Logarithmic Perspective Shadow Maps

Konni Hartmann

Universität Göttingen

Sem. Computergrafik, 2009

Ziele des Vortrags

- Überblick zu verschieden Shadow-Mapping-Techniken
- Herleitung einer Fehlermetrik für Punkt- und direktionale Lichtquellen
- Herleiten einer Parametrisierung (LogP) zur Minimierung von Aliasing Fehlern unabhängig von Szenengeometrie
- Übersicht zur praktischen Umsetzung von LogPSM

Gliederung

1. Shadow Maps
   - Uniform Shadow Maps
   - Warping
   - Partitionierung
2. Fehler-Metrik
   - Herleitung Aliasing-Fehler-Metrik
   - Parametrisierung von m
3. Vereinfachte Parametriesung der Verteilungsfunktion δ_b
   - Einfache Parametrisierungen
   - Logaritmisch-Perspektivische Parametrisierung
   - LogPSM Algorithmus
4. Ausblick
   - Implementierung auf aktueller Hardware
   - Vergleich

Gliederung

Uniform Shadow Mapping

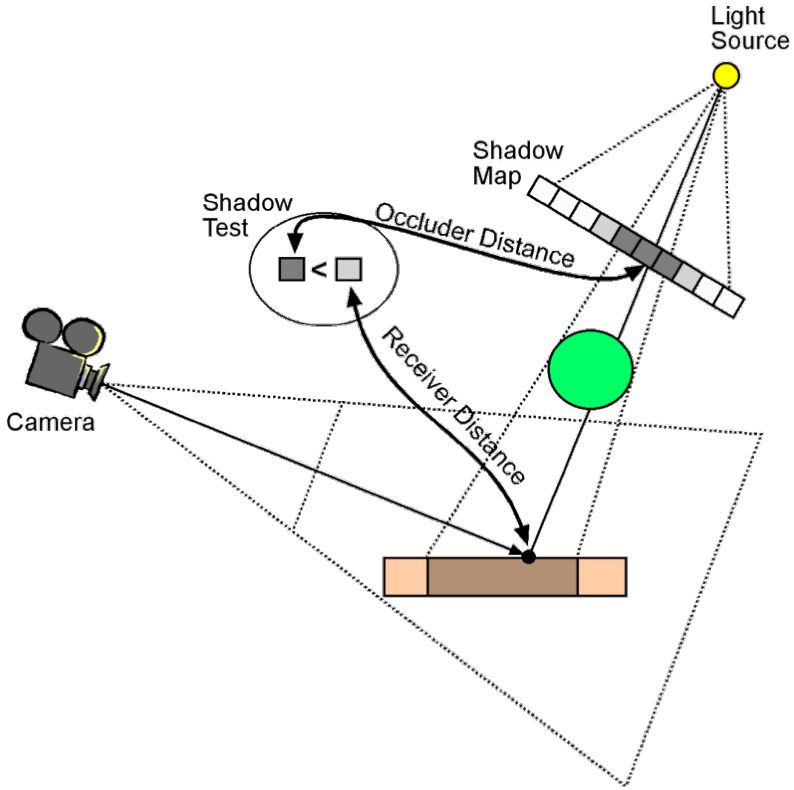

Abbildung: Shadow Mapping
[Brabec2002, Fig.1, S.2]

«Casting curved shadows on curved surfaces«

Uniform Shadow Maps [Williams1987]

- Unabhängig vom Betrachter
- Unabhängig von Szenengeometrie

Probleme mit Uniform Shadow Maps

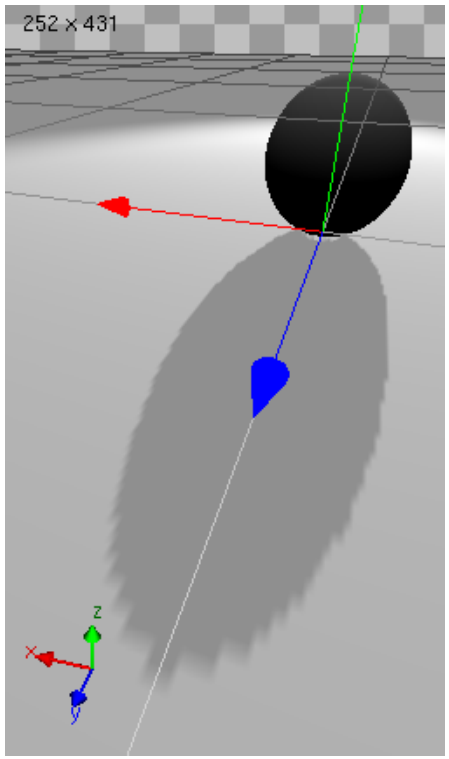

Abbildung: Beispiel für Aliasing Fehler in GroIMP

Auflösungsbedingt sind Aliasing-Artefakte zu sehen

- Bei Vergrößern eines Szene-Auschnittes
- Bei starker Abweichung zwischen Licht- und Kamera-Perspektive[a]

[a] Extremfall: »duelling Frustra«

Gliederung

Warping: Idee

- Gleichmäßige Verteilung der Schatten Texel führt zu nicht optimaler Nutzung
  - Ungenutze Texel nahe des Betrachters (st-Auflösung)
  - Höchste Auflösung nahe der Lichtquelle, nicht nahe des Betrachters (z-Auflösung)
- Bestimme Transformation in Abhängigkeit von Kameraposition und Lichtquelle um Aliasing zu Reduziern

Beispiele: PSM, LiPSM, TSM

Perspektivisches Warping

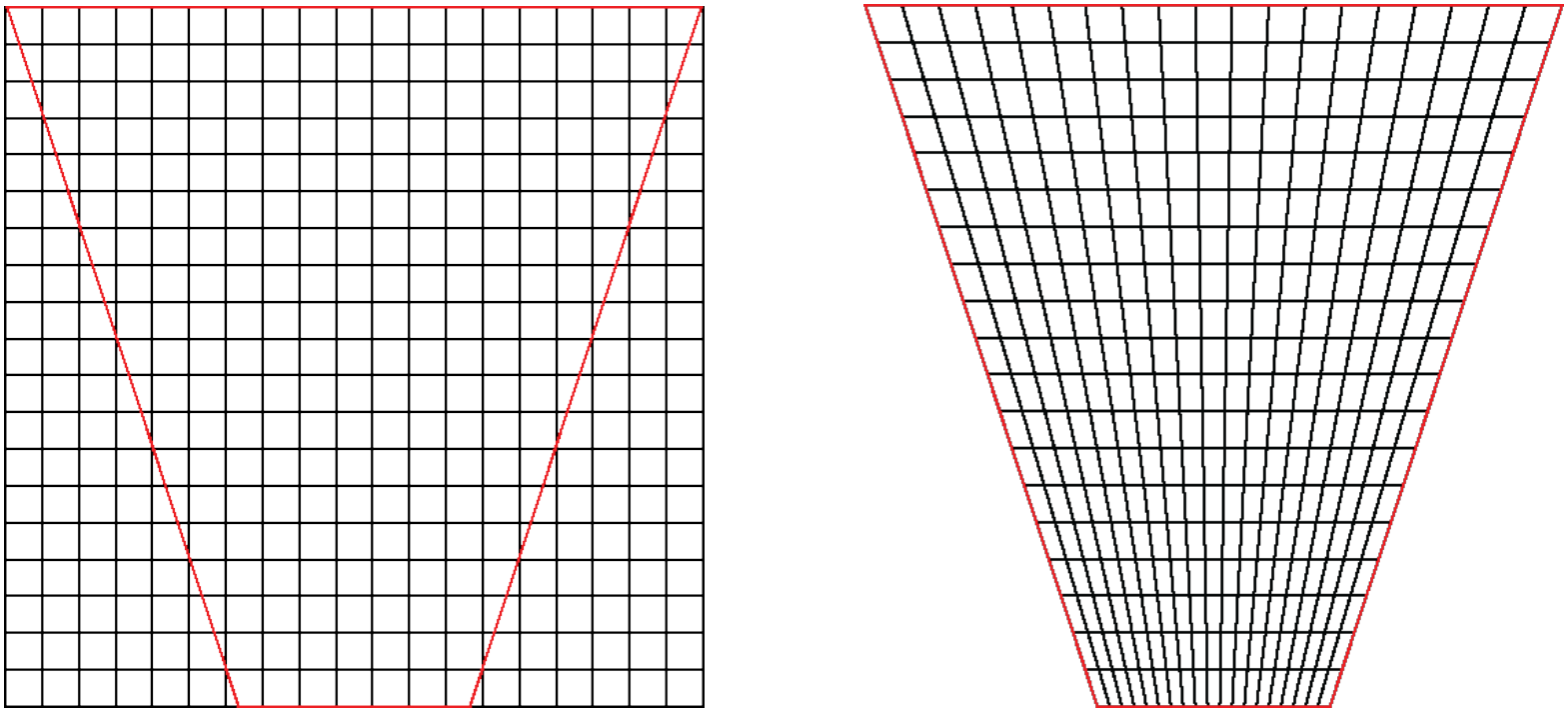

Abbildung: Warping der Texel in s-Richtung um effektive Auflösung zu erhöhen

Gliederung

Partitionierung: Idee

1. Teile Szene in »Regions-of-Interesst«
2. Erstelle je Partition eine Shadow Map
3. Schattentest je nach Partionierung durch einfache Indexfunktion abbildbar (Textur-Arrays!)

- Vorteile:
  - kann durch fixed-pipeline abgebildet werden.
    somit auch auf älterer Hardware nutzbar
  - auf vielen Grafikkarten hardwarebeschleunigt (inkl. PCF[1])

Beispiele: Cascaded Shadow Maps, Adaptive Shadow Maps ...

1

Percentage Closer Filtering ähnlich bilinearer Filterung um Aliasing zu reduzieren

z-Partitionierung

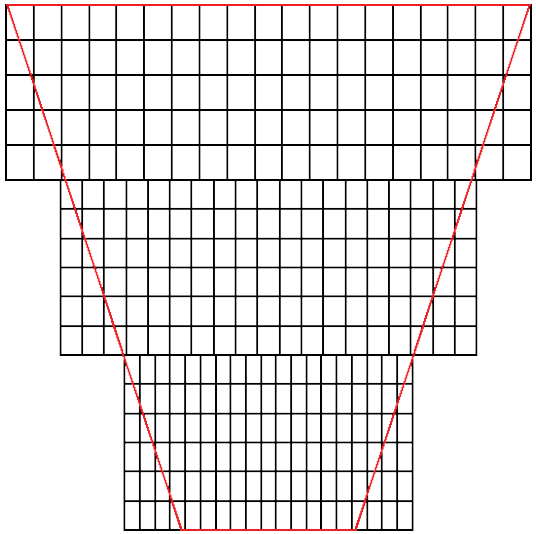

Abbildung: z-Partitionierung

- Teile Szene aus Kamerasicht in verschiedene z-Ebenen
- Bestimme für jede z-Ebene »enges« Shadow Frustrum
- einfach zu implementieren
- Cascaded Shadow Maps (für nicht interaktives Rendern genutzt)

Face-Partitionierung

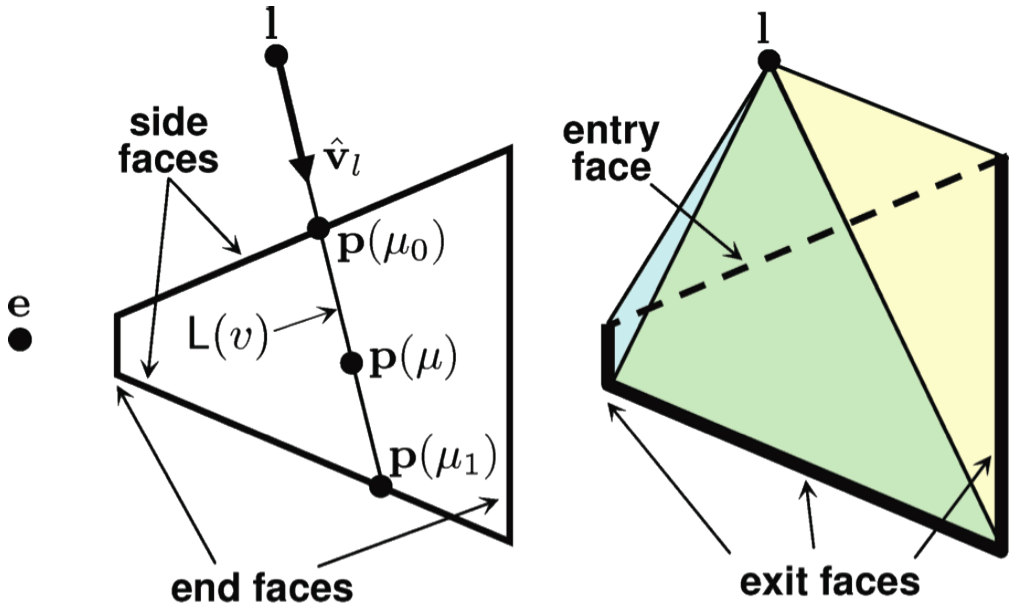

Abbildung: Face-Partitionierung [Lloyd2008, Fig.4, S.8]

- Erstelle Shadow Maps abhängig von Kamera- und Licht-Frustrum
- Geringe Anzahl an Shadow Maps (im Durchschnitt 3-4)
- Mit passendem Warping werden Aliasing-Fehler stark reduziert (siehe FP+LogPSM)

Gliederung

Fehler Quantifizieren

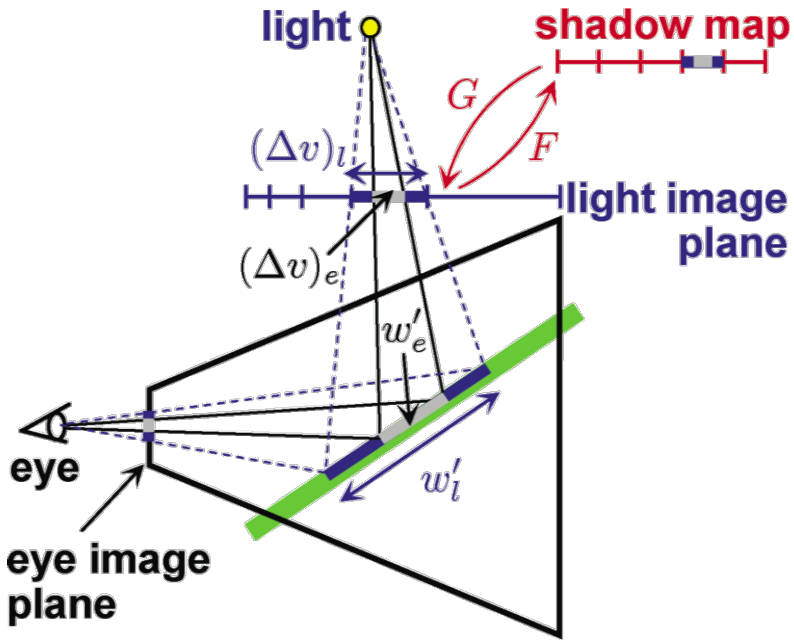

Abbildung: Metrik für Shadow Map [Lloyd2008, Fig.2, S.4]

Erhalte Metrik durch

$$m = \frac{r_j}{r_t}\frac{\mathrm{d}j}{\mathrm{d}t}$$

- $m > 1$ undersampled und $m < 1$ oversampled

r_j : Auflösung des Bildes
r_t : Auflösung der Shadow Map

Fehler Ableiten (geometrisch, 2D)

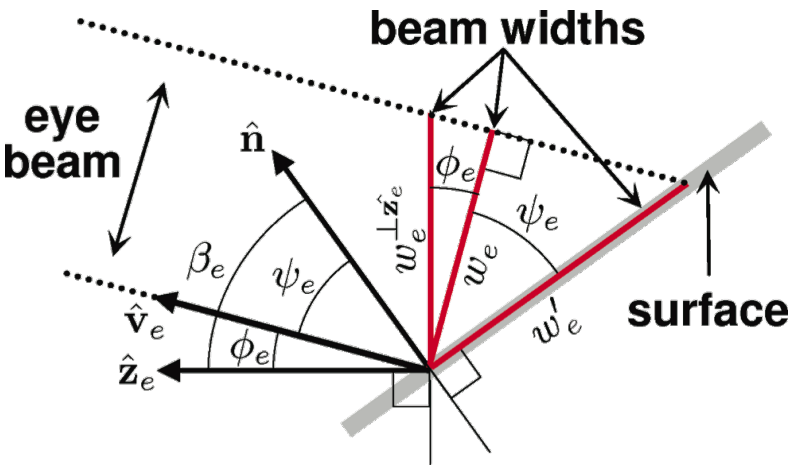

Abbildung: Geometrische Herleitung der Metrik m [Lloyd2008, Fig.3, S.6]

$$m = \frac{w'_l}{w'_e} \tag{1}$$

$$w_e^{\perp \hat{\mathbf{z}}_e} = \frac{W_e}{r_j}\frac{d_e}{n_e} \tag{2}$$

$$w'_e = \frac{w_e}{\cos\psi_e} = \frac{W_e}{r_j}\frac{d_e}{n_e}\frac{\cos\phi_e}{\cos\psi_e} \tag{3}$$

$$w'_l = \cdots = \frac{W_l}{r_t}\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\frac{\cos\phi_l}{\cos\psi_l} \tag{4}$$

$$m = \frac{r_j}{r_t}\underbrace{\frac{\mathrm{d}G}{\mathrm{d}t}\frac{W_l}{W_e}\frac{n_l}{n_e}\frac{d_l}{d_e}\frac{\cos\phi_l}{\cos\phi_e}\frac{\cos\psi_l}{\cos\psi_e}}_{=\mathrm{d}_j/\mathrm{d}_t}$$

Gliederung

Szenen-unabhängige Approximierung

$$m = \frac{r_j}{r_t} \underbrace{\left(\frac{\mathrm{d}G}{\mathrm{d}t}\right)}_{\delta_l} \underbrace{\left(\overbrace{\left(\frac{W_l}{W_e}\frac{n_l}{n_e}\frac{d_l}{d_e}\frac{\cos\phi_l}{\cos\phi_e}\right)}^{1/\tilde{\delta}_e} \frac{\cos\psi_l}{\cos\psi_e} \right)}_{1/\delta_e}$$

$\frac{\cos\psi_l}{\cos\psi_e}$ basiert auf Ausrichtung der Oberflächen in der Szene und kann i.A. nicht Parametrisiert werden.[2]

$$\tilde{m} = \frac{r_j}{r_t}\frac{\delta_l}{\tilde{\delta}_e} = \frac{w_e}{w_l}$$

- wollen: $\tilde{m} = 1$ über die gesammte Szene
- einfacher $\tilde{m} \leq 1$ (oversampling)

[2] In Ausnahmefällen möglich. Erlaubt 1-zu-1 matching von Schatten-Texeln zu Bild-Pixeln

Strenge untere Grenze

- suchen δ_b die strenge untere Grenze der perspektivischen Faktoren von δ_e darstellt:

$$\delta_b(v) \in \left[\frac{1}{C}\min\left(\tilde{\delta}_e(v)\right), \min\left(\tilde{\delta}_e(v)\right)\right], C > 1$$

- Nutze Face-Partitionierung um δ_b für einzelne Faces zu bestimmen.
- Parametrisierung aus δ_b immer noch zu komplex. Liefert aber untere Schranke für weitere Approximation

Gliederung

Übersicht End- und Side-Faces

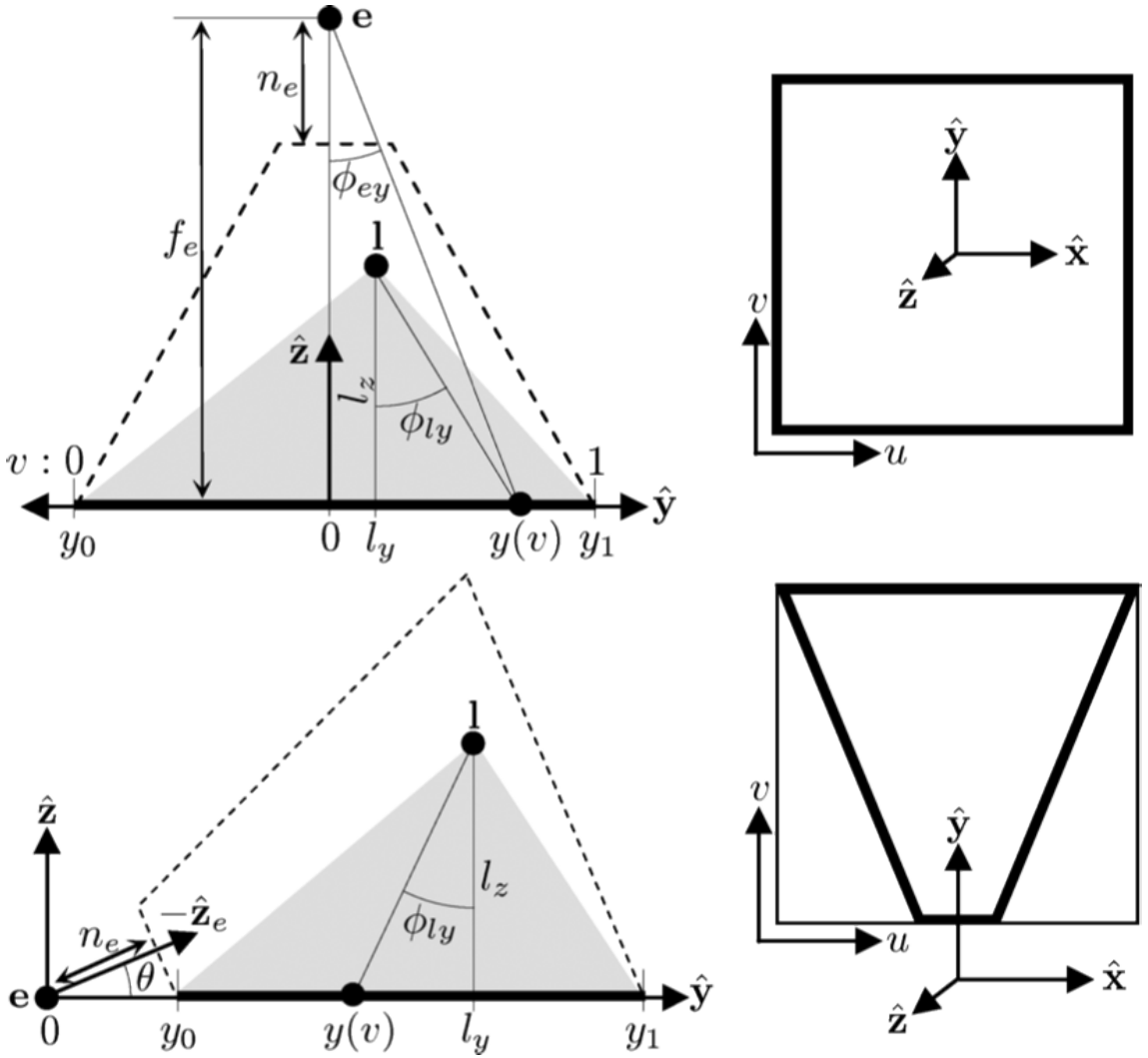

Abbildung: End- und Side-Face [Lloyd2008, Fig.5, S.9]

Parametrisierungen

Gleichverteile Parametrisierung führt zu Uniform-Shadow-Maps:

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{un}(u,v) = \begin{bmatrix} s \\ t \end{bmatrix} \tag{5}$$

Perspektivische Parametrisierung wie in PSM und LiPSM:

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{p}(u,v) = \begin{bmatrix} \frac{p_0 x(u)+p_1(y(v)+a)}{y(v)+a} \\ \frac{p_2(y(v)+a)+p_3}{y(v)+a} \end{bmatrix} \tag{6}$$

$$p_0, p_1, p_2, p_3 = \dots$$

Logarithmische Parametrisierung in t-Richtung

$$t = \mathbf{F}_{l}(v) = \frac{\log\left(\frac{y(v)+a}{y_0+a}\right)}{\log\left(\frac{y1+a}{y_0+a}\right)} \tag{7}$$

Gliederung

LogP-Parametrisierung

Kombiniere Logarithmische- und Perspektivische Parametrisierung ([6] und [7]):

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{lp}(u,v) = \begin{bmatrix} F_{p,s}(u,v) \\ c_0 \log\left(c_1 F_{p,t}(u,v) + 1\right) \end{bmatrix} \tag{8}$$

$$c_0 = \frac{-1}{\log\left(\frac{y_1+a}{y_0+a}\right)}$$

$$c_1 = -\frac{y_1 - y_0}{y_1 + a}$$

- bietet gute Approximierung von δ_b sowohl in s- als auch in t-Richtung

Vergleich F_{un} mit F_{lp}

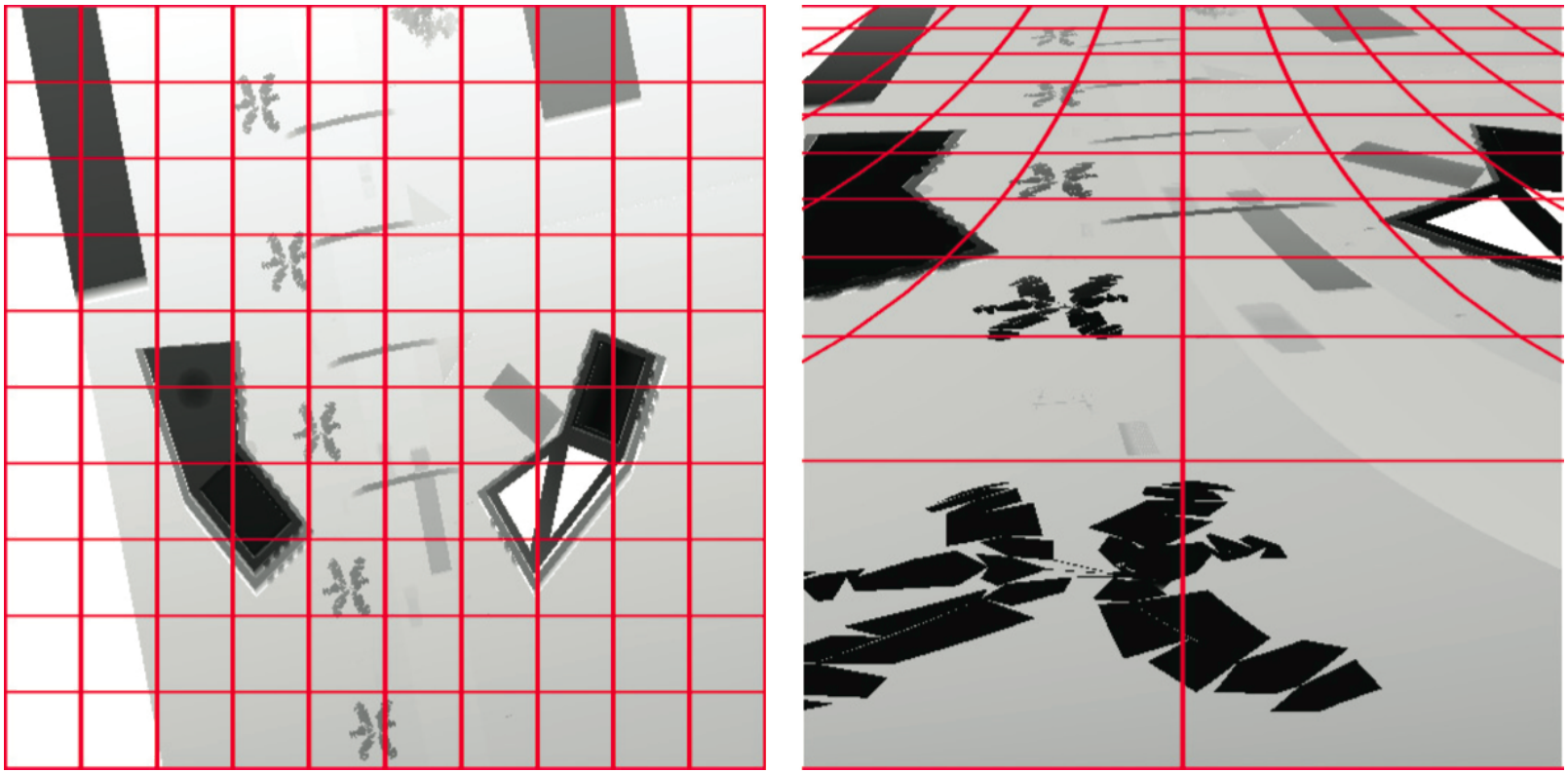

Abbildung: log-perspektivische Rasterisierung nach F_{lp} [Lloyd2008, Fig.8, S.15]

Error Bound

Tabelle: Maximale Fehler [Lloyd2008, Table III, S.11]

Parametrization	R_s^{end}	R_s^{side}	R_t^{side}	S^{side}
Error Bound (F_b)	$O(1)$	$O(1)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$
Uniform (F_{un})	$O(1)$	$O\left(\frac{f_e}{n_e}\right)$	$O\left(\frac{f_e}{n_e}\right)$	$O\left(\left(\frac{f_e}{n_e}\right)^2\right)$
Perspective (F_p)	-	$O(1)$	$O\left(\sqrt{\frac{f_e}{n_e}}\right)$	$O\left(\frac{f_e}{n_e}\right)$
Logarithmic (F_l)	-	-	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$	-
LogPSM (F_{lp})	-	$O(1)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$

Mit kritischen Auflösungen:

$R_t = \max_v \left(\frac{\delta_l(v)}{\delta_b(v)}\right)$

$R_s^{side} = \max_{(u,v)\in\mathcal{F}} \left(\frac{\delta_l(u,v)}{\delta_b^{side}(u,v)}\right)$

n_e, f_e near- und far-Plane

Storage Faktor:

$S = R_s \times R_t$

Gliederung

LogPSM Algorithmus

1. Berechne Parametrisierung der Shadow Maps für jede Partition. Für Side-Faces LogPSM, für End-Faces UniformSM
2. Verteile Auflösung basierend auf maximalem Fehler der Partitionen.
    1. Bestimme je Partition δ_l/δ_b und so R_s, R_t und S.
    Bei direktionalen Lichtquellen ist Fehler über gesammte Partition monoton. Es genügt Auswertung an den Vertices.
    Bei Punktlichtquellen Fehler Approximieren durch Abstandsvektor von Face und Lichtquelle sowie Auswertung an den Vertices der Partition.
    2. Verteile innerhalb Partition Auflösung nach R_s und R_t
3. Rendere Shadow Maps
4. Rendere das Bild mit Shadow Maps

Optimierung der Darstellung

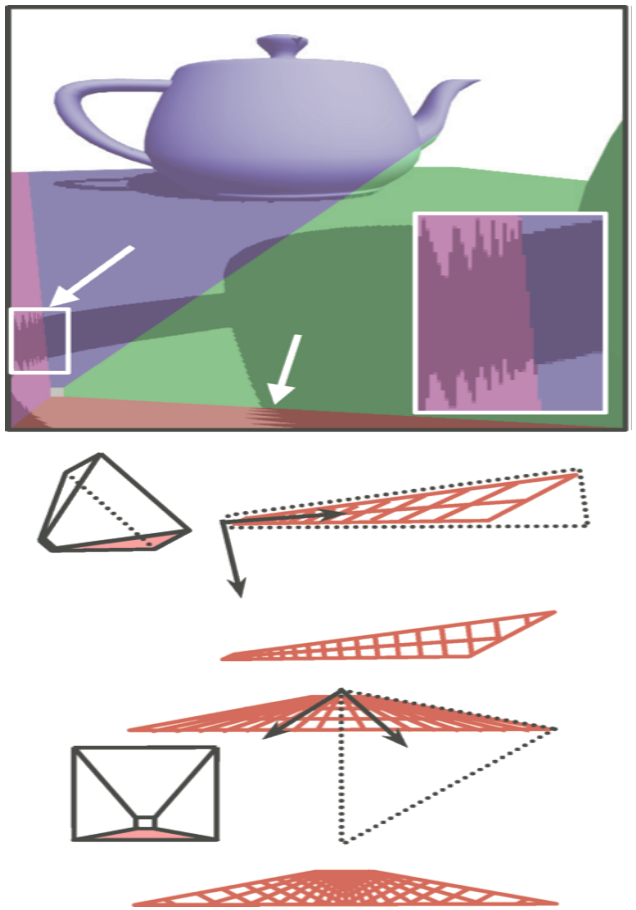

Abbildung: FPc und FPcs
[Lloyd2008, Fig.10, S.16f]

Verteilung der Auflösung kann zu Scherungsartefakten führen.

FPc Face-Partitionierung mit coordinate frame Anpassung

FPcs Face-Partitionierung mit coordinate frame Anpassung und Face splitting

Gliederung

Implementierung

- Aktuelle Hardware unterstützt nur lineare Rasterisierung.
- Hardwareerweiterung kann LogPSM zu vergleichbaren Geschwindigkeiten mit anderen Shadow Map Algorithmen helfen.
- LogPSM benötigt wesentlich geringere Auflösung um Ergebnisse vergleichbar mit anderen SM Algorithmen zu liefern.
- Log-Transformation muss emuliert werden:

Log-Transformation

Partitioning

- Hohe Anzahl an Partitionen simuliert Log-Transformation
- entspricht Cascaded Shadow Maps
- Verwaltungsaufwand

Vertex-Warping

- Hohe Anzahl an Vertices
- Adaptive Anpassung durch geometry-Shader
- Viele Vertextransformationen nötig

Fragment-Discarding

- Optimales Ergebnis
- Discard verringert Leistung
- um Faktor 10 langsamer als normales Sampling

Gliederung

Vergleich: Aliasing Fehler

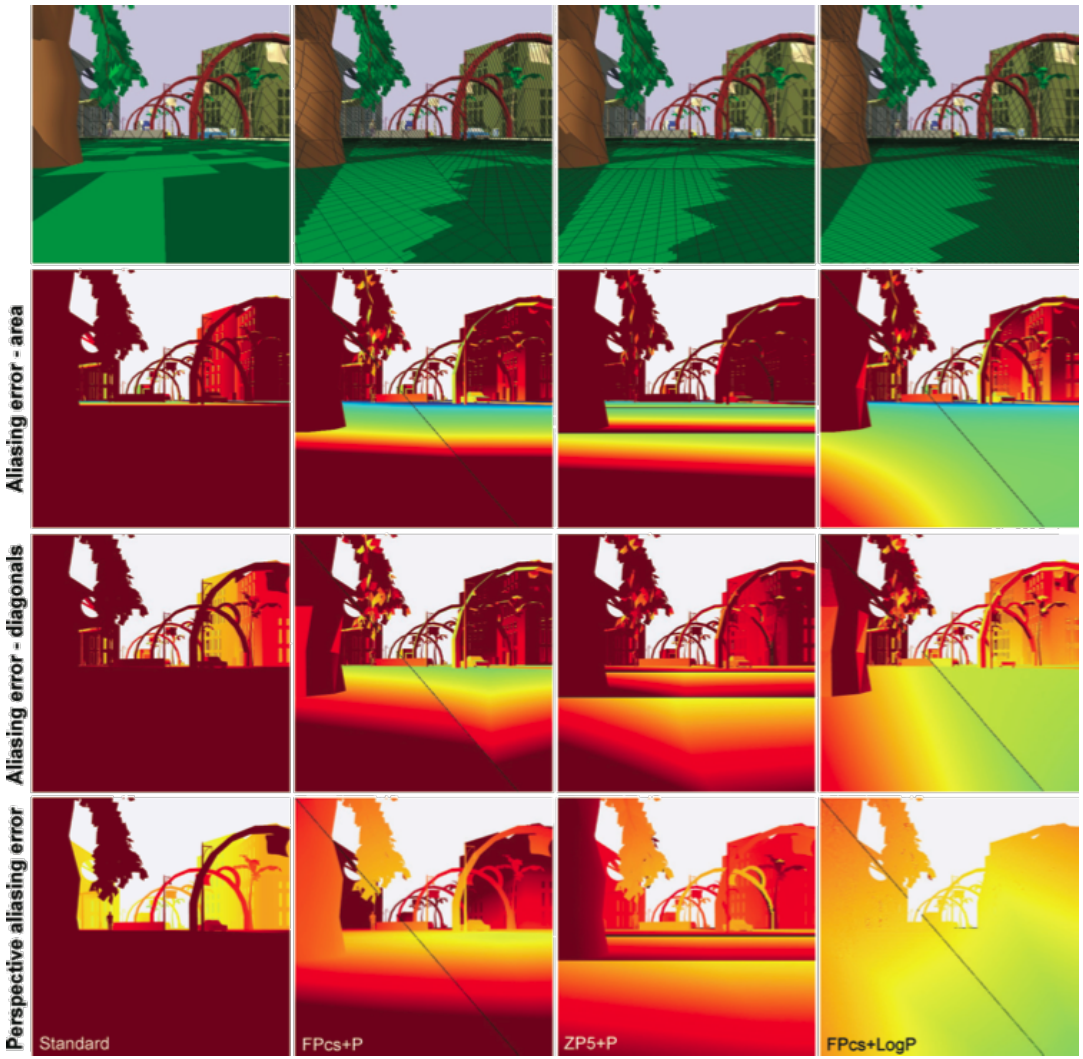

Abbildung: Farbkodierte Darstellung der Metrik m [Lloyd2008, Fig.12, S.18]

Vergleich: Benötigte Auflösung

Tabelle: Storage Faktor für direktionale Lichtquelle über alle Eintrittswinkel [Lloyd2008, Fig.12, S.18]

Algorithmus	$\min S$	$\max S$	$\min / \max$	$\varnothing S$	$\varnothing \#$
Standart	8.65×10^5	2.00×10^6	2.31	1.32×10^6	1
FPcs+P	865	3170	3.66	1880	3.8
ZP5+P	12.9	159	12.3	53.7	5
FPcs+LogP	5.98	27.7	4.62	15.6	3.8

Zusammenfassung

Vorteile:

- Storage Faktor: $O\left(\log\left(\frac{f_e}{n_e}\right)\right)$ für beliebige Szene
- Verfahren mit bisher niedrigsten Aliasing Fehlern (ausgenommen Raytracing-ähnliche-Algorithmen)
- Lässt sich Problemlos mit verschiedenen Partitionierungs-Verfahren nutzen

Nachteile:

- Auf aktueller Hardware zu langsam: 2-3 fps[3]
- Spezielle Hardwareerweiterung notwendig (Logarithmische Rasterisierung)

Ansätze können auch für andere Shadow Mapping Algorithmen von Vorteil sein. Siehe Appendix [Lloyd2008, S.23ff]

[3] GeForce 7800GT; 2 GHz processor [Lloyd2008, S.15]

Ende

Vielen Dank für Ihre Aufmerksamkeit

Referenzen


Brandon Lloyd, Naga Govindaraju, Cory Quammen, Steve Molnar, Dinesh Manocha.
Logarithmic Perspective Shadow Maps
ACM Trans. Graph. 27, 4, Article 106 (October 2008)


L. Williams.
Casting curved Shadows on curved surfaces
ACM SIGGRAPH Computer Graphics. Vol. 12. 270–274 (1978)


Stamminger, Drettakis.
Perspective Shadow Maps
ACM SIGGRAPH. 557–562. (2002)


Stefan Brabec, Thomas Annen, Hans-Peter-Seidel.
Practical Shadow Mapping
J. Graph. Tools 7 (2002)