgedeeld, wordt de kd-boom verder opgedeeld voor de puntdaten. Net zoals bij de HAPM wordt een groot gedeelte van de photons opgedeeld met de VVH ((d) en (h)).

Bij de overgang van 1k naar 1M photons, zien we dat de HSPM consistent is wat betreft de opdeling van de geometrische primitieven. De lokale photondensiteit

7. RESULTATEN



FIGUUR 7.12: Histogrammen van de diepte van de ropes (van niet-lege bladknopen) in de photonmap voor verschillende photonaantallen en querystralen voor **room**. Een diepte van nul (blauw) komt overeen met de wortel van de photonmap. Voor de andere dieptes (groen) doet de HCPM het beter dan de gewone photonmap. De photonmap is steeds gebouwd met de VVH.

verandert grotendeels lineair. Dit heeft een gelijkaardige opdeling van de geometrische primitieven tot gevolg.

7.3.3 Performantie

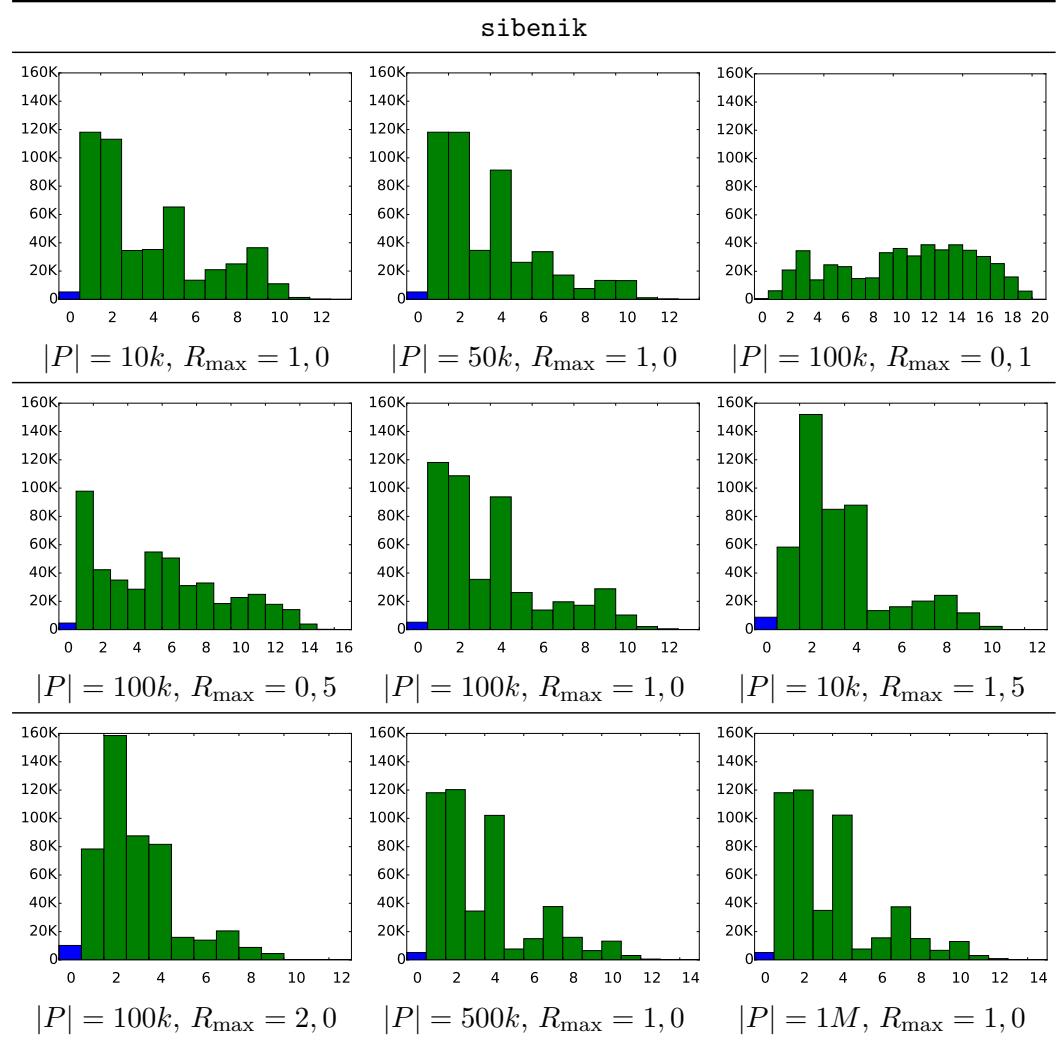
In Tabellen 7.7-7.14 zijn de totale rendertijden weergegeven voor de testscenes, gebruikmakende van de niet-hybride en hybride photonmappen. De totale ray tracing tijd en k-nearest neighbor querytijd behoren tot deze totale rendertijd. De

$ P = 1k$									
scene	a	b	c	d	e	f	g	h	i
cornell	27	34	1k	50k	0,3k	0,5k	0,5k	50k	48
room	34	46	2k	44k	140k	5k	121k	41k	18k
sibenik	36	49	5k	50k	1,07M	18k	926k	41k	137k
sponza	36	48	3k	36k	1,23M	20k	1,05M	27k	169k
scene	j	k	l	m	n	o	p	q	r
cornell	5k	10	2,1	6	2,4	6	2,6	1	2
room	406k	8	2,2	24	3,6	26	3,2	1	2
sibenik	4,11M	23	3,8	32	5,4	48	4,3	1	2
sponza	3,94M	36	3,0	42	4,9	44	3,7	1	2
$ P = 1M$									
scene	a	b	c	d	e	f	g	h	i
cornell	39	49	2k	435k	0,4k	1k	1k	436k	48
room	39	57	5k	430k	143k	10k	126k	419k	23k
sibenik	40	58	13k	431k	1,05M	41k	940k	401k	117k
sponza	40	57	18k	404k	1,15M	63k	994k	372k	139k
scene	j	k	l	m	n	o	p	q	r
cornell	7k	10	2,2	6	2,6	6	2,5	1	2
room	436k	138	2,4	24	3,7	26	3,1	1	2
sibenik	4,31M	559	4,2	32	5,1	37	4,3	1	2
sponza	4,06M	88	3,5	41	4,8	44	3,7	1	2

TABEL 7.6: Structuur van de HSPM kd-bomen bekomen met de HH (APSA+VVH) bouwheuristiek voor de testscenes voor 1k en 1M photons.

- (a) Maximale diepte waarop geometrische primitieven voorkomen
- (b) Maximale diepte waarop photons voorkomen
- (c) #Intermediaire knopen met $G_{V_k} \neq \emptyset \neq P_{V_k}$
- (d) #Intermediaire knopen met $G_{V_k} = \emptyset \neq P_{V_k}$
- (e) #Intermediaire knopen met $G_{V_k} = \emptyset = P_{V_k}$
- (f) #bladknopen met $G_{V_k} \neq \emptyset \neq P_{V_k}$
- (g) #bladknopen met $G_{V_k} \neq \emptyset = P_{V_k}$
- (h) #bladknopen met $G_{V_k} = \emptyset \neq P_{V_k}$
- (i) #bladknopen met $G_{V_k} = \emptyset = P_{V_k}$
- (j) #referenties naar geometrische primitieven
- (k) Maximaal $|G_{V_k}| / \text{intermediaire knoop met } G_{V_k} \neq \emptyset \neq P_{V_k}$
- (l) Gemiddeld $|G_{V_k}| / \text{intermediaire knoop met } G_{V_k} \neq \emptyset \neq P_{V_k}$
- (m) Maximaal $|G_{V_k}| / \text{bladknoop met } G_{V_k} \neq \emptyset \neq P_{V_k}$
- (n) Gemiddeld $|G_{V_k}| / \text{bladknoop met } G_{V_k} \neq \emptyset \neq P_{V_k}$
- (o) Maximaal $|G_{V_k}| / \text{bladknoop met } G_{V_k} \neq \emptyset = P_{V_k}$
- (p) Gemiddeld $|G_{V_k}| / \text{bladknoop met } G_{V_k} \neq \emptyset = P_{V_k}$
- (q) Maximaal $|P_{V_k}| / \text{intermediaire knoop met } G_{V_k} = \emptyset \neq P_{V_k}$
- (r) Maximaal $|P_{V_k}| / \text{bladknoop met } G_{V_k} \neq \emptyset \neq P_{V_k}$

7. RESULTATEN



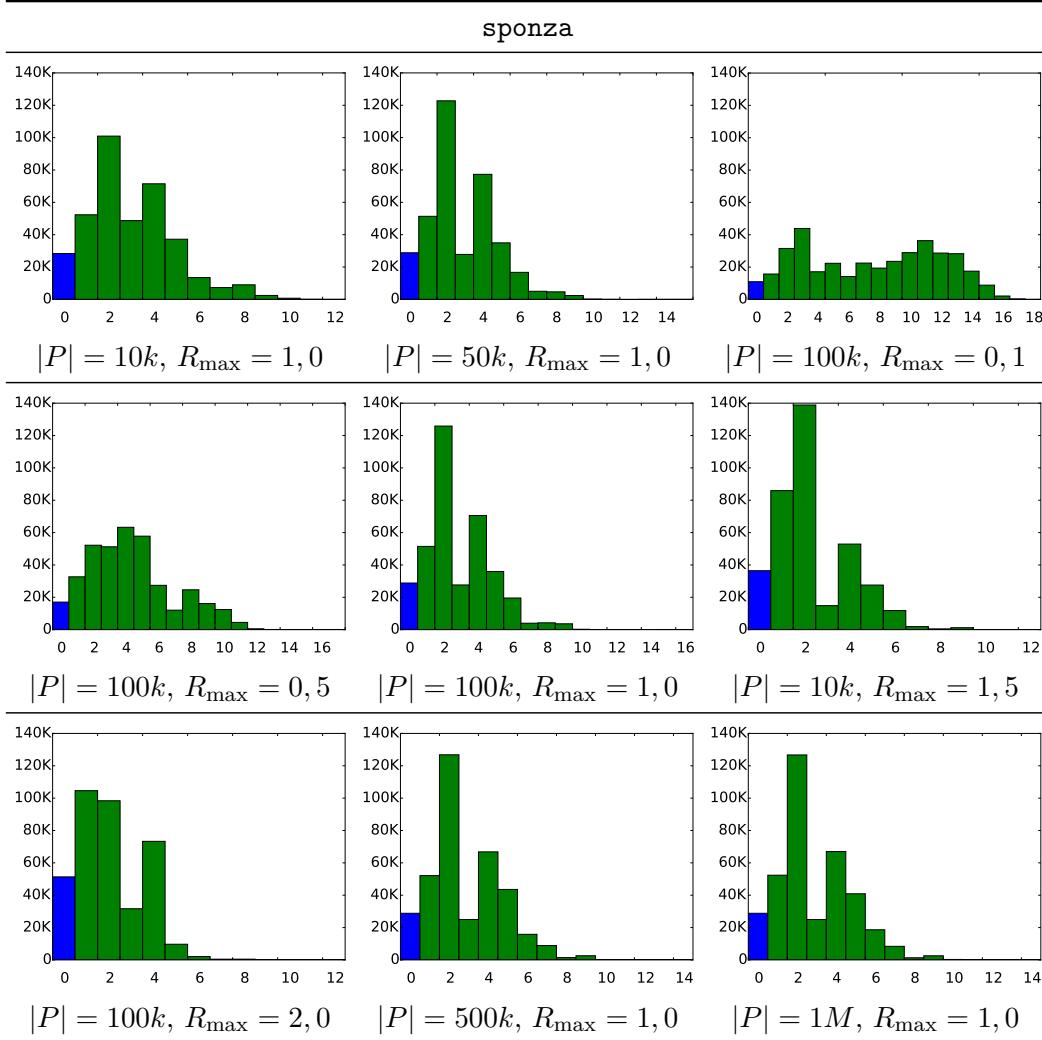
FIGUUR 7.13: Histogrammen van de diepte van de ropes (van niet-lege bladknopen) in de photonmap voor verschillende photonantallen en querystralen voor **sibenik**. Een diepte van nul (blauw) komt overeen met de wortel van de photonmap. Voor de andere dieptes (groen) doet de HCPM het beter dan de gewone photonmap. De photonmap is steeds gebouwd met de VVH.

testscenes zijn elk gerendered met 16 samples per pixel en 16 final gather rays per hitpunt. De testscenes zijn standaard gerenderd met 100k indirekte photons ($|P_V|$), een maximale querystraal (R_{\max}) van 1,0 en maximaal 32 photons per query (k). We hebben telkens één van deze drie parameters laten variëren.³

De bouw van de niet-hybride en hybride photonmappen is steeds afhankelijk van het aantal en de positie van de photons. Met uitzondering van de BH bij PM en

³Merk op dat de $|P_V| = 100k, R_{\max} = 1, 0$ en $k = 32$ kolommen hierdoor steeds identiek zijn.

7.3. Resultaten hybride photonmappen



FIGUUR 7.14: Histogrammen van de diepte van de ropes (van niet-lege bladknopen) in de photonmap voor verschillende photonaantallen en querystralen voor **sponza**. Een diepte van nul (blauw) komt overeen met de wortel van de photonmap. Voor de andere dieptes (groen) doet de HCPM het beter dan de gewone photonmap. De photonmap is steeds gebouwd met de VVH.

HAPM, is de bouw eveneens afhankelijk van de maximale querystraal. Het variëren van het aantal indirecte photons en de maximale querystraal levert dus een andere (hybride) photonmap op. Wat het variëren van het maximaal aantal photons per query betreft, blijft de (hybride) photonmap hetzelfde.

Indien het aantal photons toeneemt, wordt de niet-hybride photonmap groter. De gemiddelde query duurt hierdoor langer. Indien de maximale querystraal toeneemt, zullen de photons in een groter gebied gecontroleerd moeten worden. De gemiddelde

7. RESULTATEN

query duurt hierdoor langer. Indien we meer photons per query beschouwen, zal de querystraal en dus het zoekgebied minder verkleinen. Ook dit verlengt de duur van de gemiddelde query. Deze drie tendensen nemen we zowel waar bij de niet-hybride als hybride photonmappen. Enkel de HAPM vormt een uitzondering op de toename van het aantal indirecte photons.

HAPM. Bij de HAPM neemt de totale rendertijd af naarmate het aantal photons toeneemt. Hoe kleiner het totaal aantal photons, hoe kleiner de lokale photondensiteit en hoe kleiner het aantal photons onder de knopen met geometrische primitieven. Voor een query vanuit een hitpunt in een intermediaire knoop wordt eerst de onderliggende photonmap gecontroleerd. Indien deze photonmap een hoge densiteit aan photons heeft, kan de querystraal vlugger afnemen en zullen minder naburige voxels gecontroleerd moeten worden voor de gemiddelde query. Indien naburige gebieden toch gecontroleerd moeten worden, verlaat de query de huidige knoop en beweegt richting de wortelknoop. Bij het bottom-up doorlopen van de acceleratiestructuur worden de overige deelbomen op dit pad top-down doorlopen. De doorlooptijd neemt toe doordat we naast de photonmappen ook de acceleratiestructuur aan het doorlopen zijn. Hierdoor duurt het veel langer om de gewenste photonmappen te controleren. Naburige gebieden, bevinden zich niet noodzakelijk nabij elkaar in de boomstructuur.

We mogen deze resultaten echter niet extrapoleren. De totale rendertijd kan niet blijven afnemen. Voor hitpunten nabij de rand van de startvoxel, zullen naburige gebieden nog steeds gecontroleerd moeten worden. Naarmate het aantal photons nog meer toeneemt, zal de totale rendertijd geleidelijk aan terug toenemen. Doordat de photonmappen dieper worden, zal de doorlooptijd toenemen.

Bij een maximale querystraal van 2,0 is de toename van de doorlooptijd duidelijk waar te nemen. In dit geval is de kans groot dat een query de naburige voxels moet bezoeken met alle verplaatsingsoverhead als gevolg. Deze overhead is ook waarneembaar bij query's met maximaal 64 photons. Het zoekgebied is kleiner ($R_{\max} = 1,0$), maar door meer photons per query te zoeken, wordt de kans groter om het volledige zoekgebied te controleren. Ook nu zullen veel naburige voxels bezocht moeten worden.

De grotere totale rendertijd voor $R_{\max} = 0,1$ is een gevolg van de multiple importance sampling voor het bemonsteren van de richting van de final gather stralen (zie Opmerking 4.5). Tot het gewenste aantal photons bekomen wordt, start er telkens een nieuwe query met een opgehoogde maximale querystraal.

We besluiten dat de HAPM in het algemeen tot een groot performantieverlies leidt ten opzichte van een niet-hybride photonmap. Aangezien de photonmappen zich onder de kd-boom acceleratiestructuur bevinden in de HAPM, blijft de kost van de ray tracing hetzelfde. De totale querytijd is dus aanzienlijk toegenomen.

Dit ten gevolge van query's waarvoor veel naburige (in de ruimte, maar niet in de datastructuur) voxels gecontroleerd moeten worden.

HSPM. Aangezien zowel de geometrische primitieven als photons opgedeeld worden bij de HSPM leggen we minder restricties op dan bij de HAPM. Elke hybride kd-boom die bekomen kan worden met de HAPM kan bekomen worden met de HSPM (afhankelijk van de gebruikte bouwheuristiek), maar niet andersom. Door de grotere flexibiliteit van de HSPM verwachten we dat de totale rendertijd van de HSPM minstens even klein is dan deze van de HAPM. Dit is voor de meeste testscenes het geval. Bij de **sibenik** en **sponza** scenes met lage photonaantallen nemen we aanzienlijke verschillen waar. Door de grotere flexibiliteit van de HSPM en het hybride kostmodel van de HH neemt de adaptiviteit van de hybride kd-bomen toe en worden de splitsbeslissingen veel doordachter genomen.

Bij 1M indirekte photons, doet de HSPM in sommige gevallen iets slechter dan de HAPM. De onderliggende HH bouwheuristiek is een gulzige heuristiek. De lokaal optimale beslissing is niet noodzakelijk de globaal optimale beslissing.

Doordat de geometrische primitieven en photons door elkaar zitten in de HSPM kd-bomen, zal de structuur zowel een impact hebben op de gemiddelde ray tracing tijd als gemiddelde querytijd. De HSPM presteert echter algemeen aanzienlijk beter dan de HAPM. Nochtans is de overhead voor het doorlopen van de acceleratiestructuur er nog steeds en zijn de bekomen hybride kd-bomen veel groter en dieper. We besluiten dat de HH resulteert in aanzienlijk betere hybride kd-bomen in vergelijking met de twee lagen van de HAPM.

In vergelijking met de niet-hybride photonmappen is de toename van de totale rendertijd nog duidelijk waarneembaar. Bij een niet-hybride photonmap start een query steeds op de wortelknoop. De query daalt vervolgens af naar de dichterbijzijnde photon ten opzichte van het hitpunt. Vervolgens worden de overige deelbomen recursief bezocht. De doorloopoverhead manifesteert zich enerzijds in het (recursief) afdalen en anderzijds in het controleren van photons tijdens de afdaling die niet dicht genoeg tegen het hitpunt gelegen zijn. Merk op dat deze doorloopoverhead zeer klein is. De HSPM vermindert de afdalingstijd (ten opzichte van dezelfde kd-boom) doordat een query vroegtijdig de kd-boom kan verlaten via de minimaal omsluitende photonvoxel.

Daartegenover is de bekomen hybride photonmap groter en dieper dan de overeenkomstige niet-hybride photonmap door zowel geometrische primitieven als photons op te slaan en op te delen tot een gepaste verfijning van de kd-boom is bereikt. Per extra niveau kan het aantal knopen verdubbelen. Merk op dat deze doorloopoverhead aanzienlijk groot is. Dit ten gevolge van query's waarvoor veel naburige (in de ruimte, maar niet in de datastructuur) voxels gecontroleerd moeten worden. Dit is de reden waarom de totale rendertijd alsnog is toegenomen. Daarnaast is de

7. RESULTATEN

knoopdatastructuur groter en vergt dus meer geheugen wat resulteert in een slechtere cachecoherentie.

We besluiten dat het vroeger beëindigen van query's via de minimaal omsluitende photonvoxel slechts een beperkte impact heeft op de totale rendertijd. Het opdelen van zowel de geometrische primitieven als photons in eenzelfde kd-boom resulteert niet in de ideale opdeling van de photons. Eenmaal een query naburige voxels moet controleren, neemt de doorlooptijd aanzienlijk toe. Dit wordt bovendien in de hand gewerkt doordat de photons (puntdata) en geometrische primitieven (niet-puntdata) elkaar tegenwerken bij de constructie wat resulteert in grotere en diepere kd-bomen.

HCPM. De HCPM elimineert met zijn ropes steeds een deel van de doorloopoverhead ten opzichte van de overeenkomstige niet-hybride photonmap. De performantiewinst is echter klein (gemiddeld $\leq 1\%$) ten opzichte van de totale rendertijd.

Bouwheuristiek. De keuze van de bouwheuristiek heeft in de meeste gevallen slechts een kleine impact op de totale rendertijd. De VVH beschouwt niet alleen de meeste splitskandidaten per splitsbeslissing in vergelijking met de BH en VVH1, maar heeft ook een realistischer kostmodel ten opzichte van de BH. De grotere kwaliteit resulteert ook in een grotere bouwtijd zoals weergegeven in Tabellen 7.15-7.18.

Bouwtijd. In Tabellen 7.15-7.18 zijn de bouwtijden⁴ weergegeven voor de testsцenes. De bouwtijd voor de PM bestaat uit het bouwen van één photonmap. De bouwtijd voor de HCPM bestaat uit het bouwen van dezelfde photonmap en het berekenen van de ropes. De berekeningstijd voor de ropes is enkel waarneembaar voor de **sibenik** en **sponza** scenes met een maximale querystraal R_{\max} van 0,1. Het is voor deze grote scenes met fijne geometrische primitieven en een kleine querystraal dat de ropes het diepst gelegen zullen zijn in de photonmap. Vandaar de waarneembare berekeningstijd (0,1 seconde). De bouwtijd voor de HSPM bestaat uit het bouwen van de hybride kd-boom met de HH. De bouwtijd voor de HAPM bestaat uit het verdelen van de photons in de acceleratie kd-boom en het bouwen van de verschillende kd-boom photonmappen.

De bouwtijden van de PM en HCPM zijn gelijkaardig. De HAPM is iets sneller met meer maar kleinere photonmappen. De HSPM resulteert in grote bouwtijden voor de **sibenik** en **sponza** scenes. Dit door verschillende kosten te berekenen en te vergelijken per splitsbeslissing, meerdere splitskandidaten te beschouwen en algemeen diepere kd-bomen te bouwen.

⁴Merk op dat een bouwtijd van 0,0 seconden niet betekent dat het bouwen niet plaatsvindt. Dit is een nauwkeurigheidsbeperking van onze timer.

7.3. Resultaten hybride photonmappen

cornell						
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		8,4	8,7	8,9	9,5
	VVH		8,5	8,7	9,0	9,6
	VVH1		8,1	8,3	8,5	9,1
HCPM	BH		8,3	8,6	8,8	9,5
	VVH		8,4	8,7	8,9	9,6
	VVH1		8,1	8,3	8,4	9,0
HSPM	APSA+VVH		10,5	10,8	11,2	12,7
	APSA+VVH1		9,6	10,2	10,6	12,1
HAPM	BH		13,4	14,3	15,0	17,3
	VVH		12,1	13,0	13,7	15,7
	VVH1		12,3	13,3	13,9	15,9
R_{\max}						
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		8,9	8,9	8,9	9,0
	VVH		9,0	9,0	9,0	9,0
	VVH1		8,0	8,5	8,5	8,6
HCPM	BH		8,8	8,8	8,8	8,9
	VVH		8,9	8,9	8,9	8,9
	VVH1		8,4	8,4	8,4	8,5
HSPM	APSA+VVH		11,4	11,2	11,2	11,2
	APSA+VVH1		11,1	10,7	10,6	10,6
HAPM	BH		15,7	15,0	15,0	15,1
	VVH		14,3	13,7	13,7	13,7
	VVH1		14,3	13,9	13,9	13,9
k						
		Heuristiek	16	32	64	
PM	BH		8,9	8,9	9,0	
	VVH		8,9	9,0	9,0	
	VVH1		8,5	8,5	8,5	
HCPM	BH		8,8	8,8	8,9	
	VVH		8,9	8,9	8,9	
	VVH1		8,4	8,4	8,4	
HSPM	APSA+VVH		11,1	11,2	11,2	
	APSA+VVH1		10,5	10,6	10,6	
HAPM	BH		15,0	15,0	15,1	
	VVH		13,6	13,7	13,8	
	VVH1		13,8	13,9	14,0	

TABEL 7.7: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor cornell met 16 spp en 16 final gather rays per hitpunt.

7. RESULTATEN

		room				
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		31,2	38,8	41,9	46,1
	VVH		31,0	38,0	39,4	41,9
	VVH1		30,6	37,6	38,9	41,6
HCPM	BH		31,0	38,7	41,2	45,5
	VVH		30,9	38,0	39,4	41,7
	VVH1		30,5	37,3	38,8	41,6
HSPM	APSA+VVH		43,1	51,1	53,7	58,3
	APSA+VVH1		44,0	50,8	53,2	58,1
HAPM	BH		96,7	84,4	79,1	79,1
	VVH		96,2	87,0	78,2	76,4
	VVH1		96,7	83,2	78,7	74,9
		R_{\max}				
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		20,2	35,1	41,9	43,0
	VVH		20,3	34,6	39,4	39,9
	VVH1		20,3	35,8	38,9	42,0
HCPM	BH		19,8	34,8	41,2	42,6
	VVH		20,1	34,5	39,4	39,6
	VVH1		19,2	33,8	38,8	39,5
HSPM	APSA+VVH		24,9	44,9	53,7	57,7
	APSA+VVH1		24,9	44,7	53,2	57,6
HAPM	BH		41,9	67,6	79,1	82,3
	VVH		41,7	67,6	78,2	80,9
	VVH1		41,0	69,7	78,7	81,4
		k				
		Heuristiek	16	32	64	
PM	BH		30,4	41,9	59,7	
	VVH		28,7	39,4	58,1	
	VVH1		30,8	38,9	61,1	
HCPM	BH		30,1	41,2	59,4	
	VVH		28,5	39,4	58,0	
	VVH1		28,2	38,8	57,7	
HSPM	APSA+VVH		40,5	53,7	74,9	
	APSA+VVH1		40,2	53,2	74,6	
HAPM	BH		60,2	79,1	107,2	
	VVH		59,4	78,2	106,3	
	VVH1		59,1	78,7	105,9	

TABEL 7.8: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor room met 16 spp en 16 final gather rays per hitpunt.

7.3. Resultaten hybride photonmappen

sibenik1						
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		170,7	177,2	183,0	201,9
	VVH		169,8	174,7	182,3	198,4
	VVH1		169,2	176,1	182,9	197,8
HCPM	BH		169,8	175,7	183,0	201,1
	VVH		168,9	174,6	181,7	198,0
	VVH1		168,8	174,3	181,2	196,0
HSPM	APSA+VVH		242,3	258,7	276,8	306,7
	APSA+VVH1		241,9	257,9	277,4	305,5
HAPM	BH		491,5	436,9	415,5	348,5
	VVH		490,9	437,3	418,2	349,1
	VVH1		496,3	437,2	417,9	348,8
R_{\max}						
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		165,2	172,3	183,0	191,2
	VVH		164,6	171,5	182,3	188,8
	VVH1		163,6	169,9	182,9	187,9
HCPM	BH		164,4	172,1	183,0	190,4
	VVH		163,8	170,3	181,7	188,6
	VVH1		162,9	169,8	181,2	187,8
HSPM	APSA+VVH		226,6	245,7	276,8	302,1
	APSA+VVH1		217,1	238,8	277,4	287,3
HAPM	BH		259,2	317,4	415,5	456,3
	VVH		258,7	317,7	418,2	457,2
	VVH1		259,3	318,4	417,9	457,0
k						
		Heuristiek	16	32	64	
PM	BH		178,2	183,0	185,6	
	VVH		176,8	182,3	184,3	
	VVH1		176,7	182,9	183,6	
HCPM	BH		178,1	183,0	185,1	
	VVH		176,7	181,7	184,1	
	VVH1		176,2	181,2	183,1	
HSPM	APSA+VVH		263,3	276,8	290,7	
	APSA+VVH1		264,0	277,4	281,0	
HAPM	BH		373,2	415,5	431,7	
	VVH		373,3	418,2	430,7	
	VVH1		373,6	417,9	431,6	

TABEL 7.9: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirekte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor **sibenik1** met 16 spp en 16 final gather rays per hitpunt.

7. RESULTATEN

sibenik2						
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		298,1	309,7	318,9	352,7
	VVH		295,8	305,7	317,5	347,7
	VVH1		332,1	307,1	317,2	345,1
HCPM	BH		295,5	307,5	317,0	351,2
	VVH		293,6	303,9	314,8	345,2
	VVH1		293,7	304,2	314,6	342,6
HSPM	APSA+VVH		434,2	477,3	495,5	545,7
	APSA+VVH1		421,2	453,1	477,8	533,3
HAPM	BH		691,0	655,5	646,9	594,0
	VVH		691,3	657,7	646,3	591,7
	VVH1		691,4	654,3	647,2	595,3
R_{\max}						
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		291,6	303,3	318,9	335,4
	VVH		290,7	301,0	317,5	331,4
	VVH1		289,6	300,9	317,2	329,7
HCPM	BH		288,8	299,9	317,0	331,9
	VVH		286,8	297,9	314,8	327,8
	VVH1		287,0	297,5	314,6	328,7
HSPM	APSA+VVH		408,2	421,5	495,5	500,3
	APSA+VVH1		388,5	422,9	477,8	507,4
HAPM	BH		411,3	503,0	646,9	730,9
	VVH		409,6	510,2	646,3	755,7
	VVH1		409,8	502,3	647,2	727,9
k						
		Heuristiek	16	32	64	
PM	BH		313,8	318,9	323,1	
	VVH		311,8	317,5	321,6	
	VVH1		311,7	317,2	319,8	
HCPM	BH		311,3	317,0	320,8	
	VVH		308,6	314,8	319,0	
	VVH1		308,1	314,6	317,7	
HSPM	APSA+VVH		459,9	495,5	484,4	
	APSA+VVH1		460,4	477,8	485,7	
HAPM	BH		593,7	646,9	663,1	
	VVH		603,1	646,3	703,2	
	VVH1		592,2	647,2	664,7	

TABEL 7.10: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor sibenik2 met 16 spp en 16 final gather rays per hitpunt.

7.3. Resultaten hybride photonmappen

sponza1						
		$ P $				
	Heuristiek	10k	50k	100k	500k	1M
PM	BH	501,1	589,1	634,6	706,6	723,8
	VVH	498,6	580,1	621,9	689,8	699,7
	VVH1	498,1	579,2	623,0	691,8	703,9
HCPM	BH	499,3	583,4	630,0	705,3	718,9
	VVH	496,2	578,4	619,1	685,2	696,2
	VVH1	495,7	577,0	619,6	689,2	700,5
HSPM	APSA+VVH	712,9	828,8	890,9	1022,4	1131,4
	APSA+VVH1	712,3	833,5	888,6	1022,9	1127,4
HAPM	BH	1338,8	1134,2	1106,6	1107,2	1125,0
	VVH	1343,7	1132,4	1105,6	1097,2	1108,3
	VVH1	1341,6	1132,0	1104,1	1093,9	1107,3
		R_{\max}				
	Heuristiek	0,1	0,5	1,0	1,5	2,0
PM	BH	543,2	587,3	634,6	660,6	664,2
	VVH	541,4	580,7	621,9	635,1	641,8
	VVH1	539,2	581,7	623,0	636,3	641,2
HCPM	BH	538,8	583,5	630,0	651,6	656,3
	VVH	538,1	579,2	619,1	633,6	639,9
	VVH1	535,2	577,3	619,6	635,5	639,2
HSPM	APSA+VVH	710,7	796,5	890,9	933,5	949,7
	APSA+VVH1	703,0	793,1	888,6	930,4	950,1
HAPM	BH	773,3	966,8	1106,6	1153,3	1198,6
	VVH	771,6	968,3	1105,6	1151,8	1173,5
	VVH1	769,6	966,1	1104,1	1148,8	1173,2
		k				
	Heuristiek	16	32	64		
PM	BH	595,9	634,6	679,7		
	VVH	586,0	621,9	662,7		
	VVH1	587,0	623,0	661,3		
HCPM	BH	591,7	633,5	678,0		
	VVH	582,8	619,1	659,9		
	VVH1	584,9	619,6	659,5		
HSPM	APSA+VVH	818,7	890,9	969,7		
	APSA+VVH1	819,2	888,6	972,2		
HAPM	BH	965,0	1106,6	1259,3		
	VVH	965,3	1105,6	1262,8		
	VVH1	975,8	1104,1	1265,8		

TABEL 7.11: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor sponza1 met 16 spp en 16 final gather rays per hitpunt.

7. RESULTATEN

sponza2					
		$ P $			
Heuristiek		10k	50k	100k	500k
PM	BH	565,7	647,4	689,0	743,0
	VVH	564,0	628,1	667,3	716,4
	VVH1	563,6	626,7	664,6	714,5
HCPM	BH	563,3	643,0	687,1	739,4
	VVH	561,1	626,1	664,8	711,4
	VVH1	560,1	621,3	661,2	711,9
HSPM	APSA+VVH	822,0	891,8	947,3	1086,2
	APSA+VVH1	821,9	895,7	943,3	1092,1
HAPM	BH	1628,5	1253,8	1208,0	1177,6
	VVH	1629,3	1252,5	1202,7	1158,8
	VVH1	1630,9	1252,0	1207,3	1155,4
R_{\max}					
Heuristiek		0,1	0,5	1,0	1,5
PM	BH	578,5	645,9	689,0	698,5
	VVH	574,2	632,7	667,3	671,6
	VVH1	571,3	632,3	664,6	671,9
HCPM	BH	575,2	640,6	687,1	696,0
	VVH	573,2	629,9	664,8	671,1
	VVH1	567,9	628,1	661,2	669,4
HSPM	APSA+VVH	742,4	856,4	947,3	979,0
	APSA+VVH1	742,2	859,0	943,3	982,5
HAPM	BH	825,3	1086,1	1208,0	1239,5
	VVH	824,8	1081,7	1202,7	1236,5
	VVH1	825,5	1082,5	1207,3	1236,1
k					
Heuristiek		16	32	64	
PM	BH	638,6	689,0	759,6	
	VVH	622,0	667,3	731,4	
	VVH1	620,4	664,6	728,6	
HCPM	BH	633,9	687,1	752,9	
	VVH	621,2	664,8	727,5	
	VVH1	617,1	661,2	725,9	
HSPM	APSA+VVH	851,5	947,3	1072,3	
	APSA+VVH1	851,5	943,3	1071,8	
HAPM	BH	1033,9	1208,0	1437,7	
	VVH	1029,9	1202,7	1431,8	
	VVH1	1028,7	1207,3	1430,5	

TABEL 7.12: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor sponza2 met 16 spp en 16 final gather rays per hitpunt.

7.3. Resultaten hybride photonmappen

sponza3						
		$ P $				
Heuristiek		10k	50k	100k	500k	1M
PM	BH	465,8	524,8	557,4	642,7	677,2
	VVH	461,2	510,1	539,7	619,3	644,3
	VVH1	460,4	511,0	542,6	619,5	644,5
HCPM	BH	461,7	521,9	556,5	641,3	675,4
	VVH	460,5	509,0	537,4	609,4	639,5
	VVH1	459,8	509,3	540,2	616,7	643,1
HSPM	APSA+VVH	646,3	709,9	782,0	948,0	1035,5
	APSA+VVH1	645,4	709,5	747,9	905,9	999,9
HAPM	BH	1422,6	1037,6	996,1	1009,7	1062,6
	VVH	1428,5	1045,1	1002,7	1009,1	1045,0
	VVH1	1427,4	1030,9	988,9	1002,1	1044,1
R_{\max}						
Heuristiek		0,1	0,5	1,0	1,5	2,0
PM	BH	475,9	518,9	557,4	577,8	582,1
	VVH	467,6	504,1	539,7	552,3	559,0
	VVH1	465,0	504,7	542,6	555,3	558,7
HCPM	BH	471,3	512,7	556,5	575,7	578,8
	VVH	465,1	503,3	537,4	550,9	557,8
	VVH1	463,5	504,0	540,2	552,1	557,4
HSPM	APSA+VVH	593,3	666,9	782,0	782,1	795,4
	APSA+VVH1	592,5	667,2	747,9	780,8	794,4
HAPM	BH	687,6	851,0	996,1	1045,8	1073,3
	VVH	686,0	852,1	1002,7	1045,2	1071,2
	VVH1	685,4	852,1	988,9	1045,2	1071,3
k						
Heuristiek		16	32	64		
PM	BH	521,2	557,4	601,1		
	VVH	507,1	539,7	580,3		
	VVH1	509,9	542,6	581,0		
HCPM	BH	520,5	556,5	598,2		
	VVH	504,9	537,4	577,2		
	VVH1	508,2	540,2	579,6		
HSPM	APSA+VVH	682,4	782,0	825,1		
	APSA+VVH1	681,8	747,9	824,7		
HAPM	BH	858,8	996,1	1136,8		
	VVH	858,3	1002,7	1140,5		
	VVH1	856,7	988,9	1138,4		

TABEL 7.13: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor sponza3 met 16 spp en 16 final gather rays per hitpunt.

7. RESULTATEN

sponza4					
		$ P $			
	Heuristiek	10k	50k	100k	500k
PM	BH	496,6	567,1	604,2	674,6
	VVH	492,1	553,3	586,4	642,4
	VVH1	491,1	553,1	589,1	648,2
HCPM	BH	495,0	566,1	603,0	672,8
	VVH	491,0	552,8	584,9	640,5
	VVH1	490,2	552,2	587,3	645,7
HSPM	APSA+VVH	663,9	772,5	814,2	945,1
	APSA+VVH1	667,3	774,0	817,4	943,3
HAPM	BH	1520,5	1185,8	1126,0	1076,0
	VVH	1529,9	1183,9	1122,8	1065,0
	VVH1	1528,0	1183,5	1121,5	1072,2
R_{\max}					
	Heuristiek	0,1	0,5	1,0	1,5
PM	BH	510,4	552,7	604,2	624,2
	VVH	501,7	543,7	586,4	602,6
	VVH1	501,4	555,4	589,1	609,0
HCPM	BH	508,8	552,2	603,0	623,1
	VVH	500,9	542,9	584,9	599,5
	VVH1	498,5	540,3	587,3	600,7
HSPM	APSA+VVH	628,2	710,0	814,2	852,2
	APSA+VVH1	628,8	711,7	817,4	852,0
HAPM	BH	750,0	940,1	1126,0	1172,8
	VVH	747,9	961,2	1122,8	1176,8
	VVH1	748,0	939,4	1121,5	1191,4
k					
	Heuristiek	16	32	64	
PM	BH	566,3	604,2	645,8	
	VVH	561,9	586,4	633,9	
	VVH1	553,5	589,1	626,7	
HCPM	BH	563,8	603,0	643,7	
	VVH	552,0	584,9	624,3	
	VVH1	550,1	587,3	623,9	
HSPM	APSA+VVH	741,4	814,2	896,2	
	APSA+VVH1	744,2	817,4	891,2	
HAPM	BH	974,3	1126,0	1292,4	
	VVH	970,9	1122,8	1282,3	
	VVH1	969,1	1121,5	1284,6	

TABEL 7.14: De rendertijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$, querystraal R_{\max} en maximaal aantal photons per query k , voor sponza4 met 16 spp en 16 final gather rays per hitpunt.

cornell						
		$ P $				
	Heuristiek	10k	50k	100k	500k	1M
PM	BH	0,0	0,0	0,1	0,3	0,7
	VVH	0,1	0,2	0,4	2,6	6,1
	VVH1	0,0	0,1	0,1	0,9	2,0
HCPM	BH	0,0	0,0	0,1	0,3	0,7
	VVH	0,1	0,2	0,4	2,6	6,1
	VVH1	0,0	0,1	0,1	0,9	2,0
HSPM	APSA+VVH	0,1	0,3	0,7	3,7	8,6
	APSA+VVH1	0,1	0,3	0,5	2,6	5,9
HAPM	BH	0,0	0,0	0,0	0,2	0,5
	VVH	0,1	0,1	0,2	1,5	3,7
	VVH1	0,0	0,0	0,1	0,5	1,2
R_{\max}						
	Heuristiek	0,1	0,5	1,0	1,5	2,0
PM	BH	0,1	0,1	0,1	0,1	0,1
	VVH	0,4	0,4	0,4	0,4	0,4
	VVH1	0,1	0,1	0,1	0,1	0,1
HCPM	BH	0,1	0,1	0,1	0,1	0,1
	VVH	0,4	0,4	0,4	0,4	0,4
	VVH1	0,1	0,1	0,1	0,1	0,1
HSPM	APSA+VVH	0,7	0,6	0,7	0,6	0,6
	APSA+VVH1	0,5	0,5	0,5	0,5	0,5
HAPM	BH	0,0	0,0	0,0	0,0	0,0
	VVH	0,2	0,2	0,2	0,2	0,2
	VVH1	0,1	0,1	0,1	0,1	0,1

TABEL 7.15: De bouwtijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirekte photons $|P|$ en querystraal R_{\max} , voor **cornell**.

7. RESULTATEN

		room				
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		0,0	0,0	0,0	0,3
	VVH		0,0	0,2	0,4	2,6
	VVH1		0,0	0,1	0,1	0,9
HCPM	BH		0,0	0,0	0,0	0,3
	VVH		0,0	0,2	0,4	2,6
	VVH1		0,0	0,1	0,1	0,9
HSPM	APSA+VVH		0,6	1,0	1,5	7,2
	APSA+VVH1		0,5	0,8	1,2	4,7
HAPM	BH		0,0	0,0	0,0	0,2
	VVH		0,0	0,1	0,2	1,1
	VVH1		0,0	0,0	0,1	0,4
			R_{\max}			
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		0,0	0,0	0,0	0,0
	VVH		0,4	0,4	0,4	0,4
	VVH1		0,1	0,1	0,1	0,1
HCPM	BH		0,0	0,0	0,0	0,0
	VVH		0,4	0,4	0,4	0,4
	VVH1		0,1	0,1	0,1	0,1
HSPM	APSA+VVH		1,3	1,4	1,5	1,8
	APSA+VVH1		1,0	1,1	1,2	1,4
HAPM	BH		0,0	0,0	0,0	0,0
	VVH		0,2	0,2	0,2	0,2
	VVH1		0,1	0,1	0,1	0,1

TABEL 7.16: De bouwtijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$ en querystraal R_{\max} , voor room.

		sibenik					
		$ P $					
		Heuristiek	10k	50k	100k	500k	1M
PM	BH		0,0	0,0	0,1	0,3	0,7
	VVH		0,1	0,3	0,5	3,1	6,7
	VVH1		0,0	0,1	0,2	1,1	2,5
HCPM	BH		0,0	0,0	0,1	0,3	0,7
	VVH		0,1	0,3	0,5	3,1	6,7
	VVH1		0,0	0,1	0,2	1,1	2,5
HSPM	APSA+VVH		5,2	6,0	6,9	15,8	31,0
	APSA+VVH1		5,1	5,8	6,4	12,6	22,8
HAPM	BH		0,0	0,0	0,0	0,2	0,4
	VVH		0,0	0,1	0,1	0,5	1,1
	VVH1		0,0	0,0	0,1	0,2	0,5
		R_{\max}					
		Heuristiek	0,1	0,5	1,0	1,5	2,0
PM	BH		0,1	0,1	0,1	0,1	0,1
	VVH		0,5	0,5	0,5	0,5	0,5
	VVH1		0,2	0,2	0,2	0,2	0,2
HCPM	BH		0,2	0,1	0,1	0,1	0,1
	VVH		0,6	0,5	0,5	0,5	0,5
	VVH1		0,3	0,2	0,2	0,2	0,2
HSPM	APSA+VVH		6,9	7,0	6,9	7,0	7,2
	APSA+VVH1		6,8	6,4	6,4	6,4	6,7
HAPM	BH		0,0	0,0	0,0	0,0	0,0
	VVH		0,1	0,1	0,1	0,1	0,1
	VVH1		0,1	0,1	0,1	0,1	0,1

TABEL 7.17: De bouwtijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirekte photons $|P|$ en querystraal R_{\max} , voor **sibenik**.

		sponza				
		$ P $				
		Heuristiek	10k	50k	100k	500k
PM	BH		0,0	0,0	0,0	0,3
	VVH		0,0	0,2	0,3	2,7
	VVH1		0,0	0,0	0,1	0,8
HCPM	BH		0,0	0,0	0,0	0,3
	VVH		0,0	0,2	0,3	2,7
	VVH1		0,0	0,0	0,1	0,8
HSPM	APSA+VVH		5,1	5,5	6,7	14,9
	APSA+VVH1		4,9	5,4	6,6	11,7
HAPM	BH		0,0	0,0	0,0	0,2
	VVH		0,0	0,0	0,1	0,5
	VVH1		0,0	0,0	0,0	0,2
		R_{\max}				
		Heuristiek	0,1	0,5	1,0	1,5
PM	BH		0,0	0,0	0,0	0,0
	VVH		0,4	0,3	0,3	0,4
	VVH1		0,1	0,1	0,1	0,1
HCPM	BH		0,1	0,0	0,0	0,0
	VVH		0,5	0,3	0,3	0,4
	VVH1		0,2	0,1	0,1	0,1
HSPM	APSA+VVH		6,5	6,5	6,7	6,5
	APSA+VVH1		6,1	6,4	6,6	6,2
HAPM	BH		0,0	0,0	0,0	0,0
	VVH		0,1	0,1	0,1	0,1
	VVH1		0,0	0,0	0,0	0,0

TABEL 7.18: De bouwtijden (in seconden) voor de niet-hybride photonmap (PM) en hybride photonmappen (HCPM, HSPM, HAPM) voor verschillende bouwheuristieken, aantal indirecte photons $|P|$ en querystraal R_{\max} , voor **sponza**.

Hoofdstuk 8

Besluit

In sectie 8.1 geven we de conclusies over onze RTSAHs. In sectie 8.2 geven we de conclusies over onze hybride photonmappen. Sectie 8.3 sluit deze thesis af met mogelijke pistes van verder onderzoek op basis van onze RTSAHs.

8.1 Conclusies RTSAH

Onze RTSAHs resulteren in meer kwalitatieve kd-bomen in vergelijking met de SAH door straalterminatie in rekening te nemen. Bovendien hebben onze RTSAH bouwprocedures dezelfde totale computationele complexiteit en moeten ze dezelfde eindige verzameling van splitsingsvlakken beschouwen als de SAH.

8.2 Conclusies hybride photonmappen

Het toevoegen van de photons aan de kd-boom acceleratiestructuur (HAPM) verhoogt de doorlooptijd en dus de totale rendertijd aanzienlijk ten opzichte van de niet-hybride photonmap. De bekomen hybride kd-bomen zijn groter en dieper doordat we de photons toevoegen onder de knopen met geometrische primitieven. K-nearest neighbor query's kunnen dan wel rechtstreeks starten vanuit de voxel met het hitpunt, eenmaal naburige voxels gecontroleerd moeten worden, neemt de doorlooptijd aanzienlijk toe. Naburige voxels bevinden zich niet noodzakelijk nabij elkaar in de datastructuur. Verder wordt bij de bouw van de kd-boom acceleratiestructuur geen rekening gehouden met toekomstige photons. Dit terwijl het volgen van een straal fundamenteel verschillend is van het uitvoeren van een k-nearest neighbor query.

De HSPM probeert dit probleem te verhelpen door zowel de geometrische primitieven als photons in rekening te nemen bij de bouw van de hybride kd-boom. Uit de resultaten volgt dat de onderliggende bouwheuristiek, de HH, de totale rendertijd in het algemeen aanzienlijk kan verminderen ten opzichte van de HAPM. Een aparte kd-boom acceleratiestructuur en kd-boom photonmap blijven echter nog steeds beter presteren.

8. BESLUIT

Op basis van de resultaten van de HSPM concluderen we dat de geometrische primitieven en photons niet verzoend kunnen worden in eenzelfde kd-boom opdat de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel verminderd wordt ten opzichte van twee aparte kd-bomen, elk bekomen met een optimale bouwheuristiek voor hun specifieke data. Dit door de volgende oorzaken:

- De geometrische primitieven (niet-pundata) en photons (pundata) werken elkaar tegen wat de mate van verfijning van de kd-boom betreft.
- Het volgen van stralen en het uitvoeren van k-nearest neighbor query's zijn fundamenteel verschillende operaties. Dit heeft zowel gevolgen voor de manier waarop de hybride kd-boom doorlopen wordt als voor de (gekende versus niet-gekende) informatie die nodig is om de kostfuncties te evalueren.

Daarnaast resulteert de HH ook in aanzienlijke bouwtijden en een verhoogd geheugengebruik om alle extra parameters bij te houden.

Uit onze resultaten en argumentatie concluderen we dat geometrische primitieven en photons niet verzoend kunnen worden noch in kd-bomen noch in de veel gebruikte huidige acceleratiestructuren (reguliere roosters, BVHs, BSPs, octrees) opdat de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel verminderd wordt ten opzichte van twee aparte datastructuren.

De HCPM waarbij de acceleratiestructuur door middel van ropes verbonden wordt met de photonmap, is een nieuwe datastructuur die algemeen in een zeer kleine winst (gemiddeld $\leq 1\%$) resulteert voor de totale rendertijd (meer bepaald de totale querytijd). De HCPM heeft het bijkomende voordeel dat ze eenvoudig te implementeren is in een bestaand photonmapping algoritme en het geheugengebruik slechts beperkt laat toenemen (voor de ropes).

8.3 Toekomstig onderzoek

Met het oog op toekomstig onderzoek is het belangrijk de verschillen tussen de kd-bomen bekomen met de SAH en RTSAH verder te onderzoeken en de onderliggende assumpties en vereenvoudigingen verder te valideren. Daarnaast kunnen onze benaderde visibiliteitsprobabiliteiten gebruikt worden om de doorloopvolgorde van de voxels voor schaduwstralen te bepalen in vergelijking met de (originele) RT-SAH [IH11] en SATO [NM14]. Zowel de SAH als RTSAH nemen noch de actuele distributie van de stralen noch de aanwezigheid van interne stralen in rekening. De aangepaste SAH voor stralen gelegen binnen de omhullende box van de scene, geïntroduceerd door Fabianowski et al. [FFD09], kan baat hebben aan de straalterminatie van onze RTSAHs.

Bijlagen

Bijlage A

Poster



KATHOLIEKE UNIVERSITEIT
LEUVEN

FACULTEIT
INGENIEURSWETENSCHAPPEN

Master
Computer-
wetenschappen

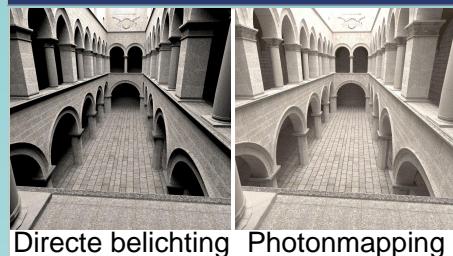
Masterproef
Matthias Moulin

Promotor
Prof. dr. ir.
Philip Dutré

Academiejaar
2014-2015

Hybride Kd-bomen voor photonmapping en het versnellen van ray tracing

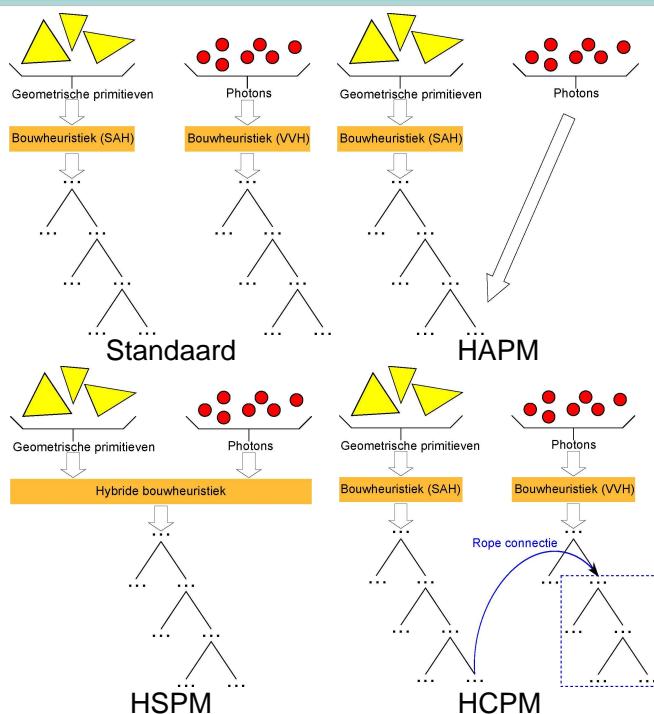
Situering



Doelstellingen

- We introduceren hybride kd-bomen waarin zowel de **geometrische primitieven** als **photons** worden ondergebracht met als doel de gecombineerde kost van **ray tracing** en **k-nearest neighbor query's** fundamenteel te verminderen

Hybride kd-bomen



• Hybride bouwheuristiek

- kapselt **RTSAH** (Moulin et al. 2015) en **VVH** (Wald et al. 2004) in
- verbindt de kost van de **ray tracing** en **k-nearest neighbor query's**

Resultaten

- **Geometrische primitieven** en **photons** zijn NIET verzoenbaar noch in eenzelfde kd-boom noch in de veel gebruikte huidige acceleratiestructuren
 - Niet-puntdata vs puntdata
 - Ray tracing vs k-nearest neighbor query's
- Enkel **HCPM** resulteert in een zeer kleine performantiewinst
 - gemiddeld $\leq 1\%$ voor de totale rendertijd



KATHOLIEKE UNIVERSITEIT
LEUVEN

FACULTEIT
INGENIEURSWETENSCHAPPEN

Master
Computer-
wetenschappen

Masterproef
Matthias Moulin

Promotor
Prof. dr. ir.
Philip Dutré

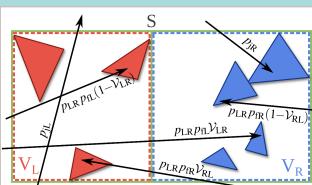
Academiejaar
2014-2015

Efficiënte visibiliteitsheuristieken voor kd-bomen gebruikmakende van de RTSAH

Situering en doelstellingen

- Acceleratiestructuren zoals kd-bomen reduceren de kost voor het volgen van stralen, wat cruciaal is voor de renderingperformantie
- De de-facto standaard om kd-bomen te bouwen, de SAH (MacDonald & Booth 1990) veronderstelt dat stralen nooit geometrische primitieven raken
- De originele RTSAH (Ize et al. 2011) bepaalt de doorloopvolgorde van voxels voor occlusiestralen en neemt straalterminatie in rekening
- We passen deze RTSAH aan om kd-bomen te bouwen

RTSAH



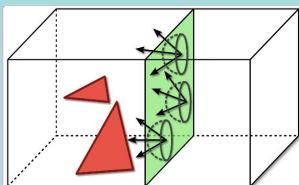
• Opsplitsing van voxel V

- in kindvoxels V_L en V_R
- door splitsingsvlak in S

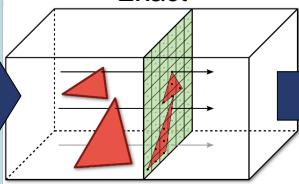
• Straalterminatie

- stralen kunnen eindigen in kindvoxel
- uitgedrukt door visibiliteitsprobabiliteiten \mathcal{V}

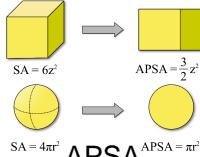
\mathcal{V} Benaderingen



Exact



APOD



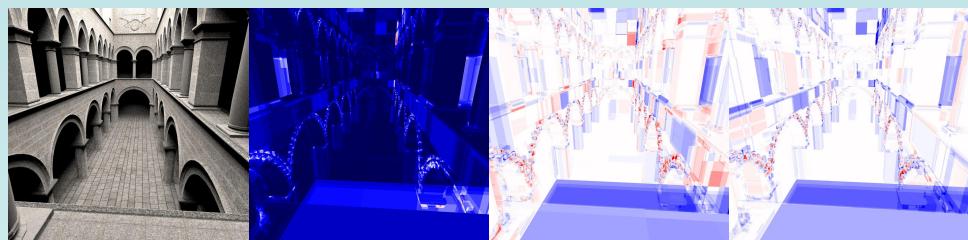
Resultaten (vs SAH)

• 2 bouwprocedures

- met dezelfde computationele complexiteit
- met dezelfde eindige verzameling van splitskandidaten

• Reductie in intersectietesten

- tot 47% voor primaire stralen
- tot 41% voor schaduwstralen



Scene

SAH

Δ APOD RTSAH

Δ APSA RTSAH

Bijlage B

Artikel

Efficient visibility heuristics for kd-trees using the RTSAH

Matthias Moulin, Niels Billen, Philip Dutré

Department of Computer Science, KU Leuven, Belgium

Abstract

Acceleration data structures such as kd-trees aim at reducing the per-ray cost which is crucial for rendering performance. The de-facto standard for constructing kd-trees, the Surface Area Heuristic (SAH), does not take ray termination into account and instead assumes rays never hit a geometric primitive. The Ray Termination Surface Area Heuristic (RTSAH) is a cost metric originally used for determining the traversal order of the voxels for occlusion rays that takes ray termination into account. We adapt this RTSAH to building kd-trees that aim at reducing the per-ray cost of rays. Our build procedure has the same overall computational complexity and considers the same finite set of splitting planes as the SAH. By taking ray termination into account, we favor cutting off child voxels which are not or hardly visible to each other. This results in fundamentally different and more qualitative kd-trees compared to the SAH.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

High quality acceleration data structures for ray-tracing such as kd-trees aim to reduce the cost of tracing a ray. The standard heuristic for constructing these kd-trees is the Surface Area Heuristic (SAH) [MB90], which assigns to each candidate voxel a cost equal to the product of the cost of processing this voxel and the probability of actually processing this voxel. A top-down construction greedily minimizes this cost. In order to calculate these probabilities for each candidate voxel, the SAH assumes rays to be infinitely long, to start outside the scene's bounding box and the origins and directions of the rays to be uniformly distributed. Despite its popularity, the assumptions underlying the SAH usually do not hold in practice. In many scenes, most of the rays originate from inside the scene's bounding box and terminate when hitting a surface. By not taking ray termination into account, costs are overestimated, leading to a suboptimal ranking of the splitting planes and even to the decision of not splitting at all.

In this paper, we try to revise this assumption by revisiting the Ray Termination Surface Area Heuristic (RTSAH) [IH11], a cost metric originally used for determining the traversal order of the voxels for occlusion rays taking ray termination into account. We adapt this RTSAH to build

view-independent kd-tree acceleration structures. Our build procedures have the same overall computational complexity and need to consider the same finite set of splitting planes as the SAH.

The contributions of this paper are as follows:

- We present a novel way of using the RTSAH for building kd-trees.
- We present two fast building algorithms for qualitative kd-trees based on the orthogonal projected and average projected surface area of the individual geometric primitives onto the splitting plane.
- We achieve reductions in intersection tests up to 47% for primary rays and up to 41% for shadow rays (when traversing the voxels in order) compared to the SAH.

2. Related Work

Acceleration data structures. Rays are intersected with the geometric primitives in the scene in order to determine which surfaces are visible and to resolve occlusion. The number of intersection tests can be kept between theoretical boundaries by using appropriate acceleration data structures (for a survey, see [WMG*09]).

The most well-known adaptive acceleration data structures are Bounding Volume Hierarchies (BVH) and kd-trees. BVHs and kd-trees are usually constructed using a greedy divide-and-conquer strategy for partitioning the three-dimensional space or the geometric primitives. Different traversal approaches exist for BVHs [HDW^{*}11] and kd-trees [HH11].

Heuristics for building kd-trees. Heuristics are used for partitioning the three-dimensional space in a kd-tree. The best known and commonly used heuristic for generating qualitative kd-trees is the SAH. The SAH is originally introduced by Goldsmith and Salmon [GS87] for building BVHs and is adapted by MacDonald and Booth [MB90] for building kd-trees. The SAH assigns to each candidate voxel a cost equal to the product of the cost of processing this voxel and the probability of visiting this voxel. In order to calculate these probabilities for each voxel, the SAH assumes that:

1. the ray origins are uniformly distributed outside the scene's bounding box;
2. the ray directions are uniformly distributed;
3. the rays are infinitely long.

Several improvements have been made to the construction of the SAH. Havran et al. [HB02] introduced an efficient automatic termination criterion and clipping algorithm. Wald and Havran [WH06] encouraged cutting off large empty child voxels by reducing the expected cost of these splitting planes by a constant factor. This improvement usually results in slightly higher quality kd-trees. Hunt [Hun08] included mail-boxing, a ray-tracing optimization that removes redundant intersection tests. Wald and Havran [WH06] introduced a robust $\mathcal{O}(N \log N)$ build algorithm for kd-trees, where N is the total number of geometric primitives.

In addition, several fundamentally different heuristics are introduced to obtain a more accurate cost model since the assumptions of the SAH do not hold well in practice. Fabianowski et al. [FFD09] proposed the Scene Interior Ray Origins Heuristic which assumes ray origins to be uniformly distributed in the space inside the scene's bounding box. Havran and Bittner introduced build methods which take the actual distribution of rays into account for a fixed origin or direction [HB99] and by subsampling the rays [BH09]. Choi et al. [CCI12] proposed the Voxel Visibility Heuristic which considers the non-uniform distribution of rays by taking the occlusion of rays by geometric primitives into account. Reinhard et al. [RJK96] and Havran [Hav00] considered ray termination inside a voxel by including a blocking factor. Vinkler et al. [VHS12] presented a visibility-driven modification of the SAH for building BVHs.

All these build heuristics use a greedy divide-and-conquer approach. When these local heuristics are extended to a global heuristic which considers the positioning of the splitting planes at all levels of the tree simultaneously, the problem of building the optimal kd-tree becomes NP-hard.

Traversal order for shadow rays. Shadow rays need not to be traced in front to back order to find the first intersection because their sole function consist of reporting occlusion. Ize and Hansen [IH11] introduced the RTSAH (based on Havran [Hav00]) for determining the traversal order for occlusion rays in BVHs and BSPs. Their heuristic assigns to each voxel a probability that a ray terminates upon traversal through the voxel. When tracing a shadow ray, voxels with a higher termination probability are traversed first. For simplicity, non-empty leaf voxels are assumed to be completely opaque. The visibility of intermediate nodes is recursively obtained from its child nodes. Nah and Manocha [NM14] introduced the Surface Area Traversal Order metric which gives a higher traversal priority to the child voxel with the largest surface area.

Note that these algorithms, which change the traversal order, are orthogonal and fully compatible with our modified build heuristic.

Contribution. In this paper, we adapt the RTSAH to building kd-trees that reduce the cost of primary rays. By taking ray termination into account in the cost metric, we favor cutting off child voxels which are not or hardly visible to each other. This leads to a different ranking of the splitting planes which results in fundamentally different, more refined and more qualitative kd-trees compared to those obtained with the standard SAH.

For our RTSAH we introduce two new practical blocking factor approximations based on rasterization and the average projected surface area of the geometric primitives (not the axis-aligned bounding boxes).

3. Theoretical Framework

Hierarchical acceleration data structures that partition three-dimensional space such as kd-trees are often used to accelerate ray-tracing. To construct efficient acceleration structures, splitting planes that minimize the per-ray cost need to be chosen at each level of the tree. For computational efficiency, this is usually achieved by using appropriate (greedy) heuristics. In this section we briefly recall the SAH and introduce the modification of the original RTSAH [IH11] for building kd-trees.

3.1. Surface Area Heuristic

The SAH estimates the cost of splitting a voxel V with a splitting plane positioned in S into a left V_L and a right V_R child voxel as:

$$\mathcal{C}_{\text{SAH}}(S : V \rightarrow \{V_L, V_R\}) = C_t + p_L \cdot \mathcal{C}(V_L) + p_R \cdot \mathcal{C}(V_R) \quad (1)$$

The heuristic consists of the cost of traversing the parent voxel, C_t , and the costs of visiting the child voxels, $\mathcal{C}(V_L)$ and $\mathcal{C}(V_R)$, multiplied by the probability of visiting these voxels, p_L and p_R . Figure 1 illustrates the partitioning of a voxel V .

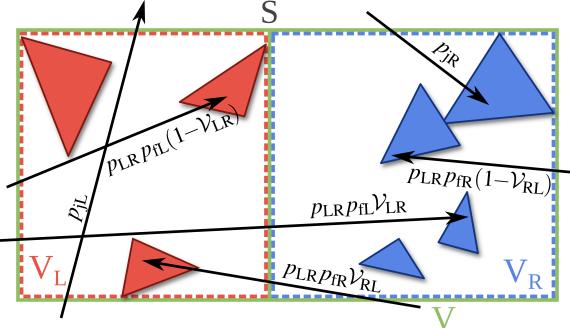


Figure 1: The types of rays considered by the RTSAH with their corresponding probabilities: rays piercing just one child voxel (with probabilities p_{jL} and p_{jR}), rays piercing both child voxels while hitting (with probabilities $p_{LR}p_{fL}(1-\mathcal{V}_{LR})$ and $p_{LR}p_{fR}(1-\mathcal{V}_{RL})$) or not hitting (with probabilities $p_{LR}p_{fL}\mathcal{V}_{LR}$ and $p_{LR}p_{fR}\mathcal{V}_{RL}$) a geometric primitive inside the voxel that is pierced first. The RTSAH reduces to the SAH when $\mathcal{V}_{LR} = \mathcal{V}_{RL} = 1$.

The cost $\mathcal{C}(V)$ of visiting a voxel is defined as the cost of testing a ray for intersection with all the geometric primitives contained in V . The probabilities p_L and p_R are the conditional probabilities of respectively piercing the left or right child voxel after piercing the parent voxel. When the ray origins and directions are uniformly distributed outside the scene's bounding box, these probabilities are exactly equal to:

$$p_L = p(V_L \text{ pierced} | V \text{ pierced}) = \frac{SA_{V_L}}{SA_V} \quad (2)$$

$$p_R = p(V_R \text{ pierced} | V \text{ pierced}) = \frac{SA_{V_R}}{SA_V} \quad (3)$$

where SA_{V_k} is the surface area of V_k .

The cost of not splitting a voxel is defined as the cost of intersecting all the geometric primitives contained in this voxel:

$$\mathcal{C}_{\text{no split}}(V) = \mathcal{C}(V). \quad (4)$$

A voxel will only be split into two child voxels when the cost of splitting is lower than the cost of not splitting.

3.2. Ray Termination Surface Area Heuristic

By not taking the presence of geometric primitives in the probabilities p_L and p_R into account, the SAH does not distinguish between rays piercing both child voxels while hitting or not hitting a geometric primitive inside the voxel that is pierced first. The RTSAH extends the SAH by distinguishing between these types of rays and by taking ray termination into account. The RTSAH cost function is defined as:

$$\begin{aligned} \mathcal{C}_{\text{RTSAH}}(S : V \rightarrow \{V_L, V_R\}) &= \mathcal{C}_t + p_{jL} \cdot \mathcal{C}(V_L) + p_{jR} \cdot \mathcal{C}(V_R) + \\ &p_{LR} \left(p_{fL} \left(\mathcal{C}(V_L) + \mathcal{V}_{LR} \cdot \mathcal{C}(V_R) \right) + p_{fR} \left(\mathcal{C}(V_R) + \mathcal{V}_{RL} \cdot \mathcal{C}(V_L) \right) \right) \end{aligned} \quad (5)$$

p_{jL} , p_{jR} and p_{LR} express the probability of piercing just the left child, just the right child and both child voxels respectively. When both child voxels are pierced, we distinguish the case where either the left or right child voxel is pierced first using the probabilities p_{fL} and p_{fR} . Rays which pierce both child voxels can potentially be terminated by a geometric primitive in the child voxel that is pierced first. The visibility probability, \mathcal{V}_{LR} (\mathcal{V}_{RL}), expresses the fraction of rays, piercing the left (right) child voxel first and then the right (left) child voxel, that reach the splitting plane without being terminated by a geometric primitive inside the left (right) child voxel. The types of rays with their corresponding probabilities are illustrated in Figure 1.

We obtain expressions for the probabilities p_{jL} , p_{jR} and p_{LR} using Equation 2 and 3:

$$\begin{aligned} p_{jL} &= p(\text{just } V_L \text{ pierced} | V \text{ pierced}) \\ &= 1 - p_R \end{aligned} \quad (6)$$

$$\begin{aligned} p_{LR} &= p(V_L \text{ and } V_R \text{ pierced} | V \text{ pierced}) \\ &= 1 - p_{jL} - p_{jR} \end{aligned} \quad (7)$$

$$\begin{aligned} p_{fL} &= p(V_L \text{ pierced first} | V_L \text{ and } V_R \text{ pierced}) \\ &= \frac{1}{2} \end{aligned} \quad (8)$$

This is analogously for p_{jR} and p_{fR} . The visibility probabilities between the child voxels are defined as:

$$\mathcal{V}_{LR} = p(\text{no hit in } V_L | V_L \text{ pierced first, then } V_R) \quad (9)$$

$$\mathcal{V}_{RL} = p(\text{no hit in } V_R | V_R \text{ pierced first, then } V_L) \quad (10)$$

Note that the RTSAH cost becomes equal to the SAH cost when $\mathcal{V}_{LR} = \mathcal{V}_{RL} = 1$.

The cost of not splitting a voxel is the same as for the SAH (Equation 4). The empty space bonus [WH06] and its underlying motivation can also be applied to the RTSAH.

4. Practical Algorithm

In order to obtain a practical build algorithm based on the RTSAH, we need to evaluate the splitting cost (Equation 5) for a finite set of splitting planes.

The probabilities of a ray intersecting a single or both child voxels can be trivially calculated using Equations 6 to 8. However, the exact calculation of the visibility probabilities \mathcal{V}_{LR} and \mathcal{V}_{RL} requires the integration of the visibility over the hemisphere of incoming directions and the area of the splitting plane (see Figure 2a). Due to the dependence on the geometrical distributions in the child voxels no closed-form expression can be found for the visibility probabilities.

While the visibility probabilities can be calculated via Monte Carlo integration techniques, this is impractical due to the absence of an acceleration data structure during the construction phase. Furthermore, performing a Monte Carlo sampling of the visibility probabilities at each possible splitting plane would result in prohibitively large build times. The number of samples needs to be low when lots of geometric primitives overlap the voxel that must be split, which is the case for the first split decisions of the kd-tree. On the contrary, the choice of splitting plane is more important and decisive at these first split decisions and thus asks for an accurate estimator.

In order to maintain a tractable build time, we aim at a practical algorithm which for each split decision:

1. considers only a finite set of splitting planes including the best splitting plane according to our approximated RTSAH;
2. allows for an incremental calculation of the visibility probabilities.

Both conditions are hard to achieve due to the directional dependence of the visibility probabilities. Therefore, we suggest to eliminate this dependence by considering only a single direction: the orthogonal projection direction (section 4.1) and the average projected direction (section 4.2) onto the splitting plane. In section 4.3, we validate these approximations. In sections 4.4 and 4.5, we show that these approximations to the visibility probabilities lead to a practical build algorithm based on the RTSAH for which both conditions are satisfied.

4.1. All Points One Direction RTSAH

The All Points One Direction (APOS) RTSAH considers all the points on the splitting plane and ignores the directional dependence by considering only the direction orthogonal to the splitting plane. This corresponds to projecting the geometric primitives onto the splitting plane. The visibility is then equal to the area of the splitting plane which is not covered by the projected primitives divided by the total area of the splitting plane. This can be calculated efficiently using rasterization. To perform the rasterization, the splitting plane is partitioned into a finite number of square rasterization cells of which the center is called a rasterization point. A rasterization cell is covered by a geometric primitive if and only if the orthogonal projection of the geometric primitive covers the rasterization point of this cell. Figure 2b illustrates this idea.

The idea of calculating the visibility between two neighbouring child voxels with an orthogonal projection onto the splitting plane is similar to calculating the blocking factor of a child voxel in [RKJ96]. In contrast to [RKJ96], we do not use the axis aligned bounding boxes of the geometric primitives, but use the actual geometric primitives during rasterization. Furthermore, we implicitly avoid overlapping of geo-

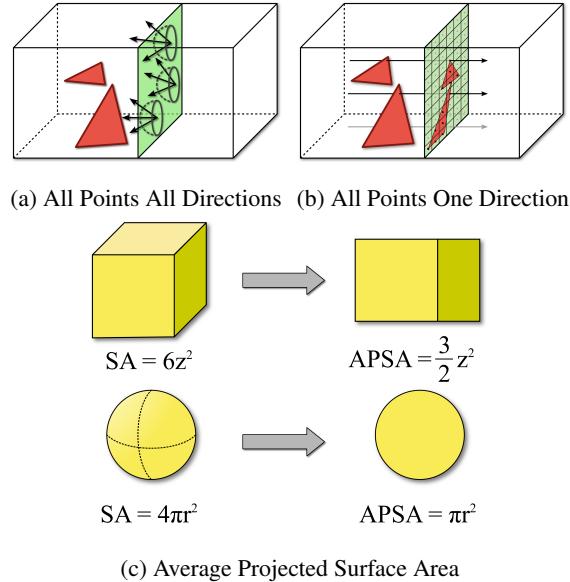


Figure 2: The ideas behind the different approximations to the RTSAH. (a) shows the ground truth, where we integrate over the splitting plane's area and the hemisphere of incoming directions. (b) shows our APOS approximation, where we integrate over every point but only take the direction orthogonal to the splitting plane into account. (c) the average projected surface area of a convex solid.

metric primitives by rasterizing and obtain both a cost model and build procedure.

The rasterization can be seen as a special form of sampling, resulting in a biased estimator. The rasterization of geometric primitives is however computationally cheaper than testing random rays for intersection, reuses intersection information while shifting the splitting plane in the orthogonal direction and does not require a second kd-tree compared to [Hav00].

4.2. Average Projected Surface Area RTSAH

Cauchy showed that the average projected area of a three-dimensional convex solid is one fourth of its surface area (see Figure 2c). This property is used for obtaining the probabilities p_L and p_R of Equation 2 and 3.

The Average Projected Surface Area (APSA) RTSAH approximates the visibility probabilities as the fraction of the area of the splitting plane that remains after subtracting the average projected surface area of the geometry contained in a child voxel:

$$\mathcal{V}_{LR} \approx 1 - \min \left(\frac{SA_{G_{V_L}}}{4SA_S}, 1 \right), \mathcal{V}_{RL} \approx 1 - \min \left(\frac{SA_{G_{V_R}}}{4SA_S}, 1 \right) \quad (11)$$

Where $\text{SA}_{G_{V_k}}$ is the surface area of all the geometric primitives contained in voxel V_k and SA_S the surface area of the splitting plane. Note that when the sets G_{V_L} and G_{V_R} represent a convex solid fully contained in V_L and V_R respectively, the above approximations are exact.

4.3. Validation

To validate our approximations of the visibility probabilities, we constructed an artificial scene consisting of a variable number of stratified placed icosahedrons. The visibility probabilities in this context express the probability that a ray passes through the different planes of the scenes' bounding box. To obtain ground truth visibility probabilities, the scenes' bounding box got pierced by 100M rays which are uniformly distributed on the scenes' bounding sphere and are guaranteed to pierce the scenes' bounding box. For the APOD we used different numbers (16^2 , 32^2 , 64^2 , 128^2 , 256^2 , 512^2 , 1024^2) of rasterization points. The results are shown in Figure 3.

The APSA consistently results in the smallest difference between the approximated and exact visibility probabilities for the 6 planes of a scene's bounding box compared to the APOD. Note the slow linear increase of these underestimations compared to the exponential increase in the number of icosahedrons. The APOD which rasterizes onto 1024^2 points does slightly worse. Unfortunately, the APOD needs a small number of rasterization points to result in a fast build procedure. For these scenes this is not problematic since the other APOD approximations differ at most 5% with an exception for the approximation which uses 16^2 rasterization points. Considering only 16 rasterization points in 1D is not accurate enough given the number of geometric primitives in these scenes. Furthermore, the result becomes too sensitive to small shifts of the rasterization points. The APSA results on average in an underestimation of 12.7% in the scene with 2048 icosahedrons which is still a reasonable approximation given the difficulty of calculating the exact visibility probabilities for this scene.

We conclude that for these difficult (with regard to visibility) scenes, the APSA results in satisfactory approximations of the visibility probabilities. The APOD on the other hand is less accurate and depends on the number of rasterization points used. In the following section, we use at least 16 and at most 64 rasterization points in 1D depending on the number of geometric primitives that overlap the voxel that must be split.

4.4. Split candidates

For simplicity, we do not clip geometric primitives that overlap multiple voxels when rasterizing (APOD) or calculating surface areas (APSA). Therefore, the APOD and APSA RTSAH cost functions increase or decrease monotonously in the region between two consecutive geometric primitive

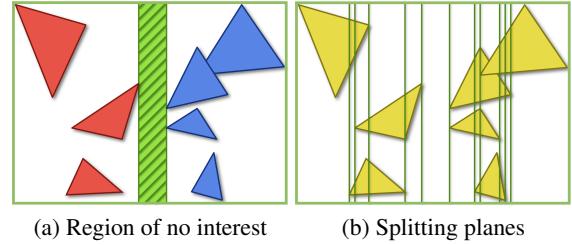


Figure 4: (a) Shows a region (green) where the SAH, APOD and APSA RTSAH cost functions are monotonously increasing/decreasing. (b) The finite set of splitting planes (green) along a single direction considered by the SAH, APOD RTSAH and APSA RTSAH to obtain the optimal splitting plane according to their respective metrics.

boundaries along the same axis (illustrated in Figure 4a). Within such a region, the approximations of the visibility probabilities \mathcal{V}_{LR} , \mathcal{V}_{RL} remain constant and p_{jL} , p_{jR} are both linear functions of the position of the splitting plane. This means that we only have to consider splitting planes at the boundaries of the geometric primitives in order to obtain the best splitting plane according to our cost metric (see Figure 4b). This is the same set of splitting planes considered by the SAH.

4.5. Build algorithm

We only have to make some small modifications to the build procedure for the SAH without changing the overall computational complexity. Obtaining p_{jL} , p_{jR} , p_{LR} is very similar to calculating p_L and p_R . The visibility probabilities are incrementally updated during a single pass through the splitting candidates for each primary direction. When moving the splitting plane from the left to the right, this requires incrementally decreasing \mathcal{V}_{LR} and increasing \mathcal{V}_{RL} . The calculation of the visibility probabilities depends on the approximation used:

APOD RTSAH: For \mathcal{V}_{LR} we need to keep track which rasterization points are not overlapped by geometric primitives contained in V_L (invariant). For \mathcal{V}_{RL} we need to keep track how many geometric primitives contained in V_R cover each rasterization point (invariant). Therefore, all geometric primitives contained in V need to be rasterized onto the splitting plane prior to determining the costs of each split candidate.

APSA RTSAH: To calculate \mathcal{V}_{LR} and \mathcal{V}_{RL} , we have to keep track of the sum of the surface area of all geometric primitives contained in the left and right child voxel respectively (invariants). This can be done incrementally by initializing \mathcal{V}_{RL} with the sum of the surface area's of all the geometric primitives contained in voxel V .

The data structures that needs to guarantee these invari-

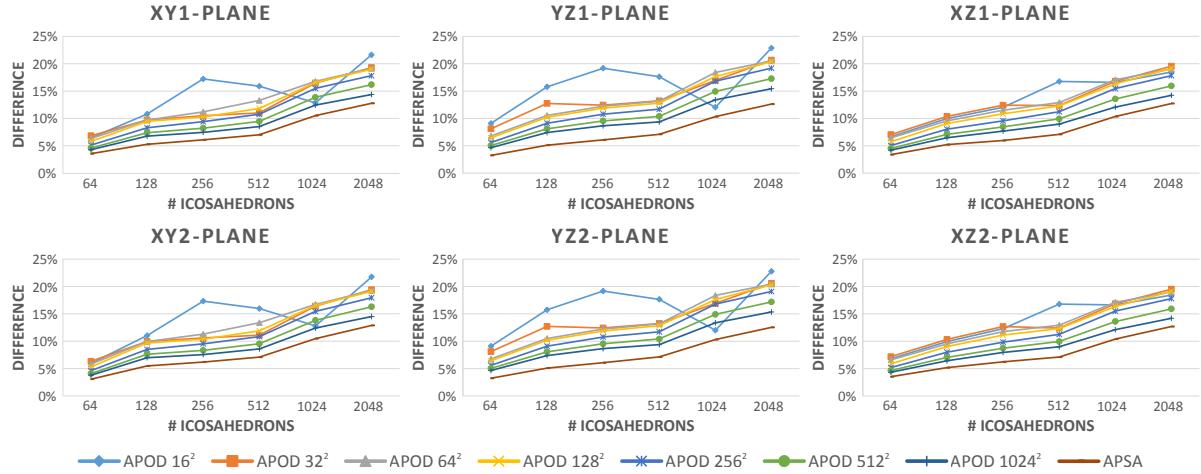


Figure 3: The difference between the approximated and exact (100M rays) visibility probabilities for the 6 planes of the scene’s bounding box. A positive (negative) difference corresponds to an underestimation (overestimation).

ants, must be updated (incremented/decremented) appropriately when a geometric primitive starts overlapping the left child voxel or stops overlapping the right child voxel.

5. Experimental Results

We have implemented the APOD and APSA RTSAH in PBRT [PH10]. Our test scenes are shown in Figure 5. We compare the obtained kd-tree structures in 5.1 and the performance with regard to the number of intersection tests and traversal steps in Section 5.2 as opposed to the SAH.

5.1. Kd-tree properties

Table 1 summarizes the properties of the obtained kd-trees. The kd-trees obtained with the APSA RTSAH contain less geometric primitives per leaf as opposed to the SAH. Therefore leaves are traversed faster on average. On the other hand, the subtrees are more refined, resulting in 30 to 60% as many voxels and more duplicated references to geometric primitives for the same maximal depth. Therefore the average number of traversed voxels will be higher, more memory needs to be allocated, but also fundamentally better kd-trees can be obtained as opposed to the SAH.

We see similar results with regard to the number of (empty) leaves and geometric primitives for the kd-trees obtained with the APOD RTSAH. Only the sponza scene is less refined compared to the SAH. Despite the refinement, the number of geometric primitives per leaf as opposed to the APSA RTSAH is less ideal.

Both RTSAH build times are higher than the SAH due to the larger and deeper obtained kd-trees. Therefore, the APSA RTSAH results in minimal overhead compared to the

SAH for the same number of split decisions. The APOD RTSAH does worse, because rasterizing at the highest levels of the kd-tree is computationally expensive. Our APOD or APSA RTSAH (CPU) build procedures are not optimized and do not cache surface areas or rasterization info for geometric primitives. These improvements can still result in a speedup, but are not tested.

5.2. Performance

Table 2 shows the difference in geometric primitive intersection tests, traversed voxels for different rays and the difference in total rendering time. As expected due to the properties of the kd-trees, more voxels need to be traversed for the APOD and APSA RTSAH due to the more refined subtrees compared to the SAH.

More importantly, we clearly see a large reduction in geometric primitive intersection tests for the APSA RTSAH. We obtain reductions up to 47% for primary rays for the sibenik and sponza scenes. In these scenes, ray termination is important to consider by a build heuristic since every ray will eventually hit some surface. We obtain a small increase in intersection tests for the bunny scene consisting of a single object and a floor plate. The APSA RTSAH needs less tests on the bunny, but performs slightly worse on the floor plate. Despite only focusing on exterior rays, we even see reductions to 41% for shadow rays (when traversing the voxels in order). This is due to the larger number of duplicated references to geometric primitives.

The APSA RTSAH generally outperforms the APOD RTSAH in more complex scenes (such as the sibenik and sponza scenes). By considering only the incoming normal direction for the APOD RTSAH, we note a difference

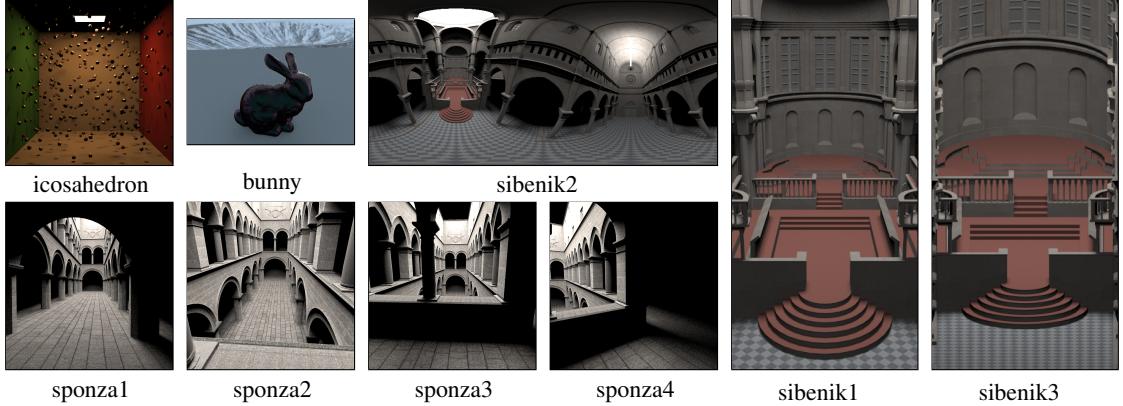


Figure 5: Test scenes. The icosahedron scene consists of approximately 7k, the bunny scene of 69k, the sibenik scenes of 80k and the sponza scenes of 66k geometric primitives.

scene	build time			#leafs			#empty leafs		
	SAH	APOD	APSA	SAH	APOD	APSA	SAH	APOD	APSA
icosahedron	0.1s	3.3s	0.2s	65 k	87 k	94 k	6 k	6 k	6 k
bunny	0.6s	30.2s	2.1s	873 k	1.17 M	1.22 M	198 k	224 k	219 k
sibenik	0.7s	31.7s	2.5s	607 k	829 k	888 k	127 k	142 k	156 k
sponza	0.6s	16.3s	2.1s	502 k	452 k	823 k	130 k	109 k	176 k
scene	#references to geometric primitives			maximum #geometric primitives/leaf			average #geometric primitives /non-empty leaf		
	SAH	APOD	APSA	SAH	APOD	APSA	SAH	APOD	APSA
icosahedron	170 k	224 k	239 k	11	8	8	2.86	2.78	2.71
bunny	1.80 M	2.46 M	2.53 M	12	11	11	2.67	2.59	2.52
sibenik	1.97 M	2.87 M	2.95 M	81	72	36	4.10	4.18	4.03
sponza	1.64 M	1.66 M	2.59 M	57	54	44	4.41	4.84	4.01

Table 1: Kd-tree structures obtained for the SAH, APOD and APSA RTSAH build heuristics for the test scenes.

between sibenik1 (perspective camera) and sibenik3 (orthographic camera). On the contrary, the APSA RTSAH results in similar gains independent of the type of camera used.

The false color images in Figure 6 show a more detailed comparison between the build heuristics. Here we can for example see that less intersection tests are performed near the pillars of the sponza scenes due to the inclusion of ray termination in our RTSAH approximations.

While we achieve significant reductions in intersection tests, the impact on the total rendering time is less pronounced. Our profiler showed that (depending on the scene and build heuristic) less than 35% of the time is spent on testing geometric primitives for intersection or traversing the acceleration data structure. Therefore, the reductions in rendering time are smaller.

6. Conclusions and further work

Our RTSAHs result in more qualitative kd-trees compared to the SAH by taking ray termination into account. Furthermore, our RTSAH build procedures have the same overall computational complexity and need to consider the same finite set of splitting planes as the SAH.

We plan to precisely investigate the differences between the acceleration data structures generated by the SAH and our RTSAHs and to further validate the underlying assumptions and simplifications. Furthermore, we want to use our approximated visibility probabilities to determine the traversal order for shadow rays as opposed to the (original) RTSAH introduced by Ize and Hansen [IH11]. Both the SAH and RTSAH do not take the actual ray distribution nor the presence of internal rays into account. The SAH for scene-interior ray origins introduced by Fabianowski et al. [FFD09] could also benefit from the ray termination of our RTSAHs.

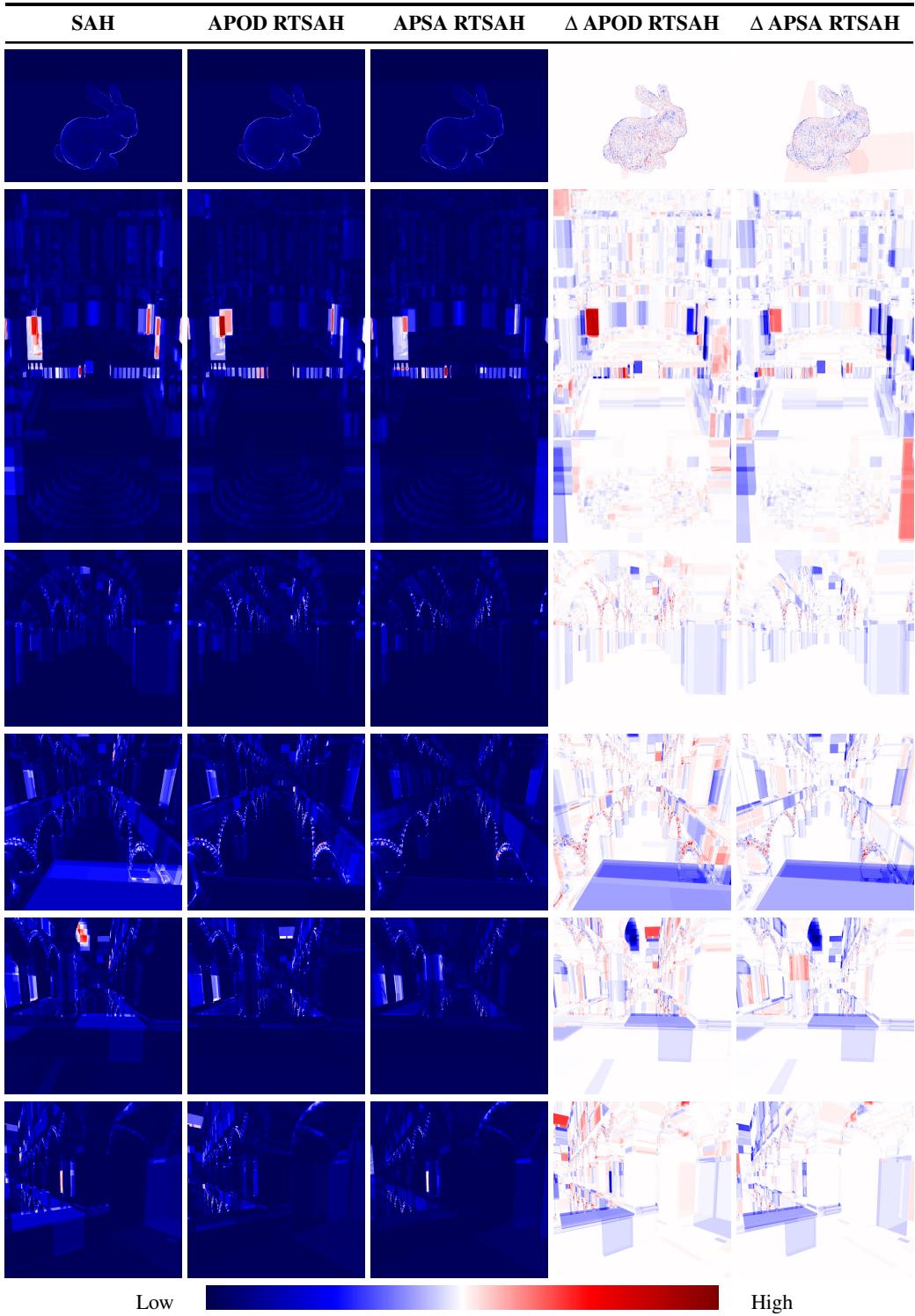


Figure 6: False color images for the number of intersection tests for primary rays for the SAH, APOD and APSA RTSAH and the difference (blue (red) corresponds to a gain (loss)) between the SAH and RTSAH approximations. All scenes are rendered with a Whitted ray tracer [Whi80] using 512 samples per pixel. The scale is linear.

scene	Δ intersection tests for primary rays		Δ intersection tests for shadow rays		Δ traversed voxels for primary rays		Δ traversed voxels for shadow rays		Δ total rendering time	
	APOD	APSA	APOD	APSA	APOD	APSA	APOD	APSA	APOD	APSA
icosahedron	-1.7%	-0.5%	-1.8%	-8.5%	1.2%	-0.6%	3.0%	-0.8%	-3.6%	-4.4%
bunny	-1.8%	0.9%	-1.5%	-3.7%	5.5%	2.6%	7.4%	2.7%	1.8%	-1.0%
sibenik1	-13.5%	-18.7%	-3.9%	-13.8%	13.4%	11.6%	15.5%	15.6%	-1.6%	-3.5%
sibenik2	-10.6%	-20.2%	-9.1%	-17.7%	8.0%	6.7%	12.3%	12.0%	-3.2%	-4.8%
sibenik3	-17.9%	-17.9%	-8.0%	-9.3%	13.7%	10.7%	14.3%	15.2%	0.4%	-0.1%
sponza1	-12.5%	-25.5%	-3.1%	-22.1%	8.2%	7.6%	-2.2%	10.6%	-3.9%	-4.8%
sponza2	-38.0%	-46.7%	-6.0%	-29.2%	3.0%	20.0%	-0.5%	14.4%	-7.4%	-7.6%
sponza3	-20.5%	-30.7%	-18.9%	-40.8%	4.6%	12.2%	1.3%	10.8%	-7.5%	-8.5%
sponza4	-12.9%	-27.4%	-14.8%	-33.8%	18.6%	7.1%	3.4%	10.4%	-0.4%	-3.1%

Table 2: The difference in geometric primitive intersection tests, traversed voxels for primary and shadow rays and total rendering time for the APOD and APSA RTSAH relative to the SAH, where a negative (positive) percentage corresponds to a decrease (increase). All scenes are rendered with a Whitted ray tracer [Whi80] using 512 samples per pixel (on 8 logical threads).

7. Acknowledgments

The scenes used in this paper are courtesy of PBRT [PH10]. Niels Billen is funded by the Agency for Innovation by Science and Technology in Flanders (IWT).

References

- [BH09] BITTNER J., HAVRAN V.: Rdh: ray distribution heuristics for construction of spatial data structures. In *Proceedings of the 25th Spring Conference on Computer Graphics* (2009), ACM, pp. 51–58. [2](#)
- [CCI12] CHOI B., CHANG B., IHM I.: Construction of efficient kd-trees for static scenes using voxel-visibility heuristic. *Computers Graphics* 36, 1 (2012), 38 – 48. [2](#)
- [FFD09] FABIANOWSKI B., FOWLER C., DINGLIANA J.: A cost metric for scene-interior ray origins. *Eurographics Short Papers* (2009), 49–52. [2, 7](#)
- [GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.* 7, 5 (May 1987), 14–20. [2](#)
- [Hav00] HAVRAN V.: *Heuristic ray shooting algorithms*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University, Prague, 2000. [2, 4](#)
- [HB99] HAVRAN V., BITTNER J.: Rectilinear bsp trees for preferred ray sets. *Proceedings of SC'99* (1999), 171–179. [2](#)
- [HB02] HAVRAN V., BITTNER J.: On improving kd-trees for ray shooting. In *Proc. of WSCG 2002 Conference* (2002), pp. 209–217. [2](#)
- [HDW*11] HAPALA M., DAVIDOVIČ T., WALD I., HAVRAN V., SLUSALLEK P.: Efficient stack-less bvh traversal for ray tracing. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2011), ACM, pp. 7–12. [2](#)
- [HH11] HAPALA M., HAVRAN V.: Review: Kd-tree traversal algorithms for ray tracing. *Computer Graphics Forum* 30, 1 (2011), 199–213. [2](#)
- [Hun08] HUNT W. A.: Corrections to the surface area metric with respect to mail-boxing. In *Interactive Ray Tracing 2008, IEEE Symposium on* (2008), IEEE, pp. 77–80. [2](#)
- [IH11] IZE T., HANSEN C.: Rtsah traversal order for occlusion rays. *Computer Graphics Forum* 30, 2 (2011), 297–305. [1, 2, 7](#)
- [MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *Vis. Comput.* 6, 3 (May 1990), 153–166. [1, 2](#)
- [NM14] NAH J.-H., MANOCHA D.: Sato: Surface-area traversal order for shadow ray tracing. *Computer Graphics Forum* (2014). [2](#)
- [PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010. [6, 9](#)
- [RKJ96] REINHARD E., KOK A. J. F., JANSEN F. W.: Cost prediction in ray tracing. In *Rendering Techniques* (1996), Springer, pp. 41–50. [2, 4](#)
- [VHS12] VINKLER M., HAVRAN V., SOCHOR J.: Visibility driven bvh build up algorithm for ray tracing. *Computers & Graphics* 36, 4 (2012), 283–296. [2](#)
- [WH06] WALD I., HAVRAN V.: On building fast kd-trees for ray tracing, and on doing that in $O(n \log n)$. In *Interactive Ray Tracing 2006, IEEE Symposium on* (2006), IEEE, pp. 61–69. [2, 3](#)
- [Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM* 23, 6 (June 1980), 343–349. [8, 9](#)
- [WMG*09] WALD I., MARK W. R., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S. G., SHIRLEY P.: State of the art in ray tracing animated scenes. *Computer Graphics Forum* 28, 6 (2009), 1691–1722. [1](#)

Bibliografie

- [AK87] ARVO J., KIRK D.: Fast ray tracing by ray classification. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 55–64.
- [Arv86] ARVO J.: Backward ray tracing. In *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing* (1986), pp. 259–263.
- [Aur91] AURENHAMMER F.: Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput. Surv.* 23, 3 (Sept. 1991), 345–405.
- [BF79] BENTLEY J. L., FRIEDMAN J. H.: Data structures for range searching. *ACM Comput. Surv.* 11, 4 (Dec. 1979), 397–409.
- [BH09] BITTNER J., HAVRAN V.: Rdh: ray distribution heuristics for construction of spatial data structures. In *Proceedings of the 25th Spring Conference on Computer Graphics* (2009), ACM, pp. 51–58.
- [Caf98] CAFLISCH R. E.: Monte carlo and quasi-monte carlo methods. *Acta Numerica* 7 (1 1998), 1–49.
- [CCI12] CHOI B., CHANG B., IHM I.: Construction of efficient kd-trees for static scenes using voxel-visibility heuristic. *Computers Graphics* 36, 1 (2012), 38 – 48.
- [CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 137–145.
- [DBBS03] DUTRÉ P., BALA K., BEKAERT P., SHIRLEY P.: *Advanced global illumination*, vol. 11. AK Peters, 2003.
- [dBCvKO08] DE BERG M., CHEONG O., VAN KREVELD M., OVERMARS M.: *Computational Geometry: Algorithms and Applications*. Springer Berlin Heidelberg, 2008.
- [FFD09] FABIANOWSKI B., FOWLER C., DINGLIANA J.: A cost metric for scene-interior ray origins. *Eurographics Short Papers* (2009), 49–52.

- [Fle09] FLEISZ M.: *Photon Mapping on the GPU*. Master thesis, School of Informatics, University of Edinburgh, 2009.
- [FTI86] FUJIMOTO A., TANAKA T., IWATA K.: Artsccelerated ray-tracing system. *IEEE Comput. Graph. Appl.* 6, 4 (Apr. 1986), 16–26.
- [Gla84] GLASSNER A.: Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications* 4, 10 (1984), 15–22.
- [GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.* 7, 5 (May 1987), 14–20.
- [Hav00] HAVRAN V.: *Heuristic ray shooting algorithms*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University, Prague, 2000.
- [HB99] HAVRAN V., BITTNER J.: Rectilinear bsp trees for preferred ray sets. *Proceedings of SCCG 99* (1999), 171–179.
- [HB02] HAVRAN V., BITTNER J.: On improving kd-trees for ray shooting. In *In Proc. of WSCG 2002 Conference* (2002), pp. 209–217.
- [HDW*11] HAPALA M., DAVIDOVIČ T., WALD I., HAVRAN V., SLUSALLEK P.: Efficient stack-less bvh traversal for ray tracing. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2011), ACM, pp. 7–12.
- [HH11] HAPALA M., HAVRAN V.: Review: Kd-tree traversal algorithms for ray tracing. *Computer Graphics Forum* 30, 1 (2011), 199–213.
- [HHS05] HAVRAN V., HERZOG R., SEIDEL H.-P.: Fast final gathering via reverse photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2005)* 24, 3 (August 2005), 323–333.
- [HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 141.
- [HKWB09] HAŠAN M., KŘIVÁNEK J., WALTER B., BALA K.: Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 143.
- [HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Transactions on Graphics (TOG)* 27, 5 (2008), 130.
- [Hun08] HUNT W. A.: Corrections to the surface area metric with respect to mail-boxing. In *Interactive Ray Tracing 2008, IEEE Symposium on* (2008), IEEE, pp. 77–80.
- [ICG86] IMMEL D. S., COHEN M. F., GREENBERG D. P.: A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 133–142.

- [IH11] IZE T., HANSEN C.: Rtsah traversal order for occlusion rays. *Computer Graphics Forum* 30, 2 (2011), 297–305.
- [ISP07] IZE T., SHIRLEY P., PARKER S.: Grid creation strategies for efficient ray tracing. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), RT ’07, IEEE Computer Society, pp. 27–32.
- [IWP08] IZE T., WALD I., PARKER S. G.: Ray tracing with the bsp tree. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (2008), IEEE, pp. 159–166.
- [JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 311–320.
- [Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques’ 96*. Springer, 1996, pp. 21–30.
- [Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.
- [Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH ’86, ACM, pp. 143–150.
- [Kap85] KAPLAN M. R.: The use of spatial coherence in ray tracing. *ACM SIGGRAPH Course Notes* 11 (1985).
- [Kel97] KELLER A.: Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 49–56.
- [KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 25.
- [LD08a] LAGAE A., DUTRÉ P.: Accelerating ray tracing using constrained tetrahedralizations. *Computer Graphics Forum* 27, 4 (2008), 1303–1312.
- [LD08b] LAGAE A., DUTRÉ P.: Compact, fast and robust grids for ray tracing. *Computer Graphics Forum* 27, 4 (2008), 1235–1244.
- [LW93] LAFORTUNE E. P., WILLEMS Y.: Bi-directional path tracing. In *Compugraphics ’93, Compugraphics ’93, Alvor, Portugal* (Dec 1993), pp. 145–153.

- [MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *Vis. Comput.* 6, 3 (May 1990), 153–166.
- [NM14] NAH J.-H., MANOCHA D.: Sato: Surface-area traversal order for shadow ray tracing. *Computer Graphics Forum* (2014).
- [NNDJ12a] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Progressive virtual beam lights. *Computer Graphics Forum* 31, 4 (2012), 1407–1413.
- [NNDJ12b] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 60.
- [OMSD87] OOI B. C., McDONELL K., SACKS-DAVIS R.: Spatial kd-tree: a data structure for geographic databases. In *Proceedings of the IEEE COMPSAC Conference* (1987).
- [PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stackless kd-tree traversal for high performance gpu ray tracing. *Computer Graphics Forum* 26, 3 (2007), 415–424.
- [PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.
- [RKC98] REINHARD E., KOK A. J., CHALMERS A.: Cost distribution prediction for parallel ray tracing. *Second Eurographics Workshop on Parallel Graphics and Visualisation* (1998), 77–90.
- [RKJ96] REINHARD E., KOK A. J. F., JANSEN F. W.: Cost prediction in ray tracing. In *Rendering Techniques* (1996), Springer, pp. 41–50.
- [Sam15] SAMAEY G.: *Numerical methods for stochastic simulation and integration*. 2015.
- [Sil86] SILVERMAN B. W.: *Density estimation for statistics and data analysis*, vol. 26. CRC press, 1986.
- [Sim96] SIMONOFF J. S.: *Smoothing methods in statistics*. Springer Science & Business Media, 1996.
- [SW00] SUYKENS F., WILLEMS Y. D.: Density control for photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, UK, 2000), Springer-Verlag, pp. 23–34.
- [Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162.

- [VG95] Veach E., Guibas L. J.: Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 419–428.
- [VG97] Veach E., Guibas L. J.: Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 65–76.
- [Wal99] Wald I.: *Photorealistic Rendering using the Photon Map*. Master thesis, Numerical Algorithms Group, University of Kaiserslautern, Germany, 1999.
- [WBKS03] Williams A., Barrus S., Keith R., Shirley M. P.: An efficient and robust ray-box intersection algorithm. *Journal of Graphics Tools* 10 (2003), 54.
- [WGS04] Wald I., Günther J., Slusallek P.: Balancing considered harmful – faster photon mapping using the voxel volume heuristic. *Computer Graphics Forum* 23, 3 (2004), 595–604.
- [WH06] Wald I., Havran V.: On building fast kd-trees for ray tracing, and on doing that in $O(n \log n)$. In *Interactive Ray Tracing 2006, IEEE Symposium on* (2006), IEEE, pp. 61–69.
- [Whi80] Whitted T.: An improved illumination model for shaded display. *Commun. ACM* 23, 6 (June 1980), 343–349.
- [WK06] Wächter C., Keller A.: Instant ray tracing: The bounding interval hierarchy. *Rendering Techniques 2006* (2006), 139–149.
- [WMG*09] Wald I., Mark W. R., Günther J., Boulos S., Ize T., Hunt W., Parker S. G., Shirley P.: State of the art in ray tracing animated scenes. *Computer Graphics Forum* 28, 6 (2009), 1691–1722.

Fiche masterproef

Student: Matthias Moulin

Titel: Hybride kd-bomen voor photonmapping en het versnellen van ray tracing

Engelse titel: Hybrid Kd-trees for Photon Mapping and Accelerating Ray Tracing

UDC: 004.92

Korte inhoud:

We introduceren drie hybride kd-bomen waarin zowel de geometrische primitieven als photons worden ondergebracht met als doel de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel te verminderen: De Hybrid Additive Photon Map (HAPM) voegt de photons toe aan de kd-boom acceleratiestructuur en gebruikt hiervoor de opdeling van de geometrische primitieven om de photons op te delen. De Hybrid Swapping Photon Map (HSPM) bouwt een hybride kd-boom op basis van de geometrische primitieven én photons met behulp van de door ons geïntroduceerde Hybrid Heuristic (HH) bouwheuristiek. De Hybrid Connected Photon Map (HCPM) behoudt de aparte kd-boom acceleratiestructuur en kd-boom photonmap en verbindt deze door middel van ropes. Onze HH kapselt zowel de Ray Termination Surface Area Heuristic (RTSAH) voor het bouwen van kd-boom acceleratiestructuren als de Voxel Volume Heuristic (VVH) [WGS04] voor het bouwen van kd-boom photonmappen in en koppelt de kost van de ray tracing en k-nearest neighbor query's. De RTSAH [IH11] is een kostmetriek origineel gebruikt voor het bepalen van de doorloopvolgorde van de voxels voor occlusiestralen door straalterminatie in rekening te nemen. We passen deze RTSAH aan om kd-boom acceleratiestructuren te bouwen. We behalen reducties in intersectietesten tot 47% voor primaire stralen en tot 41% voor schaduwstralen (wanneer de voxels van voor naar achter doorlopen worden) in vergelijking met de Surface Area Heuristic (SAH) [MB90]. De HAPM en HSPM resulteren algemeen in aanzienlijke toenamen van de totale rendertijd ten opzichte van een niet-hybride photonmap. Enkel de HCPM resulteert gemiddeld in reducties tot minder dan 1% in totale rendertijd. We concluderen dat geometrische primitieven en photons niet verzoend kunnen worden noch in eenzelfde kd-boom noch in de veel gebruikte huidige acceleratiestructuren (reguliere roosters, BVHs, BSPs, octrees) opdat de gecombineerde kost van de ray tracing en k-nearest neighbor query's fundamenteel verminderd wordt ten opzichte van twee aparte datastructuren.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotor: Prof. dr. ir. Philip Dutré

Assessoren: Ir. Jeroen Baert

Dr. Aram Hovsepyan

Begeleider: Ir. Niels Billen