

Plückerbomen: efficiënt zoeken van k dichtstbijzijnde rechten rond punt in 3D, met toepassing in photonmapping

Wouter De Keersmaecker

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

Promotor:
Prof. dr. ir. P. Dutré

Assessoren:
Prof. dr. ir. K. Meerbergen
R. Frederickx

Begeleider:
R. Frederickx

© 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

Deze masterthesis het sluitstuk van vijf fantastische jaren aan de KU Leuven. Naast een berg nieuwe kennis en ervaring heb ik ook vele fantastische mensen leren kennen. Vooraleerst bedank ik graag Prof. Dr. Ir. P. Dutré voor het aanwakkeren van mijn interesse in computer graphics, de excellente lessen, de aangeboden kansen en de begeleiding bij deze thesis. Ik bedank eveneens mijn begeleider R. Frederickx. Hij beantwoordde iedere week mijn lange mails met grondige en nuttige feedback, zowel bij technische kwesties als bij het schrijven. Zijn inzichten en advies waren essentieel voor dit werk. Mijn ouders en familie bedank ik voor hun uitgebreide steun en de kansen die ze me geboden hebben. Verder bedank ik ook graag onder meer Tom, Eleonora, Bram en Thomas voor hun warme vriendschap en motivatie in het maken van deze thesis. Ten slotte wil ik ook jou, de lezer, bedanken voor het lezen van mijn thesis.

Wouter De Keersmaecker

Inhoudsopgave

Voorwoord	i
Inhoudsopgave	ii
Samenvatting	iv
Lijst van figuren en tabellen	vi
Lijst van afkortingen en symbolen	ix
1 Inleiding	1
1.1 Computergrafiek	1
1.2 Hiërarchische spatiale datastructuren	3
1.3 Photonmapping	4
1.4 Gradient descent	5
2 Probleemstelling en aanpak	7
2.1 Zoeken naar dichtbijzijnde rechten	7
2.2 Omgaan met occlusie	9
3 Representaties van rechten	11
3.1 Projectieve geometrie	11
3.2 Projectieve geometrische algebra	13
3.3 Plückercoördinaten	14
4 Plückerbomen	17
4.1 Zoekalgoritme	17
4.2 Minimumafstandsfunctie	18
4.3 Domeinbegrenzing	19
4.4 Minimalisatie voor vaste momentvector	20
4.5 Minimalisatie over alle momentvectoren	24
4.6 Opdeling ruimte	27
4.7 Constructie	31
5 Extensies	33
5.1 Dichtste raakpunt	33
5.2 Lijnstukken	36
6 Resultaten	43
6.1 Performantie zoekopdrachten	43
6.2 Alternatieve heuristieken	45

INHOUDSOPGAVE

6.3	Invloed invoerdata	48
6.4	Alternatieve zoekopdrachten	51
6.5	Photonmapping	55
7	Besluit	61
7.1	Verder onderzoek	62
	Bibliografie	63

Samenvatting

In computer graphics betekent renderen het berekenen van afbeeldingen van virtuele scènes. Photonmapping is een renderalgoritme dat onder meer toelaat om een deel van de computationeel dure lichttransportberekeningen te hergebruiken voor meerdere afbeeldingen, bijvoorbeeld in animatie. Het bepaalt hoeveel licht invalt in een punt op een oppervlak door te zoeken naar naburige, voorberekende punten op dat oppervlak die inkomende lichtbijdragen voorstellen. De voorberekende punten worden opgeslagen in een bijhorende datastructuur, de photonmap. Deze is echter slechts bruikbaar zolang de geometrie die betrokken was bij de constructie ervan statisch blijft. Indien het photonmapping algoritme zou werken met stralen in plaats van punten, dan zou deze voorwaarde minder strikt zijn en de photonmap dus langer geldig blijven.

Concreet vereist deze aanpassing een algoritme dat, gegeven het zoekpunt \mathbf{q} op een oppervlak, de k rechten zoekt waarvan het snijpunt met het oppervlak het dichtst bij \mathbf{q} ligt. Een gerelateerd, eenvoudiger probleem is het zoeken naar de k rechten met de kortste loodrechte afstand tot het zoekpunt. Bestaande technieken hiervoor zijn beperkt tot lijnstukken, hebben een groter dan lineaire geheugencomplexiteit, of laten niet toe om op efficiënte wijze elementen toe te voegen of te verwijderen.

Deze thesis introduceert plückerbomen, een nieuwe datastructuur om te zoeken naar de k dichtstbijzijnde rechten rond een punt in \mathbb{R}^3 . Er wordt gebruik gemaakt van plückercoördinaten om rechten voor te stellen als zesdimensionale punten. Dit werk stelt vooreerst een manier voor om deze punten recursief te partitioneren. Verder wordt er een algoritme aangeboden dat efficiënt deze partities kan doorzoeken op de k rechten met kortste loodrechte afstand tot het zoekpunt. Daarna wordt een extensie voorgesteld die toelaat om de k rechten te zoeken waarvan het snijpunt met het zoekvlak het dichtst bij \mathbf{q} ligt. Ten slotte wordt er een extensie gepresenteerd die toelaat om halfrechten of lijnstukken te gebruiken.

Uit experimenten blijkt dat de voorgenoemde zoekoperaties met plückerbomen een tijdscomplexiteit vertonen rond $\mathcal{O}(n^{0.608})$, met n het aantal rechten in de boom. In de gebruikte implementatie gebeurt constructie altijd in $\mathcal{O}(n \log^2 n)$ tijd, al kan dit met verdere optimalisatie teruggebracht worden naar $\mathcal{O}(n \log n)$ tijd. Insertie en verwijdering van elementen kan steeds in $\mathcal{O}(\log n)$ tijd, en de geheugencomplexiteit is $\mathcal{O}(n)$.

Ten slotte demonstreert deze thesis het gebruik van de plückerboom in photonmapping met experimenten. Hier wordt aangetoond dat met dit algoritme de photonmap bruikbaar is na bepaalde aanpassingen van de scène waar klassieke pho-

SAMENVATTING

tonmapping zou falen. In conclusie biedt deze thesis een extensie van de levensduur van de photonmap doormiddel van een nieuwe datastructuur, waar nog ruimte voor optimalisatie is.

Lijst van figuren en tabellen

Lijst van figuren

1.1	Schematische illustratie van pathtracing.	2
1.2	Illustratie van het zoeken van de dichtste buur van punt \mathbf{q} in een 2D kd-boom.	3
1.3	Schematische illustratie van photonmapping.	5
3.1	Voorstelling van een projectieve ruimte \mathbb{RP}^2 met een punt \mathbf{s}	12
3.2	Voorstelling van een projectieve ruimte \mathbb{RP}^2 met een rechte \mathbf{l}	12
4.1	Geometrische voorstelling van domeinbegrenzing in vergelijking (4.1) . .	20
4.2	Ontbinding van h_{\perp}	21
4.3	Gevallenonderscheid van relatie tussen $ \mathbf{m} $ en $ \mathbf{q}_p $	22
4.4	Illustratie van een rechte die niet door \mathbf{q}_p gaat.	23
4.5	Heatmaps van domein minimalisatiefunctie	24
4.6	Meerdere lokale minima bij begrenzing momentdomein	26
4.7	Ongeldige combinatie van momentvector en directioneel interval	27
4.9	Gradient descent bij sferische coördinaten	30
4.10	Voorbeelden van sectoren	30
4.11	Splitsing van een richtingsinterval in twee deelintervallen	31
5.1	Verschil tussen loodrechte afstand en afstand tot raakpunt	33
5.2	Illustratie van de verschillende geometrische componenten gebruikt bij de afleiding van h_{opp} voor het zoeken van de dichtste raakpunten	34
5.3	Voorbeeld van een lijnstuk	36
5.4	Voorbeelden configuraties lijnstukken	37
5.5	Resultaat van het uitvoeren van algoritme 2 op de voorbeelden uit figuur 5.4	38
5.6	Invloed van positionering t waarden op minima	39
5.7	Invloed van teken van t waarden op minima.	40
6.1	Aantal bezochte knopen voor dichtste-buur zoekoperaties op basis van loodrechte afstand in plückerbomen met stijgende grootte gebouwd met afwisselende splitsingstypes.	45
6.2	Performantiegrafieken van alternatieve heuristieken.	46

LIJST VAN FIGUREN EN TABELLEN

6.3	Grafiek van hoe nauw takken aansluiten bij de rechten die ze omvatten. Merk op dat afwisselend splitsen sneller daalt dan de twee alternatieven.	46
6.4	Relatieve frequentie van elk splitsingstype op iedere diepte in de boom voor de verschillende heuristieken. In figuren 6.4b en 6.4c zijn iets minder splitsingen op radius te vinden.	47
6.5	Progressie van de afstand van de dichtste beschouwde rechte tot het zoekpunt naarmate de boom doorzocht wordt.	48
6.6	Performantiegrafiek van zoekopdrachten op plückerbomen van rechten met verlaagde dimensionaliteit, gebouwd met afwisselend splitsen. Deze heuristiek presteert hier minder goed, aangezien er splitsingen worden gemaakt op dimensies waarin de data geen variantie vertoont.	49
6.7	Performantiegrafiek van zoekopdrachten op plückerbomen van rechten met verlaagde dimensionaliteit, gebouwd met de maximum variantie heuristiek. Deze heuristiek presteert nog slechter dan afwisselend splitsen.	50
6.8	Performantie bij het zoeken van de buur met dichtste loodrechte afstand in scène met glazen bol. De resultaten komen grotendeels overeen met die uit sectie 6.1	51
6.9	Performantie bij het zoeken van de buur met dichtste loodrechte afstand in scène met golven. De resultaten komen overeen met die uit sectie 6.1	51
6.10	Invloed van stijgende k	52
6.11	Invloed van positionering van de oorsprong op de zoekperformantie. Het centreren van de oorsprong tussen de rechten geeft de beste performantie.	53
6.12	Performantie bij zoekoperaties uit hoofdstuk 5. Het aantal bezochte knopen bij het zoeken op dichtste raakpunt is nagenoeg identiek aan de resultaten voor dichtste loodrechte afstand. Zoeken op lijnstukken presteert iets slechter, met 0.663 en 0.657 als exponenten tegenover 0.608.	54
6.13	Invloed van de lengte van lijnstukken op de zoekperformantie. Bij kortere lijnstukken worden er minder knopen in de boom bezocht.	55
6.14	Referentieafbeelding van bolscène gemaakt met klassieke photonmapping. (~ 35 miljoen fotonen in scène, 20 fotonen in PDE)	56
6.15	Vergelijking brandpunt, zonder directe belichting, bij verschillende photonmapping technieken. (~ 128000 fotonen in scène, 20 fotonen in PDE)	56
6.16	Verschil tussen het gebruik van de loodrechte afstand en de afstand tot het dichtste raakpunt.	57
6.17	Heatmaps van het aantal bezochte knopen per pixel voor fotonen en fotonstralen.	57
6.18	Referentieafbeelding van golfscène gemaakt met klassieke photonmapping. (~ 123 miljoen fotonen in scène, 20 fotonen in PDE)	58
6.19	Vergelijking brandpunt bij verschillende photonmapping technieken. (~ 600000 fotonen in scène, 20 fotonen in PDE)	58
6.20	Vergelijking van golfscène uit figuur 6.19 met toegevoegde diffuse <i>Stanford Bunny</i> voor verschillende photonmapping technieken.	59

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

kNN	k Nearest Neighbours, oftewel zoekalgoritmes die de k dichtste buren rond een punt vinden.
PDE	Photon Density Estimation , deel van het photonmapping algoritme waar irradiantie geschat wordt op basis van de lokale densiteit van fotonen.
PGA	Projectieve Geometrische Algebra , beschreven in hoofdstuk 3

Symbolen

x	Een scalar (niet vet gedrukt)
\mathbf{v}	Een vector (vet gedrukt)
ϕ	Azimut, deel van een sferisch coördinaat
θ	Hoek gemeten van de zenit, deel van sferisch coördinaat
r	Radius, deel van sferisch coördinaat
E	Irradiantie
P	Vermogen
\mathbf{d}	Richtingsvector
\mathbf{m}	Momentvector
\mathbf{p}	Punt op rechte dat het dichtst bij de oorsprong ligt
\mathbf{q}	Zoekpunt
f_{\perp}	Minimumafstandsfunctie voor loodrechte afstand
f_{opp}	Minimumafstandsfunctie voor dichtste raakpunt op zoekvlak
f_{\sqcap}	Minimumafstandsfunctie voor lijnstukken

Hoofdstuk 1

Inleiding

In dit hoofdstuk wordt een kort overzicht gegeven van de materie waarop deze thesis verderbouwt. Hoofdstuk 2 legt vervolgens het probleem uit waarvoor een oplossing werd gezocht, legt de vereisten vast en duidt enkele alternatieve of gerelateerde technieken aan. Om een goede datastructuur te bouwen wordt een geschikte representatie van de data gekozen, waarvoor hoofdstuk 3 de wiskundige achtergrond geeft. De ontwikkelde datastructuur komt vervolgens in hoofdstuk 4 aan bod, waar ook een bijhorend zoekgoritme wordt opgebouwd. Hoofdstuk 5 formuleert enkele extensies op deze datastructuur, waarna in hoofdstuk 6 ten slotte de eigenschappen van de datastructuur en het algoritme worden geëvalueerd.

1.1 Computergrafiek

Deze thesis is gesitueerd in *computer graphics*. Dit domein van de computerwetenschappen is gespecialiseerd in het creëren, manipuleren en interpreteren van geometrische data voor visualisatie. Deze visualisatie uit zich bijvoorbeeld in beelden, zoals in visual effects, animatiefilms, medische visualisatie of video games, maar kan bijvoorbeeld ook 3D-printen betekenen. Het maken van beelden, of *renderen*, is een proces waarbij met een virtuele camera een afbeelding van een virtuele scène berekend wordt. De algoritmes die hiervoor gebruikt worden kunnen ruwweg in twee categorieën opgedeeld worden [2]. Enerzijds is er *rasterisatie*, waarbij de geometrie door middel van transformaties geprojecteerd wordt op de virtuele camerasensor. Deze techniek wordt eerder gebruikt in interactieve toepassingen waar snelheid primeert boven kwaliteit. Een tweede techniek, *ray tracing*, benadert met Monte-Carlotechnieken [17] het licht dat invalt op de virtuele camerasensor door het lichttransport door de scène te simuleren. Dit resulteert in meer realistische afbeeldingen, maar wordt meer gebruikt in bijvoorbeeld de filmindustrie omwille van de tragere performantie.

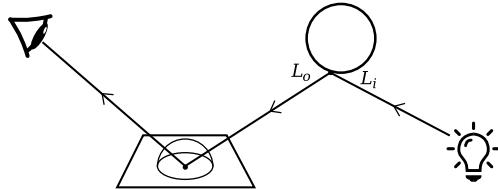
Beide technieken steunen op dezelfde theoretische achtergrond. Lichtbronnen stralen een bepaald vermogen P uit. Wanneer dit licht een object bereikt, wordt de intensiteit beschreven als vermogen per oppervlakte, of *irradiantie* E . De hoeveelheid licht die op het oppervlak invalt, of uitgestraald wordt, in een bepaalde richting wordt gemeten in vermogen per oppervlakte per ruimtehoek, of *radiantie* L [26]. De

1. INLEIDING

verhouding tussen de radiantie die invalt en vertrekt vanuit een punt wordt gegeven door de renderingvergelijking [16]

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i. \quad (1.1)$$

Om een afbeelding te berekenen wordt deze gediscretiseerd in vakjes, of *pixels*, en voor elke pixel wordt de waarde bepaald met deze vergelijking. Hierin is L_o dan de radiantie die door de pixel zichtbaar is, uitgezonden vanop punt \mathbf{x} , het punt dat door de pixel gezien wordt onder de ruimtehoek ω_o . De radiantie L_o is gelijk aan de som van de radiantie L_e die het object zelf uitstraalt, en de radiantie die door het oppervlak wordt gereflecteerd. Die gereflecteerde radiantie wordt gegeven door de integraal over alle ruimtehoeken ω_i in de hemisfeer rond punt \mathbf{x} van de radiantie L_i die vanuit ω_i het oppervlak bereikt, vermenigvuldigd met de materiaalspecifieke *bidirectional reflectance distribution function* (BRDF) f_r . Deze formule is recursief, zodat de radiantie vanuit punt \mathbf{x} afhangt van de radiantie die het ontvangt van alle andere punten, welke dan weer radiantie ontvangen van andere punten, enzovoort. Zo ontstaan er lichttransportpaden waar radiantie getransporteerd wordt vanuit een lichtbron tot de camera over punten x_1, x_2, \dots, x_n . Dit wordt geïllustreerd in figuur 1.1. Enkel de paden die een lichtbron verbinden met de camera zonder occlusie leveren een bijdrage aan de uiteindelijke belichting van de pixel. Gezien het enorm grote domein van paden dat mogelijk is, wordt L_o meestal benaderd met bijvoorbeeld *pathtracing*. Dit lichttransportalgoritme bemonstert de paden die het licht aflegt en berekent zo met Monte-Carlosimulatie een schatting van L_o . De paden worden gegenereerd door op ieder knooppunt x_i een richting te kiezen waarnaar een straal getraceerd wordt, waarvan het raakpunt met de scènegeometrie dan het volgende knooppunt x_{i+1} is.



Figuur 1.1: Schematische illustratie van pathtracing. Vanuit de camera worden paden berekend door de scène waarlangst radiantie getransporteerd wordt.

Binnen lichttransportpaden kunnen de knooppunten x_i geklasseerd worden in diffuse en speculaire knooppunten. Diffuse knooppunten hebben een materiaal dat inkomend licht weerkaatst in alle richtingen, in het ideale geval gelijkmatig over alle hoeken. Voorbeelden van objecten met een diffuus materiaal zijn mat papier, of een matte muur. Speculaire knooppunten zullen daarentegen het licht in slechts specifieke hoeken weerkaatsen, zoals bijvoorbeeld het geval is bij spiegels of glas.

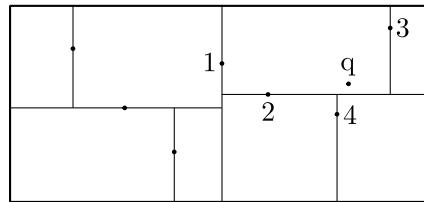
1.2 Hiërarchische spatiale datastructuren

Binnen computer graphics worden verschillende *spatiale acceleratiestructuren* gebruikt voor allerlei doeleinden. Deze structuren delen de ruimte, en de geometrie erin, op in subdomeinen waar dan eenvoudiger in gezocht kan worden. Vaak maken deze structuren gebruik van het verdeel-en-heersprincipe. Door elementen recursief te groeperen in spatiale subdomeinen, kunnen aan de hand van deze domeinen snel beslissingen gemaakt worden over grote groepen elementen. Dit is gelijkaardig aan hoe binair zoeken werkt, of zoeken in een binaire boom.

In deze thesis wordt voornamelijk voortgebouwd op kd-bomen. In deze structuur wordt een k -dimensionale ruimte recursief opgedeeld met as-gealigneerde hypervlakken [3]. De splitsing gebeurt meestal door een pivotpunt en een as te kiezen om vervolgens de elementen op te delen in de groep die op de as voor het pivotpunt ligt en een groep die erachter ligt. Het splitsingsvlak is dan een as-gealigneerd hypervlak dat door dit pivotpunt loopt. Het pivotpunt wordt een knoop in de kd-boom en het splitsingsvlak verdeelt dus de resterende elementen in twee subtakken. Bij het zoeken kan dan op basis van het domein, welke een hyperrechthoek is, beslist worden of een tak al dan niet een gezocht element bevat.

De kd-boom wordt onder meer gebruikt voor het accelereren van k -dichtste buren algoritmes (k nearest-neighbours of kNN) ¹ [19]. Hierbij worden de k punten gezocht die het dichtst bij een gegeven zoekpunt liggen volgens een bepaalde afstandsmetriek. Algoritme 1 geeft een overzicht van hoe een kNN algoritme op een kd-boom werkt, en figuur 1.2 toont een 2D voorbeeld.

Het algoritme werkt door recursief door de boom te stappen en hierbij een lijst bij te houden van de k tot nog toe dichtste punten, de kandidaten. In iedere knoop wordt eerst gecontroleerd of het punt dat in de knoop zit dichter ligt dan de verste huidige kandidaat. Indien dit het geval is, dan vervangt het punt de verste kandidaat. Vervolgens worden dan de minimumafstanden van de kindknopen hun subdomeinen tot het zoekpunt berekend. Dit kan door het zoekpunt bijvoorbeeld te projecteren op de grens van het subdomein, en vervolgens de afstand tussen het zoekpunt en de projectie te bepalen. De kindknopen worden dan door middel van recursie bekeken, van kleinste naar grootste minimumafstand, indien hun minimumafstand kleiner is dan die van de verste kandidaat.



Figuur 1.2: Illustratie van het zoeken van de dichtste buur van punt q in een 2D kd-boom. De cijfers geven aan in welke volgorde de punten bekeken worden.

¹De k in kd-boom en k -dichtste buren verwijzen naar verschillende variabelen. Het symbool wordt hergebruikt om consistent te zijn met bestaande literatuur.

1. INLEIDING

```
zoek_dichtste_buren(knoop, querypunt, kandidaten):
    max_afstand ← max({afstand(punt, querypunt) | ∀ punt ∈ kandidaten})
    if #kandidaten < k of afstand(knoop.punt, querypunt) < max_afstand
        then
            voeg knoop.punt in kandidaten
            verwijder verstuurgelegen kandidaat indien #kandidaten > k
            update max_afstand
            min_afstanden ← {min_afstand(kindknoop.subdomein, querypunt) | ∀
                kindknoop ∈ knoop}
            sorteer min_afstanden en bewaar resulterende permutatie kindknopen
    foreach kindknoop, van kleinste naar grootste minimumafstand do
        if min_afstanden[kindknoop] > max_afstand then
            break
        else
            zoek_dichtste_buren(kindknoop, querypunt, kandidaten)
            update max_afstand indien kandidaten werd aangepast
    return kandidaten
```

Algoritme 1: ZoekDichtsteBuren

1.3 Photonmapping

Zoals sectie 1.1 uitlegde, kan pathtracing gebruikt worden om afbeeldingen te renderen door lichttransportpaden te bemonsteren en zo het licht dat invalt in de camera te schatten. In bepaalde scènes presteert pathtracing echter niet goed, of resulteert het in een slechte schatting. Om dit te verhelpen werden daarom alternatieve algoritmes ontworpen. Zo beschreef H.W. Jensen in 1995 een andere techniek genaamd *photonmapping* [12], waarvan figuur 1.3 een schematisch overzicht geeft. Photonmapping is een vorm van bi-directionele ray tracing. Hierbij worden er partiële lichttransportpaden opgebouwd, zowel langs de kant van de camera als de lichtbron, welke dan vervolgens worden samengevoegd. Dit garandeert dat de resulterende paden zowel een lichtbron bereiken, alsook de camera. Het samenvoegen van de paden steunt op het feit dat in diffuse punten de inkomende radiantie in alle richtingen wordt uitgestraald. Wanneer beide partiële paden op hetzelfde diffuse punt eindigen, kunnen deze dus altijd verbonden worden, ongeacht de richtingen waaruit deze het punt bereiken.

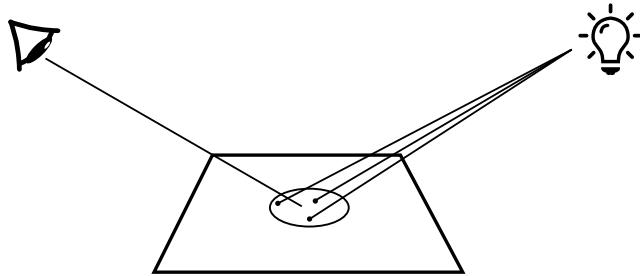
Bij photonmapping wordt in een eerste fase vanuit de lichtbron partiële paden berekend totdat deze een diffuus punt bereiken. Op deze punten wordt dan een zogenaamde *foton* opgeslagen. Deze bevat de positie van het raakpunt, de richting van de fotonstraal en een deel van het vermogen van de lichtbron. De verzameling van al deze fotonen wordt een *photonmap* genoemd. In een tweede fase worden dan vanuit de camera lichtpaden berekend totdat deze het diffuse oppervlak raken. Vervolgens worden dan op het oppervlak de k fotonen opgezocht die het dichtst bij het einde van het camerapad liggen. Met deze wordt dan een schatting gemaakt van de irradiantie op dat punt van het diffuse oppervlak. Deze schatting, ook *photon density estimation*

(PDE) genoemd, bepaalt de irradiantie door de densiteit en intensiteit van de fotonen in de buurt te meten. Om de dichtstbijzijnde fotonen te vinden wordt een kNN algoritme gebruikt, zoals werd beschreven in sectie 1.2, waarvoor de fotonen dan ook in een driedimensionale kd-boom worden opgeslagen.

In PDE wordt de assumptie gemaakt dat de k gevonden fotonen lokaal uniform verdeeld liggen in een cirkel met radius r . Door het totale vermogen van de fotonen te delen door de oppervlakte waarin ze liggen, wordt een benadering van de irradiantie bekomen. Formeel wordt deze dan gegeven door

$$E \approx \frac{\sum_i^k P_i}{\pi r^2}, \quad (1.2)$$

waarin P_i het vermogen is van de i de gevonden foton [12]. Merk op dat de irradiantie hier afhangt van zowel het vermogen van de fotonen, als de straal waarin ze liggen. Indien een groter aantal fotonen per lichtbron gebruikt wordt, dan neemt de densiteit toe en daalt dus r , maar neemt tegelijk ook het vermogen P_i per foton af. In de limiet zijn er oneindig veel fotonen aanwezig in de scène en bereikt de benadering van de irradiantie de werkelijke waarde.



Figuur 1.3: Schematische illustratie van photonmapping. Vanuit de lichtbron worden partiële paden berekend tot op een diffuus oppervlak, waar fotonen worden opgeslagen. Deze worden in een tweede fase gebruikt om de irradiantie te schatten.

De irradiantiewaarde kan vervolgens gebruikt worden in de renderingvergelijking om de uitgaande radiantie van het diffuse oppervlak te benaderen. Perfect diffuse oppervlakken stralen per definitie evenveel radiantie uit in iedere richting [26]. Ze hebben bijgevolg een constante BRDF van ρ/π , met ρ de reflectantieparameter van het diffuse materiaal. De uitgaande radiantie wordt dan gegeven door

$$L_o(\mathbf{x}, \omega_o) \approx L_e(\mathbf{x}, \omega_o) + \frac{\rho}{\pi} \frac{\sum_i^k p_i}{\pi r^2}. \quad (1.3)$$

1.4 Gradient descent

Wanneer een optimum van een functie berekend moet worden, en dit optimum niet analytisch gevonden kan worden, dan kunnen er numerieke methodes gebruikt worden. *Gradient descent* algoritmes doorzoeken iteratief het domein van de functie om dit optimum te vinden [4]. In iedere iteratie wordt naast de functiewaarde ook

1. INLEIDING

de gradiënt berekend, welke dan gebruikt wordt om te bepalen in welke richting de functie het meest stijgt of daalt, en dus het optimum ligt. Er kan dan een punt in het domein gekozen worden door een stap in deze richting te nemen, welke dan gebruikt wordt in de volgende iteratie. De belangrijkste parameters zijn hier de grootte van de stap in iedere iteratie, en de stopcondities die bepalen wanneer het optimum bereikt is. Indien de gradiëntfunctie niet vorhanden is, of deze moeilijk te berekenen is, dan is een gradientloos optimalisatiealgoritme een mogelijk alternatief. Deze klasse van algoritmes zal het optimum zoeken met enkel de functiewaarden. Dit kan bijvoorbeeld door de gradiënt numeriek te benaderen door middel van eindige differenties.

Hoofdstuk 2

Probleemstelling en aanpak

Een interessant neveneffect van het gebruik van photonmapping is dat de photonmap die gegenereerd wordt in de eerste fase hergebruikt kan worden. Zolang de punten van het partiële lichttransportpad dat leidde tot de foton en de visibiliteit tussen deze punten onveranderd blijven, kan de rest van de scène aangepast worden zonder dat de photonmap ongeldig wordt. Aangezien deze photonmap informatie bevat over de lichttransportpaden, kan dit dus ook beschouwd worden als een manier om werk uit te sparen in de lichttransportberekening van gelijkaardige opeenvolgende afbeeldingen. Dit is bijvoorbeeld interessant in animatiefilms waar afbeeldingen, ook *frames*, elkaar zeer snel opeenvolgen en de scène dus niet veel veranderd tussen twee frames.

2.1 Zoeken naar dichtbijzijnde rechten

De bruikbaarheid van de photonmap wordt beperkt door zijn geldigheid. Het is dan ook interessant om deze zo lang mogelijk geldig te houden. Dit is moeilijk, aangezien een kleine aanpassing van de lichtbron of een speculair object een groot effect kan hebben op de bijdrage van de transportpaden. Wanneer een knoop op het pad verandert, dan blijft echter het transport gelijk over alle knopen die voorgaan richting de lichtbron. Enkel het transport in de aangepaste knoop en de knopen die volgen richting de diffuse ontvanger veranderen.

Wanneer enkel de diffuse ontvanger verandert, dan zou de photonmap hersteld kunnen worden door bij alle partiële paden vanuit de voorlaatste knoop opnieuw een fotonstraal te traceren. In scènes met veel geometrie is dit echter een dure operatie met een tijdscomplexiteit van $\mathcal{O}(n \log g)$, met n het aantal fotonen en g het aantal geometrie-elementen in de scène.¹ Een alternatief idee, dat in deze thesis onderzocht wordt, is om de fotonstralen te bewaren in de photonmap in plaats van de raakpunten van de fotonstralen met diffuse oppervlakken. Deze fotonstralen zijn immers nog geldig, enkel hun raakpunt vervalt. De photonmap moet dan niet gecorrigeerd worden bij een verandering in de diffuse ontvanger. Het algoritme voor de photon density

¹Voor alle n fotonen moet een straal getraceerd worden, wat gemiddeld $\mathcal{O}(\log g)$ tijd kost in bijvoorbeeld een bounding volume hierarchy [18].

2. PROBLEEMSTELLING EN AANPAK

estimation moet echter wel aangepast worden om gebruik te maken van fotonstralen in plaats van fotonen. Dit kan, aangezien van occlusie, door kNN op de fotonrechten uit te voeren in plaats van fotonen. Het probleem wordt in feite geïnverteerd, van ‘wat is het raakpunt van deze fotonstraal?’ naar ‘welke fotonstraal raakt dit punt?’.

Het aanpassen van PDE voor gebruik van fotonstralen vereist een algoritme dat, gegeven een verzameling van fotonstralen, de k stralen vindt die resulteren in de dichtstbijzijnde fotonen. Dit zijn met andere woorden de fotonstralen waarvan het snijpunt met het oppervlak het dichtst bij het zoekpunt liggen. Een eenvoudigere variant van dit probleem kan als volgt geformuleerd worden. Gegeven een zoekpunt \mathbf{q} en een verzameling van rechten, zoek de k rechten die het dichtst bij \mathbf{q} lopen.

Een naïef algoritme is om simpelweg de afstand van alle rechten tot het zoekpunt te meten. Dit heeft een tijd- en ruimtecomplexiteit van $\mathcal{O}(n)$. De rechten worden hierbij in een lijst opgeslagen en toevoegen of verwijderen van elementen kan dan in $\mathcal{O}(1)$ tijd. Aangezien een grote densiteit van fotonen nodig is, en dus een groot aantal, is zoeken in $\mathcal{O}(n)$ tijd niet snel genoeg.

In literatuur rond computationele geometrie worden alternatieve methodes beschreven voor lijnstukken. De *edge quadtree* en *polygonal map quadtree* [22] zijn datastructuren die lijnstukken in twee dimensies opslaan, maar kunnen uitgebreid worden naar drie dimensies door er *octrees* van te maken. Deze partitioneren dan recursief de driedimensionale ruimte totdat ieder deel, of *voxel*, slechts een beperkt aantal lijnstukken bevat. In iedere voxel worden alle lijnstukken bewaard die de voxel doorkruisen. Zoeken naar de dichtste buur kan snel in $\mathcal{O}(\log n)$ tijd met een algoritme dat gelijkaardig is aan kNN in kd-bomen. Deze algoritmes hebben echter enkele nadelen. Vooreerst ondersteunen ze enkel lijnstukken, geen halfrechten of rechten. Verder worden lijnstukken op meerdere plaatsen in de datastructuur opgeslagen, en het geheugengebruik is dus groter dan lineair. De combinatie van asgealigneerde voxels en georiënteerde lijnstukken maakt dat de boomstructuur soms erg veel knopen moet hebben zodoende er genoeg resolutie is om de lijnstukken op ieder punt te onderscheiden. Ten slotte maakt de sterke duplicatie van elementen in het geheugen het ook moeilijk om efficiënt elementen toe te voegen en te verwijderen. Hiervoor moeten immers alle knopen bezocht worden die het lijnstuk snijden, en is het mogelijk dat er vele knopen toegevoegd of verwijderd moeten worden. Een ideale datastructuur biedt snelle insertie, verwijdering en zoekoperaties aan, en heeft een geheugengcomplexiteit van $\mathcal{O}(n)$.

Er zijn ook een aantal alternatieve strategieën mogelijk om de photonmap langer te kunnen gebruiken. Een eenvoudigere, nog te onderzoeken strategie zou kunnen zijn om bij verplaatsing van het diffuse oppervlak alle fotonen te projecteren op het oppervlak. Het resultaat is dan niet exact correct, maar de fout zou bij een kleine perturbatie van het oppervlak visueel imperceptibel kunnen zijn. Verder zijn er in de literatuur ook andere gerelateerde technieken te vinden zoals *photon splatting* [11], of benaderingen van de photonmap met *spherical harmonics* [8]. Deze vereisen echter dat alle fotonen beschouwd worden, of zijn benaderend. Verder is de ontwikkeling van een datastructuur voor het opzoeken van de dichtste rechten bij een punt ook een interessant probleem op zich. Een goede datastructuur zou toepassing kunnen vinden in allerlei domeinen die werken met geometrische data. Hier wordt enkel

photonmapping bekijken, maar ook in bijvoorbeeld *ray marching* [10] zou het gebruikt kunnen worden om de afstand van een punt tot de dichtste geometrie te vinden.

2.2 Omgaan met occlusie

In klassieke photonmapping geeft een foton ook impliciet informatie over occlusie, welke zich hier manifesteert als schaduw. Immers, een foton wordt enkel opgeslagen op het eerste raakpunt van de fotonstraal met de scène. Bij volgende intersecties van de fotonstraal wordt geen foton opgeslagen, en deze ontvangen dus geen belichting van de fotonstraal. Wanneer enkel de straal opgeslagen wordt, is deze informatie niet beschikbaar. Dit maakt de photonmap breder bruikbaar, aangezien een verandering van de occlusie van de diffuse ontvanger de photonmap niet ongeldig maakt. De keerzijde is dat deze occlusie dan alsnog berekend moet worden in de PDE. Een volledig correcte oplossing moet dus occlusie in rekening nemen en bij ieder punt de fotonstralen geven die het oppervlak dichtbij intersecteren maar ook niet eerder al een object sneden.

Er zijn een aantal, al dan niet benaderende, oplossingen voor occlusie in dit probleem. Er wordt hier de assumptie gemaakt dat er geen occlusie optreed tussen de lichtbron en de eerste speculaire knoop, of tussen de speculaire knopen onderling. Occlusieberekeningen zijn niet de focus van dit onderzoek, maar in wat volgt worden enkele oplossingen voorgesteld die onderzocht zouden kunnen worden.

Een eerste oplossing is om bij het zoeken de kandidaat fotonstralen te controleren door ze te traceren door de scène. Indien de straal dan iets raakt voordat deze de beschouwde diffuse ontvanger bereikt, dan wordt deze gediskwalificeerd. Er worden dan altijd de k dichtste stralen gevonden die ook zichtbaar zijn op het zoekpunt. Hiervoor moeten echter alsnog stralen getraceerd worden door de scène. Dit verhoogt ook het aantal stralen dat bekeken moet worden in de datastructuur voor punten die in schaduw liggen, aangezien er geen rekening is gehouden met occlusie in de constructie van de datastructuur. Mogelijks traceert deze methode minder stralen dan wanneer alle fotonen zouden geüpdatet worden door hun bijhorende stralen te hertraceren. Immers test deze methode enkel de stralen die mogelijk zichtbaar in een pixel. Verder kan voor occlusietesten een vereenvoudigde versie van de scène gebruikt worden, wat de kost van het zoeken naar intersectie verlaagt [23].

Een tweede oplossing, die in deze thesis kort onderzocht wordt, is om eerst de k dichtste rechten te zoeken, en dan de occlusie te controleren. In de PDE berekening van vergelijking (1.2) worden dan enkel de $m \leq k$ fotonstralen gebruikt die zichtbaar zijn in het punt. Zo hoeven er geen extra knopen in de boom bezocht worden. Het nadeel hiervan is dat in sommige scenario's het aantal gebruikte fotonen m erg laag kan komen te liggen, wat zichtbare artefacten geeft. Dit zal gedemonstreerd worden in het resultatenhoofdstuk, in figuur 6.20.

De vorige oplossing vereist nog steeds dat de occlusie van de k fotonstralen wordt gecontroleerd. Indien k hoog ligt, kan het een optie zijn om een stochastische benadering te maken van de PDE vergelijking (1.2) met Monte-Carlomethodes [17].

2. PROBLEEMSTELLING EN AANPAK

Dit kan door eerst vergelijking (1.2) te herformuleren als

$$\begin{aligned} E &\approx \frac{\sum_{i=1}^k P_i V_i}{\pi r^2} \\ r &= \max_{i \in [1, k]} (|\mathbf{s}_i - \mathbf{q}| V_i), \end{aligned} \quad (2.1)$$

waarin \mathbf{s}_i het snijpunt is van fotonstraal i met het oppervlak. V_i geeft de visibiliteit van de fotonstraal i , en heeft waarde 1 indien deze zichtbaar is in zoekpunt \mathbf{q} , en anders 0. Deze expressie kan vervolgens benaderd worden door m willekeurige fotonstralen te kiezen uit de k gevonden stralen, om dan vervolgens E te schatten met

$$\begin{aligned} E &\approx \frac{(k/m) \sum_{i=1}^m P_i V_i}{\pi r^2} \\ r &\approx \frac{m+1}{m} \max_{i \in [1, m]} (|\mathbf{s}_i - \mathbf{q}| V_i), \end{aligned} \quad (2.2)$$

waar r een schatting is van het maximum van de afstand van de zichtbare fotonstralen tot \mathbf{q} [14]. Deze techniek wordt ook gedemonstreerd in figuur 6.20.

Een derde oplossing benadert de occlusie met de dieptebuffer die vele toepassingen berekenen [2]. De dieptebuffer is een afbeelding waarin iedere pixel de afstand bevat tot het eerste raakpunt van de straal door de pixel. Door de fotonstraal te projecteren op deze afbeelding wordt een lijn van dieptewaarden bekomen. Voor ieder van deze pixels kan de diepte d_{foton} berekend worden waarop de fotonstraal ligt en deze waarde kan dan vergeleken worden met waarde d_{scene} uit de dieptebuffer. Indien het zoekpunt en het startpunt van de straal beide in de dieptebuffer zichtbaar zijn, en voor alle pixels ertussen geldt dat $d_{foton} < d_{scene}$, dan is er zeker geen occlusie. Dit kan verder benaderd worden door niet de volledige lijn te testen, maar enkele punten op de lijn te kiezen. Dit is gelijkaardig aan de occlusietest voor directe belichting in [20]. Indien een gelijkaardige buffer beschikbaar is voor de lichtbron, zoals een schaduwbuffer, kan een gelijkaardige methode gebruikt worden om occlusie te testen tussen de lichtbron en de eerste speculaire knoop van het pad.

Een vierde optie is om de occlusie te modelleren door de fotonstralen op te slaan als lijnstukken in plaats van halfrechten. De lijnstukken worden dan begrensd door het startpunt van de straal, en het eerste raakpunt van de fotonstraal. Indien er meerdere diffuse ontvangers achter elkaar liggen, dan intersecteren de lijnstukken zo enkel het eerste oppervlak, en niet diegene die erachter liggen. Dit geeft een geldig resultaat zolang de eerste ontvanger niet verder weg beweegt van het startpunt, en er geen andere ontvangers op het lijnstuk komen te liggen.

Ten slotte kan het in bepaalde toepassingen of scènes een optie zijn om de occlusie te negeren, bijvoorbeeld wanneer het effect van de occlusie imperceptibel is.

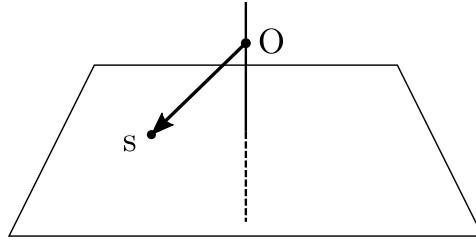
Hoofdstuk 3

Representaties van rechten

Om een algoritme te bouwen dat efficiënt kan zoeken in een verzameling van rechten, is het belangrijk om eerst een goede representatie te kiezen voor deze rechten. Een eenvoudige voorstelling is als een paar van punten waar de rechte doorloopt, of als een richtingsvector samen met een punt op de rechte. Het is echter niet eenvoudig om in deze representaties de rechten efficiënt te groeperen op basis van hun minimumafstand tot een zoekpunt. Deze vormen zijn extensies van de klassieke lineaire algebra, welke ruimtes definieert die bestaan uit vectoren. Geometrie zoals punten, rechten of vlakken worden beschreven als combinaties van vectoren, maar zijn geen deel van de algebra zelf. Er bestaat echter een algebra waarin dit wel het geval is. *Projectieve geometrische algebra* (PGA) laat toe om rechten voor te stellen als een element van de algebra, net zoals vectoren deel zijn van klassieke vectorruimtes. Deze voorstelling zal in hoofdstuk 4 gebruikt worden om een datastructuur te definiëren. De volgende secties geven een korte introductie en intuïtie voor projectieve geometrie, alsook de bijhorende algebra. De rest van dit hoofdstuk is een samenvatting van de relevante stukken uit [9], [6] en [13].

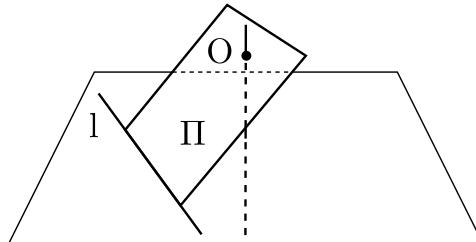
3.1 Projectieve geometrie

Projectieve meetkunde is een deel van de wiskunde die projectieve ruimtes bestudeert. In deze ruimtes ontstaat geometrie door projectie van hogerdimensionale objecten op lagere dimensies. Dit principe kan eenvoudig gedemonstreerd worden met een voorbeeld. Figuur 3.1 toont een *projectieve ruimte* \mathbb{RP}^2 waarin een tweedimensionaal projectief vlak ligt. Loodrecht op dit vlak staat de projectieve as. De oorsprong, die niet in het projectief vlak ligt, is aangegeven met **O**. Vanuit de oorsprong loopt een vector **v** die het projectief vlak snijdt. Dit snijpunt bepaalt de positie van het punt **s**. De driedimensionale vector **v** definieert hier dus het tweedimensionale punt **s**.



Figuur 3.1: Voorstelling van een projectieve ruimte $\mathbb{R}P^2$ met een punt s geproduceerd door intersectie van vector v met het projectief vlak.

Figuur 3.2 toont dezelfde opstelling, maar met een vlak Π door de oorsprong in plaats van een vector. De intersectie van Π met het projectief vlak resulteert in een rechte. Aangezien de projectieve ruimte hier slechts drie dimensies heeft, is een rechte hier het element met de hoogst-mogelijke dimensie. Het is echter mogelijk om een gelijkaardige constructie te maken voor ruimtes met een arbitrair aantal dimensies. Zo kan dus ook een vierdimensionale projectieve ruimte $\mathbb{R}P^3$ opgesteld worden die driedimensionale geometrie voorbrengt. Een vector geeft dan een punt, een tweedimensionale deelruimte resulteert in een rechte, en een driedimensionale deelruimte produceert een vlak. Algemeen geldt voor $\mathbb{R}P^n$ dat deelruimtes van graad k gebruikt worden om geometrie van graad $k - 1$ voor te stellen.



Figuur 3.2: Voorstelling van een projectieve ruimte $\mathbb{R}P^2$ met een rechte l geproduceerd door intersectie van vlak Π met het projectief vlak.

Aangezien een vlak door de oorsprong volledig gedefinieerd wordt door zijn normaalvector, kunnen zowel tweedimensionale punten als rechten volledig beschreven worden door een projectieve vector. Dit is een gevolg van *Poincaré dualiteit*. Voor een projectieve ruimte $\mathbb{R}P^n$ die geometrie uit \mathbb{R}^{n-1} voorstelt, kan een isomorfe *duale ruimte* opgesteld worden. In de duale ruimte worden deelruimtes van graad k gebruikt om geometrie van graad $n - k$ voor te stellen. Zo stellen in de duale ruimte van $\mathbb{R}P^2$ deelruimten van graad 1 dan rechten voor, en deelruimten van graad 2 punten. In de duale ruimte van $\mathbb{R}P^3$ stellen deelruimten van graad 1 vlakken voor, en deelruimten van graad 3 punten. Merk op dat in 3D rechten dus altijd voorgesteld worden door deelruimten van graad 2, ook in de duale ruimte, aangezien zowel $k - 1 = 2 - 1 = 1$ als $n - k = 3 - 2 = 1$. Met andere woorden zijn rechten in 2D dual aan punten, terwijl rechten in 3D dual zijn aan zichzelf. Dit duidt mogelijk aan waarom het zoeken van de dichtstbijzijnde rechten een eenvoudiger probleem is in twee dimensies

dan in drie dimensies.

Een andere interessante eigenschap van de projectieve ruimte is dat deze meer punten bevat dan de corresponderende lagerdimensionale Euclidische ruimte, en dus ook meer kan voorstellen. Zo is bijvoorbeeld geometrie die oneindig ver ligt deel van de ruimte. In het tweedimensionale voorbeeld van figuur 3.1 kan zo een punt op oneindig worden voorgesteld. Een intuïtie hiervoor is dat, naarmate de projectieve vector wegroteert van het projectieve vlak, het bijhorende punt steeds verder weg ligt van de oorsprong. Uiteindelijk ligt de vector parallel aan het vlak, is er geen intersectie meer met het projectief vlak, en ligt het bijhorend punt op oneindig. Deze eigenschap laat onder meer toe dat de intersectie tussen eender welke twee rechten berekend kan worden, zelfs als deze parallel liggen.

3.2 Projectieve geometrische algebra

Sectie 3.1 introduceerde hoe uit projectie geometrie kan ontstaan. Om deze geometrie te beschrijven, te meten of op te opereren is echter een bijhorende algebra nodig. Hiervoor wordt gebruik gemaakt van de projectieve geometrische algebra (PGA), een toepassing van de geometrische algebra. Breed gezien bestaat een algebra uit een verzameling van symbolen samen met operaties op deze symbolen. De PGA is een rijke algebra en biedt krachtige operatoren aan, zoals het exterieur product \wedge dat onder meer toelaat om geometrie te verbinden of om de intersectie tussen geometrie te berekenen met een simpele productoperatie. De operatoren zijn echter niet vereist voor deze thesis en ze uitleggen zou dan ook te diep gaan. Hieronder wordt daarom vooral besproken hoe PGA de verschillende geometrische elementen symbolisch beschrijft.

De PGA is in feite een verzamelnaam van verschillende algebras die gebouwd kunnen worden door een signatuur te kiezen. Een vector in \mathbb{R}^n is een lineaire combinatie van basisvectoren. Zo kan een vector $\mathbf{v} \in \mathbb{R}^2$ bijvoorbeeld beschreven worden als $\mathbf{v} = ax + by$. Analoog kunnen elementen uit PGA ontbonden worden volgens de basis $\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_n$. In PGA wordt echter ook het product van deze basiselementen gedefinieerd. Het kwadraat van ieder basiselement \mathbf{e}_i wordt per axioma vastgelegd op 1, -1 of 0. De signatuur van een PGA specificeert hoeveel basiselementen van elke soort er gebruikt worden, en wordt ook genoteerd als $\mathbf{P}(\mathbb{R}_{\#pos,\#neg,\#nul})$. In deze thesis wordt enkel gebruik gemaakt van $\mathbf{P}(\mathbb{R}_{3,0,1})$, welke dus drie positieve en één nul basiselement heeft. Deze basiselementen kunnen intuïef geïnterpreteerd worden als de \mathbf{x}, \mathbf{y} en \mathbf{z} assen, met als vierde de projectieve as. Per conventie wordt het nulelement \mathbf{e}_0 genoemd, en de drie positieve assen $\mathbf{e}_1, \mathbf{e}_2$ en \mathbf{e}_3 . Door voor de projectieve as een nulelement te kiezen, kan met een combinatie van producten een translatie bepaald worden. Dit is nauw gerelateerd aan hoe homogene coördinaten werken. De $\mathbf{P}(\mathbb{R}_{3,0,1})$ algebra heeft alle ingrediënten die nodig zijn in dit onderzoek om driedimensionale euclidische geometrie te modelleren.

Hierboven werd reeds vermeld dat het product van basiselementen \mathbf{e}_i gedefinieerd is, en dat het kwadraat axiomatisch vastligt. Er kunnen echter ook twee verschillende basiselementen $\mathbf{e}_i, \mathbf{e}_j$ vermenigvuldigd worden, en dit product vormt een zogenaamde

3. REPRESENTATIES VAN RECHTEN

2-blade $\mathbf{e}_i\mathbf{e}_j$ of kortweg \mathbf{e}_{ij} . Net zoals een element van \mathbb{R}^2 dat ontbindt over de \mathbf{x} en \mathbf{y} assen een vector genoemd wordt, zo is een element van PGA met een 2-blade een *bivector*. Een bivector kan geïnterpreteerd worden als een georiënteerde oppervlakte die opgespannen wordt langst de assen \mathbf{e}_i en \mathbf{e}_j . Het is georiënteerd, aangezien het product anti-commutatief is, en dus $\mathbf{e}_{ij} = -\mathbf{e}_{ji}$. Wanneer de 2-blade vervolgens vermenigvuldigd wordt met een derde basiselement \mathbf{e}_k , $k \neq i, k \neq j$, dan ontstaat een 3-blade \mathbf{e}_{ijk} . Een element met een 3-blade wordt een *trivector* genoemd en kan dan gezien worden als een georiënteerd volume. Analoog kan nog een *quadvector* gevormd worden in $\mathbf{P}(\mathbb{R}_{3,0,1})$, waarna alle basiselementen opgebruikt zijn. In andere algebras met meer elementen kan dit echter dus nog verder groeien. Figuur 3.1 toont een tabel met alle mogelijke producten in $\mathbf{P}(\mathbb{R}_{3,0,1})$. Het is belangrijk om op te merken dat deze n -vectors evenwaardig zijn in de algebra aan de vector. Ieder element uit de algebra kan dus ontbonden worden als een lineaire combinatie van blades.

1	e_0	e_1	e_2	e_3	e_{01}	e_{02}	e_{03}	e_{12}	e_{31}	e_{23}	e_{021}	e_{013}	e_{032}	e_{123}	e_{0123}
e_0	0	e_{01}	e_{02}	e_{03}	0	0	0	$-e_{021}$	$-e_{013}$	$-e_{032}$	0	0	0	e_{0123}	0
e_1	$-e_{01}$	1	e_{12}	$-e_{31}$	$-e_0$	e_{021}	$-e_{013}$	e_2	$-e_3$	e_{123}	e_{02}	$-e_{03}$	e_{0123}	e_{23}	e_{032}
e_2	$-e_{02}$	$-e_{12}$	1	e_{23}	$-e_{021}$	$-e_0$	e_{032}	$-e_1$	e_{123}	e_3	$-e_{01}$	e_{0123}	e_{03}	e_{31}	e_{013}
e_3	$-e_{03}$	e_{31}	$-e_{23}$	1	e_{013}	$-e_{032}$	$-e_0$	e_{123}	e_1	$-e_2$	e_{0123}	e_{01}	$-e_{02}$	e_{12}	e_{021}
e_{01}	0	e_0	$-e_{021}$	e_{013}	0	0	0	e_{02}	$-e_{03}$	e_{0123}	0	0	0	$-e_{032}$	0
e_{02}	0	e_{021}	e_0	$-e_{032}$	0	0	0	$-e_{01}$	e_{0123}	e_{03}	0	0	0	$-e_{013}$	0
e_{03}	0	$-e_{013}$	e_{032}	e_0	0	0	0	e_{0123}	e_{01}	$-e_{02}$	0	0	0	$-e_{021}$	0
e_{12}	$-e_{021}$	$-e_2$	e_1	e_{123}	$-e_{02}$	e_{01}	e_{0123}	-1	e_{23}	$-e_{31}$	e_0	e_{032}	$-e_{013}$	$-e_3$	$-e_{03}$
e_{31}	$-e_{013}$	e_3	e_{123}	$-e_1$	e_{03}	e_{0123}	$-e_{01}$	$-e_{23}$	-1	e_{12}	$-e_{032}$	e_0	e_{021}	$-e_2$	$-e_{02}$
e_{23}	$-e_{032}$	e_{123}	$-e_3$	e_2	e_{0123}	$-e_{03}$	e_{02}	e_{31}	$-e_{12}$	-1	e_{013}	$-e_{021}$	e_0	$-e_1$	$-e_{01}$
e_{021}	0	e_{02}	$-e_{01}$	$-e_{0123}$	0	0	0	e_0	e_{032}	$-e_{013}$	0	0	0	e_{03}	0
e_{013}	0	$-e_{03}$	$-e_{0123}$	e_{01}	0	0	0	$-e_{032}$	e_0	e_{021}	0	0	0	e_{02}	0
e_{032}	0	$-e_{0123}$	e_{03}	$-e_{02}$	0	0	0	e_{013}	$-e_{021}$	e_0	0	0	0	e_{01}	0
e_{123}	$-e_{0123}$	e_{23}	e_{31}	e_{12}	e_{032}	e_{013}	e_{021}	$-e_3$	$-e_2$	$-e_1$	$-e_{03}$	$-e_{02}$	$-e_{01}$	-1	e_0
e_{0123}	0	$-e_{032}$	$-e_{013}$	$-e_{021}$	0	0	0	$-e_{03}$	$-e_{02}$	$-e_{01}$	0	0	0	$-e_0$	0

Tabel 3.1: Tabel met alle mogelijke vermenigvuldigingen van basiselementen uit $\mathbf{P}(\mathbb{R}_{3,0,1})$ [5].

3.3 Plückercoördinaten

In sectie 3.1 werd uitgelegd dat deelruimtes van graad k in de projectieve ruimte gebruikt worden om geometrie van graad $k-1$ voor te stellen. Deze deelruimtes kunnen voorgesteld worden door de k -vectors uit sectie 3.2. Zo kan bijvoorbeeld een tweedimensionale deelruimte, wat na projectie een rechte geeft, voorgesteld worden door een bivector. Meer formeel is de relatie tussen bivectoren \mathbf{v} en rechten

$$\mathbf{v} = \sum a_{ij} \mathbf{e}_{ij} \text{ is een rechte} \iff a_{01}a_{23} + a_{02}a_{31} + a_{03}a_{12} = 0. \quad (3.1)$$

De coördinaten van deze elementen die hier aan voldoen worden ook *plückercoördinaten* genoemd.

Plückercoördinaten stellen dus rechten voor op een basis van 2-blades, namelijk $\{\mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}\}$. Aangezien ze een zestupel zijn, kunnen ze ook geïnterpreteerd worden als zesdimensionale punten in de projectieve ruimte P^5 ,

waar ze de plückerkwadriek vormen. De meest intuïtieve vorm is te vinden in een geometrische interpretatie. Zo kunnen ze gezien worden als een paar van \mathbb{R}^3 vectoren (\mathbf{d}, \mathbf{m}). Hierin geeft \mathbf{d} de richting van de rechte aan, en is \mathbf{m} een momentvector. Uit vergelijking (3.1) kan verder ook afgeleid worden dat het scalair product tussen deze twee nul moet zijn, en bijgevolg staan deze loodrecht op elkaar. Deze geometrische interpretatie zal nuttig blijken om in hoofdstuk 4 een datastructuur op te bouwen.

Hoofdstuk 4

Plückerbomen

Een plückerboom is een hiërarchische spatiale datastructuur op basis van plückercoördinaten. Zoals een kd-boom een hiërarchie van punten vormt volgens hun carthesische coördinaten[3], zo organiseert een plückerboom rechten volgens hun plückercoördinaten zoals besproken in hoofdstuk 3. Iedere knoop in de boom bevat één rechte en partitioneert de resterende verzameling van rechten in twee deelverzamelingen voor zijn kindknopen. Voor iedere deelverzameling kan dan een omvattend zes-dimensionaal deeldomein in P^5 bepaald worden. Verdeel en heers-zoekalgoritmen bekijken deze domeinen en beslissen zo of een tak in de boom de gezochte rechten kan bevatten. De rest van dit hoofdstuk gaat verder in op de verschillende aspecten van het bouwen en zoeken in plückerbomen.

Plückerbomen gebruiken in feite slechts een deelverzameling van de plückercoördinaten. De afbeelding van een rechte naar een corresponderend plückercoördinaat is namelijk geen injectie. Alle coördinaten $\{(c\mathbf{d}, c\mathbf{m})|c \in \mathbb{R} \setminus \{0\}\}$ beschrijven dezelfde rechte. De rest van dit document gebruikt enkel de plückercoördinaten waarin \mathbf{d} een eenheidsvector is. Een rechte komt dan nog overeen met twee coördinaten, (\mathbf{d}, \mathbf{m}) en $(-\mathbf{d}, -\mathbf{m})$. Deze redundantie zal echter nuttig blijken om naast richting ook een zin van de rechte uit te drukken.

4.1 Zoekalgoritme

In sectie 1.2 werd beschreven hoe kd-bomen gebruikt kunnen worden om de k dichtste buren van een punt te vinden. Dezelfde principes kunnen gebruikt worden bij plückerbomen om de k dichtstbijzijnde rechten rond een punt te zoeken. Een naïef algoritme kan dit doen in een lineaire tijdscomplexiteit, maar dit volstaat niet voor grote datasets.

Gegeven een verzameling van plückercoördinaten en een zoekpunt \mathbf{q} is het doel om zo snel en zoveel mogelijk rechten uit te sluiten. Dit kan door op voorhand de rechten in een hiërarchische datastructuur te steken zodat er snel beslissingen kunnen gemaakt worden over de afstand van grote groepen van rechten, analoog aan hoe een kd-boom verzamelingen van punten bevat. Het bouwen van deze datastructuur komt er op neer dat de verzameling van rechten gepartitioneerd wordt in twee partities

zodat rechten die geometrisch gelijkaardig zijn gegroepeerd worden. De plückerboom ontstaat dan wanneer dit proces recursief uitgevoerd wordt. In sectie 4.6 wordt dieper ingegaan op hoe deze partitionering gekozen wordt. Iedere knoop in de boom heeft een bijhorend zes-dimensionaal deeldomein in P^5 dat alle plückerpunten van die tak bevat, net zoals iedere tak in een kd-boom een geassocieerd domein heeft dat alle euclidische punten erin omvat.

Zoeken naar de k dichtste rechten rondom een zoekpunt \mathbf{q} werkt dan door de plückerboom te doorlopen. Hierbij wordt een lijst van kandidaten bijgehouden, dit zijn de rechten die tot nu toe het dichtst bij \mathbf{q} liggen. Bij het bezoeken van een knoop wordt eerst de rechte in de knoop vergeleken met de kandidaten. Indien de rechte dichterbij ligt, dan vervangt deze de verste kandidaat. Vervolgens worden dan de deeldomeinen van de twee kindknopen bekeken. Dit domein geeft informatie over welke rechten er in de tak kunnen zitten, en op basis daarvan kan de kleinste mogelijke afstand tot \mathbf{q} bepaald worden. Dit gebeurt met een *minimumafstandsfunctie*, welke verder beschreven wordt in 4.2. Indien één van de kindknopen een minimumafstand heeft die groter is dan de afstand tot de verste kandidaat, dan kan deze tak onmogelijk rechten bevatten die dichter liggen en kan deze dus genegeerd worden. Als er geen knoop geschrapt kan worden, dan wordt eerst de tak met de kleinste minimumafstand bezocht. Het is mogelijk dat daarin een rechte met korte afstand tot \mathbf{q} zit, zodat alsnog de andere tak weggesnoeid kan worden. Dit proces wordt recursief herhaald totdat alle knopen in de boom bezocht of weggesnoeid zijn. Op dat moment zal de lijst met kandidaten de k dichtste rechten bij \mathbf{q} bevatten.

4.2 Minimumafstandsfunctie

Zoeken in een boom is efficiënt omdat het zoekalgoritme selectief bepaalde takken overslaat. Hoe meer takken er zo weggesneden kunnen worden, hoe meer werk er bespaard wordt en bijgevolg is de beslissing om takken al dan niet weg te snijden dus essentieel. Bij kNN houdt het algoritme een kandidatenlijst bij van de k bekende elementen die tot dusver het dichtst lagen. Een tak wordt genegeerd indien het enkel elementen bevat die verder liggen dan de verste kandidaat. Dit kan beslist worden aan de hand van het deeldomein van de tak, bijvoorbeeld door de kortste afstand van het zoekpunt tot de rand van het domein te meten [19]. Deze beslissing kan dus gemaakt worden zonder dat alle elementen in de tak beschouwd worden. Concreet heeft het algoritme dus een minimumafstandsfunctie nodig. Deze berekent de kleinste mogelijke afstand van elementen in een gegeven domein tot een bepaald zoekpunt.

Opstellen van een minimumafstandsfunctie voor plückerbomen is complexer dan voor conventionele kd-bomen. Klassieke kd-bomen bevatten veeleer standaard euclidische punten. Algoritmes die in deze bomen de dichtste buren zoeken vergelijken datapunten met een zoekpunt uit eenzelfde ruimte. Bij een minimumafstandsfunctie voor klassieke kd-bomen zal het domein en het zoekpunt zich dus ook in dezelfde ruimte bevinden. Hier wordt dan bijvoorbeeld de kortste Euclidische afstand gebruikt. Daarentegen wordt er bij plückerbomen een zes-dimensionaal domein uit P^5 vergeleken met een punt uit \mathbb{R}^3 . De volgende secties tonen hoe zo een functie

opgesteld kan worden.

De keuze van minimumafstandsfunctie hangt vooraleerst af van welke afstand er gemeten wordt. In kNN algoritmes op klassieke kd-bomen worden hiervoor verschillende opties gebruikt. Veelvoorkomende keuzes zijn hier Manhattan-afstand, Euclidische afstand en Chebyshev afstand [1]. Deze keuze hangt voornamelijk af van wat er precies gemeten moet worden. In een algoritme dat met kNN de k meest gelijkaardige woorden zoekt, zou bijvoorbeeld de Hammingafstand gebruikt kunnen worden. In wat volgt zal eerst g_{\perp} , de kortste loodrechte afstand tussen punt en rechte, beschouwd worden. Hierbij zal dan f_{\perp} de minimumafstandsfunctie zijn die g_{\perp} minimaliseert over een P^5 domein. Dit is echter niet correct voor gebruik in photon density estimation, waar gezocht wordt naar de k rechten die het oppervlak het dichtst snijden rond een zoekpunt. Rechten die bijvoorbeeld dicht bij het zoekpunt liggen, maar parallel lopen aan het oppervlak, zullen hier foutief als dichtbij beschouwd worden. De loodrechte afstand is echter een eenvoudiger basisgeval waarop verder kan gebouwd worden. Secties 5.1 en 5.2 gaan hier verder op in en formuleren uitbreidingen die een oplossing bieden voor het gebruik in photonmapping.

4.3 Domeinbegrenzing

In sectie 4.2 werd het belang van een goede minimumafstandsfunctie f_{\perp} aangeduid. Voordat de berekening van f_{\perp} uitgewerkt wordt, moeten eerst de invoerparameters bepaald worden. Sectie 4.2 vermeldde ook reeds dat f_{\perp} de minimumafstand van de rechten in een P^5 domein tot een \mathbb{R}^3 zoekpunt berekent. Een goede specificatie van dit domein is belangrijk voor de numerieke stabiliteit, eenvoudige splitsing in de boom en de computationele efficiëntie. De definitie die werd gekozen voor dit werk is als volgt.

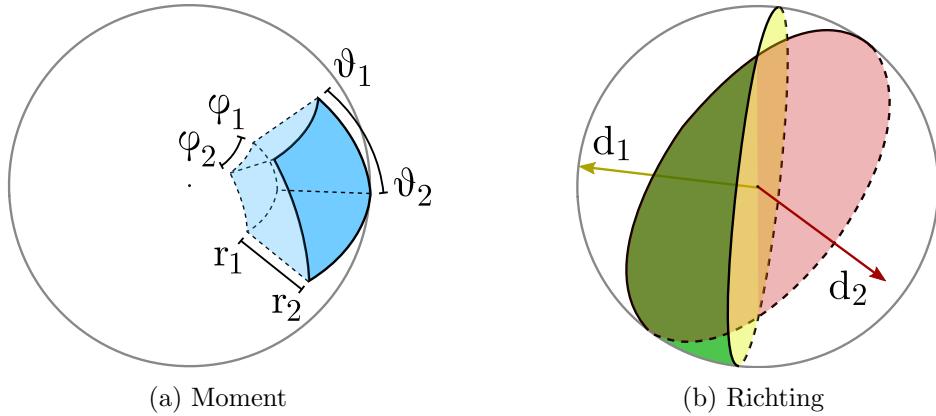
Een domein $V \in P^5$ wordt afgebakend door twee vlakken met normaalvectoren $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{R}^3$ en twee sferische momentvectoren $\mathbf{m}_1 = (r_1, \phi_1, \theta_1), \mathbf{m}_2 = (r_2, \phi_2, \theta_2) \in \mathbb{R}^3$.

Voor een plückercoördinaat (\mathbf{d}, \mathbf{m}) geldt dat

$$\begin{aligned} (\mathbf{d}, \mathbf{m}) \in V(\mathbf{d}_1, \mathbf{d}_2, \mathbf{m}_1, \mathbf{m}_2) \iff & \mathbf{d} \cdot \mathbf{d}_1 \geq 0 \wedge \mathbf{d} \cdot \mathbf{d}_2 \geq 0 \wedge \\ & r_1 \leq m_r \geq r_2 \wedge \\ & \phi_1 \leq m_{\phi} \geq \phi_2 \wedge \\ & \theta_1 \leq m_{\theta} \geq \theta_2. \end{aligned} \tag{4.1}$$

Hierin zijn m_r, m_{ϕ} en m_{θ} de coördinaten van de sferische representatie van \mathbf{m} . Dit onderscheid is belangrijk aangezien in plückercoördinaten enkel de carthesische waarden geldig zijn, maar de onderverdeling van deze ruimte dus in sferische coördinaten gebeurt. Verdere verantwoording voor deze keuzes wordt in latere paragrafen gegeven.

De geometrische interpretatie van deze grenzen is weergegeven in figuur 4.1. Links wordt de ruimte van de momentvectoren opgedeeld in sferische balken, rechts vormen de richtingsvectoren lunes.



Figuur 4.1: Geometrische voorstelling van domeinbegrenzing in vergelijking (4.1). Links: begrenzing van momentvectoren. Rechts: begrenzing van richtingsvectoren.

4.4 Minimalisatie voor vaste momentvector

De gezochte minimumafstandsfunctie f_{\perp} kan nu formeel beschreven worden. Gegeven domein $V(\mathbf{d}_1, \mathbf{d}_2, \mathbf{m}_1, \mathbf{m}_2)$ is de minimumafstand tot punt $\mathbf{q} \in \mathbb{R}^3$ gedefinieerd als

$$\begin{aligned} f_{\perp}(\mathbf{m}_1, \mathbf{m}_2, \mathbf{d}_1, \mathbf{d}_2, \mathbf{q}) &= \min_{\substack{\mathbf{m} \in [\mathbf{m}_1, \mathbf{m}_2] \\ \mathbf{d} \cdot \mathbf{d}_1 \geq 0 \wedge \mathbf{d} \cdot \mathbf{d}_2 \geq 0}} g_{\perp}(\mathbf{m}, \mathbf{d}, \mathbf{q}) \\ &= \min_{\mathbf{m} \in [\mathbf{m}_1, \mathbf{m}_2]} h_{\perp}(\mathbf{m}, \mathbf{q}), \end{aligned} \quad (4.2)$$

waarin h_{\perp} de minimumafstand voor vaste \mathbf{m} en een variabele \mathbf{d} bepaalt. Deze sectie toont hoe h_{\perp} tot stand komt. Sectie 4.5 zal vervolgens h_{\perp} gebruiken om tot een oplossing voor f_{\perp} te komen.

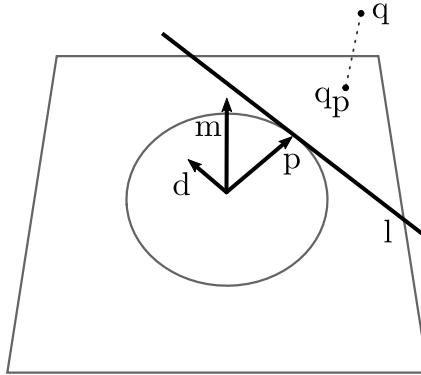
Aangezien \mathbf{m} gegeven is in h_{\perp} , komt de minimalisatie uit vergelijking (4.2) overeen met het vinden van de optimale parameter \mathbf{d} . De onderlinge relatie tussen \mathbf{d} en \mathbf{m} beperkt het domein van \mathbf{d} in h_{\perp} tot één dimensie, wat het analytisch oplossen vereenvoudigt. Per definitie geldt namelijk dat $\mathbf{m} = \mathbf{p} \times \mathbf{d}$ en moet \mathbf{d} dus loodrecht staan op \mathbf{m} . Bovendien werd eerder vastgelegd dat \mathbf{d} een eenheidsvector moet zijn. Dit beperkt het zoekdomein van \mathbf{d} tot een cirkel. Een ander gevolg is dat alle \mathbf{p}, \mathbf{d} , en dus ook de bijhorende rechten, coplanair zijn.

Vooraleer over te gaan op de optimalisatie van \mathbf{d} , is het handig om de minimumafstand in termen op te splitsen en een overzicht op te bouwen van de situatie. Gezien alle rechten met momentvector \mathbf{m} in eenzelfde vlak liggen, kan de minimumafstand ontbonden worden met de stelling van Pythagoras. Dit geeft

$$h_{\perp}(\mathbf{m}, \mathbf{q}) = \sqrt{u^2 + v^2}. \quad (4.3)$$

Hierin is u de afstand van \mathbf{q} naar de loodrechte projectie \mathbf{q}_p op het vlak met normaal $\hat{\mathbf{m}}$. Omdat de rechten coplanair zijn, is \mathbf{q}_p het dichtste punt waar ze door kunnen lopen. Afstand u is daarom constant en fungereert als een onvoorwaardelijke ondergrens

voor de afstand tussen de rechten en \mathbf{q} . Lengte v is de minimale afstand van deze rechten naar \mathbf{q}_p . Figuur 4.2 geeft een grafische voorstelling van deze opstelling.



Figuur 4.2: Illustratie van de ontbinding van h_{\perp} . Rechte l met richting \mathbf{d} loopt door punt \mathbf{P} . \mathbf{q}_p is de loodrechte projectie van \mathbf{q} op het vlak met normaal $\hat{\mathbf{m}}$. $\mathbf{d}, \mathbf{p}, l$ en \mathbf{q}_p zijn coplanair.

Afstand u in vergelijking (4.3) wordt gegeven door

$$\begin{aligned}\mathbf{q}_p &= \mathbf{q} - \hat{\mathbf{m}}(\mathbf{q} \cdot \hat{\mathbf{m}}) \\ u &= |\mathbf{q}_p - \mathbf{q}|.\end{aligned}\tag{4.4}$$

De optimale waarde voor \mathbf{d} is nu diegene waarbij l zo dicht mogelijk bij \mathbf{q}_p loopt, en dus v zo klein mogelijk is. Aangezien \mathbf{d} afhangt van \mathbf{q}_p , kan deze beschreven worden als een rotatie van $\hat{\mathbf{q}}_p$ rond $\hat{\mathbf{m}}$ met hoek γ . Stel dat er geen verdere beperkingen zijn op \mathbf{d} , d.w.z. de volledige cirkel aan richtingsvectoren valt binnen het zoekdomein. Er onderscheiden zich dan drie gevallen. Deze zijn grafisch uitgedrukt in figuur 4.3.

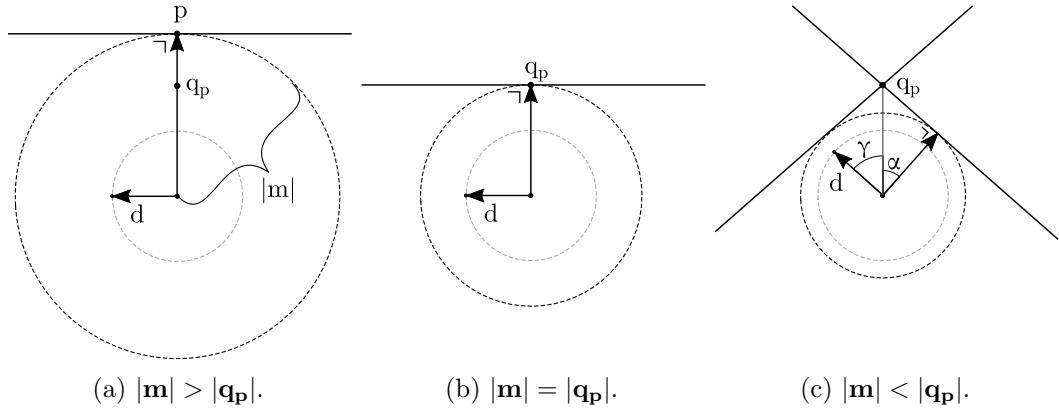
1. $|\mathbf{m}| > |\mathbf{q}_p|$: Er is geen rechte mogelijk die \mathbf{q}_p snijdt. De dichtstbijzijnde rechte gaat door $\mathbf{p} = |\mathbf{m}|\hat{\mathbf{q}}_p$. De optimale richtingsvector ligt op 90 graden van \mathbf{p} , en dus geldt $\sin \gamma = 1$
2. $|\mathbf{m}| = |\mathbf{q}_p|$: Er is precies één rechte die \mathbf{q}_p snijdt. Deze rechte is precies dezelfde die gevonden werd in het eerste geval en dus geldt ook hier $\sin \gamma = 1$.
3. $|\mathbf{m}| < |\mathbf{q}_p|$: Er zijn twee rechten die \mathbf{q}_p snijden. Indien α de hoek is tussen \mathbf{p} en \mathbf{q}_p , dan volgt dat

$$\sin \gamma = \sin \left(\alpha + \frac{\pi}{2} \right) = \cos \alpha = \frac{|\mathbf{m}|}{|\mathbf{q}_p|}.\tag{4.5}$$

Samengevat is deze relatie hetzelfde als

$$\sin \gamma = \max \left(\frac{|\mathbf{m}|}{|\mathbf{q}_p|}, 1 \right).\tag{4.6}$$

4. PLÜCKERBOMEN

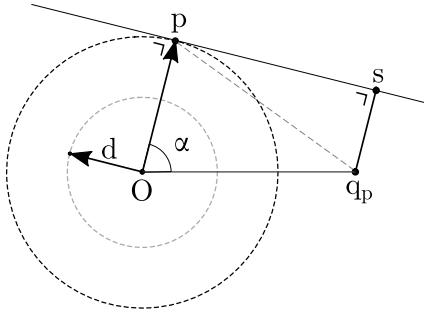


Figuur 4.3: Gevalenonderscheid van relatie tussen $|m|$ en $|q_p|$. In (a) liggen de rechten te ver van de oorsprong om q_p te snijden. In (b) is er precies één rechte die q_p snijdt. In (c) zijn er twee rechten die q_p snijden.

Het toepassen van de rotatie geeft dan de twee effectieve ideale richtingsvectoren \mathbf{d}_a en \mathbf{d}_b . Deze zijn dus gelijk als en slechts als $|m| \geq |q_p|$. Ze worden berekend met

$$\begin{aligned} \mathbf{d}_a &= \hat{\mathbf{q}}_p \cos \gamma + (\hat{\mathbf{m}} \times \hat{\mathbf{q}}_p) \sin \gamma \\ \mathbf{d}_b &= -\hat{\mathbf{q}}_p \cos \gamma + (\hat{\mathbf{m}} \times \hat{\mathbf{q}}_p) \sin \gamma. \end{aligned} \quad (4.7)$$

Een volledige implementatie van h_\perp vereist echter dat de waarden van \mathbf{d} ook begrensd kunnen worden. Het is dus mogelijk dat \mathbf{d}_a en \mathbf{d}_b buiten het toegelaten domein liggen. Indien dit het geval is, dan zal de beste beschikbare richtingsvector op de grenzen van het domein liggen. Deze grensvectoren zijn immers de vectoren die het dichtst bij \mathbf{d}_a en \mathbf{d}_b liggen. Ze zijn optimaal aangezien de afstand monotoon stijgt wanneer γ afwijkt van de bovenstaande waarde, totdat deze het punt halverwege \mathbf{d}_a en \mathbf{d}_b bereikt. Dit kan als volgt aangetoond worden met een geometrische constructie die geïllustreerd is in figuur 4.4. Indien de hoek α tussen \mathbf{q}_p en \mathbf{p} afwijkt van $\gamma - 90^\circ$, en dus l \mathbf{q}_p niet snijdt, dan ontstaat er een vierhoek tussen de oorsprong \mathbf{O} , \mathbf{q}_p , \mathbf{p} en \mathbf{s} , het punt op l het dichtst bij \mathbf{q}_p . Per constructie geldt dat $|\angle \mathbf{Ops}| = 90^\circ$. Hieruit volgt dat $|\angle \mathbf{Ops}| = |\angle \mathbf{Opq}_p + \angle \mathbf{q}_p \mathbf{ps}| = 90^\circ$. Wanneer α groeit, dan moet $\angle \mathbf{Opq}_p$ krimpen aangezien de som van de hoeken van $\triangle \mathbf{Opq}_p$ 180° moet blijven en $|\mathbf{q}_p|$ en $|\mathbf{p}|$ niet veranderen. Aangezien $|\angle \mathbf{Ops}| = 90^\circ$ moet de hoek $\angle \mathbf{q}_p \mathbf{ps}$ dan groeien. De afstand tussen \mathbf{q}_p en \mathbf{s} moet dan groeien omdat $|\angle \mathbf{q}_p \mathbf{sp}| = 90^\circ$ (loodrechte projectie) en $|\mathbf{q}_p - \mathbf{p}|$ per definitie stijgt bij groeiende α tot 180° .


 Figuur 4.4: Illustratie van een rechte die niet door \mathbf{q}_p gaat.

Als bij het berekenen van h_{\perp} blijkt dat de ideale richtingsvectoren buiten het domein liggen, dan moeten deze ingeperkt worden waarvoor eerst de grensvectoren bepaald worden. De grenzen van het domein werden eerder in (4.1) vastgelegd. De gezochte grensvectoren \mathbf{b}_1 en \mathbf{b}_2 moeten dan liggen in de vlakken met respectievelijke normaal \mathbf{d}_1 en \mathbf{d}_2 . Verder moeten ze uiteraard ook liggen in het vlak met normaal $\hat{\mathbf{m}}$. Een intersectie tussen de vlakken geeft telkens twee mogelijke eenheidsvectoren. Slechts één van deze voldoet aan beide voorwaarden. Een extra factor is daarom nodig om de correcte zin te selecteren. Vectoren \mathbf{b}_1 en \mathbf{b}_2 worden dan gegeven door

$$\begin{aligned}\mathbf{b}_1 &= \text{sgn}(\mathbf{d}_2 \cdot (\mathbf{d}_1 \times \mathbf{m})) \frac{\mathbf{d}_1 \times \mathbf{m}}{|\mathbf{d}_1 \times \mathbf{m}|} \\ \mathbf{b}_2 &= \text{sgn}(\mathbf{d}_1 \cdot (\mathbf{d}_2 \times \mathbf{m})) \frac{\mathbf{d}_2 \times \mathbf{m}}{|\mathbf{d}_2 \times \mathbf{m}|}.\end{aligned}\quad (4.8)$$

De vectoren \mathbf{d}_{α} en \mathbf{d}_{β} zijn nu de ideale richtingsvectoren voor h_{\perp} die ook binnen het domein liggen. Hieronder toont (4.9) hoe deze bepaald worden. Wanneer \mathbf{d}_a of \mathbf{d}_b buiten het domein valt, dan wordt \mathbf{b}_1 of \mathbf{b}_2 gekozen, afhankelijk van welke dichterbij is. Ze worden dus berekend met

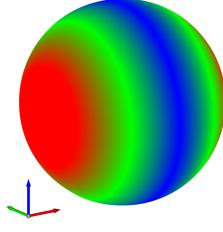
$$\begin{aligned}\mathbf{d}_{\alpha} &= \begin{cases} \mathbf{d}_a & \text{als } (\mathbf{d}_a \cdot \mathbf{d}_1 \geq 0 \wedge \mathbf{d}_a \cdot \mathbf{d}_2 \geq 0) \\ \mathbf{b}_1 & \text{als } (\mathbf{d}_a \cdot \mathbf{d}_1 < 0 \vee \mathbf{d}_a \cdot \mathbf{d}_2 < 0) \wedge (\mathbf{d}_a \cdot \mathbf{d}_1 \leq \mathbf{d}_a \cdot \mathbf{d}_2) \\ \mathbf{b}_2 & \text{als } (\mathbf{d}_a \cdot \mathbf{d}_1 < 0 \vee \mathbf{d}_a \cdot \mathbf{d}_2 < 0) \wedge (\mathbf{d}_a \cdot \mathbf{d}_1 > \mathbf{d}_a \cdot \mathbf{d}_2) \end{cases} \\ \mathbf{d}_{\beta} &= \begin{cases} \mathbf{d}_b & \text{als } (\mathbf{d}_b \cdot \mathbf{d}_1 \geq 0 \wedge \mathbf{d}_b \cdot \mathbf{d}_2 \geq 0) \\ \mathbf{b}_1 & \text{als } (\mathbf{d}_b \cdot \mathbf{d}_1 < 0 \vee \mathbf{d}_b \cdot \mathbf{d}_2 < 0) \wedge (\mathbf{d}_b \cdot \mathbf{d}_1 < \mathbf{d}_b \cdot \mathbf{d}_2) \\ \mathbf{b}_2 & \text{als } (\mathbf{d}_b \cdot \mathbf{d}_1 < 0 \vee \mathbf{d}_b \cdot \mathbf{d}_2 < 0) \wedge (\mathbf{d}_b \cdot \mathbf{d}_1 \geq \mathbf{d}_b \cdot \mathbf{d}_2). \end{cases}\end{aligned}\quad (4.9)$$

Ten slotte kan afstand v uit (4.3) berekend worden voor beide richtingsvectoren met

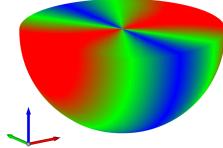
$$\begin{aligned}v_{\alpha} &= |(\mathbf{q}_p \times \mathbf{d}_{\alpha}) - \mathbf{m}| \\ v_{\beta} &= |(\mathbf{q}_p \times \mathbf{d}_{\beta}) - \mathbf{m}| \\ v &= \min(v_{\alpha}, v_{\beta}).\end{aligned}\quad (4.10)$$

De kortst mogelijke afstand tot \mathbf{q}_p is dan het minimum van dit paar. Dit, in combinatie met (4.3), geeft de totale minimale afstand.

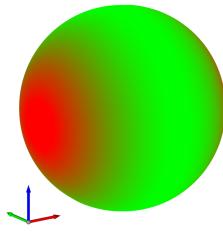
4.5 Minimalisatie over alle momentvectoren



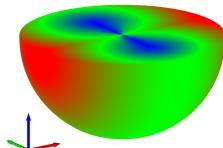
(a) Alle \mathbf{m} vectoren binnen radius 1 voor $\mathbf{q} = (1, 0, 0)$, geen beperking op \mathbf{d}



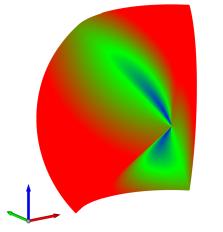
(b) Doorsnede van (a)



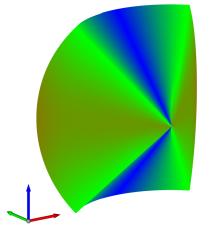
(c) Alle \mathbf{m} vectoren binnen radius 1 voor $\mathbf{q} = (0.5, 0, 0)$, geen beperking op \mathbf{d}



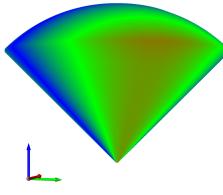
(d) Doorsnede van (c)



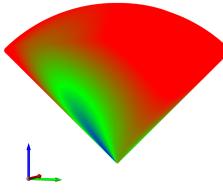
(e) $\mathbf{m} \in [(0, \frac{\pi}{4}, \frac{\pi}{4}); (1, \frac{3\pi}{4}, \frac{3\pi}{4})]$,
 $\mathbf{d}_1 = (1, 0, 0)$, $\mathbf{d}_2 = (0, 0, 1)$, $\mathbf{q} = (1, 0, 0)$



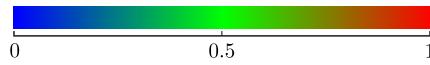
(f) Zelfde \mathbf{m} en \mathbf{q} als (e), maar $\mathbf{d}_1 = (-1, 0, 0)$,
 $\mathbf{d}_2 = (0, 0, -1)$



(g) $\mathbf{m} \in [(0, -\pi, 0); (1, \pi, \frac{\pi}{4})]$,
 $\mathbf{d}_1 = (1, 0, 0)$, $\mathbf{d}_2 = (0, 1, 0)$, $\mathbf{q} = (1, 0, 0)$



(h) Zelfde \mathbf{m} en \mathbf{q} als (g), maar $\mathbf{d}_1 = (-1, 0, 0)$,
 $\mathbf{d}_2 = (0, -1, 0)$



Figuur 4.5: Heatmaps van domein minimalisatiefunctie f_{\perp} voor verscheidene queries. De rode, groene en blauwe vectoren tonen de oriëntatie van de x -, y - en z -as.

Zoals aangegeven in vergelijking (4.2) is f_{\perp} het minimum van h_{\perp} over een domein van momentvectoren. Figuur 4.5 toont dit domein voor enkele verschillende queries. Figuren (a) tot (d) tonen aan dat, zonder beperkingen op \mathbf{d} , de minimumvectoren een ring vormen. Deze ring heeft een radius gelijk aan $|\mathbf{q}|$ en ligt in het vlak loodrecht op \mathbf{q} . Intuïtief is dit te verklaren doordat alle rechten met een momentvector in lijn met \mathbf{q} in een vlak liggen dat loodrecht ligt op \mathbf{q} en dit punt dus niet kunnen snijden. Omgekeerd geldt voor de momentvectoren loodrecht op \mathbf{q} dat \mathbf{q} in het vlak van rechten ligt en dus $\mathbf{q} = \mathbf{q}_p$ waardoor afstand u in vergelijking (4.3) nul wordt. De totale afstand is dan nul indien er een rechte is die door \mathbf{q}_p gaat, wat het geval is wanneer $|\mathbf{q}_p| \geq |\mathbf{m}|$ zoals te zien is in figuur 4.3. Verder zijn de waarden continu, op een discontinuitéit in de oorsprong na. In de oorsprong is \mathbf{m} immers nul, en bijgevolg is \mathbf{q}_p uit vergelijking (4.4) ongedefinieerd net als \mathbf{b}_1 en \mathbf{b}_2 uit vergelijking (4.8). Wanneer de momentvector nul is, met andere woorden, dan geldt de beperking tot een cirkel van richtingsvectoren niet en is iedere richting mogelijk. Afbeeldingen (e) tot (h) tonen het effect van een beperking op de richtingsvectoren. Door de manier waarop de richtingen beperkt worden, kan maar een beperkte ruimtehoek getoond worden. Het domein van momentvectoren is hier dan ook beperkt tot hoeken met $\phi \leq \frac{\pi}{2}, \theta \leq \frac{\pi}{2}$. Meer informatie hierover is te vinden in sectie 4.6. De figuren demonstreren dat, onder deze condities, de functie continu blijft. Verder is ook te zien dat het beperken van \mathbf{d} een significant effect heeft op de minimumafstanden. Dit is voordelig, aangezien meer contrast leidt tot een grotere kans op het wegsnoeien van takken tijdens zoekopdrachten.

Er zijn verschillende manieren om de minimalisatie van h_{\perp} in vergelijking (4.2) uit te voeren. Vermits een analytische oplossing niet haalbaar bleek, werd er geoptimaliseerd voor numerieke optimalisatie [4]. Een eenvoudige methode is het gebruik van een lokaal optimalisatiealgoritme zonder afgeleiden. Afdalen in het domein van de doelfunctie zonder gradiënt kan bijvoorbeeld door deze te benaderen met monsters van de functiewaarden. Bijgevolg moet hier dus enkel de functie, een startwaarde en de stopconditie gespecificeerd worden. Sommige algoritmes in deze categorie kunnen optioneel ook binnen bepaalde grenzen optimaliseren, wat essentieel is aangezien we de minimale afstand zoeken voor alle rechten binnen een domein V zoals gedefinieerd in vergelijking (4.1). Binnen het onderzoek van deze masterthesis werd gebruik gemaakt van de applicatiebibliotheek NLOpt[15] omwille van de grote variëteit aan optimalisatiealgoritmes die hierin geïmplementeerd zijn. Enkele verschillende algoritmes werden met dezelfde dataset proefgedraaid. Experimenteel bleek hieruit *Splx*, een implementatie van *Subplex* [21], het meest geschikt omwille van zijn robuustheid en snelle convergentie. Een alternatieve methode, *Constrained Optimization BY Linear Approximations (COBYLA)*, is ook beschikbaar in NLOpt. Deze functioneert meestal ook, maar trager. Echter is er een openstaande fout in NLOpt waardoor het algoritme met bepaalde invoerparameters nooit eindigt.

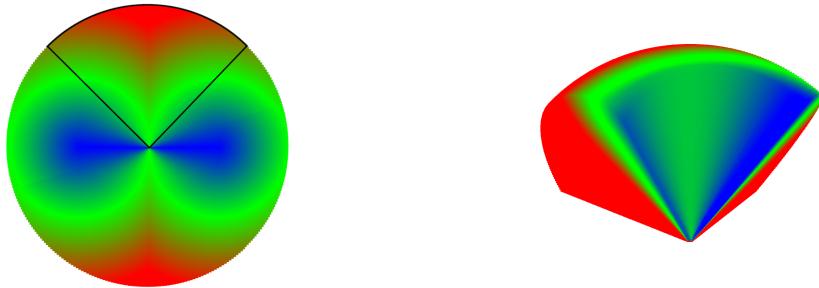
Gradiënt-gebaseerde methodes zijn vaak efficiënter dan methodes zonder afgeleiden [4]. Dit komt omdat ze de gradiënt gebruiken om preciezer af te afdalen zonder veel extra functiewaarden te moeten berekenen. Hier staat tegenover dat er een efficiënte gradiëntfunctie beschikbaar moet zijn. h_{\perp} is afleidbaar, en zou dus geminimaliseerd kunnen worden met deze methodes. Dit onderzoek gaat echter

4. PLÜCKERBOMEN

dieper in op de complexiteit van de zoekopdrachten in functie van het aantal bezochte knopen, eerder dan de performantie van de optimalisaties.

Bij gradient descent algoritmes moeten steeds de stopcondities gespecificeerd worden. De twee hoofdparameters zijn hier de absolute en de relatieve stopwaarde. De afdaling stopt wanneer een waarde bereikt wordt die kleiner is dan de absolute stopwaarde, of wanneer het verschil tussen twee opeenvolgende iteraties kleiner is dan de relatieve stopwaarde. De keuze van deze parameters is belangrijk voor de efficiëntie. Indien ze te klein zijn, dan zal het lang duren voordat de gewenste waarde gevonden wordt. Als ze te groot zijn, dan zal de minimalisatie resulteren in een minimumafstand die groter is dan in werkelijkheid. Dit kan er voor zorgen dat er onterecht knopen overgeslagen worden die wel degelijk een optimaal resultaat kunnen bevatten. In praktijk valt dit echter mee. Een maximumfout ϵ betekent immers dat het gevonden resultaat maximaal ϵ verder ligt dan de werkelijke rechte met minimale afstand. De implementatie die geschreven werd om experimenten uit te voeren hanteert een waarde van 10^{-3} . Dit was voldoende om geen fouten te krijgen tijdens het uitvoeren van de experimenten. Verder kan het zoekalgoritme ook conservatiever gemaakt worden door deze marge in rekening te nemen in de keuze om knopen over te slagen. Een grotere marge betekent wel dat er minder takken weggeweerd worden, wat een negatieve impact op de performantie heeft.

Het is nodig dat de minimalisatie meerdere keren wordt uitgevoerd omdat er een lokaal optimalisatiealgoritme wordt gebruikt en er mogelijks meerdere minima in het domein zijn. In figuur 4.5 is te zien dat de minima een continue ringvormige structuur aannemen. Wanneer er echter snedes genomen worden van de ruimte, dan is het mogelijk dat dit meerdere lokale minima geeft. Dit wordt geïllustreerd in figuur 4.6.



Figuur 4.6: Bij begrenzing van het domein kunnen er meerdere lokale minima zijn.

Het is cruciaal om te voorkomen dat het minimalisatiealgoritme een lokaal maar niet globaal minimum vindt. De minimalisatie wordt daarom acht keer uitgevoerd, één maal per hoek van het domein. De kleinste waarde die gevonden wordt, is het werkelijke minimum. Dit volstaat aangezien, in de situaties waar er meerdere disjuncte lokale minima aan de grenzen van het domein liggen, er geen minimum in het midden van het domein kan liggen zoals te zien is op figuur 4.6.

Vaak is het mogelijk alsnog het minimum te vinden met minder dan acht minimalisaties. Na het minimum voor een knoop te berekenen, is immers geweten in

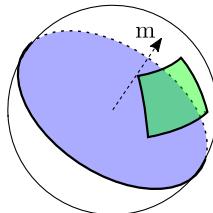
welke van de twee kindknopen deze minimale rechte ligt. Het minimum van deze kindknoop ligt dan meteen vast. Voor de andere knoop geldt dat de minimumwaarde niet lager kan zijn dan het minimum van de ouder. Wanneer vanuit een hoek deze waarde gevonden wordt, dan kunnen de overige hoeken overgeslagen worden.

4.6 Opdeling ruimte

De vorige secties duidden hoe de minimumafstand van een domein V uit 4.3 berekend wordt, maar hierbij resteert de vraag hoe deze domeinen tot stand komen. De keuzes die gemaakt worden omtrent de opdeling van de ruimte zijn een belangrijk deel van de ontwikkeling van een spatiale zoekstructuur. Ze beïnvloeden de numerieke stabiliteit van de minimalisatie, complexiteit van de splitsingen in de boom en de algemene computationele efficiëntie.

Vooraleerst werd gekozen om de ruimte op te delen op basis van de sferische coördinaten van de momentvectoren \mathbf{m} . Een ideale onderverdeling splitst de ruimte perfect in deeldomeinen met lage of hoge minimumafstand. Hierboven werd in figuur 4.5 aangetoond dat de minimumvectoren voor \mathbf{m} zich groeperen in ringvormige structuren. Gemiddeld gezien zullen sferische balken hier nauwer bij aansluiten dan carthesische. Verder wordt er, analoog aan kd-bomen, in ieder knoop maximaal één \mathbf{m} dimensie gesplitst. Dit vereenvoudigt de constructie van de boom door het aantal mogelijke splitsingen in te perken. Een laatste belangrijke opmerking bij de momentvectoren is dat geen enkel van de deeldomeinen de oorsprong mag bevatten. Zoals werd beschreven bij figuur 4.5 is de minimumafstandsfunctie niet gedefinieerd in de oorsprong. Voor de beginradius van de momentdomeinen wordt daarom een kleine waarde ϵ gekozen in plaats van nul.

Naast het moment kan er ook gesplitst worden op richting. Aangezien het totale domein hier de verzameling van alle eenheidsvectoren is, lijkt het passend om deze op te delen op azimut en elevatie. Dit brengt echter enkele nadelen met zich mee. Bij een beperking van de richting tot een arbitrair sferisch interval is het mogelijk dat er voor een gegeven momentvector geen enkele geldige richtingsvector is. Een voorbeeld hiervan is te zien in figuur 4.7.



Figuur 4.7: Voorbeeld van een ongeldige combinatie van momentvector en directioneel sferisch interval. Enkel de richtingsvectoren op de blauwe cirkel zijn geldig voor de momentvector \mathbf{m} , maar geen daarvan zit in het domein voorgesteld door het groene gebied.

Deze combinaties zijn ongeldig en moeten dan bij constructie vermeden worden,

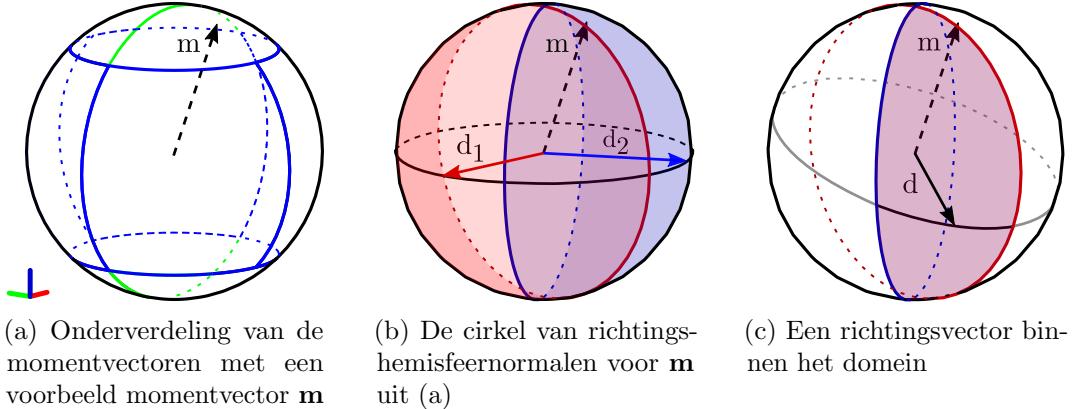
4. PLÜCKERBOMEN

of ze vormen lege knopen in de boom. Deze strategie vereist daarnaast ook dat de richtingsvectoren \mathbf{d}_a en \mathbf{d}_b uit vergelijking (4.7) van carthesische naar sferische coördinaten worden geconverteerd voor de inperking beschreven in formule (4.9). Vervolgens moeten deze dan terug in carthesische coördinaten gezet worden voor de afstandsberekeningen in vergelijking (4.10). Dit is niet alleen inefficiënt, het veroorzaakt ook problemen met numerieke fouten. Wanneer \mathbf{d}_a of \mathbf{d}_b wijst naar $(0, 0, \pm 1)$, dan is de azimut ongedefinieerd. Bovendien wijken deze in praktijk zo goed als altijd af van $(0, 0, \pm 1)$, omwille van numerieke precisie. Door afrondingsfouten krijgt de azimut dan een arbitraire waarde, wat problemen kan geven bij het inperken in formule (4.9).

Een andere optie zou kunnen zijn om het domein van de richtingsvectoren als ééndimensionaal te beschouwen. Gegeven een momentvector is er namelijk slechts een cirkel van richtingsvectoren mogelijk. Het is echter moeilijk om de richtingen op deze cirkel te parametriseren. Een hoek is immers enkel betekenisvol als er een consistente referentievector is rond welke de hoek genomen wordt. Het toewijzen van referentievectoren aan momentvectoren op een consistente manier die goede splitsingen toelaat is een uitdaging bij deze werkwijze.

Zoals in sectie 4.3 reeds werd aangegeven, werd er gekozen om het domein van de richtingsvectoren af te bakenen als de intersectie van twee richtingshemisferen. Hierin zijn \mathbf{d}_1 en \mathbf{d}_2 de normaalvectoren van de vlakken die deze richtingshemisferen bepalen. Dit heeft een aantal voordelen. Zo vereist deze aanpak geen conversie naar sferische coördinaten. Controleren of een richting binnen de grenzen valt vergt slechts twee scalaire producten, en indien dit het geval is dan wordt de vector gebruikt zonder verdere operaties. Indien \mathbf{d}_1 en \mathbf{d}_2 goed gekozen worden is er een geldige richting voor iedere momentvector en worden numerieke problemen vermeden.

Een goede keuze van deze richtingshemisferen is dus van belang. Bij de berekening van de grensvectoren \mathbf{b}_1 en \mathbf{b}_2 in (4.8) wordt het kruisproduct tussen \mathbf{d}_1 , \mathbf{d}_2 en \mathbf{m} gebruikt. Indien deze vectoren zouden samenvallen, precies of bij benadering, dan zal de lengte van dit kruisproduct naar nul toegaan. De normering die daarna plaatsvindt zal dan aanleiding geven tot numerieke fout. Een oplossing hiervoor is om in de wortel van de boom het domein te splitsen in 32 zogenaamde sectoren. De boom zal dus 32 wortelknopen hebben met deze sectoren als domein. De volgende alinea legt samen met figuur 4.8 uit hoe deze tot stand komen.

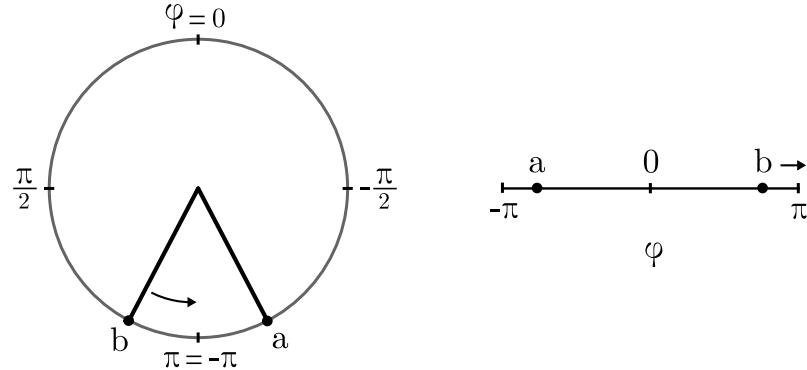


Figuur 4.8: Onderverdeling van het domein

- Eerst worden de momentvectoren in zes gepartitioneerd. Dit wordt geïllustreerd met de blauwe lijnen in 4.8a. Alle momentvectoren met een elevatie kleiner dan 45° vormen een noordpool. Analoog worden de vectoren met een elevatie groter dan 135° gegroepeerd in een zuidpool. Alle momentvectoren die resteren worden opgedeeld in vier partities op basis van azimut. Het midden van iedere partitie functioneert als de normaalvector van de cirkel waaruit de normalen van de richtingshemisferen gekozen worden. Zo toont 4.8a een momentvector op de noordpool, en loopt in 4.8b de bijhorende cirkel van richtingshemisfeernormalen over de evenaar. De selectie van een richtingshemisfeer met normaal \mathbf{d}_1 of \mathbf{d}_2 op deze cirkel garandeert dat er minstens 45° en maximum 135° zit tussen \mathbf{m} en \mathbf{d}_1 of \mathbf{d}_2 . Dit voorkomt een explosie van de relatieve fout door annulatie wanneer het kruisproduct berekend wordt, welke anders na normering voor een grote absolute fout zou zorgen. Figuur 4.8c toont dan een richtingsvector die in het domein van \mathbf{d}_1 en \mathbf{d}_2 ligt.
- De noord- en zuidpool worden nogmaals in tweeën gesplitst op basis van azimut. Dit geeft de groene lijnen in 4.8a. Normale gradient descent houdt namelijk geen rekening met de geometrische structuur van sferische coördinaten, zoals te zien is op figuur 4.9. Zo komt het voor dat het minimum te vinden is op $\phi = -175^\circ$, maar dat het algoritme afdaalt naar $\phi = 180^\circ$. Deze punten liggen in feite dicht bij elkaar, maar voor het algoritme is dit de volledig andere kant van het domein. Door de polen op te splitsen in twee delen, $\phi_1 \in [-\frac{\pi}{2}, 0]$ en $\phi_2 \in [0, \frac{\pi}{2}]$ respectievelijk, kan deze fout niet meer plaatsvinden. Het is dan geen probleem dat er niet wordt afgedaald van $-\pi$ naar π , aangezien dit dan ook de grens van het domein is.
- Ten slotte worden de 32 sectoren gevormd door voor ieder van de acht momentpartities in 4.8a vier onderverdelingen te maken op basis van richting. Figuur 4.10 toont twee voorbeelden van zulke sectoren. De intersectie van twee richtingshemisferen kan maximaal opnieuw een hemisfeer geven, dus zijn er alvast twee sectoren per partitie nodig om alle richtingen voor te stellen. Echter

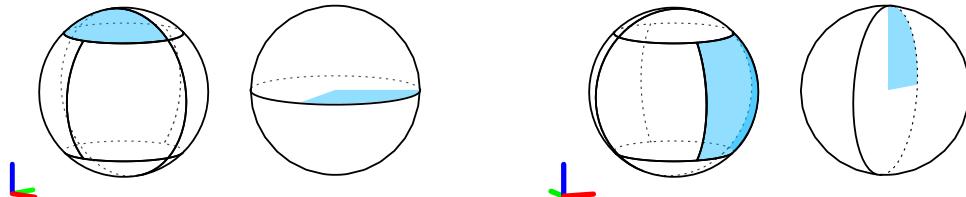
4. PLÜCKERBOMEN

treed er een bijkomend probleem op indien de volledige hemisfeer geselecteerd is en dus $\mathbf{d}_1 = \mathbf{d}_2$. In (4.8) zal bijvoorbeeld $\mathbf{d}_2 \cdot (\mathbf{d}_1 \times \mathbf{m})$ dan nul zijn. Dit geeft problemen bij het berekenen van \mathbf{b}_1 en \mathbf{b}_2 . Een eenvoudige oplossing is dan om de richtingen in vier in plaats van twee op te delen.



Figuur 4.9: Standaard gradient descent houdt geen rekening met de geometrische structuur van sferische coördinaten.

Figuur 4.10 toont twee voorbeelden van sectoren in een grafische representatie.



(a) Een sector op de noordpool. $\phi \in [0, -\frac{\pi}{2}], \theta \in [0, \frac{\pi}{4}], \mathbf{d}_1 = (1, 0, 0), \mathbf{d}_2 = (0, -1, 0)$ (b) Een sector over de evenaar. $\phi \in [0, -\frac{\pi}{2}], \theta \in [\frac{\pi}{4}, \frac{3\pi}{4}], \mathbf{d}_1 = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0), \mathbf{d}_2 = (0, 0, 1)$

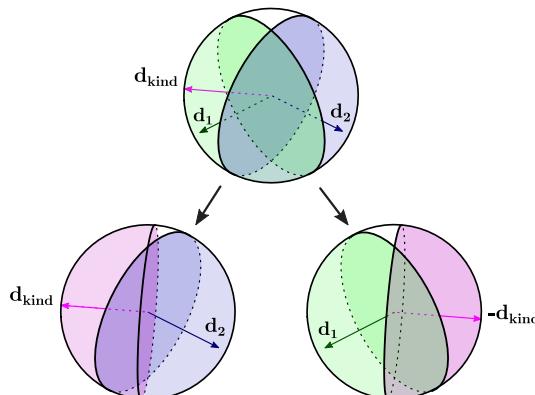
Figuur 4.10: Twee voorbeelden van sectoren. In beide staat links het domein van momentvectoren en rechts het bijhorende domein van richtingshemisfeernormalen.

Het opdelen van de wortel van de boom in 32 sectoren roept misschien vragen op rond de impact op de performantie. Immers, indien een aantal sectoren met een hoge minimumafstand kunnen gegroepeerd worden, dan kunnen deze allemaal uitgesloten worden door enkel het domein van de overkoepelende knoop te beschouwen. Echter is er een zeker niveau van opdeling nodig voordat de minimumafstand stijgt. Indien er te weinig opdeling is, en het domein dus groot is, dan is de kans immers ook groot dat het domein een stuk van de minimumvectoren bevat. Zo was in figuur 4.5 namelijk te zien dat de minimumvectoren een ringvormige structuur aannemen die een significante ruimtehoek beslaat. Verder zijn in praktische toepassingen de bomen groot genoeg dat 32 minimalisatieoperaties verwaarloosbaar zijn.

4.7 Constructie

Gegeven een verzameling van rechten wordt de bijhorende plückerboom als volgt geconstrueerd. Eerst worden de rechten gesorteerd volgens de sector waarin ze zich bevinden. Vervolgens wordt voor iedere sector met de bijhorende rechten een wortelknoop gebouwd. Het samenstellen van een knoop gaat als volgt. Indien er geen rechten toegewezen zijn aan de knoop, dan is de knoop de lege verzameling \emptyset . Anders worden de rechten gesplitst in twee deelverzamelingen met behulp van een heuristiek. De heuristiek kiest om te splitsen op de richtingsvector of één van de momentvectordimensies geleid door welke het meest waarschijnlijk een efficiënte boom geeft. De verschillende heuristieken worden hieronder besproken. Ten slotte worden de rechten effectief onderverdeeld aan de hand van de gekozen splitsing. Met deze deelverzamelingen worden dan recursief de kindknopen gebouwd.

Splitsen van een domein op moment- of richtingsvector verloopt verschillend. Een partitionering op basis van de momentvector kan simpelweg door één van de sferische coördinaten van het domein in twee te delen. Opdelen volgens richtingsvector houdt in dat er een nieuwe vector d_{kind} bepaald wordt. Een knoop met richtingsgrenzen (d_1, d_2) , zoals gedefinieerd in 4.3, krijgt dan twee kindknopen met respectievelijke richtingsgrenzen (d_{kind}, d_2) en $(d_1, -d_{kind})$. Dit wordt gedemonstreerd in figuur 4.11.



Figuur 4.11: Splitsing van een richtingsinterval in twee deelintervallen

d_{kind} wordt uit hetzelfde vlak genomen als d_1 en d_2 . De normaalvector van dit vlak krijgt hier de naam d_n . Eerst wordt een pivotelement uit de verzameling van rechten gekozen. Dit kan door de richtingsvectoren van de rechten te nemen, deze te projecteren op het vlak met normaal d_n en dan terug te normeren, om ten slotte van deze de lengte van het kruisproduct met d_1 te berekenen. Deze sinuswaardes geven dan een idee van de rotatieafstand van iedere rechte tegenover d_1 . Wanneer we de rechten sorteren op deze waarde, dan kunnen we de mediaan kiezen als pivotelement. d_{kind} is dan gelijk aan $\frac{d_1 \times d_{pivot}}{|d_1 \times d_{pivot}|}$.

Er zijn veel mogelijke heuristieken voor het kiezen van splitsingen. In het kader van dit onderzoek werden enkele verschillende strategieën uitgetest.

4. PLÜCKERBOMEN

- Afwisselend: de splitsing wordt gekozen op basis van de diepte van de knoop. Bijvoorbeeld: niveau 1,2,3,4 en 5 splitsen respectievelijk op radius, ϕ , θ , richting en opnieuw radius.
- Maximum variantie: er wordt gesplitst op de dimensie waarover de positie van de elementen de grootste variantie vertonen. De variantie geeft een idee van de spreiding van de waarden. Het idee is hier dan ook om knopen met veel spreiding te splitsen in knopen waar de waarden compacter liggen.
- Maximum gemiddelde absolute afwijking (Mean Absolute Deviation of MAD): gelijkaardig aan het splitsen op maximum variantie, maar in plaats van variantie wordt de gemiddelde absolute afwijking gebruikt. Deze waarde geeft ook een indicatie van spreiding, maar is minder gevoelig voor uitschieters.

Er werd gekozen om, onafhankelijk van de heuristiek, één vierde van de splitsingen op richting uit te voeren. Bovenstaande heuristieken werden dus enkel gebruikt om een splitsingsdimensie voor de momentvectoren te kiezen. Dit laat toe om de heuristieken te testen, onafhankelijk van de berekening van spreidingsmaten van de richtingsvectoren. Deze is namelijk niet triviaal, gezien de formulering van de grenzen \mathbf{d}_1 en \mathbf{d}_2 van het richtingsdomein in sectie 4.3. Een mogelijke manier om de spreiding van de richtingen te meten is om alle richtingsvectoren te projecteren op de cirkel waarop \mathbf{d}_1 en \mathbf{d}_2 liggen. Vervolgens worden dan de hoeken tussen deze en een grensvector gemeten. Van deze verzameling van hoekwaarden kunnen dan spreidingsmaten berekend worden.

De heuristieken op basis van spreiding worden berekend met de sferische representatie van de momentvectoren. Aangezien deze waarden in verschillende domeinen liggen, moeten ze genormaliseerd worden. De variantie van n waarden x_i in bereik $[a, b]$ met gemiddelde μ wordt genormaliseerd met

$$\begin{aligned}
 \text{Var}\left(\frac{x_i - a}{b - a}\right) &= \frac{\sum\left(\frac{x_i - a}{b - a} - \frac{\mu - a}{b - a}\right)^2}{n} \\
 &= \frac{\sum\left(\frac{x_i - \mu}{b - a}\right)^2}{n} \\
 &= \frac{1}{(b - a)^2} \frac{\sum(x_i - \mu)^2}{n} \\
 &= \frac{\text{Var}(x_i)}{(b - a)^2}.
 \end{aligned} \tag{4.11}$$

De normalisatie van de MAD gebeurt door deling met $b - a$ in plaats van $(b - a)^2$, aangezien deze correleert met de standaarddeviatie, wat de wortel van de variantie is.

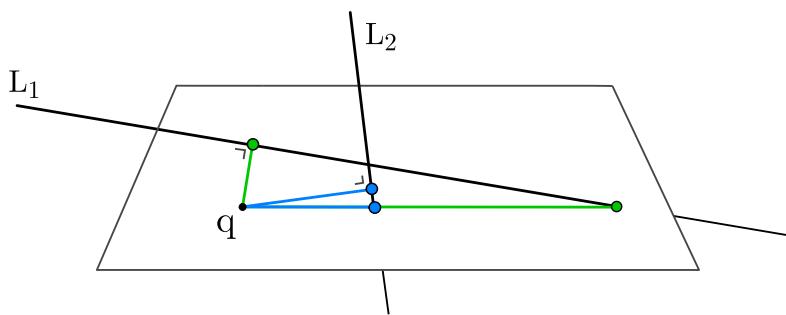
De performantie van deze heuristieken wordt besproken in hoofdstuk 6.

Hoofdstuk 5

Extensies

5.1 Dichtste raakpunt

Tot nu toe werd steeds de rechte met de kortste loodrechte afstand gezocht, maar het is ook interessant om andere afstanden te beschouwen. Zoals reeds vermeld op het einde van sectie 4.2, is de loodrechte afstand niet correct voor gebruik in PDE. Stralen die parallel met het oppervlak staan zullen dit oppervlak nooit raken en dus geen bijdrage leveren aan de iradiantie, hoe dichtbij ze ook lopen. De gewenste afstand is hier de lengte tussen \mathbf{q} en de intersectie van iedere rechte met het oppervlak, ook zoekvlak genaamd, door \mathbf{q} met normaal $\hat{\mathbf{n}}$. Hierbij wordt de assumptie gemaakt dat het oppervlak rond \mathbf{q} vlak is. Dit is redelijk aangezien bij een stijgend aantal fotonstalen, en een constant aantal gezochte stralen, de regio waarin de resulterende snijpunten liggen krimpt. In de limiet geeft dit dan een infinitesimaal klein oppervlak. Figuur 5.1 demonstreert het verschil in de afstand.



Figuur 5.1: Verschil tussen loodrechte afstand en afstand tot raakpunt

Het is niet noodzakelijk om de minimumafstandsfunctie f_{\perp} aan te passen om te zoeken op dichtste raakpunt. Bij het zoeken wordt een tak overgeslagen indien f aanduidt dat de minimumafstand van deze tak hoger ligt dan de verste huidige kandidaat, en de tak dus geen dichtere rechten kan bevatten. Indien f de minimumafstand onderschat, maar nooit overschat, dan zal er nooit een tak ontrect worden overgeslagen. De performantie kan wel dalen doordat er onnodig knopen worden

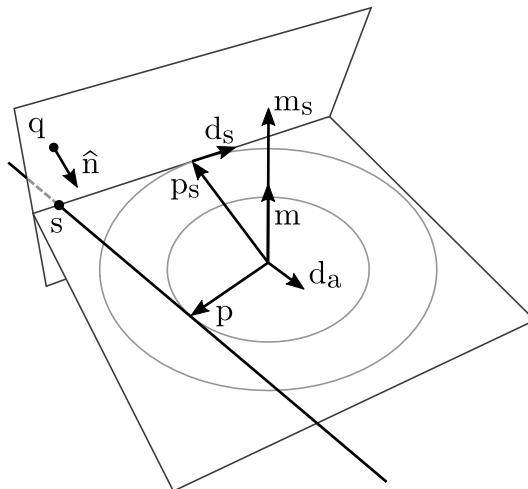
5. EXTENSIES

bezocht die met een nauwer aansluitende f weggesneden zouden worden. Dit is van toepassing in dit geval, aangezien de loodrechte afstand altijd kleiner of gelijk is aan de afstand tot het raakpunt. De loodrechte afstand is immers de kortst mogelijke afstand tussen het punt en de rechte. Het is wel noodzakelijk dat de uiteindelijke afstands berekening voor individuele rechten wordt aangepast. De afstand tussen het raakpunt van een rechte (\mathbf{d}, \mathbf{m}) en een punt \mathbf{q} op zoekvlak met normaal $\hat{\mathbf{n}}$ wordt beschreven met de formule

$$\begin{aligned} g_{opp}(\mathbf{m}, \mathbf{d}, \mathbf{q}, \hat{\mathbf{n}}) &= |\mathbf{s}(\mathbf{m}, \mathbf{d}, \mathbf{q}, \hat{\mathbf{n}}) - \mathbf{q}| \\ \mathbf{s}(\mathbf{m}, \mathbf{d}, \mathbf{q}, \hat{\mathbf{n}}) &= \mathbf{p} + \left(\mathbf{d} \frac{(\mathbf{q} - \mathbf{p}) \cdot \hat{\mathbf{n}}}{\mathbf{d} \cdot \hat{\mathbf{n}}} \right) \\ \mathbf{p} &= \mathbf{d} \times \mathbf{m}, \end{aligned} \quad (5.1)$$

waarin \mathbf{p} de loodrechte projectie van de oorsprong op de rechte is, \mathbf{s} het snijpunt van de rechte met het vlak door \mathbf{q} met normaal $\hat{\mathbf{n}}$ en g_{opp} de gezochte afstand.

Het is echter ook mogelijk om een minimumafstandsfunctie f_{opp} op te stellen, die de werkelijk kleinste afstand tot raakpunt geeft van een domein uit de plückerboom. Hiervoor moet enkel een optimaliseringfunctie h_{opp} opgesteld worden, analoog met h_{\perp} uit vergelijking (4.2), die de minimale afstand voor een vaste momentvector berekent. Figuur 5.2 illustreert de situatie en verschillende variabelen.



Figuur 5.2: Illustratie van de verschillende geometrische componenten gebruikt bij de afleiding van h_{opp} voor het zoeken van de dichtste raakpunten

De optimale rechte in h_{opp} , namelijk diegene met momentvector \mathbf{m} waarvan het snijpunt met het zoekvlak met normaal $\hat{\mathbf{n}}$ het dichtst bij \mathbf{q} ligt, snijdt dit zoekvlak in punt \mathbf{s} . Alle rechten in het domein liggen in het rechtenvlak met normaal $\hat{\mathbf{m}}$. Aangezien \mathbf{s} in zowel het zoekvlak als het rechtenvlak moet liggen, zal h_{opp} eerst de snijdingslijn tussen deze twee berekenen. De plückercoördinaten $(\mathbf{d}_s, \mathbf{m}_s)$ van deze

snijdingslijn worden berekend met

$$\begin{aligned}\mathbf{d}_s &= \frac{\hat{\mathbf{m}} \times \hat{\mathbf{n}}}{|\hat{\mathbf{m}} \times \hat{\mathbf{n}}|} \\ \mathbf{m}_s &= \hat{\mathbf{m}} \left(\frac{\hat{\mathbf{n}} \cdot \mathbf{q}}{|\hat{\mathbf{m}} \times \hat{\mathbf{n}}|} \right).\end{aligned}\tag{5.2}$$

Hierbij is het belangrijk te controleren of $|\hat{\mathbf{m}} \times \hat{\mathbf{n}}| > 0$. Indien dit niet het geval is, dan liggen de vlakken parallel en zal er dus geen enkele rechte snijden. Punt \mathbf{s} , het snijpunt van de optimale rechte met het zoekvlak, is de projectie van \mathbf{q} op deze lijn en kan gevonden worden met

$$\begin{aligned}\mathbf{p}_s &= \mathbf{d}_s \times \mathbf{m}_s \\ \mathbf{s} &= \mathbf{p}_s + \mathbf{d}_s \mathbf{d}_s \cdot (\mathbf{q} - \mathbf{p}_s).\end{aligned}\tag{5.3}$$

De richtingsvectoren \mathbf{d}_a en \mathbf{d}_b van de optimale rechten kunnen nu berekend worden zoals in vergelijking (4.7), buiten dat nu \mathbf{s} gebruikt wordt in plaats van \mathbf{q}_p aangezien dat hier het punt is waar de rechte hoort door te lopen. Hierna moeten deze richtingsvectoren ingeperkt worden tot het richtingendomein. Grensvectoren $\mathbf{b}_1, \mathbf{b}_2$ en ingeperkte richtingsvectoren \mathbf{d}_α en \mathbf{d}_β worden hierbij net zoals in vergelijkingen (4.8) en (4.9) berekend. Het is mogelijk dat na het inperken de richtingsvector \mathbf{d}_α of \mathbf{d}_β parallel staat met het zoekvlak. Dit is voor \mathbf{d}_α het geval bij indien $|\mathbf{d}_\alpha \cdot \mathbf{d}_s| = 1$. De bijhorende rechte zal dan het vlak dus niet snijden en deze hoeft dan ook niet verder berekend te worden. Analoog voor \mathbf{d}_β .

Nu \mathbf{d}_α en \mathbf{d}_β gekend zijn, kunnen de afstanden van de overeenkomstige rechten tot het zoekpunt \mathbf{q} berekend worden. Eerder werd in vergelijking (5.2) al de snijdingslijn $(\mathbf{d}_s, \mathbf{m}_s)$ gevonden. Punt \mathbf{s} was de intersectie van deze lijn met de rechten met richting \mathbf{d}_a of \mathbf{d}_b . Om de afstanden te berekenen tot de rechten met de ingeperkte richtingsvectoren, moeten nu de intersecties van $(\mathbf{d}_\alpha, \mathbf{m})$ en $(\mathbf{d}_\beta, \mathbf{m})$ met de snijdingslijn bepaald worden.

De intersectie tussen twee rechten L_1 en L_2 voorgesteld door plückercoördinaten $(\mathbf{d}_1, \mathbf{m}_1)$ en $(\mathbf{d}_2, \mathbf{m}_2)$ wordt gegeven door [6]

$$\begin{aligned}L_1 \cap L_2 = & |\mathbf{d}_1 \times \mathbf{d}_2|^2 e_0 + \mathbf{d}_1 \mathbf{m}_2 \cdot (\mathbf{d}_1 \times \mathbf{d}_2) + \mathbf{d}_2 \mathbf{m}_1 \cdot (\mathbf{d}_2 \times \mathbf{d}_1) \\ & + \frac{1}{2}(\mathbf{d}_2 \cdot \mathbf{m}_1 - \mathbf{d}_1 \cdot \mathbf{m}_2)(\mathbf{d}_1 \times \mathbf{d}_2),\end{aligned}\tag{5.4}$$

wat een PGA punt is met e_0 de projectieas, zoals uitgelegd in sectie 3.3. Dit kan omgezet worden in een vector uit de klassieke \mathbb{R}^3 lineaire algebra met

$$\begin{aligned}L_1 \cap L_2 = & \frac{1}{|\mathbf{d}_1 \times \mathbf{d}_2|^2} \left(\mathbf{d}_1 \mathbf{m}_2 \cdot (\mathbf{d}_1 \times \mathbf{d}_2) + \mathbf{d}_2 \mathbf{m}_1 \cdot (\mathbf{d}_2 \times \mathbf{d}_1) \right. \\ & \left. + \frac{1}{2}(\mathbf{d}_2 \cdot \mathbf{m}_1 - \mathbf{d}_1 \cdot \mathbf{m}_2)(\mathbf{d}_1 \times \mathbf{d}_2) \right).\end{aligned}\tag{5.5}$$

Vermits volgens vergelijking (5.2) $\mathbf{m}_s = c\mathbf{m}$, $c \in \mathbb{R}$, staan \mathbf{d}_α en \mathbf{m}_s loodrecht, net als \mathbf{d}_s en \mathbf{m} . Zowel $\mathbf{d}_\alpha \cdot \mathbf{m}_s$ als $\mathbf{d}_s \cdot \mathbf{m}$ zijn bijgevolg nul, en het intersectiepunt \mathbf{s}_α

tussen de snijdingslijn $(\mathbf{d}_s, \mathbf{m}_s)$ en $(\mathbf{d}_\alpha, \mathbf{m})$ volgt dan uit

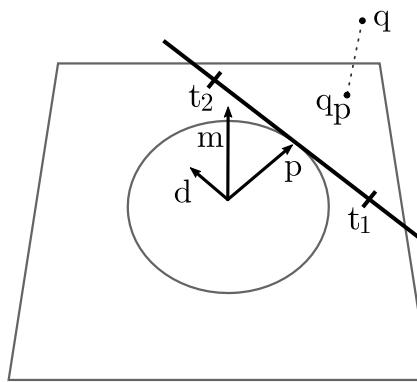
$$\mathbf{s}_\alpha = \frac{1}{|\mathbf{d}_s \times \mathbf{d}_\alpha|^2} (\mathbf{d}_s \mathbf{m} \cdot (\mathbf{d}_s \times \mathbf{d}_\alpha) + \mathbf{d}_\alpha \mathbf{m}_s \cdot (\mathbf{d}_\alpha \times \mathbf{d}_s)). \quad (5.6)$$

De afstand tussen het raakpunt \mathbf{s}_α en het zoekpunt \mathbf{q} is dan $|\mathbf{s}_\alpha - \mathbf{q}|$. Analoog is zo de afstand te berekenen tussen \mathbf{q} en het raakpunt van $(\mathbf{d}_\beta, \mathbf{m})$ met het zoekvlak. De gezochte minimumafstand h_{opp} is dan het minimum van deze twee afstanden.

5.2 Lijnstukken

Zowel sectie 4.4 als 5.1 vertrekken van een verzameling van rechten, maar in sommige toepassingen zijn lijnstukken of halfrechten meer geschikt. In het voorgaande werden bijvoorbeeld steeds rechten gebruikt als geometrisch object van de fotonstralen voor photonmapping. Zoals werd beschreven in sectie 1.3, hebben deze stralen evenwel een startpunt en leveren ze dus geen bijdrage aan de irradiantie van de oppervlakken die voor dat startpunt liggen. Daarnaast ontvangt ook enkel het eerste oppervlak dat geraakt wordt een bijdrage. Halfrechten of lijnstukken omvatten dus beter alle aspecten van fotonstralen.

Plückercoördinaten kunnen uitgebreid worden om lijnstukken en halfrechten te ondersteunen. Hiertoe voegen we parameters $t_1, t_2 \in \mathbb{R} \cup \{-\infty, +\infty\}$ toe, met $t_1 < t_2$. Lijnstukken lopen dan van $\mathbf{p} + t_1 \mathbf{d}$ tot $\mathbf{p} + t_2 \mathbf{d}$, met $\mathbf{p} = \mathbf{d} \times \mathbf{m}$. Hierin is (\mathbf{d}, \mathbf{m}) het plückercoördinaat van de onderliggende rechte waar het lijnstuk een deel van is. Halfrechten ontstaan wanneer t_1 of t_2 op respectievelijk $-\infty$ en ∞ worden gezet. Figuur 5.3 toont een voorbeeld van een lijnstuk met de bijhorende parameters.



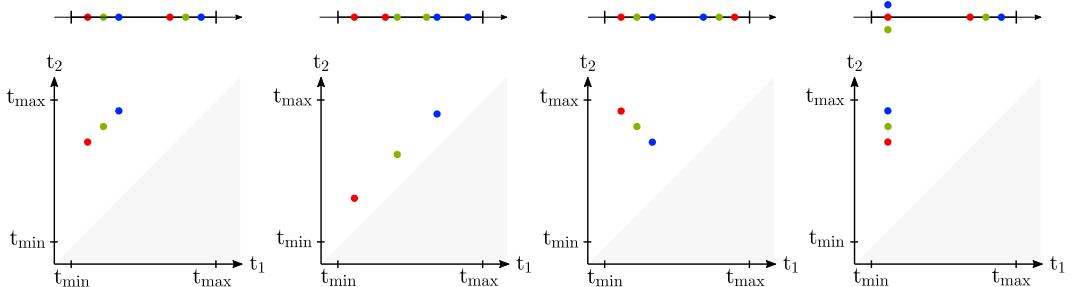
Figuur 5.3: Voorbeeld van een lijnstuk $(\mathbf{d}, \mathbf{m}, t_1, t_2)$

De aanpassing aan de coördinaten wordt ook doorgevoerd in de structuur van de boom. In sectie 4.3 werd voor iedere knoop in de boom een deeldomein gespecificeerd dat alle plückercoördinaten in die tak omvat. Hier wordt de definitie van dat domein uitgebreid met een minimum- en maximumwaarde voor respectievelijk t_1 en t_2 .

Formeel geeft dit

$$\begin{aligned}
 (\mathbf{d}, \mathbf{m}, t_1, t_2) \in V(\mathbf{d}_1, \mathbf{d}_2, \mathbf{m}_1, \mathbf{m}_2, t_{min}, t_{max}) \iff & \mathbf{d} \cdot \mathbf{d}_1 \geq 0 \wedge \mathbf{d} \cdot \mathbf{d}_2 \geq 0 \wedge \\
 & r_1 \leq m_r \geq r_2 \wedge \\
 & \phi_1 \leq m_\phi \geq \phi_2 \wedge \\
 & \theta_1 \leq m_\theta \geq \theta_2 \wedge \\
 & t_{min} \leq t_1 \leq t_{max} \wedge \\
 & t_{min} \leq t_2 \leq t_{max} \wedge \\
 & t_1 \leq t_2.
 \end{aligned} \tag{5.7}$$

Bij constructie kan, naast splitsingen op moment of richting, het domein nu ook de t-waarden gesplitst worden. Deze splitsing is echter niet zo triviaal aangezien de t-intervallen van verschillende lijnstukken kunnen overlappen. Indien dit het geval is, dan is het niet mogelijk om de lijnstukken te partitioneren volgens hun t-parameter zodat ze volledig bevatten zitten in disjuncte t-deelintervallen. Figuur 5.4 toont enkele voorbeelden van verzamelingen van lijnstukken, samen met een bijhorende tweedimensionale plot van deze configuraties waarin t_1 en t_2 als de 2D coördinaten van een punt worden geïnterpreteerd. Enkel het tweede voorbeeld kan opgedeeld worden in disjuncte delen. Wanneer bij de andere voorbeelden een t-interval gekozen wordt dat een lijnstuk omvat, dan zal dit interval altijd ook minstens een deel van een ander lijnstuk bevatten. Merk ook op dat indien een lijnstuk a zich op de plot rechts-onder een ander lijnstuk b bevindt, a omvat wordt door b .



Figuur 5.4: Bovenaan: enkele mogelijke configuraties van drie lijnstukken op een rechte. Onderaan: bijhorende plots van de lijnstukken volgens hun t_1 en t_2 parameters.

Bij een splitsing op t-interval wordt eerst het pivotelement bepaald, welke wordt opgeslagen in de knoop en dus in geen van beide kindknopen ingedeeld zal worden. Deze is op de plot uiterst links-boven te vinden. Vervolgens moeten er dan twee t_1, t_2 intervallen beslist worden, één voor elke kindknoop. Deze zullen elk langst een kant van het domein starten en naar de andere kant groeien totdat beide evenveel lijnstukken bevatten en alle lijnstukken in een kindknoop ingedeeld zijn. Merk op dat, hoewel de resulterende intervallen kunnen overlappen, de individuele lijnstukken

5. EXTENSIES

altijd in slechts één kind worden ingedeeld. Algoritme 2 geeft een overzicht van deze procedure.

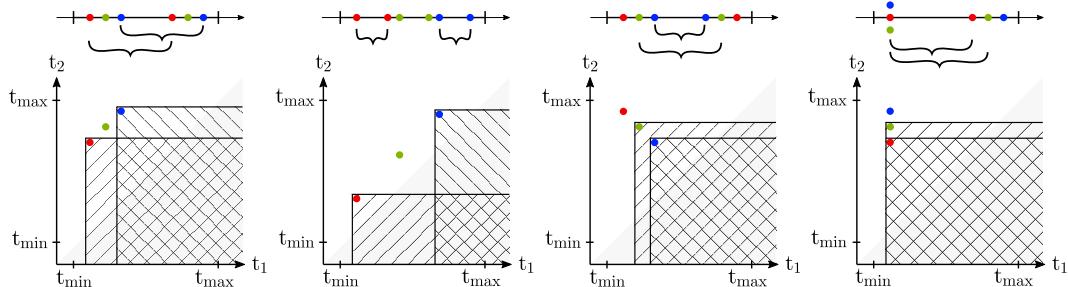
```

splits_t_interval(lijnstukken):
    pivot  $\leftarrow$  verwijder element uit lijnstukken waarvoor afstand van  $(t_1, t_2)$  tot
     $(t_{1\text{-min}}, t_{2\text{-max}})$  het kleinst is
    sorteer lijnstukken op  $t_1$ 
     $c_1t_1 \leftarrow \text{lijnstukken}[0].t_1$ 
     $c_1t_2 \leftarrow \text{lijnstukken}[0].t_2$ 
     $c_2t_1 \leftarrow \text{lijnstukken}[\#\text{lijnstukken} - 1].t_1$ 
     $c_2t_2 \leftarrow \text{lijnstukken}[\#\text{lijnstukken} - 1].t_2$ 
    for  $i = 0 \rightarrow \#\text{lijnstukken}/2$  do
         $\text{links} \leftarrow \text{lijnstukken}[i]$ 
         $\text{rechts} \leftarrow \text{lijnstukken}[\#\text{lijnstukken} - i - 1]$ 
         $c_1t_2 \leftarrow \max(c_1t_2, \text{links}.t_2)$ 
         $c_2t_1 \leftarrow \min(c_2t_1, \text{rechts}.t_1)$ 
         $c_2t_2 \leftarrow \max(c_2t_2, \text{rechts}.t_2)$ 
    indien  $\#\text{lijnstukken}$  oneven is, breidt dan één van de twee intervallen uit om
    dit element te omvatten
    return  $[c_1t_1, c_1t_2], [c_2t_1, c_2t_2]$ 

```

Algoritme 2: Splitsing t-interval

Het resultaat van dit algoritme op de voorbeelden uit figuur 5.4 is te zien in figuur 5.5. De gearceerde zones zijn de t intervallen van elke kindknoop en tonen alle lijnstukken die binnen het interval vallen.



Figuur 5.5: Resultaat van het uitvoeren van algoritme 2 op de voorbeelden uit figuur 5.4

Net zoals in sectie 5.1 geldt ook hier dat de minimumafstandsfunctie f_{\perp} , naar de definitie in sectie 4.2, onveranderd gebruikt kan worden. De kortste afstand van een punt tot een lijnstuk is immers altijd langer of gelijk aan de loodrechte afstand tot de onderliggende rechte waarop het lijnstuk ligt. In sectie 5.1 werd wel de afstandsfunctie van punt tot individuele rechte aangepast naar g_{opp} . Voor lijnstukken wordt dan g_{\perp} opgesteld, hier zodat de loodrechte afstand enkel gebruikt wordt indien de projectie van zoekpunt \mathbf{q} op de onderliggende rechte een t -waarde geeft die binnen het lijnstuk valt. Anders wordt de afstand gebruikt tot het punt met

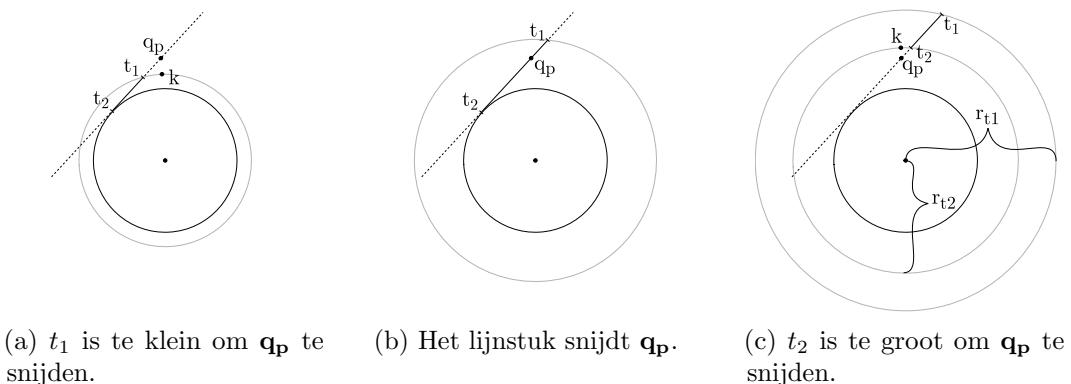
de eerstvolgende t-waarde die wel element is van het lijnstuk. Dit vertaalt zich in

$$\begin{aligned}
 g_{\perp}(\mathbf{m}, \mathbf{d}, t_1, t_2, \mathbf{q}) &= |\mathbf{s} - \mathbf{q}| \\
 \mathbf{s} &= \mathbf{p} + t\mathbf{d} \\
 t &= \min(\max(\mathbf{d} \cdot \mathbf{q}, t_1), t_2) \\
 \mathbf{p} &= \mathbf{d} \times \mathbf{m},
 \end{aligned} \tag{5.8}$$

waarin \mathbf{s} het punt op het lijnstuk dichtst bij \mathbf{q} is en \mathbf{p} het punt waar de onderliggende rechte doorloopt. Indien niet de loodrechte afstand gezocht wordt, maar de afstand tot het dichtste raakpunt van het lijnstuk met een zoekvlak, analoog aan sectie 5.1, dan verandert deze expressie in

$$\begin{aligned}
 g_{opp\perp}(\mathbf{m}, \mathbf{d}, t_1, t_2, \mathbf{q}) &= \begin{cases} |\mathbf{s} - \mathbf{q}| & \text{als } t_1 \leq t \leq t_2 \\ \text{geen raakpunt} & \text{anders} \end{cases} \\
 \mathbf{s} &= \mathbf{p} + t\mathbf{d} \\
 t &= \frac{(\mathbf{q} - \mathbf{p}) \cdot \hat{\mathbf{n}}}{\mathbf{d} \cdot \hat{\mathbf{n}}} \\
 \mathbf{p} &= \mathbf{d} \times \mathbf{m},
 \end{aligned} \tag{5.9}$$

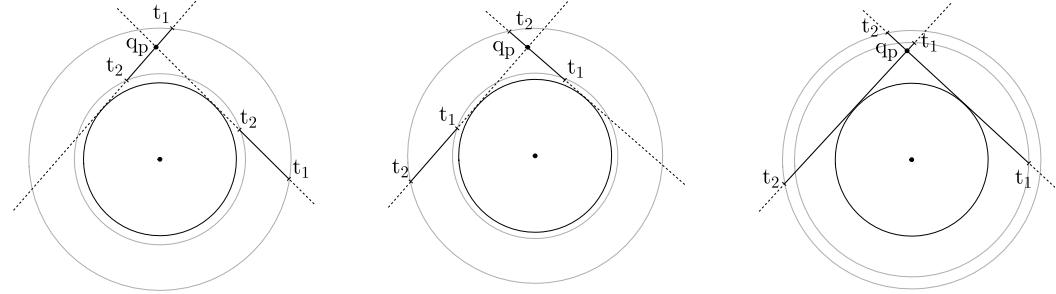
In sectie 5.1 werd minimumafstandsfunctie f_{opp} bepaald voor het zoeken van het dichtste raakpunt bij een punt op een zoekvlak, en ook hier kan een functie f_{\perp} opgesteld worden die het werkelijke minimum geeft bij lijnstukken. Net zoals bij f_{opp} moet voor f_{\perp} ook enkel een functie h_{\perp} geconstrueerd worden die de minimumafstand bij een vaste momentvector bepaalt. Bij sectie 4.4 werden twee optimale richtingsvectoren uitgewerkt waarvan de rechtes door het geprojecteerde zoekpunt \mathbf{q}_p lopen en dus een minimale loodrechte afstand tot het zoekpunt \mathbf{q} hebben. Aangezien in deze sectie met lijnstukken gewerkt wordt, is het mogelijk dat de lijnstukken in het domein te kort zijn of te ver liggen om door \mathbf{q}_p te gaan. Dit wordt gedemonstreerd op figuur 5.6.



Figuur 5.6: Invloed van positionering t waarden op minimum van het linkse lijnstuk. Een gespiegelde situatie is van toepassing op het minimum van het rechtse lijnstuk.

5. EXTENSIES

Verder geldt ook dat, indien t_1 en t_2 allebei positief of negatief zijn, dat er slechts één optimale richtingsvector overblijft. Dit is zichtbaar op figuur 5.7.



(a) $t_1, t_2 \leq 0$, enkel het linkse lijnstuk vormt een optimum.
(b) $t_1, t_2 \geq 0$, enkel het rechtse lijnstuk vormt een optimum.
(c) $t_1 < 0, t_2 > 0$, zowel het linkse als het rechtse lijnstuk geven een optimum.

Figuur 5.7: Invloed van teken van t waarden op minima.

Parameters t_1 en t_2 bepalen, samen met de lengte van \mathbf{m} , hoe dicht of ver punten kunnen liggen van de oorsprong alvorens er een lijnstuk is dat ze snijdt. Deze afstanden genaamd r_{t_1} en r_{t_2} zijn op figuur 5.6 en 5.7 terug te vinden als de grijze cirkels. Ze worden berekend aan de hand van de stelling van Pythagoras, namelijk

$$\begin{aligned} r_{t_1} &= \sqrt{t_1^2 + |\mathbf{m}|^2} \\ r_{t_2} &= \sqrt{t_2^2 + |\mathbf{m}|^2}. \end{aligned} \tag{5.10}$$

De minimumafstandsfunctie voor vaste momentvector kan nu gedefinieerd worden met een gevalsonderscheid. Stel dat $t_1, t_2 \leq 0$, dan is er slechts één richting die resulteert in een afstandsminimum en dus optimaal is. Dan bepaalt t_1 de grootste afstand die het lijnstuk kan raken en t_2 de kleinste. Indien ook $r_{t_1} < |\mathbf{q}_p|$, dan is de optimale richtingsvector diegene waarmee het t_1 uiteinde van het lijnstuk in lijn komt te liggen met \mathbf{q}_p . Dit is te zien op figuur 5.6a, waar de cirkel door alle punten overeenkomend met t_1 het dichtst bij \mathbf{q}_p staat in het punt \mathbf{k} , welke in lijn staat met \mathbf{q}_p . Net zoals in vergelijking (4.6) γ werd berekend zodat de overeenkomstige rechte door \mathbf{q}_p loopt, zo wordt γ hier gekozen zodat het lijnstuk \mathbf{k} raakt. Dit leidt tot $\sin \gamma = \min(\frac{|\mathbf{m}|}{\sqrt{r_{t_1}}}, 1)$, waarmee we dan de optimale richtingsvector \mathbf{d}_b kunnen berekenen zoals in vergelijking (4.7). Analoog is het zo dat, indien $r_{t_2} > |\mathbf{q}_p|$, het t_2 uiteinde van het lijnstuk met kortste afstand tot \mathbf{q}_p ook in lijn ligt met \mathbf{q}_p . Hier geldt dan dat $\sin \gamma = \min(\frac{|\mathbf{m}|}{\sqrt{r_{t_2}}}, 1)$. In de andere gevallen kan het lijnstuk \mathbf{q}_p snijden en wordt γ berekend zoals in vergelijking (4.6).

De afleiding voor γ bij $t_1, t_2 \geq 0$ is hetzelfde, enkel wisselen de rollen van t_1 en t_2 om en is de optimale richtingsvector \mathbf{d}_a . De formulering hiervan is te zien in vergelijking (5.11) en (5.12).

Als $t_1 < 0$ en $t_2 > 0$, zoals in figuur 5.7c, dan zijn er twee lijnstukken met richtingsvectoren \mathbf{d}_a en \mathbf{d}_b die een minimale afstand hebben tot \mathbf{q}_p . Het is mogelijk

dat t_1 niet even groot is als t_2 , en dan hebben de optimale richtingsvectoren een verschillende hoek met \mathbf{q}_p . Bijgevolg zijn er dus twee waarden γ_a en γ_b voor \mathbf{d}_a en \mathbf{d}_b respectievelijk. Aangezien $t_1 < 0$ en $t_2 > 0$, kunnen de lijnstukken niet ‘te ver’ liggen om \mathbf{q}_p te snijden, zoals het geval was in figuur 5.6c. Er moet dan enkel nog gecontroleerd worden of t_2 dan wel t_1 te klein zijn voor intersectie. Deze afleiding wordt samengevat in vergelijkingen (5.11) en (5.12),

$$\sin \gamma_b = \begin{cases} \text{ongedefinieerd} & \text{als } t_1 \geq 0 \wedge t_2 \geq 0 \\ \min(|\mathbf{m}|/\sqrt{r_{t1}}, 1) & \text{als } r_{t1} < |\mathbf{q}_p| \\ \min(|\mathbf{m}|/\sqrt{r_{t2}}, 1) & \text{als } r_{t2} > |\mathbf{q}_p| \\ \min(|\mathbf{m}|/|\mathbf{q}_p|, 1) & \text{anders} \\ \min(|\mathbf{m}|/\sqrt{r_{t1}}, 1) & \text{als } r_{t1} < |\mathbf{q}_p| \\ \min(|\mathbf{m}|/|\mathbf{q}_p|, 1) & \text{anders} \end{cases}, \quad (5.11)$$

$$\sin \gamma_a = \begin{cases} \min(|\mathbf{m}|/\sqrt{r_{t2}}, 1) & \text{als } r_{t2} < |\mathbf{q}_p| \\ \min(|\mathbf{m}|/\sqrt{r_{t1}}, 1) & \text{als } r_{t1} > |\mathbf{q}_p| \\ \min(|\mathbf{m}|/|\mathbf{q}_p|, 1) & \text{anders} \\ \text{ongedefinieerd} & \text{als } t_1 \leq 0 \wedge t_2 \leq 0 \\ \min(|\mathbf{m}|/\sqrt{r_{t2}}, 1) & \text{als } r_{t2} < |\mathbf{q}_p| \\ \min(|\mathbf{m}|/|\mathbf{q}_p|, 1) & \text{anders} \end{cases}. \quad (5.12)$$

Nadat \mathbf{d}_a , \mathbf{d}_b of beide gevonden zijn, kunnen deze ingeperkt worden zoals beschreven in formule (4.9). Vervolgens kan dan de minimumafstand berekend worden door het minimum te nemen van vergelijking (5.8) toegepast op de gevonden lijnstukken.

Hoofdstuk 6

Resultaten

In dit hoofdstuk worden de performantiekarakteristieken van plückerbomen besproken en gestaafd met experimentele data. Het is hierbij interessant om te kijken naar de tijd- en ruimtecomplexiteit van de belangrijkste operaties. Concreet zijn dit de constructie van de datastructuur, het toevoegen of verwijderen van elementen en de zoekoperaties.

De tijd- en ruimtecomplexiteit van de constructie is hetzelfde als bij klassieke kd-bomen. Tijdens de constructie wordt recursief iedere tak gesplitst door de verzameling op te delen op basis van een heuristiek. De exacte tijdscomplexiteit hangt hierbij af van de gebruikte heuristiek, maar in de meeste gevallen moet er dan bij iedere splitsing de verzameling van rechten in de tak gesorteerd worden. Dit komt dan neer op een tijdscomplexiteit van $\mathcal{O}(n \log^2 n)$ met n het aantal rechten, al bestaan er ook algoritmes die $\mathcal{O}(n \log n)$ halen [27]. Aangezien net zoals bij kd-bomen ieder element wordt voorgesteld door één knoop in de boom, en de knopen een constante grootte hebben, is de ruimtecomplexiteit van plückerbomen $\mathcal{O}(n)$.

Wat betreft het toevoegen of verwijderen van elementen is de complexiteit ook hetzelfde als bij binaire bomen. Iedere rechte komt overeen met precies één knoop in de boom, en deze operaties vereisen dan dat in de boom gezocht wordt waar deze knoop zich moet bevinden. Anders dan in de zoekalgoritmes beschreven in hoofdstuk 4 en 5, wordt in deze zoekoperatie een rechte vergeleken met de knopen van de boom, en kan op iedere niveau meteen beslist worden in welke kindknoop het coördinaat moet zitten. Dit geeft dan steeds een tijdscomplexiteit van $\mathcal{O}(\log n)$, indien bij aanpassingen de boom altijd gebalanceerd blijft [24].

6.1 Performantie zoekopdrachten

De laatste operaties op plückerbomen zijn zoekopdrachten, en de rest van dit hoofdstuk zal hier verder op ingaan. Vooreerst wordt de performantie bekeken van het zoeken op de k rechten met dichtste loodrechte afstand tot een punt, als beschreven in hoofdstuk 4. In sectie 4.7 werden verschillende heuristieken voorgesteld, waarvan in deze sectie eerst ‘afwisselend’ wordt getest aangezien deze het meest eenvoudig is.

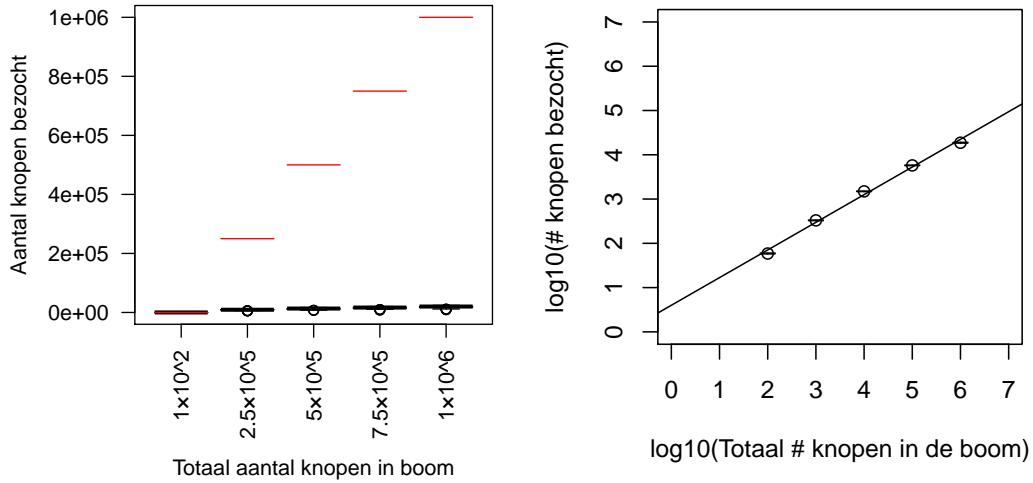
6. RESULTATEN

Het experiment wordt als volgt uitgevoerd. Genereer een willekeurig punt \mathbf{q} in \mathbb{R}^3 en een verzameling S van N willekeurige rechten. De momentvectoren van de rechten hebben een maximumlengte van 100, zodat alle rechten een afstand tot de oorsprong hebben van 100 of minder. Analoog wordt \mathbf{q} gekozen zodat de afstand tot de oorsprong kleiner of gelijk is aan 100. Construeer een plückerboom met de rechten uit S en zoek hierin de dichtste buur tot \mathbf{q} . Hierbij wordt het aantal knopen geteld dat bezocht worden. Herhaal dit proces honderd keer zodat er verschillende monsters beschikbaar zijn voor bomen met grootte N . Herhaal ten slotte deze procedure voor stijgende N .

Het resultaat van dit experiment is te zien in figuur 6.1. Figuur 6.1a is een grafiek waarop het aantal bezochte knopen uitgeplot is in boxplots voor lineair stijgende N . De figuur toont dat het aantal bekeken knopen ver onder de totale grootte van de boom ligt, en dat de verhouding tussen deze mogelijks sublineair is. De precieze relatie is betere te zien op figuur 6.1b, welke gelijkaardige data uitzet op een dubbellogaritmische grafiek. Op zulke grafieken wordt een monoom $y = ax^b$ geplot als $\log y = b \log x + \log a$. Indien de relatie tussen het aantal bezochte knopen en de grootte van de boom N polynomiaal is, dan zal deze bij benadering een rechte vormen op de plot waarbij de richtingscoëfficiënt de macht b aangeeft. Voor figuur 6.1b werden zoekopdrachten uitgevoerd waarbij N stijgende machten van 10 aannam. Hiervan is dan voor elke N het gemiddelde aantal bezochte knopen uitgezet op de grafiek. Een lineaire regressie op deze punten geeft een richtingscoëfficiënt van 0.608, wat overeenkomt met een tijdscomplexiteit van $\mathcal{O}(n^{0.608})$.

Deze coëfficiënt is geverifieerd door honderd keer de berekening opnieuw uit te voeren met datasets gegenereerd met *bootstrap sampling* [7]. Dit houdt in dat uit een dataset van m elementen er m willekeurige datapunten gekozen worden, met teruglegging. Dit resulteert in een standaarddeviatie van 0.0017 op de exponent.

Aangezien traditionele kd-bomen een tijdscomplexiteit van $\mathcal{O}(\log n)$ hebben bij het zoeken van de dichtste buur [3], presteert de plückerboom hier slechter, al gaat het hier wel om een ander type data dat doorzocht moet worden. Zoals hoofdstuk 2 reeds uitlegde biedt een efficiënte datastructuur voor rechten mogelijkheden die niet beschikbaar zijn in klassieke kd-bomen.



(a) Lineaire grafiek. De rode lijnen geven de totale grootte van de boom aan.
(b) Dubbellogaritmische grafiek met lineaire regressie.

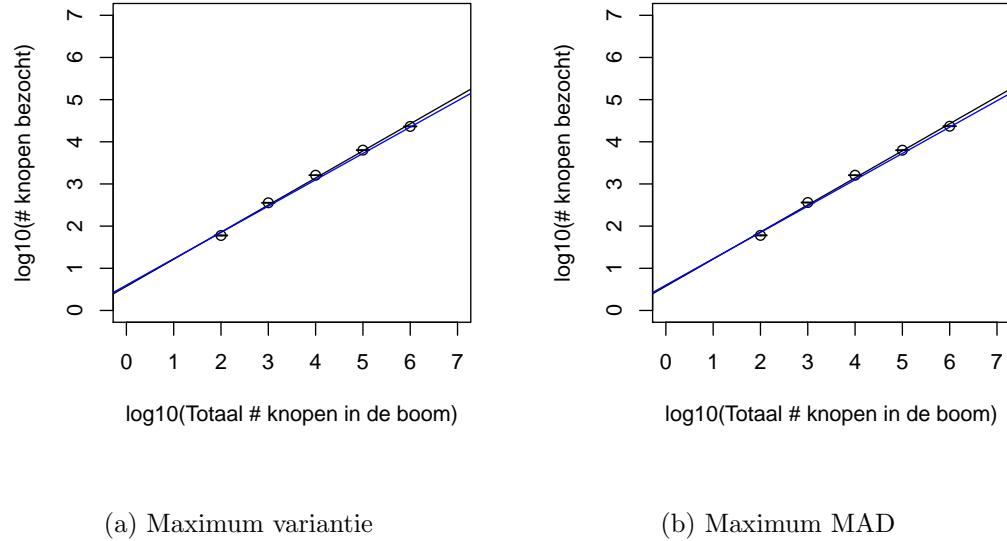
Figuur 6.1: Aantal bezochte knopen voor dichtste-buur zoekoperaties op basis van loodrechte afstand in plückerbomen met stijgende grootte gebouwd met afwisselende splitsingstypes. De relatie is sublineair, met een complexiteit van $\mathcal{O}(n^{0.608})$.

6.2 Alternatieve heuristieken

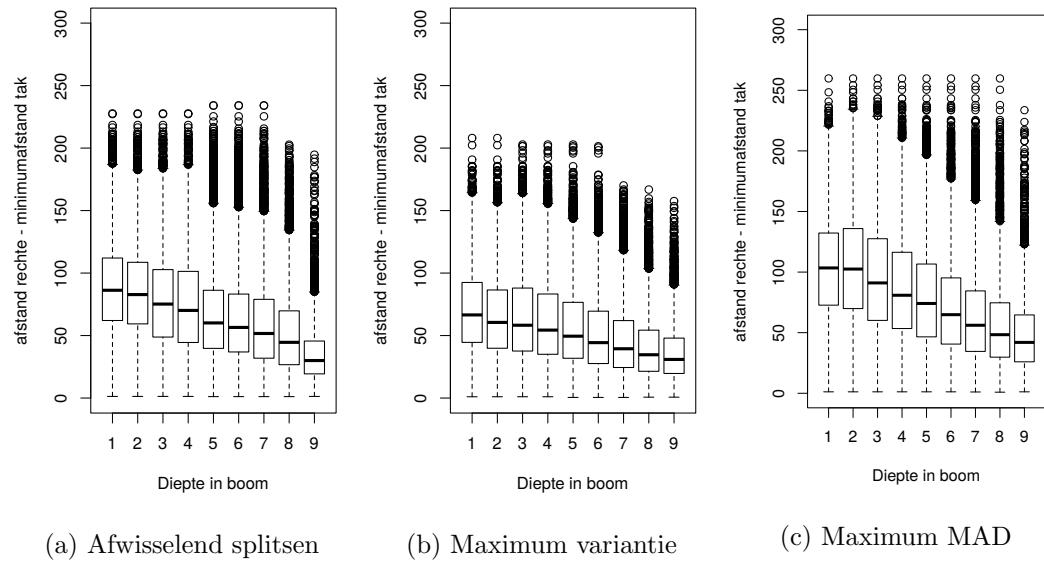
Sectie 4.7 specificeert verder nog twee alternatieve heuristieken: maximale variantie en maximale MAD. Figuur 6.2 vergelijkt hun performantie met het afwisselend splitsen in hetzelfde experiment als beschreven in sectie 6.1. Op de figuur is te zien dat de performantie lichtjes slechter is met een exponent van 0.642 ± 0.0025 en 0.642 ± 0.0020 voor respectievelijk maximum variantie en maximum MAD.

Aangezien de rechten willekeurig en uniform werden gegenereerd, is afwisselend splitsen hier een goede heuristiek, aangezien de rechten ongeveer evenveel variatie vertonen over alle assen. De alternatieve heuristieken resulteren in een boom waarvan de takken een minimumafstand tot het zoekpunt hebben die minder goed aansluit bij de minimumafstand van de rechten die in die tak zitten, wat de slechtere performantie veroorzaakt. Takken met veel verafgelegen rechten die desondanks toch een kleine minimumafstand hebben, kunnen moeilijker uitgesloten worden tijdens de zoekoperatie en vertragen dus het algoritme. Dit wordt geïllustreerd op figuur 6.3, wat een grafiek is van het verschil in iedere tak tussen de afstanden tot het zoekpunt van de beschouwde rechten in die tak en de minimumafstand van het domein van de tak. Hoe dichter en hoe sneller dit nul benadert, hoe beter de subdomeinen van de takken aansluiten bij de rechten die ze bevatten en hoe meer takken gesnoeid kunnen worden wat de prestaties verbetert. De figuur toont dan ook dat de alternatieve heuristieken een lichtjes minder sterke daling vertonen dan afwisselend splitsen.

6. RESULTATEN



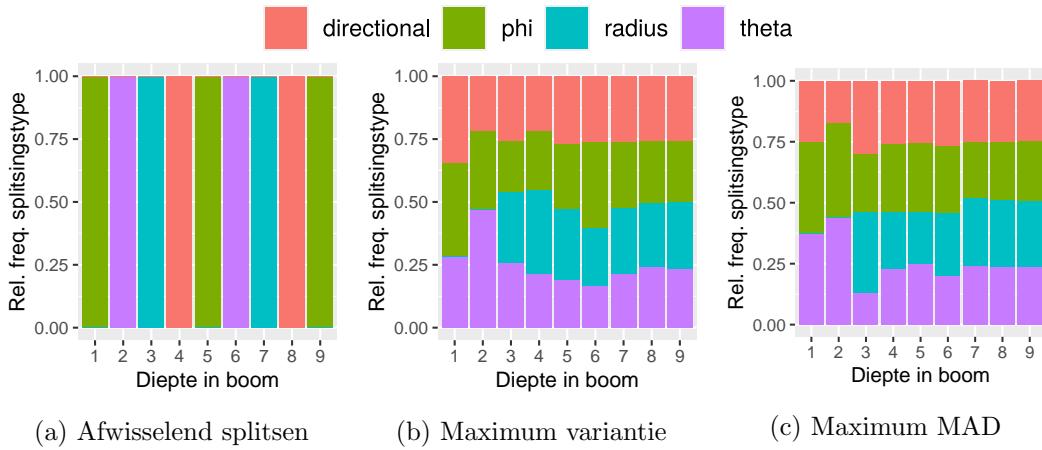
Figuur 6.2: Performantiegrafieken van alternatieve heuristieken. De blauwe lijn is de functie die gevonden werd in figuur 6.1b. Het aantal bezochte knopen is hier ongeveer gelijk als bij het experiment van sectie 6.1.



Figuur 6.3: Grafiek van hoe nauw takken aansluiten bij de rechten die ze omvatten. Merk op dat afwisselend splitsen sneller daalt dan de twee alternatieven.

Figuur 6.4 visualiseert het verschil tussen de bomen geconstrueerd met de verschillende heuristieken. Voor ieder diepteniveau is uitgeplot wat de verhouding is van de splitsingstypes gebruikt voor de knopen op dat niveau. Maximum variantie en

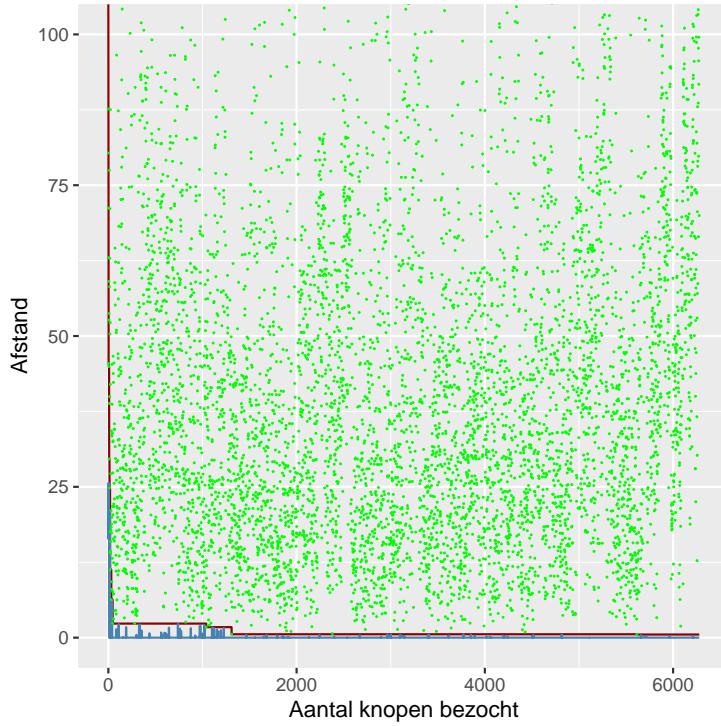
maximum MAD geven hier gelijkaardige resultaten, maar splitsen pas vanaf diepte 3 op radius en hebben in deze dimensie dus iets minder splitsingen. Dit betekent dat in de resulterende boom de domeinen van de knopen een iets grotere radiusdimensie zullen hebben, wat een lagere minimumafstand waarschijnlijk maakt, en dus de performantie negatief beïnvloed. Verder splitsen ze ongeveer evenveel op iedere dimensie, wat in lijn ligt met de uniformiteit van de data.



Figuur 6.4: Relatieve frequentie van elk splitsingstype op iedere diepte in de boom voor de verschillende heuristieken. In figuren 6.4b en 6.4c zijn iets minder splitsingen op radius te vinden.

Indien de toepassing vereist dat de gevonden rechten dichtbij liggen, maar niet dat ze strikt de dichtstbijzijnde zijn, dan is het mogelijk om een benaderend resultaat te bekomen door de zoekoperatie vroegtijdig te stoppen. Figuur 6.5 toont de progressie van de kortste afstand die tot nu toe gevonden is naarmate het algoritme de boom doorzoekt. Er is te zien dat er met weinig iteraties reeds een relatief dichte rechte gevonden wordt, maar dat er daarna een trage convergentie naar het strikte minimum volgt.

6. RESULTATEN



Figuur 6.5: Progressie van de afstand van de dichtste beschouwde rechte tot het zoekpunt naarmate de boom doorzocht wordt. De groene stippen zijn de afstand van de rechte die op dat moment bekijken is. De rode lijn geeft de beste afstand die op dat moment gevonden is. De blauwe lijn geeft de minimumafstand van het domein in de huidige knoop tot het zoekpunt. Merk op dat er reeds met weinig iteraties een dichtbijzijnd element gevonden wordt, maar er daarna nog veel iteraties nodig zijn om het absoluut dichtste te vinden.

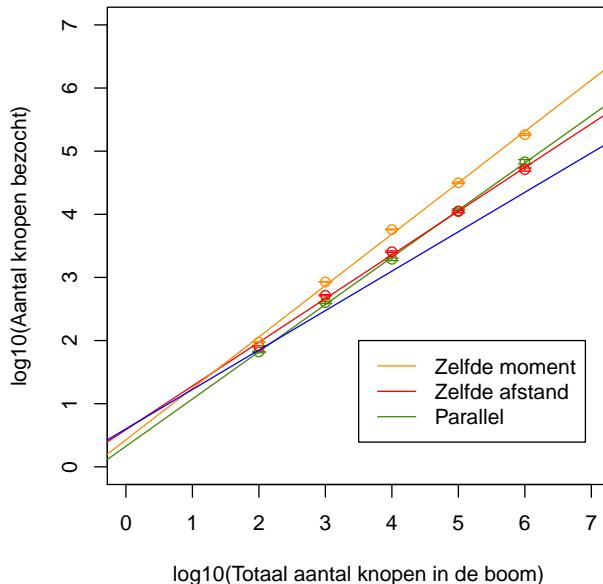
6.3 Invloed invoerdata

Tot nu toe werden de zoekoperaties steeds uitgevoerd op een verzameling van willekeurige rechten. Dit is een gunstige situatie wanneer er afwisselend gesplitst wordt, aangezien de rechten dan uniform verdeeld zijn in de ruimte. Figuur 6.6 toont het resultaat dat verkregen wordt indien hetzelfde experiment van sectie 6.1 wordt uitgevoerd, maar dan voor andere verzamelingen van rechten.

- **Zelfde afstand:** De loodrechte afstand van de rechten tot de oorsprong is hetzelfde. Dit betekent dat de richting van de moment- en richtingsvector willekeurig is, maar de lengte van de momentvector is identiek voor alle rechten.
- **Parallel:** de rechten gaan door willekeurige punten in de ruimte, maar liggen allemaal parallel aan elkaar. Dit beperkt de keuze van de momentvector tot een vlak, en legt de richtingsvector vast.

- Zelfde moment: De momentvector is volledig gelijk bij alle rechten. De richtingsvector is willekeurig, maar ligt noodzakelijk steeds op dezelfde cirkel loodrecht aan de momentvector.

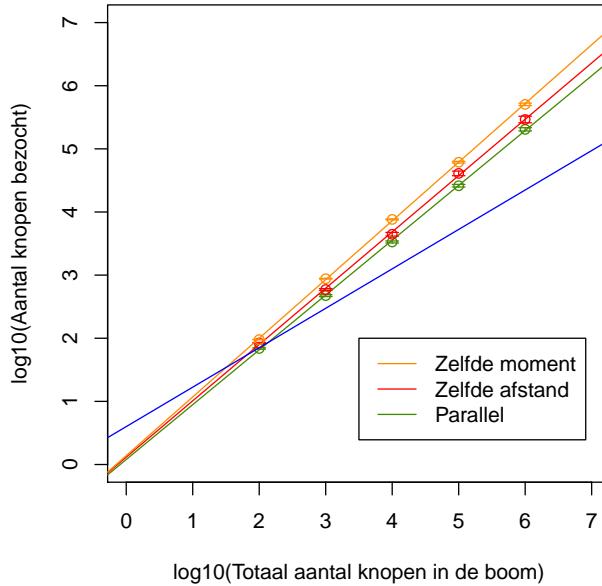
De figuur toont dat de performantie van het algoritme daalt bij deze situaties omdat de rechten dimensionaliteit verliezen vergeleken met volledig willekeurige rechten. De exponenten voor zelfde afstand, parallel en zelfde moment zijn respectievelijk 0.692 ± 0.005 , 0.748 ± 0.007 en 0.814 ± 0.004 tegenover 0.608 ± 0.002 bij het eerder uitgevoerde experiment van sectie 6.1. De slechtere prestatie wordt veroorzaakt door het gebruik van de heuristiek die afwisselend splitst, welke hier dus splitsingen maakt in dimensies zonder variantie. De onderlinge verschillen tussen de experimenten komt doordat ze een stijgend aantal dimensies beperken. Hoe lager de dimensionaliteit van de rechten, hoe slechter de performantie. Een betere heuristiek zou dit oplossen door hier rekening mee houden en enkel de waardevolle splitsingen te kiezen.



Figuur 6.6: Performantiegrafiek van zoekopdrachten op plückerbomen van rechten met verlaagde dimensionaliteit, gebouwd met afwisselend splitsen. Deze heuristiek presteert hier minder goed, aangezien er splitsingen worden gemaakt op dimensies waarin de data geen variantie vertoont.

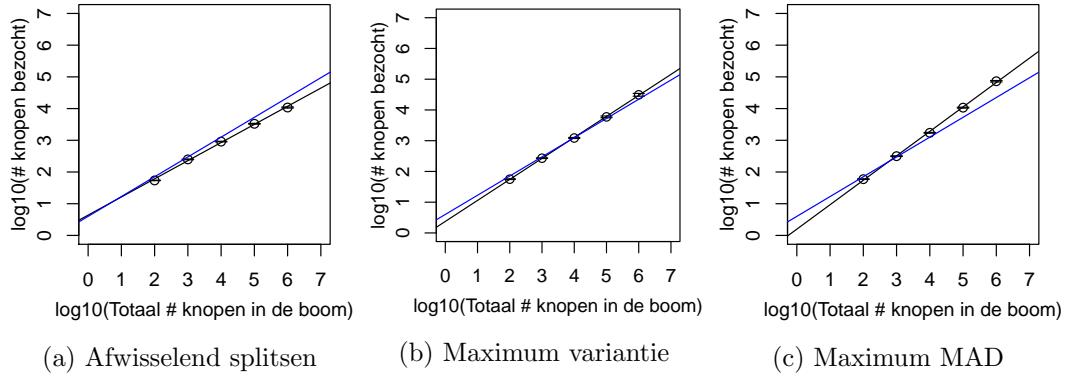
In figuur 6.7 werd hetzelfde experiment uitgevoerd, maar de boom werd gebouwd op basis van de maximum variantie. Ook hier presteert deze heuristiek slechter dan het afwisselend splitsen, ondanks dat de rechten nu niet meer uniform verdeeld liggen. Mogelijk heeft dit te maken met de vorm van de subdomeinen en hoeveel subdomeinen zo gesneden worden door de minimumzone bij een arbitraire zoekopdracht.

6. RESULTATEN

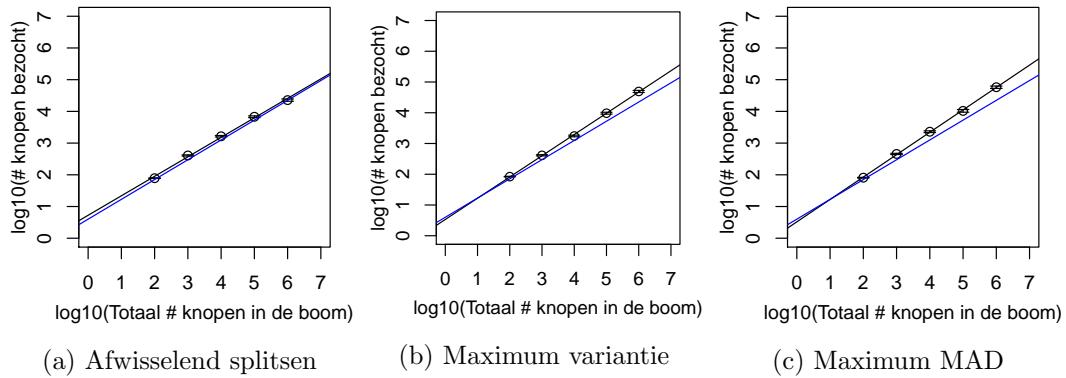


Figuur 6.7: Performantiegrafiek van zoekopdrachten op plückerbomen van rechten met verlaagde dimensionaliteit, gebouwd met de maximum variantie heuristiek. Deze heuristiek presteert nog slechter dan afwisselend splitsen.

De vorige zoekopdrachten gebruikten kunstmatig genereerde verzamelingen van rechten. Figuren 6.8 en 6.9 tonen de performantie van het zoeken van de buur met kortste loodrechte afstand in meer realistische photonmappingscènes. De scènes bevatten een puntlichtbron, een object met een perfect speculair en refractief materiaal en een diffuse ontvanger. De rechten in deze experimenten zijn de fotonstralen die na speculaire transmissie de diffuse ontvanger zullen raken, en daar een brandpunt zullen creëren. Sectie 6.5 toont afbeeldingen van deze scenes. Het experiment bij figuur 6.8 werd uitgevoerd op een scène met een glazen bol op een diffuus oppervlak. De lichtbron, die naast en boven de bol staat, creëert dan een sterk geconcentreerd brandpunt op het oppervlak. Voor figuur 6.9 werd een scène gebruikt waarin de fotonstralen na transmissie door een golfoppervlak van water lijnen vormen op de diffuse ontvanger. Met deze scènes werden experimenten uitgevoerd die gelijkaardig zijn aan die beschreven in sectie 6.1. Hiervoor werden willekeurig N rechten uit de volledige verzameling van fotonstralen gekozen, waarmee dan de plückerboom werd opgebouwd en de rechte met kortste loodrechte afstand opgezocht. Dit werd honderd maal herhaald voor iedere geteste waarde van N .



Figuur 6.8: Performantie bij het zoeken van de buur met dichtste loodrechte afstand in scène met glazen bol. De resultaten komen grotendeels overeen met die uit sectie 6.1



Figuur 6.9: Performantie bij het zoeken van de buur met dichtste loodrechte afstand in scène met golven. De resultaten komen overeen met die uit sectie 6.1

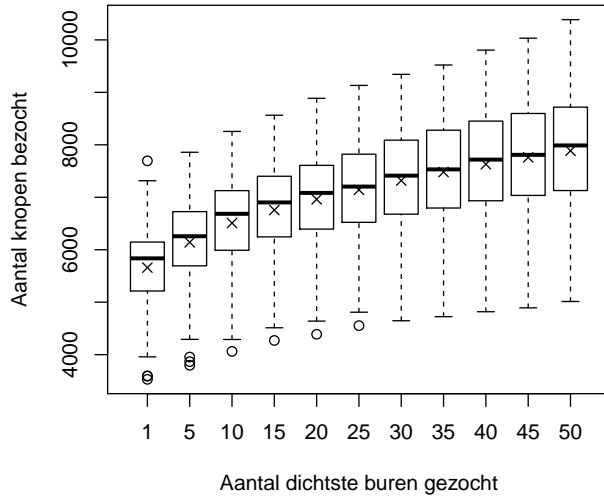
De resultaten van deze experimenten liggen in lijn met de bevindingen uit de vorige secties. Afwisselend splitsen resulteert bij figuur 6.8a in performantie die iets beter is dan wat te zien was in sectie 6.1. Bij figuur 6.9a is het aantal bezochte knopen zeer vergelijkbaar met de resultaten uit sectie 6.1. Dit kan verklaard worden doordat de fotonstralen in de golfscène meer verspreid liggen en meer diverse richtingen hebben, waardoor deze meer gelijkaardig is aan de dataset met willekeurige rechten. De heuristieken op basis van maximum variantie en maximum MAD presteren ook hier slechter dan afwisselend splitsen.

6.4 Alternatieve zoekopdrachten

Hoewel in de vorige experimenten steeds de enkele dichtste buur werd gezocht, is het ook mogelijk om de k dichtste buren te zoeken. Figuur 6.10 toont het effect op het

6. RESULTATEN

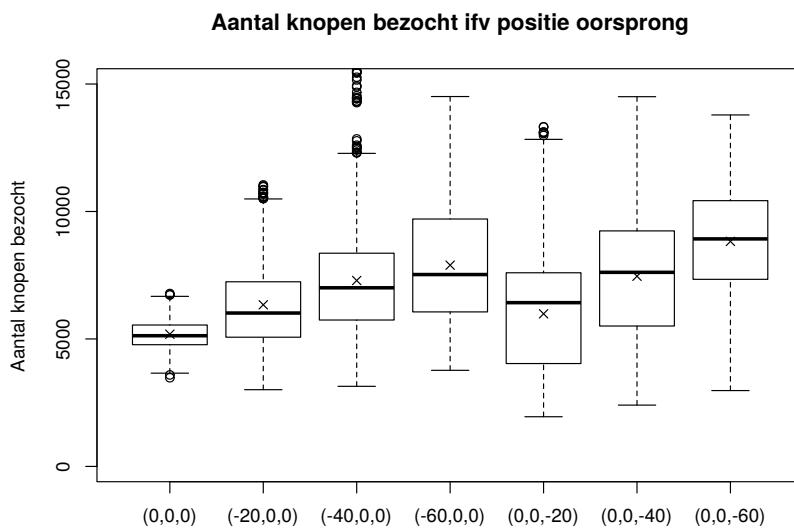
aantal bezochte knopen van een stijgend aantal k bij het zoeken in een plückerboom met 10^5 willekeurige lijnen. Op de figuur is te zien dat bij een stijgende k er meer rechten bekeken worden. Een stijging van k van 5 naar 10 geeft een stijging in het gemiddeld aantal bezochte knopen van 6140 naar 6508, oftewel stijging van $\sim 6\%$. Relatief tot het totaal aantal rechten is dit een stijging van 0.0037% . Wanneer k stijgt van 10 naar 15, gaat het gemiddeld aantal bezochte knopen van 6508 naar 6755, oftewel een kleinere stijging van $\sim 3.8\%$. Dit is ook op de figuur te zien, de stijging neemt af naarmate k stijgt. Intuïtief kan dit als volgt verklaart worden. Bij het zoeken van iedere additionele buur $k + 1$ bestaat de kans dat het algoritme deze al is tegengekomen bij het zoeken naar dichtste k buren. De additionele kost ontstaat dan wanneer dit niet het geval is. Naarmate k stijgt, neemt het aantal bezochte knopen toe, en stijgt dus ook de kans dat $k + 1$ al gevonden is. Wanneer $k = n$, dan worden alle n knopen één keer bekeken.



Figuur 6.10: Invloed van stijgende k bij het zoeken naar de k buren met dichtste loodrechte afstand in een plückerboom met 10^5 willekeurige rechten. Een stijging in k betekent ook een stijging van het aantal bezochte knopen. De kruisjes geven het gemiddelde aan.

De keuze van de oorsprong heeft ook een invloed op de performantie van het zoekalgoritme. Dit is te zien in figuur 6.11, waar de dichtste buren van 100 zoekpunten werden gezocht in een verzameling van 10^5 willekeurige rechten. Vervolgens werden deze rechten allemaal getransleerd, de plückerboom opnieuw geconstrueerd en de test opnieuw uitgevoerd. Dit werd dan herhaald voor een aantal verschillende translaties. Na translatie neemt de uniformiteit van de coördinaten af, wat een gelijkaardig effect geeft als te zien was in figuur 6.6. Het verschil tussen translatie op de x -as en de z -as kan verklaard worden door het aantal en de vorm van de sectoren waarin de rechten terecht komen na translatie. Herinner dat voor iedere rechte \mathbf{p} , het punt

op de rechte het dichtst bij de oorsprong, de normaalvector is van het vlak met de bijkomende momentvector. Wanneer dit punt bijvoorbeeld verplaatst in de richting van de x -as, dan zal de momentvector meer in het yz -vlak komen liggen. Analoog geldt bij een zoekoperatie dat de sectoren die het vlak bevatten dat loodrecht staat op de translatie-as, de laagste minimumafstand zullen hebben en dus eerst bezocht worden. Een translatie volgens de z -as zal dan de rechten en zoekoperaties focussen op de sectoren rond de evenaar. Een translatie volgens de x -as focust deze op een deel van de sectoren op de evenaar, en de sectoren op de polen, welke een verschillende vorm hebben. Analoog aan wat te zien was in figuur 6.6 zou ook hier een betere heuristiek dit effect kunnen temperen.

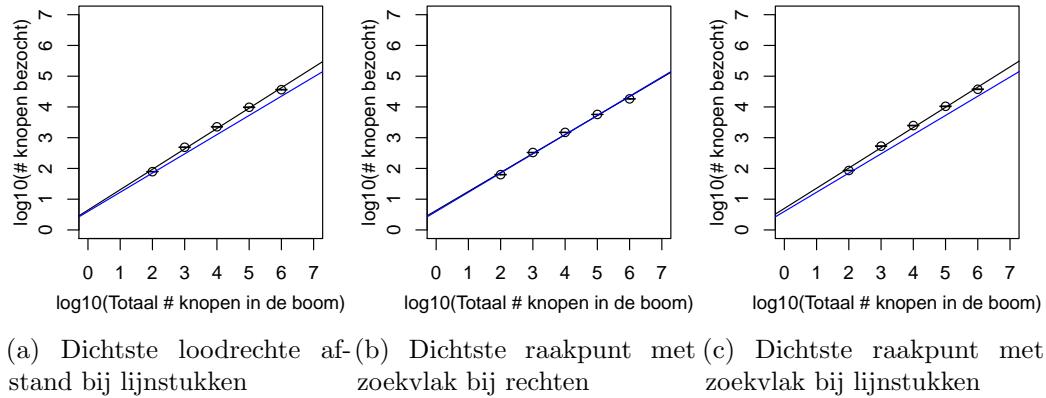


Figuur 6.11: Invloed van positionering van de oorsprong op de zoekperformantie. Het centreren van de oorsprong tussen de rechten geeft de beste performantie.

Alle vorige experimenten gingen uit van de loodrechte afstand tussen rechte en zoekpunt. Figuur 6.12 toont de performantie van de extensies gedefinieerd in hoofdstuk 5. Voor figuur 6.12a werd gezocht in een plückerboom met lijnstukken naar het element met de kortste loodrechte afstand tot het zoekpunt. Om een willekeurig lijnstuk te genereren werd eerst een willekeurige rechte gecreëerd, waarna dan twee willekeurige waarden in $[-100; 100]$ werden bepaald waarvan de kleinste t_1 werd en de grootste t_2 . Figuur 6.12b toont de performantie bij een zoekopdracht naar de rechte die resulteert in het dichtste raakpunt op een zoekvlak. Ten slotte werd voor figuur 6.12c gezocht naar het lijnstuk dat het dichtste raakpunt op een zoekvlak geeft. Verder werden de experimenten uitgevoerd op dezelfde wijze als in sectie 6.1. Op de figuur is te zien dat het aantal bezochte knopen gelijkaardig is aan wat bepaald werd in figuur 6.1b. De gevonden exponenten zijn respectievelijk 0.663 ± 0.003 , 0.616 ± 0.002 en 0.657 ± 0.003 . De zoekoperaties met lijnstukken lijken hier iets

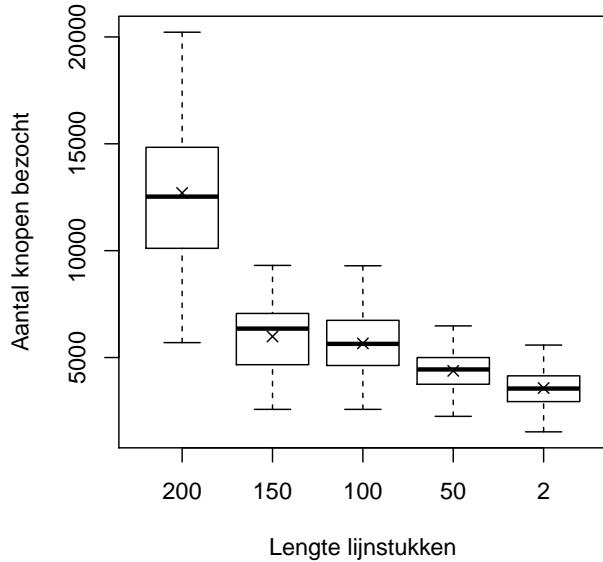
6. RESULTATEN

minder efficiënt te zijn dan die met rechten. Mogelijks wordt dit veroorzaakt doordat bij een splitsing van het t -interval, de twee deelintervallen kunnen overlappen.



Figuur 6.12: Performantie bij zoekoperaties uit hoofdstuk 5. Het aantal bezochte knopen bij het zoeken op dichtste raakpunt is nagenoeg identiek aan de resultaten voor dichtste loodrechte afstand. Zoeken op lijnstukken presteert iets slechter, met 0.663 en 0.657 als exponenten tegenover 0.608.

Figuur 6.12 toonde hoe zoekoperaties op lijnstukken iets slechter kunnen presteren dan bij rechten, maar dit hangt ook af van de lengte van de lijnstukken. Figuur 6.13 toont dit aan als volgt. Voor verzameling van 10^5 willekeurige lijnstukken werd ieder lijnstuk geschaald rond zijn middelpunt zodat de lengte overeen kwam met een constante waarde. Vervolgens werden daarna de lijnstukken met dichtste loodrechte afstand gezocht voor een vaste verzameling van 100 willekeurige zoekpunten. De zoekpunten werden zo gekozen dat hun afstand tot de oorsprong kleiner of gelijk is aan 100. Dit proces werd uitgevoerd voor verschillende lengtes van de lijnstukken. In de figuur is te zien dat bij zoekoperaties op kortere lijnstukken er minder knopen bezocht worden dan bij langere lijnstukken. Enerzijds kunnen kortere lijnstukken netter opgedeeld worden op basis van t waarden in (bijna) disjuncte stukken. Anderzijds zullen de subdomeinen ook een korts t interval krijgen, wat gemiddeld gezien een grotere minimumafstand geeft voor de bijhorende takken, welke dan gemakkelijker uitgesloten kunnen worden.

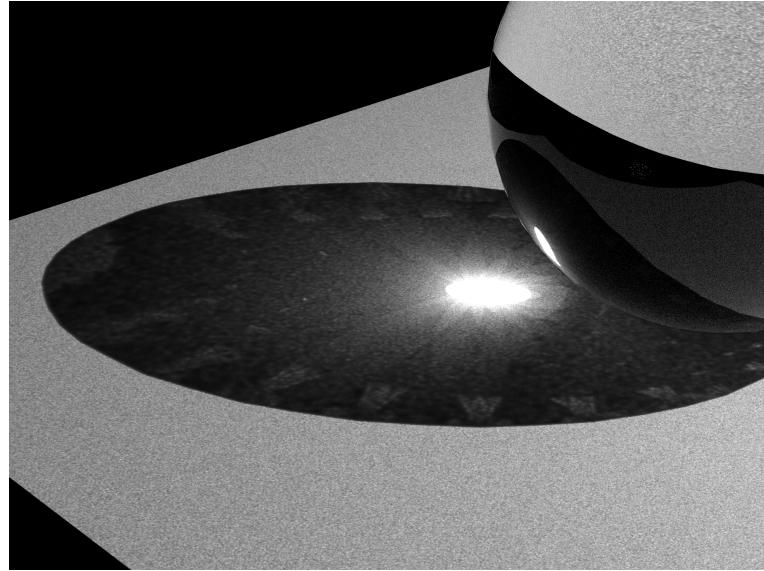


Figuur 6.13: Invloed van de lengte van lijnstukken op de zoekperformantie. Bij kortere lijnstukken worden er minder knopen in de boom bezocht.

6.5 Photonmapping

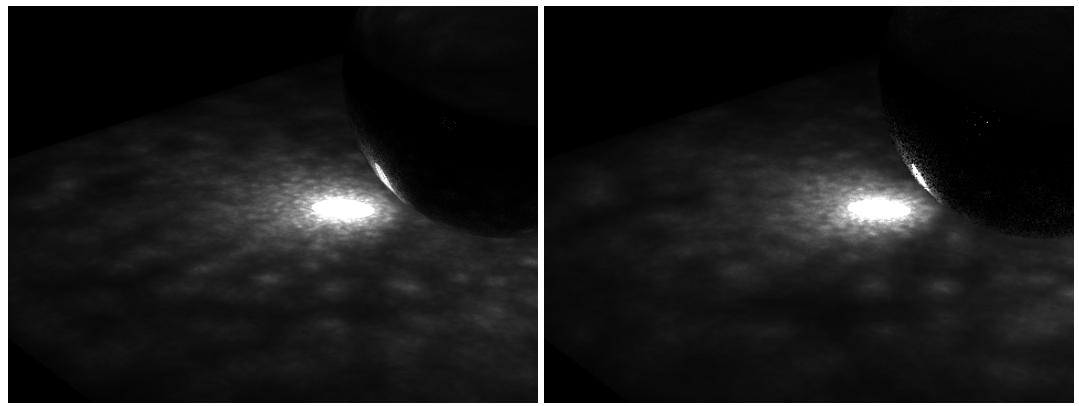
In deze sectie wordt het gebruik van plückerbomen in photonmapping grafisch bekeken. Zoals vermeld in de vorige sectie wordt hier gebruik gemaakt van twee scènes. De eerste scène bestaat uit een glazen bol op een diffuus oppervlak. Figuur 6.14 toont het resultaat van klassieke photonmapping op deze scène. Hierbij werd een photonmap met 35 miljoen fotonen gebruikt. Bij de PDE-stap werden telkens de 20 dichtstbijzijnde fotonen opgezocht. Door refractie ontstaat er een intens brandpunt in de schaduw van de bol.

6. RESULTATEN



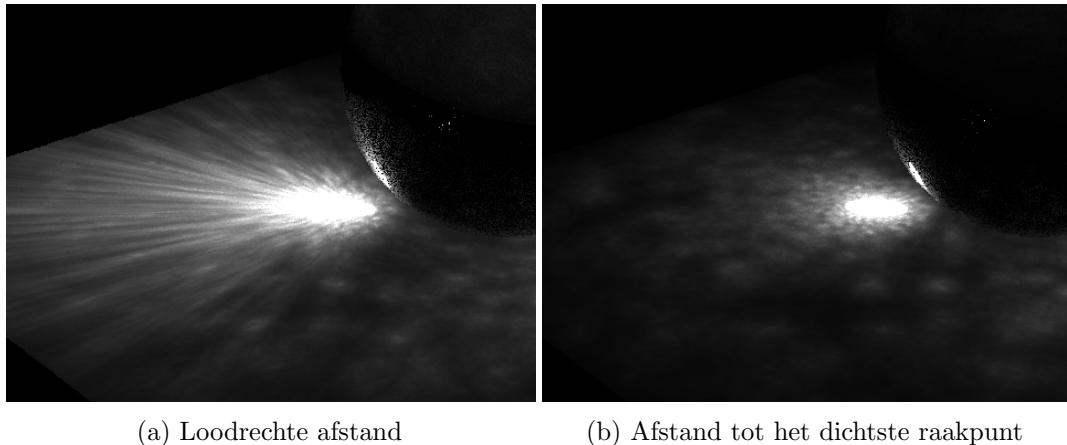
Figuur 6.14: Referentieafbeelding van bolscène gemaakt met klassieke photonmapping.
(~ 35 miljoen fotonen in scène, 20 fotonen in PDE)

Voor figuur 6.15 werd dezelfde scène gerenderd met zowel klassieke photonmapping als photonmapping met fotonstralen. Hierbij werd in de PDE-stap gebruik gemaakt van plückerbomen om ook hier de 20 dichtste fotonen, of dus fotonstralen met dichtste raakpunten aan het zoekvlak, te vinden. In deze afbeelding werden enkel de speculaire fotonen gebruikt om duidelijk het brandpunt in beeld te brengen. De ‘vlekken’ die ontstaan aan de rand zijn het gevolg van een verlaging van het aantal fotonen in de scène om kortere rendertijd te bekomen. In de afbeelding is te zien dat, afgezien van kleine variaties omwille van het gebruik van stochastische variabelen, de afbeeldingen identiek zijn wat duidt op een correcte werking.



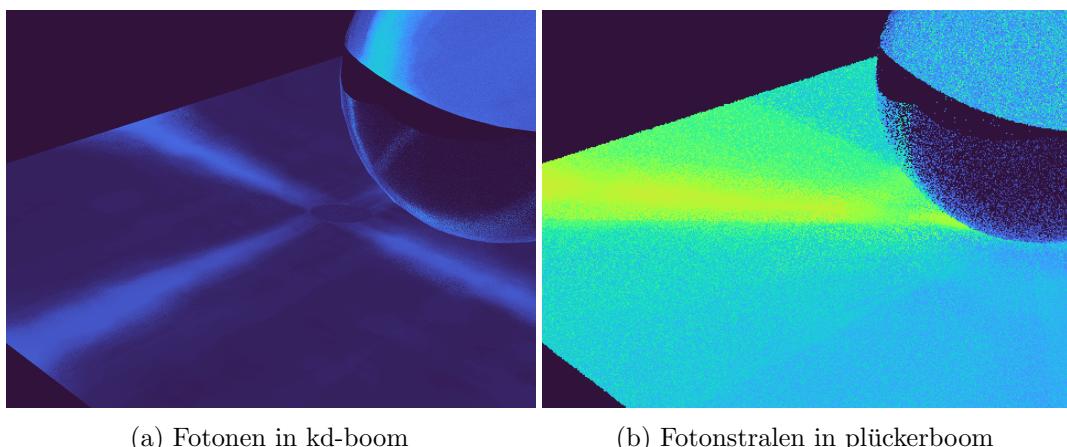
Figuur 6.15: Vergelijking brandpunt, zonder directe belichting, bij verschillende photonmapping technieken. (~ 128000 fotonen in scène, 20 fotonen in PDE)

In sectie 5.1 werd uitgelegd dat het gebruik van de loodrechte afstand tot zoekpunt voor photonmapping niet correct is. Figuur 6.16 demonstreert het verschil tussen het gebruik van de loodrechte afstand tot zoekpunt en de afstand tot het dichtste raakpunt met het zoekvlak. Op de eerste afbeelding is de fout zichtbaar in de vorm van strepen op het diffuse vlak. De fotonstralen van het brandpunt raken het oppervlak en lopen daarna onder het vlak verder, waar ze in de strepen foutief meegeteld worden als nabije fotonen.



Figuur 6.16: Verschil tussen het gebruik van de loodrechte afstand en de afstand tot het dichtste raakpunt.

Figuur 6.17 toont een heatmap van het aantal bezochte knopen in iedere pixel bij het renderen van de afbeeldingen van figuur 6.15.

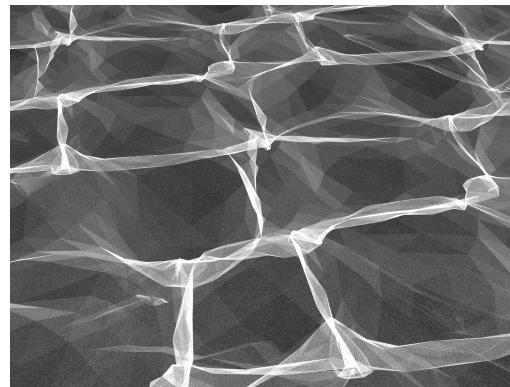


Figuur 6.17: Heatmaps van het aantal bezochte knopen per pixel voor fotonen en fotonstralen. Blauw is kleinste, oranje is grootste. Merk op dat de intensiteit bij kd-bomen de as volgt, terwijl deze bij de plückerboom in een hoek rond de oorsprong ligt.

6. RESULTATEN

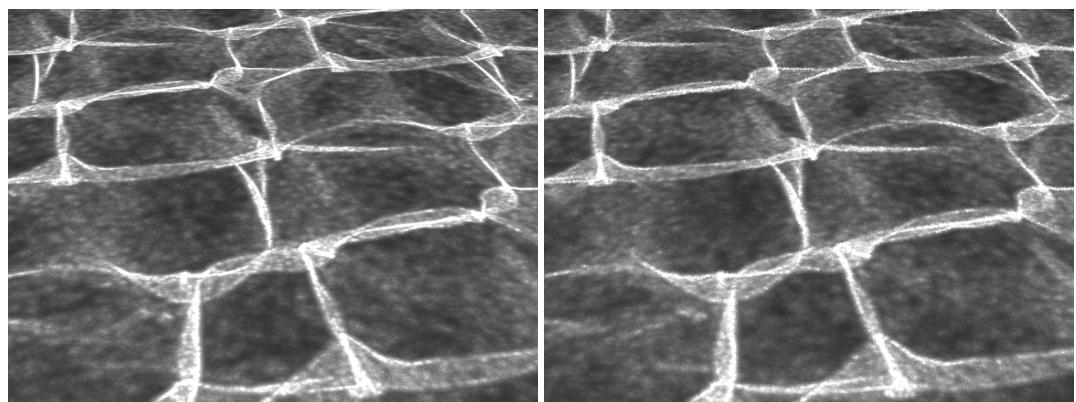
Er is te zien dat er meer knopen zijn bezocht bij de plückerboom. Verder toont de figuur ook hoe de zones met hoge intensiteit de spatiale vorm van de datastructuur aannemen. In figuur 6.17a zijn de lijnen van de splitsingen volgens de assen in de kd-boom te zien. Figuur 6.17b toont hoe een groot aandeel van de fotonstralen een gelijkaardige richting hebben. De punten die gesneden kunnen worden door stralen met deze richting vertonen dan een hogere intensiteit, aangezien in deze punten een lagere minimumafstand gevonden wordt tot de subdomeinen van een groot aandeel van de takken in de boom.

De tweede scène bevat een puntlichtbron boven een golvend wateroppervlak, waaronder een diffuus oppervlak ligt. Het golvende oppervlak veroorzaakt een meer uitgespreid patroon op het oppervlak, zoals te zien is in figuur 6.18.



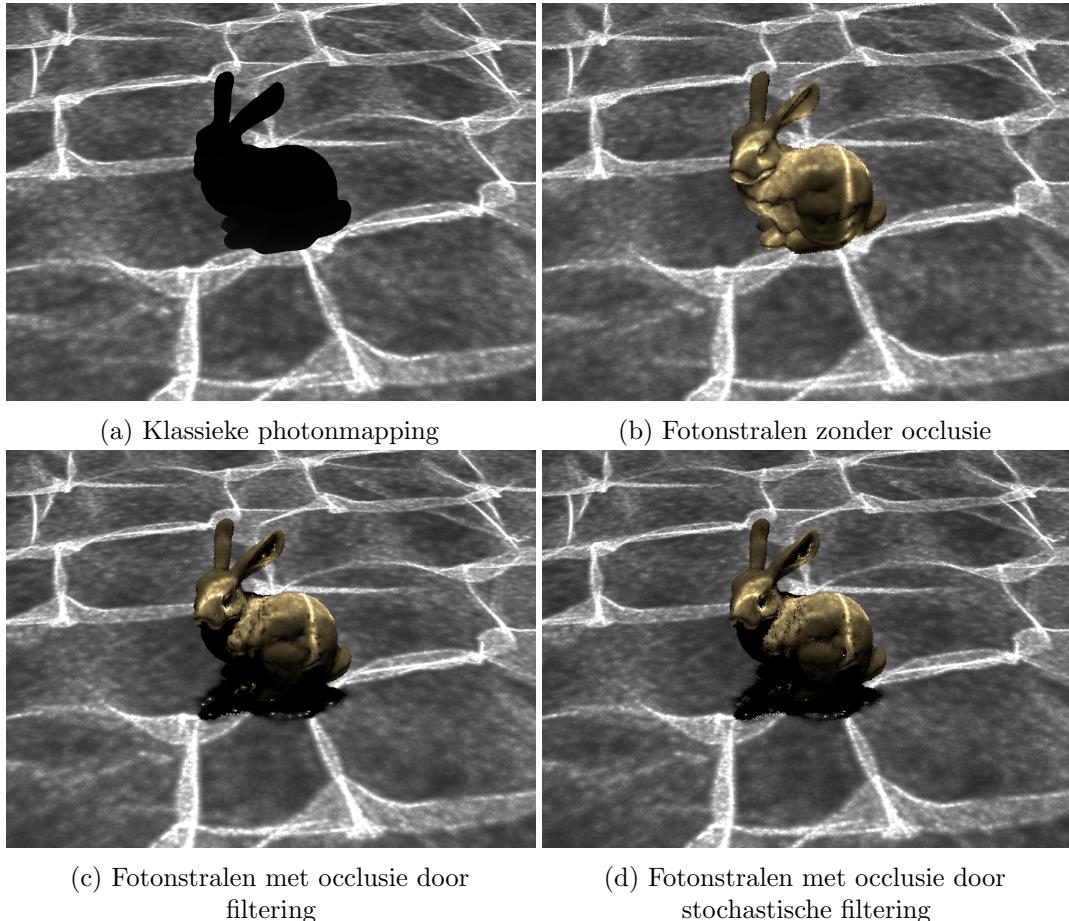
Figuur 6.18: Referentieafbeelding van golfscène gemaakt met klassieke photonmapping. (~ 123 miljoen fotonen in scène, 20 fotonen in PDE)

Voor figuur 6.19 werd dezelfde scène met een lager aantal fotonen gerenderd met zowel klassieke photonmapping als photonmapping met fotonstralen. Deze figuur toont ook dat het resultaat hetzelfde is met beide algoritmes.



Figuur 6.19: Vergelijking brandpunt bij verschillende photonmapping technieken. (~ 600000 fotonen in scène, 20 fotonen in PDE)

Ten slotte werd voor het laatste experiment een goudkleurig diffuus model, *Stanford Bunny* [25], in de scène geplaatst. Hierna werd de scène opnieuw gerenderd, met dezelfde photonmap die gebruikt werd in figuur 6.19. Het resultaat hiervan is te zien in figuur 6.20.



Figuur 6.20: Vergelijking van golfscène uit figuur 6.19 met toegevoegde diffuse *Stanford Bunny* voor verschillende photonmapping technieken uit sectie 2.2. Voor (d) werden willekeurig 10 van de 20 fotonen gekozen.

In figuur 6.20a werd klassieke photonmapping gebruikt, en is het model volledig onbelicht aangezien er geen fotonen op het oppervlak van het model gevonden worden. Naast klassieke photonmapping werden drie varianten van photonmapping met fotonstralen gebruikt, welke geïntroduceerd werden in sectie 2.2. Figuur 6.20b toont het resultaat wanneer er geen occlusietest wordt uitgevoerd. Het model is hier belicht, maar schaduw ontbreekt.

In figuur 6.20c werden uit de gevonden fotonstralen enkel diegene gebruikt die niet eerder al een object raakten. Er is te zien dat hier wel correct schaduw vormt, al vormen er zich artefacten in de penumbra, de regio die slechts gedeeltelijk in schaduw staat, in de vorm van heldere stippen. De oorzaak hiervan is dat op sommige plaatsen

6. RESULTATEN

in de penumbra slechts een klein aantal fotonen voldoen aan de occlusietest. Hierdoor wordt de irradientie op deze punten benaderd met bijvoorbeeld 1 foton in plaats van 20. De intensiteit hangt dan volledig af van het vermogen en de afstand tot deze ene foton. Indien steeds k zichtbare fotonstralen zouden gezocht worden, in plaats van achteraf op zichtbaarheid te filteren, dan zouden deze artefacten verdwijnen, al stijgt hiermee ook het aantal knopen dat bezocht moet worden in de plückerboom.

Figuur 6.20d toont het resultaat wanneer een stochastische benadering van de PDE wordt gebruikt, zoals gegeven in vergelijking (2.2). Voor iedere pixel werden willekeurig 10 van de 20 gevonden fotonen gekozen en getest, en op basis daarvan werd een benadering gemaakt. Het resultaat is grotendeels gelijkaardig aan figuur 6.20c, maar het verschil tussen de twee is terug te vinden in de penumbra. Hier ontstaat ruis, veroorzaakt door een grotere variantie in visibiliteit tussen de gevonden fotonstralen. In complete schaduw, of volledige belichting, is geen ruis te vinden aangezien daar de visibiliteit en vermogen van alle fotonstralen hetzelfde zijn. Indien de fotonen een verschillend vermogen zouden hebben, bijvoorbeeld door het toevoegen van een sterker of zwakkere lichtbron, dan zou hier ook ruis optreden.

Hoofdstuk 7

Besluit

Het doel van deze thesis was een datastructuur te ontwikkelen die toelaat efficiënt de k dichtstbijzijnde rechten rond een 3D punt te zoeken, om hiermee een photonmapping algoritme op basis van fotonrechten te ontwikkelen. Efficiënt betekent hierin dat zoekopdrachten en het invoegen of verwijderen van elementen een lage tijdscomplexiteit hebben, terwijl het geheugenverbruik $\mathcal{O}(n)$ niet overstijgt.

Het resultaat van dit onderzoek is een nieuwe datastructuur, de plückerboom, met bijhorende zoekalgoritmes. Om deze structuur op te bouwen werd eerst een recursieve opdeling van het plückercoördinatendomein gespecificeerd dat toelaat om een numeriek stabiele minimumafstandsfunctie op te stellen. Voor deze minimumafstandsfunctie werd een gedeeltelijke analytische oplossing gevonden die het zoekdomein voor minimalisatie terugbrengt naar drie dimensies. Voor de rest van het domein werd een geschikte numerieke optimalisatiemethode geïdentificeerd. Met deze minimumafstandsfunctie werd vervolgens een zoekalgoritme gespecificeerd. Er werd een constructiemethode opgesteld en verschillende heuristieken voor het splitsen van het domein vergeleken. Vervolgens werden er extensies gedefinieerd die de datastructuur en het zoekalgoritme geschikt maken voor gebruik in photonmapping. Hiervoor werd een extensie gedefinieerd voor het vinden van de k rechten waarvan het snijpunt met een zoekvlak het dichtst bij het zoekpunt ligt. Ten slotte werd een extensie gepresenteerd voor het gebruik van halfrechten en lijnstukken.

Uit experimenten blijkt dat de voorgenoemde zoekoperaties met plückerbomen een tijdscomplexiteit vertonen rond $\mathcal{O}(n^{0.608})$. Een vergelijkbare datastructuur, de kd-boom, heeft hier een tijdscomplexiteit van $\mathcal{O}(\log n)$, maar deze werkt enkel op punten en biedt geen ondersteuning voor rechten. In computationele geometrie zijn de *edge quadtree* en *polygonal map quadtree* gerelateerde datastructuren voor zoekoperaties [22]. Deze hebben voor het zoeken ook een tijdscomplexiteit van $\mathcal{O}(\log n)$, maar ondersteunen enkel lijnstukken, hebben een groter dan lineair geheugenverbruik, en het toevoegen/verwijderen van elementen vereist potentieel de aanpassing van een groot deel van de boom. In plückerbomen gebeurt het toevoegen en verwijdering van elementen in $\mathcal{O}(\log n)$, en is de geheugencomplexiteit $\mathcal{O}(n)$.

In de resultaten werd ook aangetoond dat door fotonstralen in de photonmap op te slaan, deze bruikbaar is na bepaalde aanpassingen van de scène waar klassieke

7. BESLUIT

photonmapping zou falen. Dit breidt de levensduur van de photonmap uit, al staat hier tegenover dat er een grotere kost is om zoekoperaties uit te voeren.

7.1 Verder onderzoek

De plückerboom is een nieuw concept en er is nog veel ruimte voor verder onderzoek. Vooreerst is er nog ruimte voor optimalisatie in de minimumafstandsfunctie. Deze heeft een tijdscomplexiteit van $\mathcal{O}(1)$, maar in praktijk is dit een significante kost door het gebruik van een gradiëntloze numerieke optimalisatiemethode. Een analytische oplossing zou hier ideaal zijn, maar verdere optimalisatie van de numerieke methode door bijvoorbeeld het gebruik van een gradiënt zou ook een sterke verbetering kunnen geven. Een andere opportuniteit voor optimalisatie is te vinden in de constructieheuristieken. Deze werden hier slechts beperkt behandeld, maar in de resultaten kwam reeds aan bod dat een goede heuristiek zou kunnen zorgen voor een stabielere performantie over verschillende configuraties van rechten. Ook in het zoekgoritme is ruimte voor verder onderzoek. Zo zou van iedere knoop naast de minimumafstand ook de maximumafstand berekend kunnen worden, wat mogelijk toelaat om beter te kiezen welke knopen eerst bezocht moeten worden. Verder werd hier gebruik gemaakt van een depth-first strategie, analoog aan het zoeken in een kd-boom, maar andere volgordes zouden sneller kunnen zijn.

Ten slotte zou deze thesis kunnen dienen als een basis voor verder onderzoek rond andere problemen. Binnen computergrafiek zou het bijvoorbeeld interessant kunnen zijn om plückerbomen te gebruiken in raymarching [10] om op ieder punt de dichtste geometrie te vinden. Dit onderzoek focuste zich op rechten met plückercoördinaten, maar met geometrische algebra kan ook complexere geometrie gemodelleerd worden, voor welke ook er zoekstructuren gebouwd zouden kunnen worden. Gezien dat deze thesis een algemeen geometrisch probleem behandelt, is er mogelijk ook potentieel in andere domeinen die geometrie behandelen, zoals fysicasimulaties of machine learning.

Bibliografie

- [1] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. S. Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data*, 7(4):221–248, Dec 2019.
- [2] T. Akenine-Mller, E. Haines, and N. Hoffman. *Real-Time Rendering, Fourth Edition*. A. K. Peters, Ltd., USA, 4th edition, 2018.
- [3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.
- [4] A. Bultheel. *Inleiding tot de numerieke wiskunde*. Leuven : Acco, 2006.
- [5] S. De Keninck. 3d projective geometric algebra cayley table. <https://bivector.net/tools.html>.
- [6] L. Dorst, D. Fontijne, and S. Mann. *Geometric Algebra for Computer Science (Revised Edition)*. Elsevier, 2009.
- [7] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.
- [8] O. Good and Z. Taylor. Optimized photon tracing using spherical harmonic light maps. page 53, 01 2005.
- [9] C. G. Gunn and S. De Keninck. Geometric algebra and computer graphics. *ACM SIGGRAPH 2019 Courses*, Jul 2019.
- [10] J. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12, 06 1995.
- [11] R. Herzog, V. Havran, S. Kinuwaki, K. Myszkowski, and H.-P. Seidel. Global illumination using photon ray splatting. *Computer Graphics Forum*, 26(3):503–513, 2007.
- [12] H. W. Jensen and N. J. Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers and Graphics*, 19(2):215 – 224, 1995.
- [13] Y.-B. Jia. Plücker coordinates for lines in the space. Aug. 2019.

- [14] R. W. Johnson. Estimating the size of a population. *Teaching Statistics*, 16(2):50–52, 1994.
- [15] S. G. Johnson. The nlopt nonlinear-optimization package. <https://github.com/stevengj/nlopt>.
- [16] J. T. Kajiya. The rendering equation. In *Computer Graphics*, pages 143–150, 1986.
- [17] M. H. Kalos and P. A. Whitlock. *Monte carlo methods*. John Wiley & Sons, 2009.
- [18] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. *Computer graphics (New York, N.Y.)*, 20(4):269–278, 1986.
- [19] K. Nordhausen. The elements of statistical learning: Data mining, inference, and prediction, second edition by trevor hastie, robert tibshirani, jerome friedman. *International Statistical Review*, 77(3):482–482, 2009.
- [20] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D ’09, page 75–82, New York, NY, USA, 2009. Association for Computing Machinery.
- [21] T. H. Rowan. *Functional Stability Analysis of Numerical Algorithms*. PhD thesis, USA, 1990. UMI Order No. GAX90-31702.
- [22] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Longman Publishing Co., Inc., USA, 1990.
- [23] M. Sattler, R. Sarlette, T. Mücken, and R. Klein. Exploitation of human shadow perception for fast shadow rendering. In *Proceedings of the 2nd Symposium on Applied Perception in Graphics and Visualization*, APGV ’05, page 131–134, New York, NY, USA, 2005. Association for Computing Machinery.
- [24] R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley Professional, 4th edition, 2011.
- [25] Stanford. The stanford 3d scanning repository. <https://graphics.stanford.edu/data/3Dscanrep/>.
- [26] K. Suffern. *Ray Tracing from the Ground up*. CRC Press, 2016.
- [27] I. Wald and V. Havran. On building fast kd-trees for ray tracing, and on doing that in $O(n \log n)$. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 61–69, 2006.