



Filteren van ruis voor globale belichtingsalgoritmen a.d.h.v Random Parameter Filtering

Geert Van Campenhout

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

Promotor:
Prof. dr. ir. Philip Dutré

Assessoren:
Ir. W. ???
W. ???

Begeleider:
Dr. ir. Ares Lagae

© 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

The text of the preface. A few paragraphs should follow.

The Author
1 January 2010

Inhoudsopgave

Voorwoord	i
Samenvatting	iii
1 Inleiding	1
1.1 Situering	1
1.2 Doelstellingen	1
1.3 Overzicht	1
2 Gerelateerd werk	3
2.1 Adapting sampling	3
2.2 Reconstruction	3
2.3 Alternatieve methodes voor ruisvermindering	3
3 Filtering	5
3.1 Gaussiaanse filter	5
3.2 Bilateral filter als twee Gaussiaanse filters	5
3.3 Cross bilateral filter	5
4 Mutual Information	9
4.1 Histogram en joint histogram	9
4.2 Mutual information	9
5 Random Parameter Filtering	11
5.1 Het idee	11
5.2 Het basisalgoritme	13
5.3 Iteratief algoritme	13
5.4 De scene features	21
6 Resultaten	25
6.1 Verschillende iteraties	25
6.2 Depth of field	25
6.3 Motion blur	25
6.4 Soft shadows	25
6.5 Glossy reflecties	25
6.6 Speciale scenes	25
7 Besluit	37
Bibliografie	39

Samenvatting

abstract

Hoofdstuk 1

Inleiding

1.1 Situering

In dit hoofdstuk wordt een algemene inleiding gegeven waarin het probleem gekaderd wordt, monte carlo integratie wordt kort uitgelegd aan de hand van de vergelijking, hierdoor wordt het probleem duidelijk.

$$I(i, j) = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \cdots \int_{-1}^1 \int_{-1}^1 \int_{t_0}^{t_1} f(x, y, \dots, u, v, t) dt dv du dy dx.$$

1.2 Doelstellingen

1.3 Overzicht

(**TODO** Nog minder samples per pixel gebruiken om het verschil nog duidelijker te maken? Ik gebruik nu 4 spp omdat ik dat ook gebruik voor alle RPF testen)

1. INLEIDING



Figuur 1.1: Resultaat dat aantonnt dat het filteren van een afbeelding gerenderd met een laag aantal samples per pixel met het RPF algoritme vergelijkbaar is met het renderen van dezelfde afbeelding met een hoog aantal samples per pixel. (a) Input van het RPF algoritme, gerenderd met 4 samples per pixel, (b) Resultaat na filteren met het RPF algoritme en (c) Referentieafbeelding, gerenderd met 512 samples per pixel.

Hoofdstuk 2

Gerelateerd werk

In dit hoofdstuk worden papers met gerelateerde inhoud besproken en met het RPF algoritme vergeleken, belangrijk is om verbanden te leggen tussen deze papers en ze zo op te delen in de verschillende stromingen waar ze deel van uitmaken.

(figuren? of gewoon droge tekst?) (**TODO** Bundelen met titels als “adaptive sampling” was gezegd? maar er is veel overlap met anderen, dus beter gewoon alle papers afgaan en in de tekst verwijzen?)

2.1 Adapting sampling

Adaptive Rendering with Non-Local Means Filtering [12]

Multidimensional adaptive sampling and reconstruction for ray tracing [4]

SURE-based Optimization for Adaptive Sampling and Reconstruction [7]

Adaptive Sampling and Reconstruction using Greedy Error Minimization [11]

2.2 Reconstruction

Multidimensional adaptive sampling and reconstruction for ray tracing [4]

Temporal Light Field Reconstruction for Rendering Distribution Effects [5]

Reconstructing the Indirect Light Field for Global Illumination [6]

2.3 Alternatieve methodes voor ruisvermindering

2.3.1 Filtering methodes voor specifieke gedistribueerde effecten

Axis-Aligned Filtering for Interactive Sampled Soft Shadows [8]

2. GERELATEERD WERK

2.3.2 Ruis uit de visibilityterm verminderen

High-Quality Spatio-Temporal Rendering using Semi-Analytical Visibility [3]

A Theory of Monte Carlo Visibility Sampling [10]

2.3.3 Alternatieve methode voor integraalberekening

Compressive estimation for signal integration in rendering [13]

Hoofdstuk 3

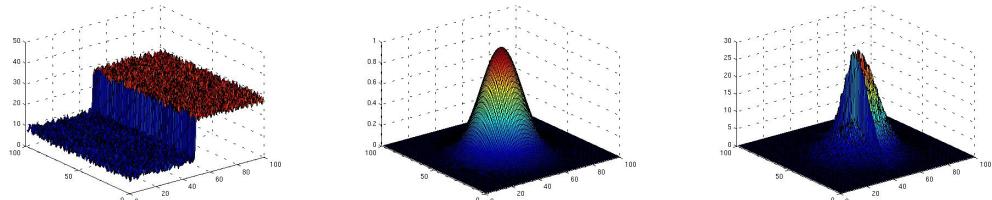
Filtering

In deze sectie wordt uitgelegd hoe de bilateral filter werkt aangezien het RPF algoritme hierop gebaseerd is. Net zoals in de paper volgt hier eerst de uitleg over een simpele Gaussiaanse filter, gevolgd door de gewone bilateral filter gevolgd door de cross bilateral filter.

[16] [2]

3.1 Gaussiaanse filter

3.2 Bilateral filter als twee Gaussiaanse filters



Figuur 3.1: Deze figuur toont aan hoe de edge-preserving eigenschap van de bilaterale filter tot stand komt. (a) stelt een stuk afbeelding met ruis en een scherpe grens tussen verschillende kleuren voor, (b) toont de gaussiaanse functie met de middelste pixel als centrum en (c) toont de gewichten van de pixels voor het bilateraal filteren van de middelste pixel.

3.3 Cross bilateral filter

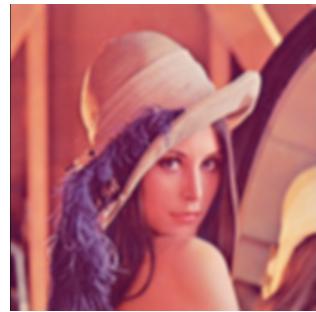
[na de uitleg en de lena afbeeldingen zeggen: als we dit nu toepassen op de errorscene uit de inleiding krijgen we het resultaat zichtbaar in figuur 3.3]

(TODO toon maar een deel van het image, anders is het verschil niet duidelijk)

3. FILTERING



(a) The unfiltered picture of a woman called called Lena.



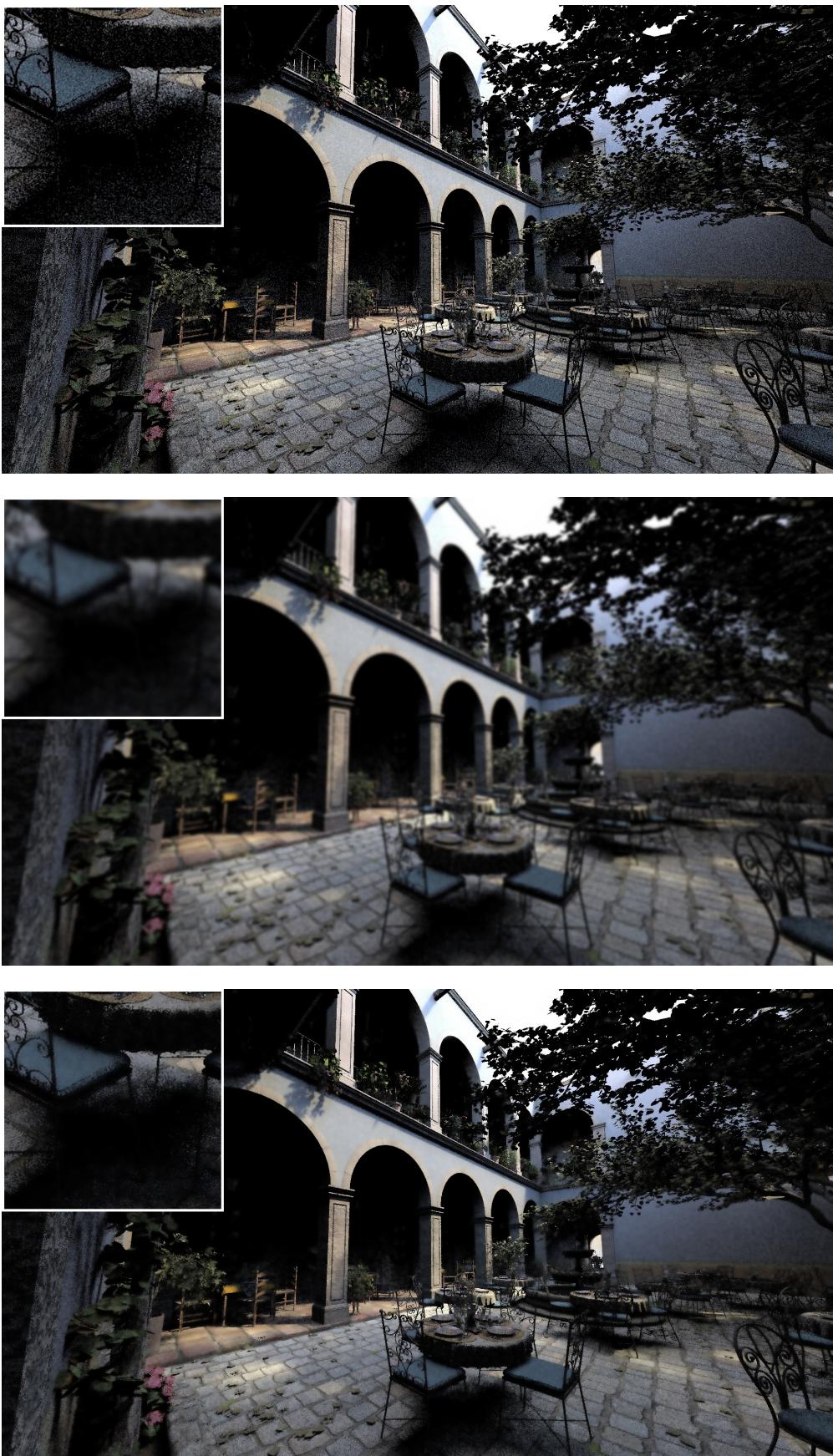
(b) Lena filtered using a Gaussian filter



(c) Lena filtered using a bilateral filter

Figuur 3.2: Pictures of Lena to compare a Gaussian and a bilateral filter, notice how the bilateral filtered image preserves the edges in the image.

3.3. Cross bilateral filter



Figuur 3.3: The san Miguel scene rendered with 4 samples per pixel (a) unfiltered, (b) filtered with a Gaussian filter and (c) filtered with a bilateral filter

Hoofdstuk 4

Mutual Information

Hier wordt uitgelegd hoe de techniek van mutual information werkt, ook het joint histogram zal hiervoor kort uitgelegd worden aangezien dit nodig is voor de berekeningen. [1] [9]

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

4.1 Histogram en joint histogram

4.2 Mutual information

⟨ **TODO** figuur van vendiagram met de entropieen op. ⟩ ⟨ **TODO** zorg dat je hier vermeld dat het optellen van individuele mutual informations een onderschatting is. ⟩

Hoofdstuk 5

Random Parameter Filtering

5.1 Het idee

Het Random Parameter Filtering Algoritme voorgesteld in [15] is een post process afbeeldingsfilter voor afbeeldingen die gerenderd worden met Monte Carlo methodes. Een post process filter wil zeggen dat de afbeelding eerst volledig gerenderd wordt en nadien pas de afbeelding gefilterd wordt. Het doel van Random Parameter Filtering is het verwijderen van ruis die veroorzaakt wordt door random parameters die gebruikt worden in Monte Carlo integratie. Het algoritme is gebaseerd op een gewogen cross bilaterale filter die bovendien in verschillende iteratiestappen kan toegepast worden. De gewichten van de filter worden berekend op basisch van de statistische afhankelijkheid tussen de inputs van Random Parameter Filtering en de random parameters. Een gewicht w_{ij} van een sample j ten opzichte van het sample i , dat gefilterd wordt, wordt berekend met vergelijking 5.1:

$$w_{ij} = \exp \left(-\frac{1}{2\sigma_p^2} \sum_{1 \leq k \leq 2} (\bar{p}_{i,k} - \bar{p}_{j,k})^2 \right) * \\ \exp \left(-\frac{1}{2\sigma_c^2} \sum_{1 \leq k \leq 3} \alpha_k (\bar{c}_{i,k} - \bar{c}_{j,k})^2 \right) * \\ \exp \left(-\frac{1}{2\sigma_f^2} \sum_{1 \leq k \leq m} \beta_k (\bar{f}_{i,k} - \bar{f}_{j,k})^2 \right) \quad (5.1)$$

De eerste term van vergelijking 5.1 neemt de afstand in de afbeeldingsruimte tussen sample j en sample i in rekening. De tweede en de derde term nemen het verschil tussen de statistische afhankelijkheden van sample j en sample i in rekening. Dit zijn de statistische afhankelijkheden tussen de random parameters en respectievelijk de kleur en de scene features van de samples. De α_k en β_k bepalen vervolgens het belang van het k^{de} kleurenkanaal of de k^{de} scene feature gebaseerd op hun statistische afhankelijkheid met de random parameters.

Deze statistische afhankelijkheden kunnen gebruikt worden om de afbeelding te fil-

teren door het effect van random parameters bij Monte Carlo integratie. Dit effect kan verduidelijkt worden aan de hand van het volgende gedachtnisvoorbeeld dat redeneert over een afbeelding met het depth of field effect: Regio's van de afbeelding die in focus zijn bevatten geen ruis aangezien een straal die door de lens gaat van een punt in focus uit de afbeelding altijd op hetzelfde punt in de scène zal terechtkomen, onafhankelijk van het random punt op de lens waar de straal doorgaat. Voor pixels die uit focus liggen anderzijds, varieert het punt in de scène waar de straal op terechtkomt, waardoor de kleur van de sample afhankelijk is van de keuze van het random punt op de lens.

Eén van de voordelen van Random Parameter Filtering is dat het algoritme geen rekening houdt de betekenis van de verschillende parameters en scene features. Als gevolg hiervan kan het algoritme ruis filteren veroorzaakt door elk mogelijk distributie effect, dit zijn effecten zoals depth of field, motion blur, zachte schaduwen en glossy BRDFs. Een voorbeeld van een algoritme dat dit voordeel niet geniet is het algoritme gepresenteerd in [8] dat alleen ruis veroorzaakt door zachte schaduwen kan filteren.

5.1.1 Inputs

Random Parameter Filtering gebruikt verschillende inputs om zijn doel te kunnen bereiken. Per sample dat genomen wordt tijdens het renderen van de afbeelding moet een vector van waarden bijgehouden worden. Deze vector ziet eruit als volgt:

$$x_i = \{c_{i,1}, c_{i,2}, c_{i,3}; p_{i,1}, p_{i,2}; r_{i,1}, \dots, r_{i,n}; f_{i,1}, \dots, f_{i,m}\} \quad (5.2)$$

De vector bevat allereerst $c_{i,1}, c_{i,2}, c_{i,3}$, dit zijn de kleuren van het sample in elk van de drie kanalen. Ook de positie van het sample binnen de afbeeldingsruimte worden bijgehouden in $p_{i,1}$ en $p_{i,2}$. Tot nu toe zijn de inputs besproken die nodig zijn om een afbeelding te filteren met een bilateral filter, maar aangezien een cross bilaterale filter gebruikt wordt, maakt Random Parameter Filtering ook gebruik van de volgende extra inputs;

- De random parameters $r_{i,1}, \dots, r_{i,n}$ die tijdens de Monte Carlo integratie procedure gebruikt worden.
 - De random positie (x, y) van het sample in de pixel.
 - De random positie (u, v) van het sample op de lens.
 - Het random tijdstip t waarin het sample genomen is.
- De scene features $f_{i,1}, \dots, f_{i,m}$, dit zijn de extra features gebruikt door de cross bilateral filter zodanig dat de afbeelding niet alleen op basis van de positie en de kleur van samples gefilterd kan worden. In het bijzonder worden de volgende scene features voorgesteld in [15]:
 - De normaal voor de eerste intersectie.

- De wereld coordinaten voor de eerste intersectie.
- De coordinaten die gebruikt worden voor texture lookups voor de eerste intersectie.
- De normaal voor de tweede intersectie bij path tracing
- De wereld coordinaten voor de tweede intersectie bij path tracing
- De coordinaten die gebruikt worden voor texture lookups voor de tweede intersectie bij path tracing

5.2 Het basisalgoritme

De structuur van het Random Parameter Filtering algoritme wordt op hoog niveau gegeven in algoritme 1. Het algoritme bevat op het hoogste niveau een for loop die het filter proces verschillende malen zal uitvoeren, zo kunnen verschillende filterstappen met verschillende filtergroottes uitgevoerd worden om zo de resultaten te verbeteren. De interne for loop loopt over alle pixels van de afbeelding om deze te filteren. Het algoritme binnen deze for loop bestaat uit de volgende drie grote delen, die elk uitgelegd worden in een aparte sectie;

1. Alle samples in de buurt N van pixel P worden verwerkt op basis van de filtergrootte van deze iteratie (zie sectie 5.3.1).
2. De statistische afhankelijkheden tussen de kleur, de scene features en de random parameters worden berekend. Deze waarden worden vervolgens gebruikt om de gewichten van de filter te berekenen (zie sectie 5.3.2).
3. De kleuren van alle samples in N worden gefilterd om de finale gefilterde kleur van elk sample in pixel P te bepalen (zie sectie 5.3.3).

Tenslotte na het iteratief uitvoeren van de filter worden alle samples van elke pixel gefilterd met een box filter zodanig dat de finale pixel kleur bepaald wordt. Het algoritme dat besproken wordt en beschikbaar is in pseudo code in algoritmen 1, 2, 3 en 4 is het algoritme zoals het geïmplementeerd is en verschilt dus van het algoritme dat besproken wordt in [14], maar is wel volledig gebaseerd hierop. Deze verschillen worden besproken in hoofdstuk 6.

5.3 Iteratief algoritme

Om een sample uit de afbeelding te filteren op basis van de statistische afhankelijkheid met de inputs moet eerst een set van naburige samples gecreëerd worden. De samples die deel uitmaken van deze set zullen bijdragen tot de gefilterde kleur van het sample. Meestal kunnen niet alle samples uit de afbeelding aan de set toegevoegd worden aangezien de afhankelijkheid van de random parameters kan veranderen over de afbeelding. Afbeelding 6.3 is een voorbeeld hiervan aangezien er regio's zijn die in focus zijn zowel als regio's die uit focus zijn. Regio's met verschillende focus

5. RANDOM PARAMETER FILTERING

Algorithm 1 Random Parameter Filtering algoritme

```
filterSize = array with filter sizes that are applied in succession, the proposed  
default is: {55, 35, 17, 7}  
for  $t = 0$  to  $filterSize.length$  do  
    for all pixels  $P$  in image  $I$  do  
        Preprocess samples in neighbourhood  $N$  of  $P$  based on  $filterSize[t]$   
        Compute statistical dependencies of sample color,  
        features inputs  $p_i$  and  $r_i$ , use them to compute filter weights.  
        Filter sample colors in  $P$  using the weighted cross bilateral filter.  
    end for  
end for  
Box filter all samples of each pixel to compute the final pixel color  
return filtered image
```

hebben ook een verschillende statistische afhankelijkheid met de inputs, zoals reeds aangehaald in het gedachten voorbeeld uit sectie 5.1.

Om de grootte van deze set van naburige samples te bepalen, moet rekening gehouden worden met het ‘dueling filter’ probleem. Dit probleem zegt dat we de filtergrootte om één reden zo groot als mogelijk willen maken en om een andere reden zo klein als mogelijk, twee tegenstrijdige idealen. Hoe groter de grootte van deze set, hoe meer informatie er beschikbaar is om de statistische afhankelijkheden te berekenen, waardoor deze afhankelijkheden nauwkeuriger zullen zijn. Hoe kleiner de grootte anderzijds, hoe kleiner de kans dat de filtergrootte verschillende regio’s met andere afhankelijkheden overlapt. Hierdoor kan meer detail uit de afbeelding bewaard worden. Om dit probleem op te lossen wordt er gebruik gemaakt van verschillende filteriteraties met elk een verschillende filtergrootte.

Door van een grote filtergrootte naar kleinere over te gaan, wordt eerst de laagfrequente ruis weggefilterd waarna steeds meer hoogfrequente ruis weggefilterd wordt en de details zoveel mogelijk bewaard blijven. In afbeelding 5.1 is te zien hoe een afbeelding gefilterd met drie iteraties meer ruis heeft kunnen wegfilteren en toch het detail zo goed mogelijk bewaart dan de afbeelding gefilterd met één iteratie.

Een tweede en gerelateerd probleem is dat een algemene ideale filtergrootte niet kan bepaald worden voor een bepaalde scène of resolutie van afbeeldingen. Elke afbeelding bevat namelijk andere regio’s met andere statistische afhankelijkheden tussen de random parameters en de inputs van Random Parameter Filtering. De filtergroottes die in [15] voorgesteld worden als standaard (55,35,17 en 7) zijn dan ook getest en geoptimaliseerd voor één bepaalde resolutie (1920x1080) en één bepaalde afbeelding (een afbeelding van de San Miguel scene).

5.3.1 Verwerken van de samples in de buurt N van pixel P

Algoritme 2 toont de pseudocode van dit gedeelte van het algoritme.

Algorithm 2 Preprocess the samples in neighbourhood N of Pixel P

Input: samples in pixel P , maximum number of samples M in neighbourhood N , filtersize b , number of samples per pixel S

$\sigma_P = b/4$
 Add all samples of pixel P to N .
 Compute the means ($m_{P,k}^f$) and standard deviations ($\sigma_{P,k}^f$) of the samples of P for all scene features k .
 /* Start adding samples that are similar to the samples in pixel P to neighbourhood N . */
 label: *outerfor*
for $p = 0$ to $M - S$ **do**
 Select a random sample j from samples inside the box with width b around P with a normal distribution with as standard deviation σ_P
 for all scene features k **do**
 /* The feature factor H is equal to 30 for the world coordinates scene feature and 3 for all the remaining scene features */
 if $\left((f_{j,k} - m_{P,k}^f) > H * \sigma_{P,k}^f \right)$ AND $\left(\left((f_{j,k} - m_{P,k}^f) > 0.1 \right) OR \left(\sigma_{P,k}^f > 0.1 \right) \right)$ **then**
 continue *outerfor*
 end if
 end for
 Add sample j to N
end for
 Compute means ($m_{N,k}^f$) and standard deviations ($\sigma_{N,k}^f$) of all samples in N for each scene feature k .
 /* Normalize the scene features of all samples in N . */
for all samples j in N **do**
 for all scene features k **do**
 $\bar{f}_{j,k} = (f_{j,k} - m_{N,k}^f) / \sigma_{N,k}^f$
 end for
end for
return Neighbourhood N



(a) Gefilterd met één iteratie met filtergrootte 7.
 (b) Gefilterd met drie iteraties met filtergroottes 35, 17 en 7.

Figuur 5.1: Een afbeelding van de sponza scene gefilterd met (a) één iteratie en (b) drie iteraties.

Per pixel die gefilterd moet worden is een set van samples om informatie uit te gebruiken nodig en aangezien deze set zoals vermeld niet even groot als de gehele afbeelding kan zijn, moet een kleinere set gecreëerd worden. Dit stuk van het algoritme krijgt onder andere de grootte van de omgeving van de pixel P , waarvan samples gebruikt zullen worden, als input. Om zowel informatie van kortbij de pixel als verder weg van de pixel te kunnen gebruiken, worden niet alle samples uit de omgeving van pixel P gebruikt, maar slechts een gedeelte. Op deze manier kan de grootte van de omgeving stijgen terwijl het rekenwerk trager zal stijgen. In deze implementatie wordt gebruik gemaakt van een maximum aantal samples in de set N gelijk aan $M = b^2/2$ wat overeenkomt met de helft van de grootte van de omgeving van pixel P .

Initieel worden alle samples van pixel P zelf toegevoegd aan N aangezien deze de meest belangrijke informatie bevatten. Vervolgens wordt het overige aantal nodige samples random gekozen uit de omgeving volgens een normaalverdeling met als standaarddeviatie $\sigma_p = b/4$. Zo krijgen samples die dichter bij pixel P liggen meer kans om in rekening gebracht te worden. Naast het feit dat het verdere rekenwerk halveert kan ook de positie term uit vergelijking 5.1 kan verwijderd worden. De reden hiervoor is dat de samples gekozen worden met behulp van een normale distributie waardoor samples verder weg van pixel P ook minder gekozen zullen worden en dus moet dit niet meer in beschouwing genomen worden bij het berekenen van de gewichten. De gewichten worden in sectie 5.3.3 dan ook berekend met de aangepaste

vergelijking 5.3:

$$w_{ij} = \exp \left(-\frac{1}{2\sigma_c^2} \sum_{1 \leq k \leq 3} \alpha_k (c_{i,k} - c_{j,k})^2 \right) * \exp \left(-\frac{1}{2\sigma_f^2} \sum_{1 \leq k \leq m} \beta_k (f_{i,k} - f_{j,k})^2 \right) \quad (5.3)$$

Elke random geselecteerde sample wordt vervolgens getest, aangezien samples met totaal verschillende scene features informatie uit regio's met andere statistische afhankelijkheden zou toevoegen. Wanneer een sample een te groot verschil toont in scene features wordt deze niet toegevoegd aan de set N . De test gebeurd op basis van het gemiddelde en de standaard deviatie van de samples uit pixel P . Als alle scene features van een nieuw gekozen sample niet binnen de drie standaarddeviaties van het gemiddelde voor die scene feature van pixel P liggen, wordt de sample niet toegevoegd. Deze factor drie wordt aangepast naar dertig in het geval dat de scene feature de wereldcoördinaten zijn. Wereldcoördinaten veranderen namelijk veel sneller en drastischer dan andere scene features waardoor bijna alle samples genegeerd zouden worden als drie de factor zou zijn. Deze test wordt tevens enkel gedaan als de standaarddeviatie van de scene feature van pixel P groter is dan 0.1 of het verschil met het gemiddelde voor deze scene feature van pixel P groter is dan 0.1. Deze voorwaarde zorgt ervoor dat wanneer de standaarddeviatie zeer klein is, zoals in regio's met constante waarden, niet alle samples genegeerd worden. Als een sample dan voldoet aan de test voor alle scene features, wordt het toegevoegd aan de set N .

Wanneer de set gevuld is met samples, worden alle scene features uit deze set ge-normaliseerd. Dit gebeurd door het gemiddelde en de standaarddeviatie voor elke scene feature uit te rekenen en vervolgens de respectievelijke waarden van de samples te verminderen met het gemiddelde en te delen door de standaarddeviatie. De reden voor normalisatie is dat de gewichten niet beïnvloed mogen worden door de grootte van de featurewaarden. Deze waarden verschillen namelijk sterk onderling, de normalen liggen bijvoorbeeld tussen 0 en 1 terwijl wereld coördinaten bijvoorbeeld tussen 0 en 1000 kunnen liggen. Normalisatie zorgt ervoor dat alle features evenwaardig behandeld worden.

5.3.2 Statistische afhankelijkheid bepalen en berekenen van filtergewichten

Algoritme 3 toont de pseudocode van dit gedeelte van het algoritme.

Nu een set van naburige samples met gelijkaardige scene features beschikbaar is moeten de kleurenkanalen en scene feature gewichten α en β berekend worden. Normalisatie van de kleurkanalen, random parameters en posities is nodig om dezelfde reden als aangehaald in sectie 5.3.1.

5. RANDOM PARAMETER FILTERING

Algorithm 3 Calculate statistical dependencies and compute feature weights

Input: number of the current filter step t , neighborhood N

Normalize the color channels, random parameters and positions of all samples inside N .

/* Compute the statistical dependencies concerning the colors. */

for all color channels i 1 to 3 **do**

for all random parameters j 1 to n **do**

$$D_{c,i}^{r,j} = I(\bar{c}_{N,i}, \bar{r}_{N,j})$$

end for

$$D_{c,i}^r = \sum_{j=0}^n D_{c,i}^{r,j}$$

for all position parameters j 1 to 2 **do**

$$D_{c,i}^{p,j} = I(\bar{c}_{N,i}, \bar{p}_{N,j})$$

end for

$$D_{c,i}^p = \sum_{j=0}^2 D_{c,i}^{p,j}$$

for all scene features j 1 to m **do**

$$D_{c,i}^{f,j} = I(\bar{c}_{N,i}, \bar{f}_{N,j})$$

end for

$$D_{c,i}^f = \sum_{j=0}^m D_{c,i}^{f,j}$$

 Calculate $W_{c,i}^r$ with equation 5.4

$$\alpha_i = \max(1 - 2(1 + 0.1t) * W_{c,i}^r, 0)$$

end for

Calculate D_c^r , D_c^p and D_c^f by summing the respective $D_{c,i}^r$, $D_{c,i}^p$ and $D_{c,i}^f$ terms over the three color channels.

/* Compute the statistical dependencies concerning the scene features. */

for all scene features i 1 to m **do**

for all random parameters j 1 to n **do**

$$D_{f,i}^{r,j} = I(\bar{f}_{N,i}, \bar{r}_{N,j})$$

end for

$$D_{f,i}^r = \sum_{j=0}^n D_{f,i}^{r,j}$$

for all position parameters j 1 to 2 **do**

$$D_{f,i}^{p,j} = I(\bar{f}_{N,i}, \bar{p}_{N,j})$$

end for

$$D_{f,i}^p = \sum_{j=0}^2 D_{f,i}^{p,j}$$

$$D_{f,i}^f = \sum_{k=0}^2 D_{f,i}^{f,k}$$

 Calculate $W_{f,i}^r$ with equation 5.5

 Calculate $W_{f,i}^f$ with equation 5.6

$$\beta_i = W_{f,i}^f * \max(1 - (1 + 0.1t) W_{f,i}^r, 0)$$

18**end for**

$$W_c^r = 1/3(W_{c,1}^r + W_{c,2}^r + W_{c,3}^r)$$

return W_c^r , vector α and vector β

Allereerst wordt berekend hoe elk van de kleurkanalen afhankelijk is van de random parameters ($D_{c,i}^{r,j}$), de positie parameters ($D_{c,i}^{p,j}$) en de scene features ($D_{c,i}^f$). Hoe een kleurkanaal afhankelijk is van de scene features kan gebruikt worden om de gewichten van de scene features die niet bijdragen tot de kleur te verlagen. Vervolgens wordt de afhankelijkheid tussen elk van de scene features en de random parameters ($D_{f,i}^{r,j}$) en positie ($D_{f,i}^{p,j}$) berekend.

De verhouding tussen de afhankelijkheid van een kleurkanaal met de random parameters en het kleurkanaal met de positie op het scherm kan vervolgens bepaald worden met vergelijking 5.4.

$$W_{c,i}^r = \frac{D_{c,i}^r}{D_{c,i}^r + D_{c,i}^p + \epsilon} \quad (5.4)$$

Deze waarde geeft aan hoe sterk een kleurkanaal afhankelijk is van de random parameters ten opzichte van de positie. Wanneer het kleurkanaal sterk afhankelijk is van de random parameters en slechts zwak van de positie ligt de waarde dichtbij 1 en als het tegenovergesteld waar is ligt ze dichtbij 0. Wanneer geen van deze twee uitersten het geval is interpoleert de vergelijking tussen 0 en 1. De ϵ in deze vergelijking zorgt ervoor dat de waarde niet te klein wordt wanneer $D_{c,i}^r$ en $D_{c,i}^p$ beide klein zijn. Op een gelijkaardige manier kan de verhouding tussen de afhankelijkheid van de scene features tot de random parameters en de afhankelijkheid van de scene features tot de positie bepaald worden met vergelijking 5.5.

$$W_{f,i}^r = \frac{D_{f,i}^r}{D_{f,i}^r + D_{f,i}^p + \epsilon} \quad (5.5)$$

Wanneer de i^{de} scene feature enkel gerelateerd is met de random parameters ligt deze waarde dichtbij 1, terwijl ze dichtbij 0 ligt als de scene feature enkel gerelateerd is met de positie. De relatie tussen de kleurkanalen en elk specifieke scene feature wordt ook berekend, met behulp van vergelijking 5.4.

$$W_{f,i}^c = \frac{D_{c,i}^f}{D_c^r + D_c^p + D_c^f} \quad (5.6)$$

Tenslotte wordt in het algoritme de algemene relatie tussen alle kleurkanalen en alle random parameters (W_c^r) berekend. Deze is nodig om de variantie van de filter aan te kunnen passen in vergelijking 5.10.

Calculation of α and β

Aan de hand van de berekende relaties kunnen de gewichten van de kleurkanalen en scene features termen in de gewichten van de filter berekend worden. Nu berekend is wat de impact van de random parameters op elk van de kleurkanalen is ($W_{c,i}^r$) kan α berekend worden aan de hand van vergelijking 5.7.

$$\alpha_i = 1 - W_{c,i}^r \quad (5.7)$$

α_k zal ervoor zorgen dat het gewicht van een kleurkanaal groot is als de random parameters weinig impact hadden op dit kleurkanaal. Op een gelijkaardige manier kan β berekend worden met de relatie tussen elke scene feature en de random parameters ($W_{f,i}^r$). Het verschil is hier dat scene features die niet bijdragen tot de kleur genegeerd moeten worden, $W_c^{f,k}$ bevat deze informatie dus wordt hiermee vermenigvuldigd (zie vergelijking 5.8).

$$\beta_i = W_c^{f,i} \left(1 - W_{f,i}^r \right) \quad (5.8)$$

In [14] werd ondervonden dat de resultaten verbeterd werden door α en β aan te passen door meer gewicht te geven aan de afhankelijkheid ten opzichte van de random parameters naar mate de filtergrootte daalt. Dit komt doordat de afhankelijkheid van de positie stijgt wanneer de filtergrootte groot is door de variaties in de afbeelding over de ruimte. Tegelijkertijd zijn de statistische berekeningen minder nauwkeurig door het overlappen van regio's met verschillen in afhankelijkheden van de random parameters. Vergelijking 5.9 toont de uiteindelijke berekening van de elementen van de vectoren α en β met t de nummer van de huidige iteratie, beginnend van 0. De maximum functie wordt gebruikt om de waarden positief te houden.

$$\begin{aligned} \alpha_i &= \max \left(1 - 2(1 + 0.1t) W_{c,i}^r, 0 \right) \\ \beta_i &= W_c^{f,i} \max \left(1 - (1 + 0.1t) W_{f,i}^r, 0 \right) \end{aligned} \quad (5.9)$$

5.3.3 Filteren van de samples met de gewogen cross bilaterale filter

Algoritme 4 toont de pseudocode van dit gedeelte van het algoritme.

Algorithm 4 Filter the color samples

Input: σ_g^2 , number of samples per pixel S , W_c^r , neighbourhood N , samples of pixel P , α , β and the previous colors of the samples c'

```

Calculate  $\sigma_c^2$  and  $\sigma_f^2$  using equation 5.10
for all samples i in P do
     $c''_i = 0$ 
     $w = 0$ 
    for all samples j in N do
        Calculate  $w_{ij}$  with equation 5.3
         $c''_i = c''_i + w_{ij}c'_j$ 
         $w = w + w_{ij}$ 
    end for
     $c''_i = \frac{c''_i}{w}$ 
end for
return The filtered sample colors  $c''$ 

```

Het filteren van de samples gebeurd met de gewogen cross bilaterale filter waarbij het gewicht van de kleur en de scene features in relatie gebracht wordt met hun afhankelijkheid met de random parameters. Dit gebeurt door de α en β factoren die in vorige sectie berekend werden. De gewichten van elke sample worden berekend met vergelijking 5.3 en zoals reeds vermeld in sectie 5.3.1 is de term die het verschil in positie in de afbeeldingsruimte tussen de samples in rekening brengt verwijderd door de manier waarop samples geselecteerd worden. Vergelijking 5.3 bevat nog twee onbekenden, namelijk de varianties σ_c^2 en σ_f^2 van de filter, deze worden berekend volgens vergelijking 5.10.

$$\sigma_c^2 = \sigma_f^2 = \frac{\sigma^2}{(1 - W_c^r)^2} \quad (5.10)$$

with

$$\sigma^2 = 8\sigma_8^2/s$$

Beide varianties krijgen dezelfde waarde toegekend aangezien geen verschil gemaakt wordt tussen de kleurkanalen en de scene features. De variantie σ_8^2 is gekozen op basis van experimenten door [15] en wordt voor scenes, waarin veer ruis aanwezig is, gelijkgesteld aan 0.02 en voor alle overige scenes gelijkgesteld aan 0.002. Door deze constante te delen door het aantal samples per pixel is het RPF algoritme een biased maar consistent algoritme aangezien wanneer het aantal samples per pixel naar oneindig gaat, de variantie van de filter 0 wordt. Dit wil zeggen dat alle gewichten 0 zijn tenzij bij $i = j$, waar het gewicht steeds 1 is, hierdoor wordt de afbeelding helemaal niet gefilterd dus convergeert de methode. Deze variantie σ^2 wordt dan gedeeld door $(1 - W_c^r)^2$ aangezien de filter zeer sterk moet filteren als de kleur sterk afhankelijk is van de random parameters en omgekeerd weinig moet filteren als de kleur weinig afhankelijk is van de random parameters.

De berekende gewichten worden vervolgens gebruikt om de samples te filteren zoals in vergelijking 5.11.

$$c''_{i,k} = \frac{\sum_{j \in N} w_{ij} c'_{j,k}}{\sum_{j \in N} w_{ij}} \quad (5.11)$$

De kleurvector c'' bevat de nieuw gefilterde kleuren terwijl de kleurvector c' ofwel een kopie is van de input kleurvector tijdens de eerste filteriteratie ofwel een tussentijds resultaat, van de iteraties die tot dan toe uitgevoerd zijn, is.

5.4 De scene features

De scene features die geïmplementeerd zijn in mijn implementatie van Random Parameter Filtering zijn de normalen en de wereldcoördinaten. In tegenstelling tot resultaten uit [15] en [14] die steeds gefilterd zijn gebruik makende van alle scene features, heb ik testen gedaan om te zien of Random Parameter Filtering ook

5. RANDOM PARAMETER FILTERING

acceptabele resultaten kan bekomen indien maar één scene feature tegelijk gebruikt wordt. In de volgende secties bespreek ik twee voorbeelden waarin filteren met één van de features betere resultaten oplevert dan filteren met de andere feature.

5.4.1 Zuil uit de Sponza scène

Afbeelding 5.2 toont de resultaten van deze test.

In afbeelding 5.2b is te zien is dat filteren, enkel gebruik makende van de normaal, ervoor zorgt dat vlakken zeer mooi egaal gefilterd worden (althoewel hier detail uit de textuur verloren is).

Afbeelding 5.2b toont daarentegen wel een schuine streep op de linkerhelft van de zuil, maar zoals in afbeelding 5.2d te zien is komt dit doordat de normalen hier plots van richting veranderen (merk de plotse kleurverandering van blauw naar roos op, men is blijkbaar vergeten rekening te houden met de richting van de normalen tijdens het opstellen van de scène).

Filteren alleen gebruik makend van de wereld coördinaten is voor deze afbeelding een slecht idee aangezien de filter geen verschil meer kan maken tussen samples die juist links van de hoek op de zuil liggen of rechts ervan (dit valt op uit afbeelding 5.2e waar de gehele zuil een bijna egale kleur krijgt). Hierdoor wordt gewoonweg over de hoek van de zuil gefilterd wat de afbeelding niet ten goede komt. Filteren met alleen de wereld coördinaten bewaart wel meer detail aangezien er wel een verschil is in wereld coördinaten van samples op een vlak terwijl de normaal van samples op een vlak exact hetzelfde is.

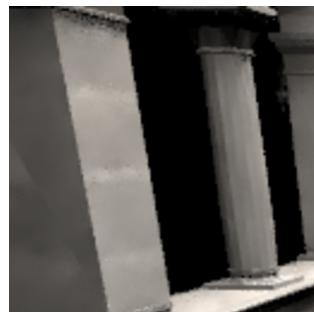
5.4.2 Menger fractalen in een Cornell box

Afbeelding 5.3 toont de resultaten van deze test.

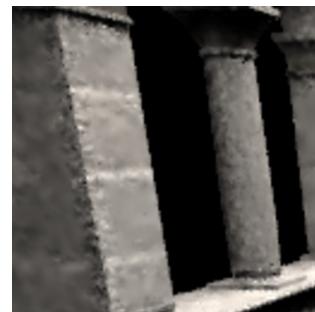
Filteren gebruik makend van enkel de wereldcoördinaten geeft een beter resultaat dan filteren met enkel de normalen. In afbeelding 5.3b is te zien hoe de grens tussen het achterste en het voorste object vervaagd aangezien de normalen op die plaatsen voor beide objecten dezelfde richting uit wijzen zoals te zien in afbeelding 5.3d. Ook de grenzen ontstaan door de gaten in de voorste object vervagen in deze afbeelding. De wereldcoördinaten zijn wel verschillend aangezien de objecten achter elkaar staan, dit is te zien in afbeelding 5.3e. De afbeelding 5.3c gefilterd met alleen de wereldcoördinaten bewaart dan ook de grens tussen de twee objecten in de afbeelding.



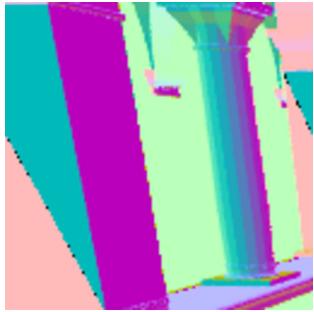
(a) Gefilterd met alle scene features.



(b) Enkel met de normalen gefilterd.



(c) Enkel met de wereld coördinaten gefilterd.



(d) False color afbeelding die de richting van de normaal op elke pixel toont.



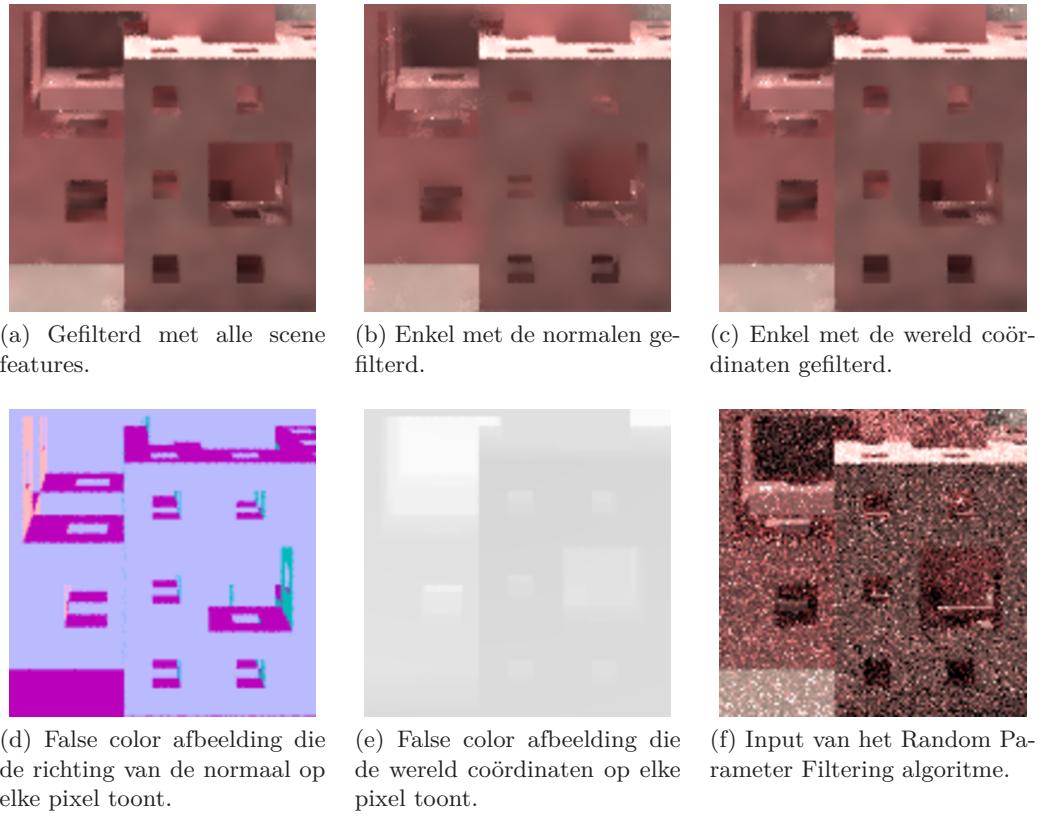
(e) False color afbeelding die de wereld coördinaten op elke pixel toont.



(f) Input van het Random Parameter Filtering algoritme.

Figuur 5.2: Een zuil uit de sponza scene gefilterd met (a) alle scene features, (b) enkel de normalen, (c) enkel de wereld coördinaten, afbeeldingen (d) en (e) tonen de normalen en wereld coördinaten en (f) toont de input van het algoritme.

5. RANDOM PARAMETER FILTERING



Figuur 5.3: Een Menger fractaal (2 iteraties) in een Cornell box gefilterd met (a) alle scene features, (b) enkel de normalen, (c) enkel de wereld coördinaten, afbeeldingen (d) en (e) tonen de normalen en wereld coördinaten en (f) toont de input van het algoritme.

Hoofdstuk 6

Resultaten

In dit hoofdstuk wordt besproken hoe het algoritme geïmplementeerd is, hoe de resultaten gerenderd zijn en wat het verschil is tussen het rpf algoritme gepresenteerd in de paper (snelheid, HDR, (features?)) en dit rpf algoritme. Ook resultaten met elk feature apart zullen besproken worden. Vergelijk de resultaten met de resultaten uit het rpf paper en resultaten van andere papers (door middel van MSE bvb). Ook toekomstig werk wordt hier besproken.

[Probeer figuren te voorzien met de verschillende effecten: DOF (dof-dragons), motion blur, zachte schaduwen, glossy reflecties]

[Resultaat met SanMiguel scene]

[Resultaat met sponza scene]

6.1 Verschillende iteraties

6.2 Depth of field

6.3 Motion blur

6.4 Soft shadows

6.5 Glossy reflecties

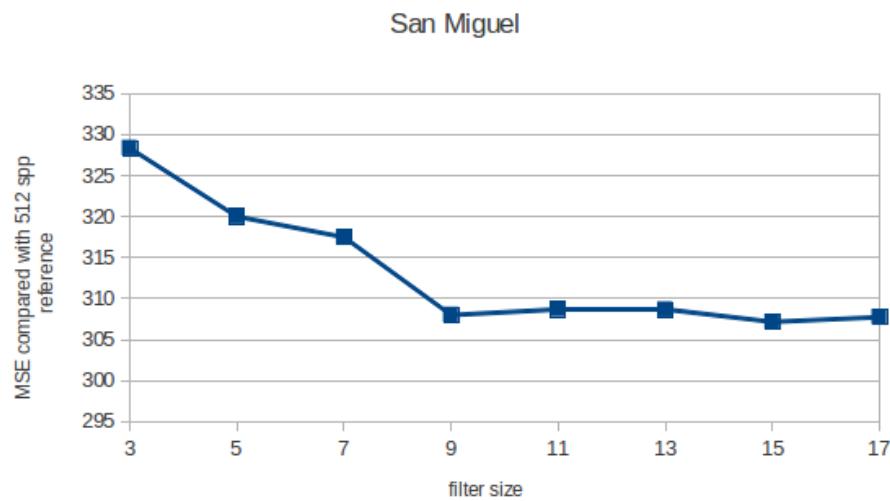
Resultaten van theepot in cornellbox

6.6 Speciale scenes

(TODO Render referentie.)

6. RESULTATEN

comment op sponza1 dat RPF problemen heeft met textures die vrij random zijn, dan kan hij geen verschil maken tussen random van de texture en de random variabelen.



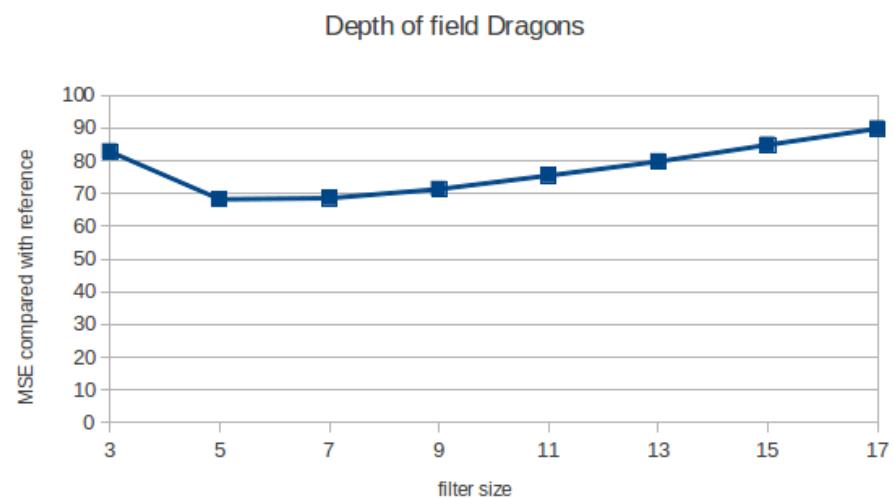
(a) Vergelijking van MSE met referentieafbeelding voor afbeeldingen gefiltered met één iteratie met filtergroottes variërend van 3 tot 17.



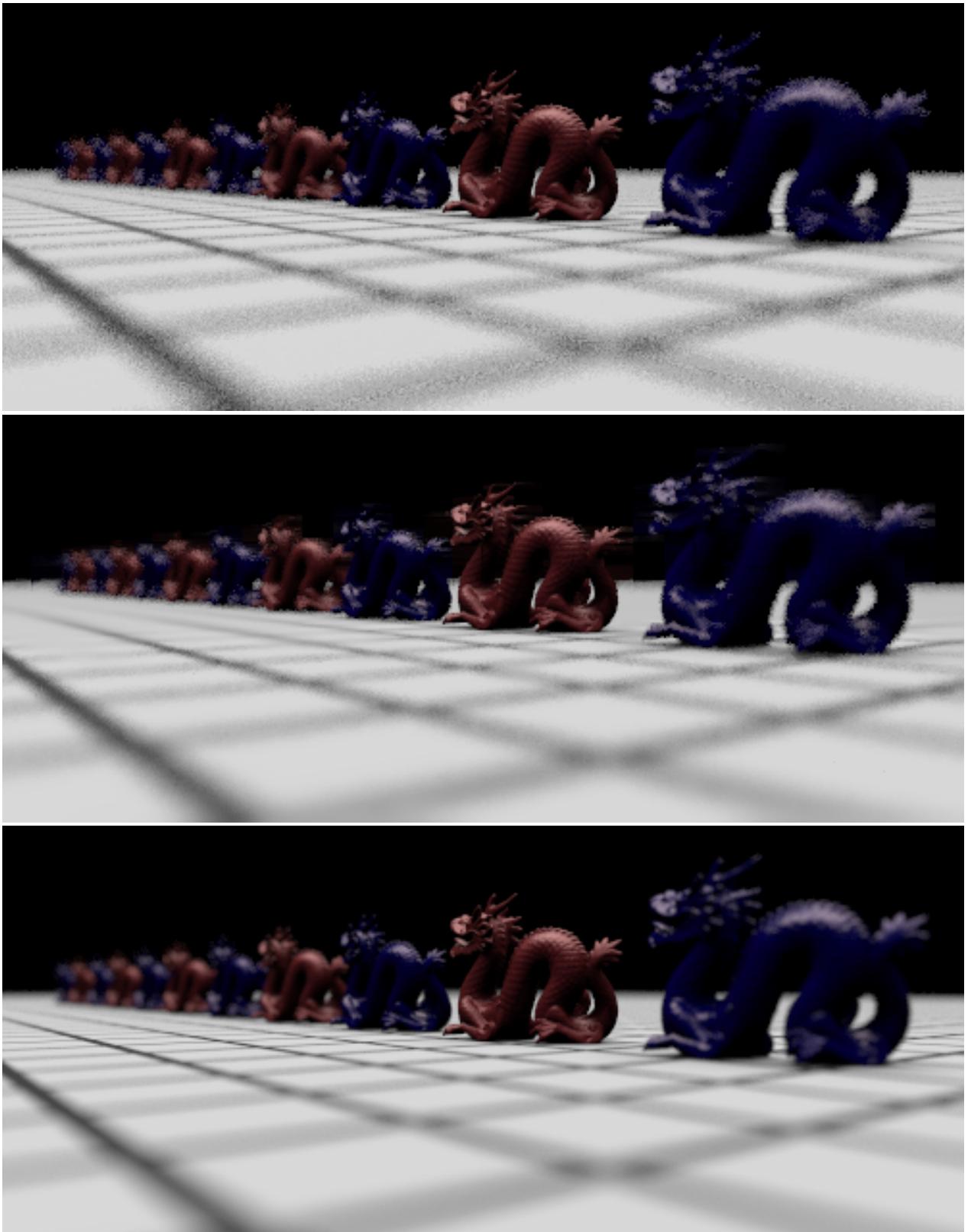
(b) Vergelijking van MSE met referentieafbeelding voor afbeeldingen gefiltered met twee iteraties met de filtergrootte van de eerste iteratie variërend van 15 tot 21 en de filtergrootte van de tweede iteratie 7 of 9.

Figuur 6.1: Vergelijking van MSE van gerenderde afbeeldingen van de sanMiguel scene gefilterd met (a) één filter iteratie en (b) twee filter iteraties.

6. RESULTATEN

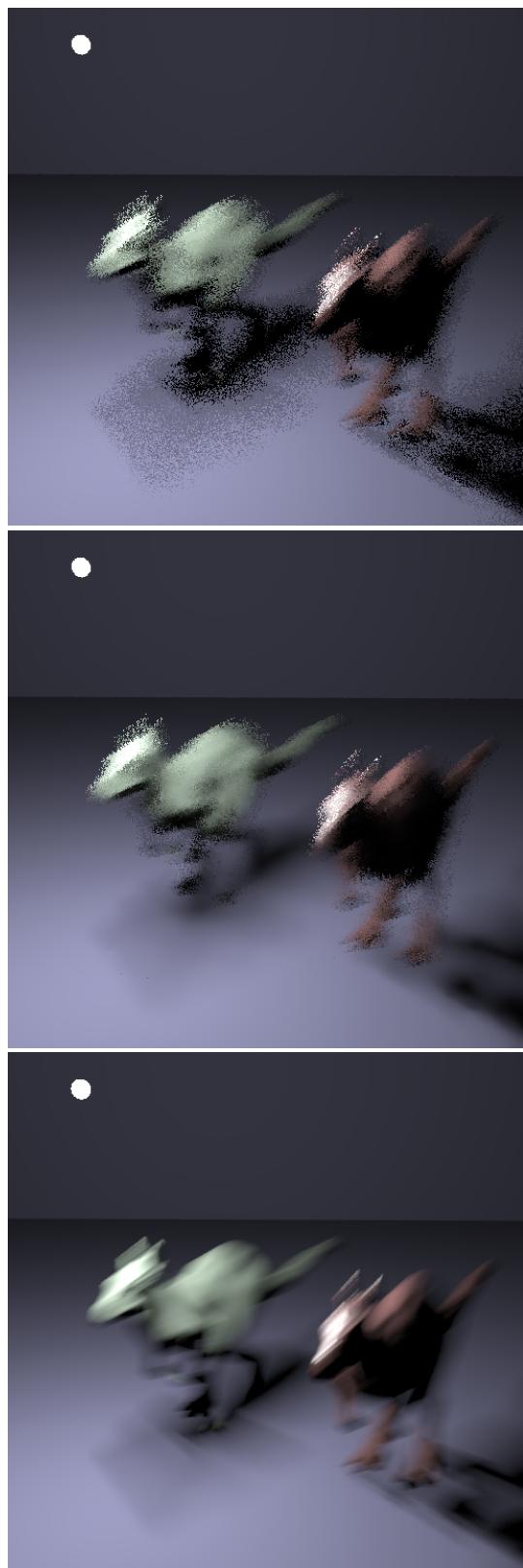


Figuur 6.2: Vergelijking van MSE van gerenderde afbeeldingen van de depth of field dragons scene gefiltered met één iteratie met filtergroottes variërend van 3 tot 17.

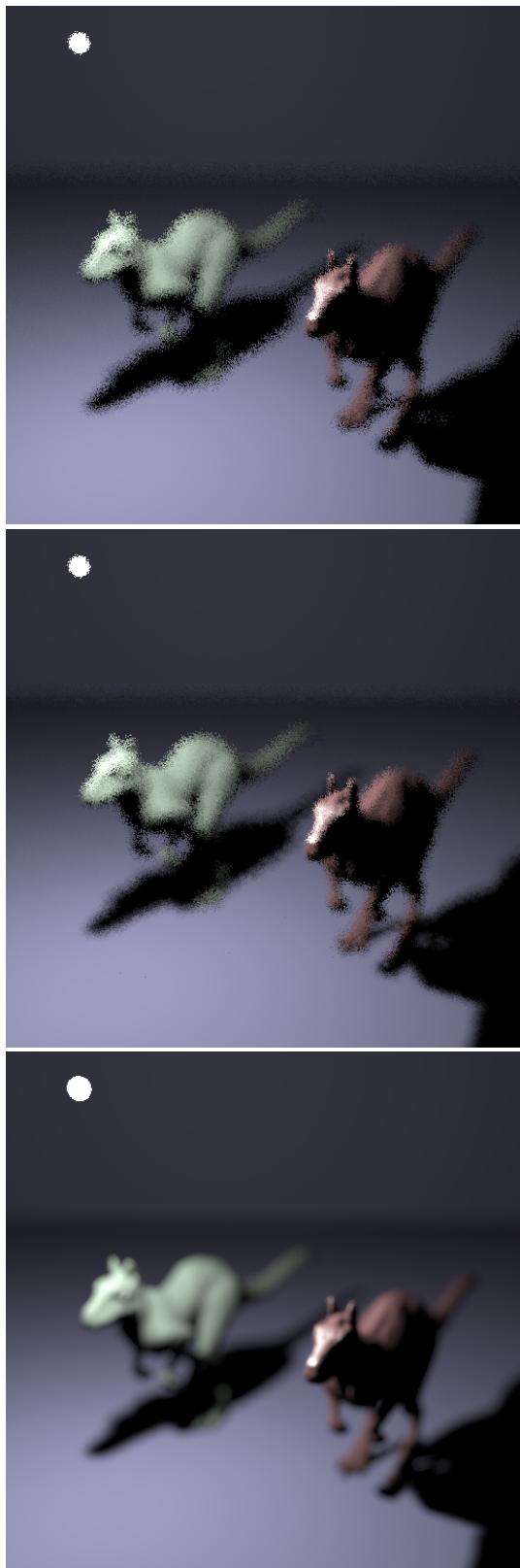


Figuur 6.3: Afbeeldingen van de dragons scene met het depth of field effect. (a) toont de input van het RPF algoritme (4 samples per pixel), (b) Het Resultaat ~~na~~ filteren met het RPF algoritme en (c) Referentieafbeelding (256 spp)

6. RESULTATEN

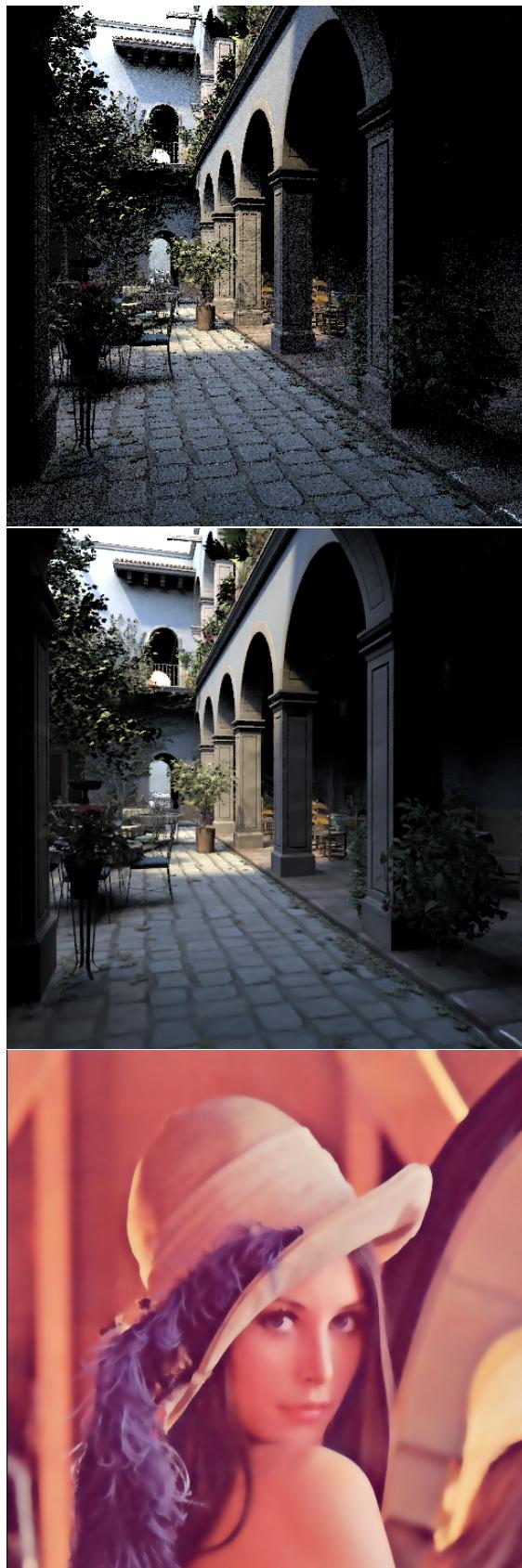


Figuur 6.4: Afbeeldingen van de moving Killeroo scene met het motion blur effect. (a) toont de input van het RPF algoritme (4 samples per pixel), (b) Het Resultaat na filteren met het RPF algoritme en (c) Referentieafbeelding (512 spp)



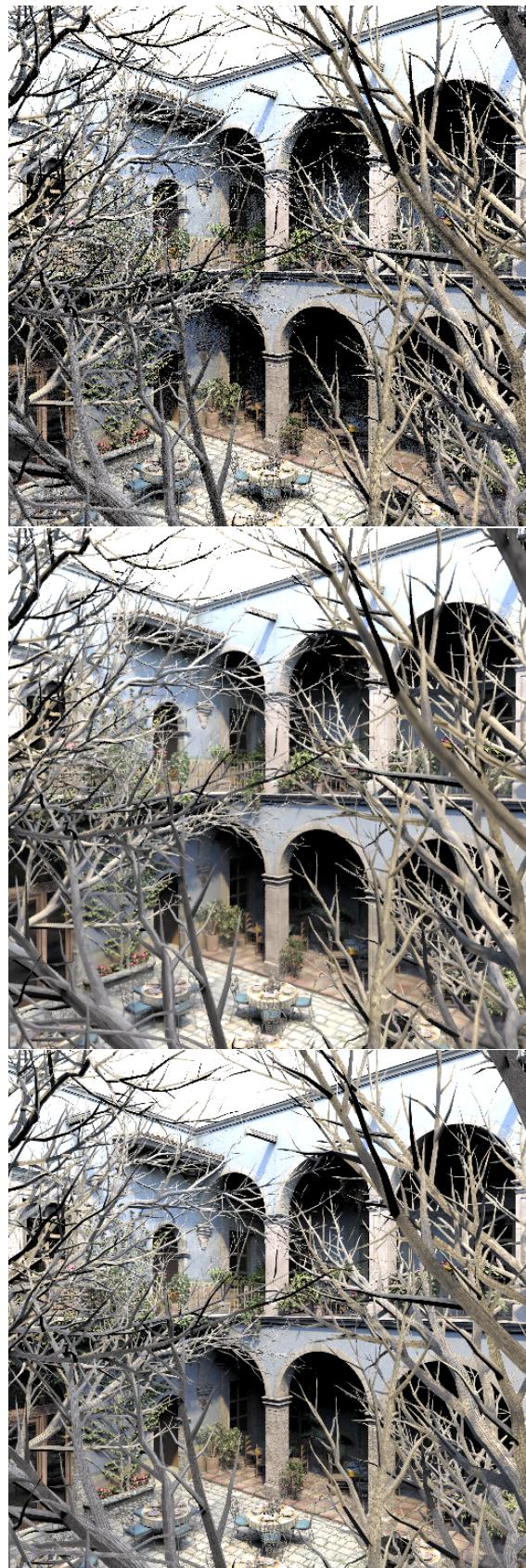
Figuur 6.5: Afbeeldingen van de Killeroo scene met het soft shadows effect. (a) toont de input van het RPF algoritme (4 samples per pixel), (b) Het Resultaat na filteren met het RPF algoritme en (c) Referentieafbeelding (512 spp)

6. RESULTATEN



32

Figuur 6.6: Enkele resultaten van afbeeldingen met een ander camerastandpunt uit de sanMiguel scene. (a) De ongefilterde afbeelding gerenderd met 4 samples per pixel, (b) Het resultaat na filtering met RPF en (c) De referentie met ???? samples per pixel.



33

Figuur 6.7: Enkele resultaten van afbeeldingen met een derde camerastandpunt uit de sanMiguel scene. (a) De ongefilterde afbeelding gerenderd met 4 samples per pixel, (b) Het resultaat na filtering met RPF en (c) De referentie met 512 samples per pixel.

6. RESULTATEN



34

Figuur 6.8: Enkele resultaten van afbeeldingen van de sponza scene. (a) De ongefilterde afbeelding gerenderd met 4 samples per pixel, (b) Het resultaat na filtering met RPF en (c) De referentie met 512 samples per pixel.



35

Figuur 6.9: Enkele resultaten van afbeeldingen van een tweede camerastandpunt uit de sponza scene. (a) De ongefilterde afbeelding gerenderd met 4 samples per pixel, (b) Het resultaat na filtering met RPF en (c) De referentie met 512 samples per pixel.

Hoofdstuk 7

Besluit

Een algemeen besluit wordt aangereikt in dit hoofdstuk.

Bibliografie

- [1] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2006.
- [2] Haarith Devarajan and Harold Nyikal. Image scaling and bilateral filtering.
<http://scien.stanford.edu/pages/labsite/2006/psych221/projects/06/imagescaling/index2.html>, 2013.
[Online, accessed 31 jan 2013].
- [3] Carl Johan Gribel, Rasmus Barringer, and Tomas Akenine-Möller. High-Quality Spatio-Temporal Rendering using Semi-Analytical Visibility. *ACM Transactions on Graphics*, 30:54:1–54:11, August 2011.
- [4] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):33:1–33:10, August 2008.
- [5] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph.*, 30(4), 2011.
- [6] Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. Reconstructing the indirect light field for global illumination. *ACM Trans. Graph.*, 31(4):51:1–51:10, July 2012.
- [7] Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. Sure-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2012)*, 31(6):186:1–186:9, November 2012.
- [8] Soham Mehta, Brandon Wang, and Ravi Ramamoorthi. Axis-aligned filtering for interactive sampled soft shadows. *ACM Trans. Graph.*, 31(6):163:1–163:10, Nov 2012.
- [9] Hanchuan Peng. Mutual information computation.
<http://www.mathworks.com/matlabcentral/fileexchange/14888>, 2007.
[Online, accessed 31 jan 2013].

BIBLIOGRAFIE

- [10] Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. A theory of monte carlo visibility sampling. *ACM Transactions on Graphics*, 2012.
- [11] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.*, 30(6):159:1–159:12, December 2011.
- [12] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.*, 31(6):195:1–195:11, November 2012.
- [13] Pradeep Sen and Soheil Darabi. Compressive estimation for signal integration in rendering. *Computer Graphics Forum*, 29(4), 2010.
- [14] Pradeep Sen and Soheil Darabi. Implementation of Random Parameter Filtering. Technical Report EECE-TR-11-0004, University of New Mexico, May 2011.
- [15] Pradeep Sen and Soheil Darabi. On Filtering the Noise from the Random Parameters in Monte Carlo Rendering. *ACM Transactions on Graphics (TOG)*, to appear, 2011.
- [16] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839 –846, jan 1998.

Fiche masterproef

Student: Geert Van Campenhout

Titel: Filteren van ruis voor globale belichtingsalgoritmen a.d.h.v
Random Parameter Filtering

Engelse titel: Filtering of Noise for Global Illumination Algorithms using random
parameter filtering

UDC: 621.3

Korte inhoud:

Hier komt een heel bondig abstract van hooguit 500 woorden. L^AT_EX commando's mogen hier gebruikt worden. Voorbeelden van letters met accenten zijn "éïçàô". (Gebruik ï in plaats van i, om geen 3 puntjes te hebben.)

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

Promotor: Prof. dr. ir. Philip Dutré

Assessoren: Ir. W. ???
W. ???

Begeleider: Dr. ir. Ares Lagae