Hard Shadows

Filtering

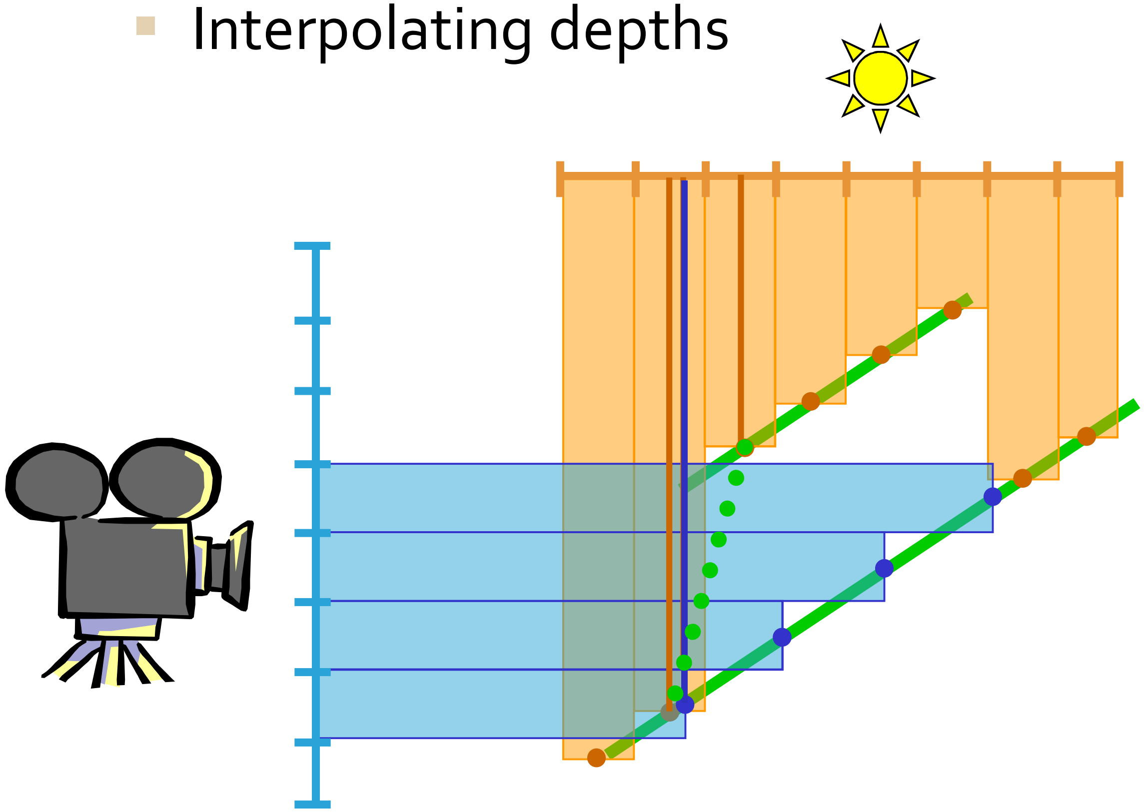
Shadow Map Filtering

Unfiltered

3x3 PCF

Shadow Map Filtering

- Depth Values are not “blendable”
  - Traditional bilinear filtering is inappropriate
  - Interpolating depths



3.2	1.5

linearFilter(3.2,1.5) = 2.8

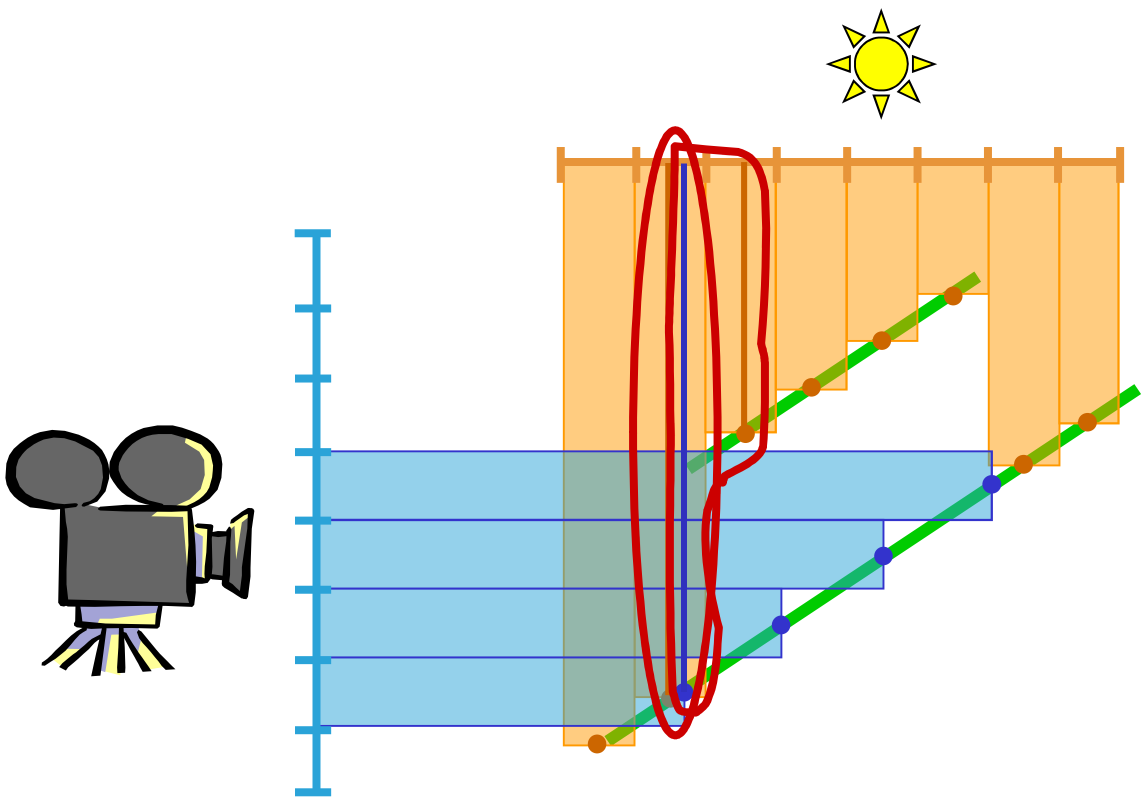
2.8 < 3

shadowed

Percentage Closer Filtering

[Reeves et al. 1987]

- Average comparison results, not depth values



3.2 > 3	1.5 > 3

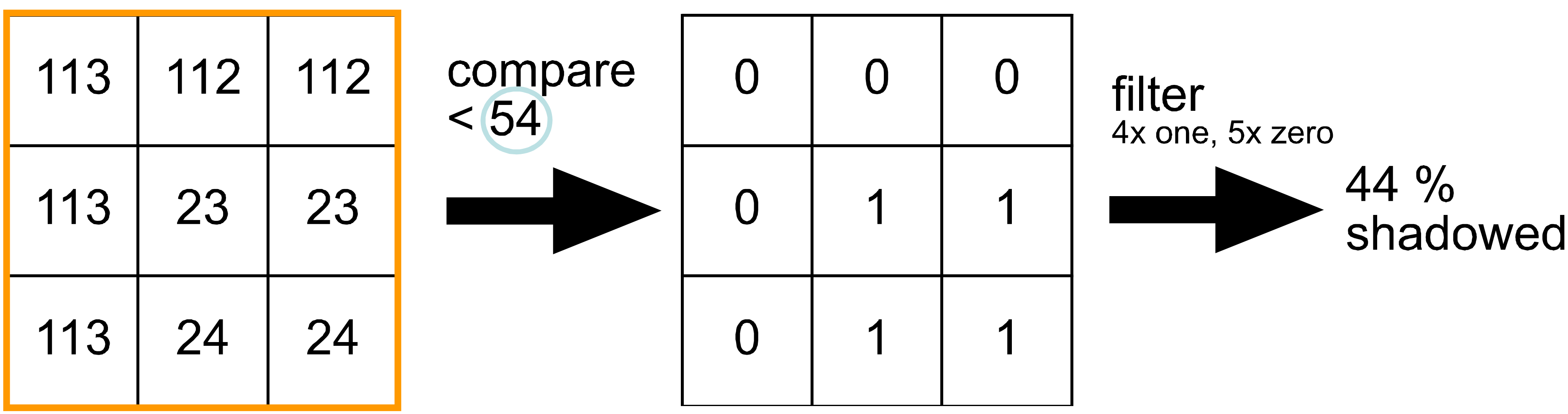
1(lit) 0 (shadowed)

(1+0)/2 = 0.5

50% shadowed

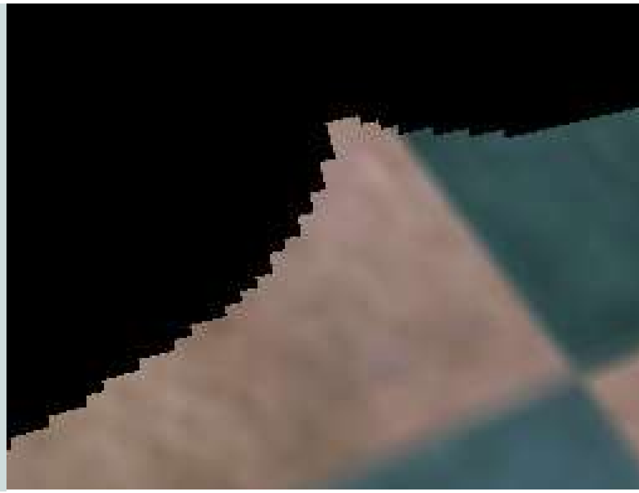
Percentage Closer Filtering - Practice

- 2D filter kernel
- Here 3x3
- Need shadow test result before filtering
  - Pre-filtering does not work
- Depth texture uses 2x2 PCF if GL_LINEAR



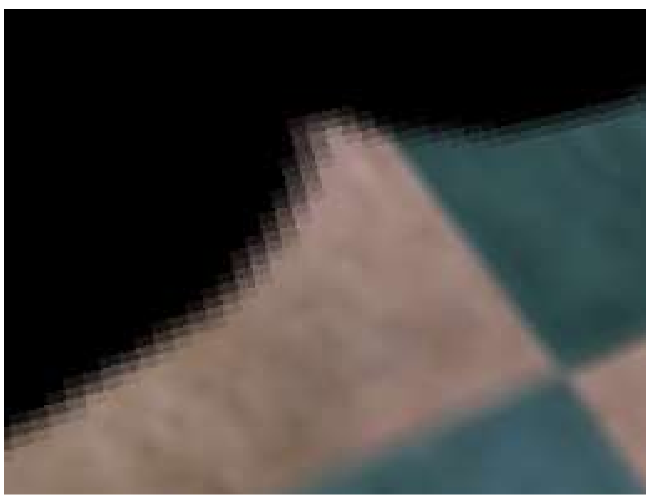
Percentage Closer Filtering - Results

scene



standard shadow maps

PCF 2x2



PCF 3x3

PCF 4x4

PCF 4x4 + bilinear lookups

PCF + Bilinear Lookups

- Nearest neighbour lookups
  $\longrightarrow$ quantization artifacts
- Bilinear lookups
  - For 2x2 kernel size straight forward
  - For bigger kernel sizes
    - Poisson disk + bilinear lookups (hack)
    - Exact bicubic convolution using 2x2 bilinear lookups [Hadwiger, GPU Gems 2]

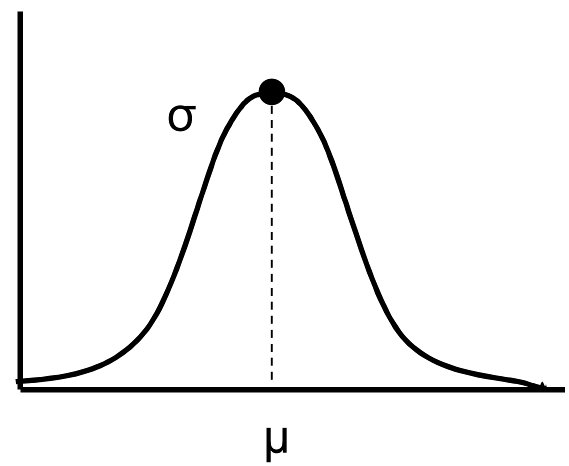
Percentage Closer Filtering

- Good quality needs big kernel size
  - Bandlimiting for oversampled areas!
- Big kernel size is slow (quadratic growth)
- Pre-filtering desirable
  - Only filter once
  - Less texture lookups
  - Big kernels cheaper

Variance Shadow Maps
[Donnelly and Lauritzen 2006]

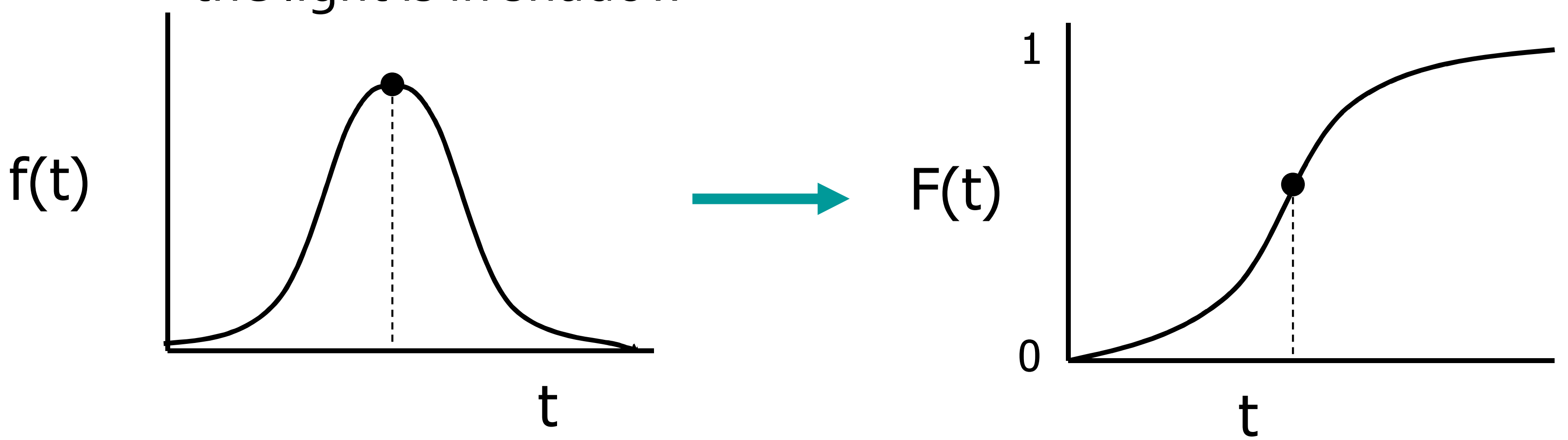
- Estimate PCF outcome with statistics
  - A representation that filters linearly
  - Use **mean** and **variance** of depth samples inside kernel
  - Can be calculated from linearized depth map

4	4	1	1	1
4	1	1	1	4
1	1	1	1	4
1	4	4	4	4



Variance Shadow Maps
[Donnelly and Lauritzen 2006]

- Estimate PCF outcome with statistics
  - Really want a cumulative distribution function (CDF) of a set of depths
  - F(t) = P(x≤t)
  - F(t) is the probability that a fragment at distance “t” from the light is in shadow



Variance Shadow Maps
[Donnelly and Lauritzen 2006]

- **Mean** = $\mu = E(x)$
- **Variance** = $\sigma^2 = E(x^2) - E(x)^2$

→ Approximate fraction of distribution that is more distant than shaded point d (*P(x >= d)*)

- Prefiltering (for a certain kernel size)
  - $E(x)$ on kernel simple to calculate from input depth texture
  - $E(x^2)$ on kernel simple to calculate from input depth texture squared
- Estimate PCF shadow test outcome with Chebyshev's Inequality

Variance Shadow Maps
[Donnelly and Lauritzen 2006]

- Chebyshev's inequality
  - Given a shadow map depth value distribution with mean and variance (*for certain kernel*), the probability P(x >= d) that a random depth value z drawn from this distribution (*this kernel*) is greater or equal to a given depth value d (*current fragment depth*) has an upper bound of

$$p(d) = \frac{\sigma^2}{\sigma^2 + (d-\mu)^2} \quad \geq \quad P(x \geq d)$$

- Percentage of fragments over the filter region that are more distant than the current fragment

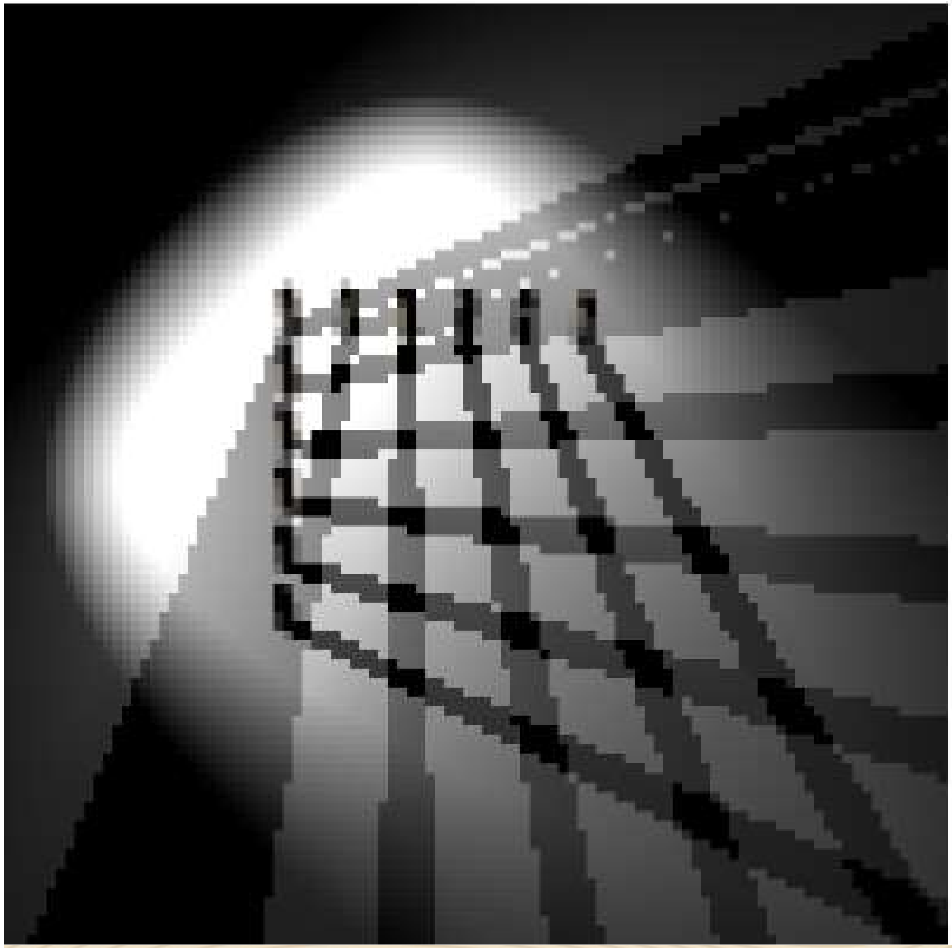
Variance Shadow Maps
[Donnelly and Lauritzen 2006]

- Inequality only gives an upper bound
  - Becomes equality for single planar occluder and receiver
  - In small neighbourhood
    occluder and receiver have constant depth and
    thus $p(d)$ will provide a close approximation to $P(x >= d)$
- In practice
  - Store $E(x)$ and $E(x^2)$
  - Pre-filter for certain kernel (mipmapping!)
  - Rendering: evaluate $p(d)$ for each fragment

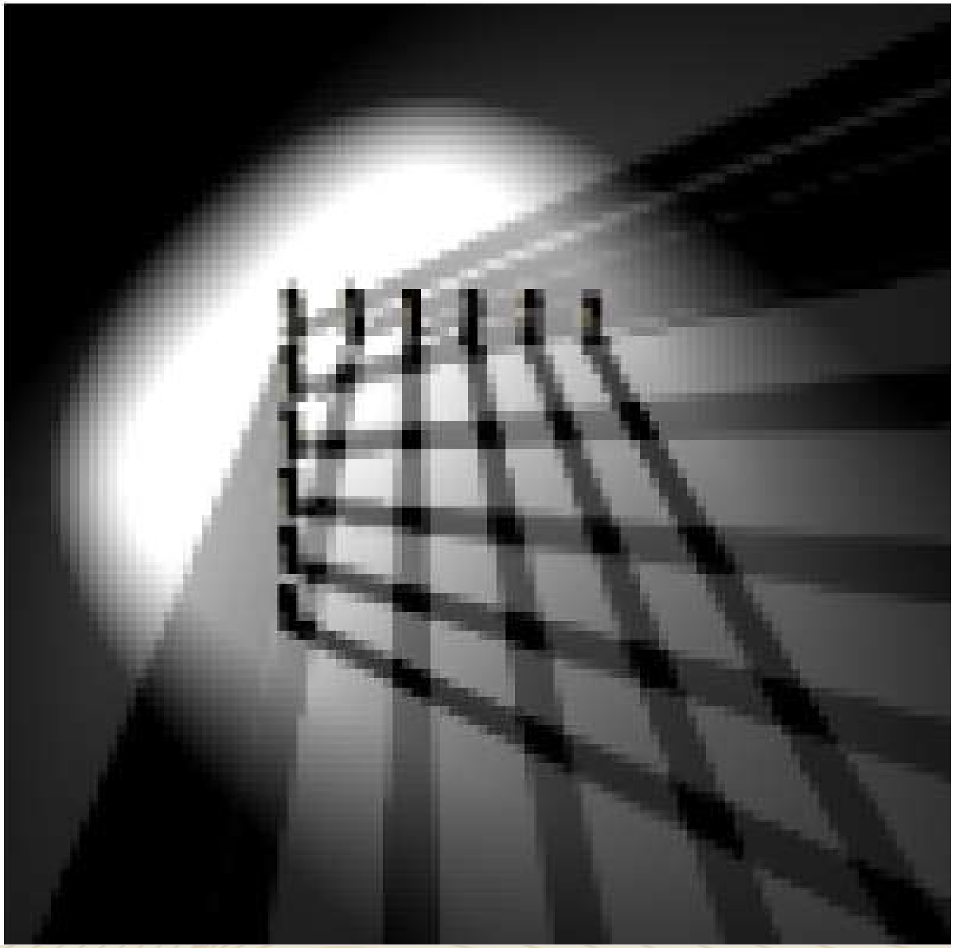
Variance Shadow Maps

[Donnelly and Lauritzen 2006]

shadow Map

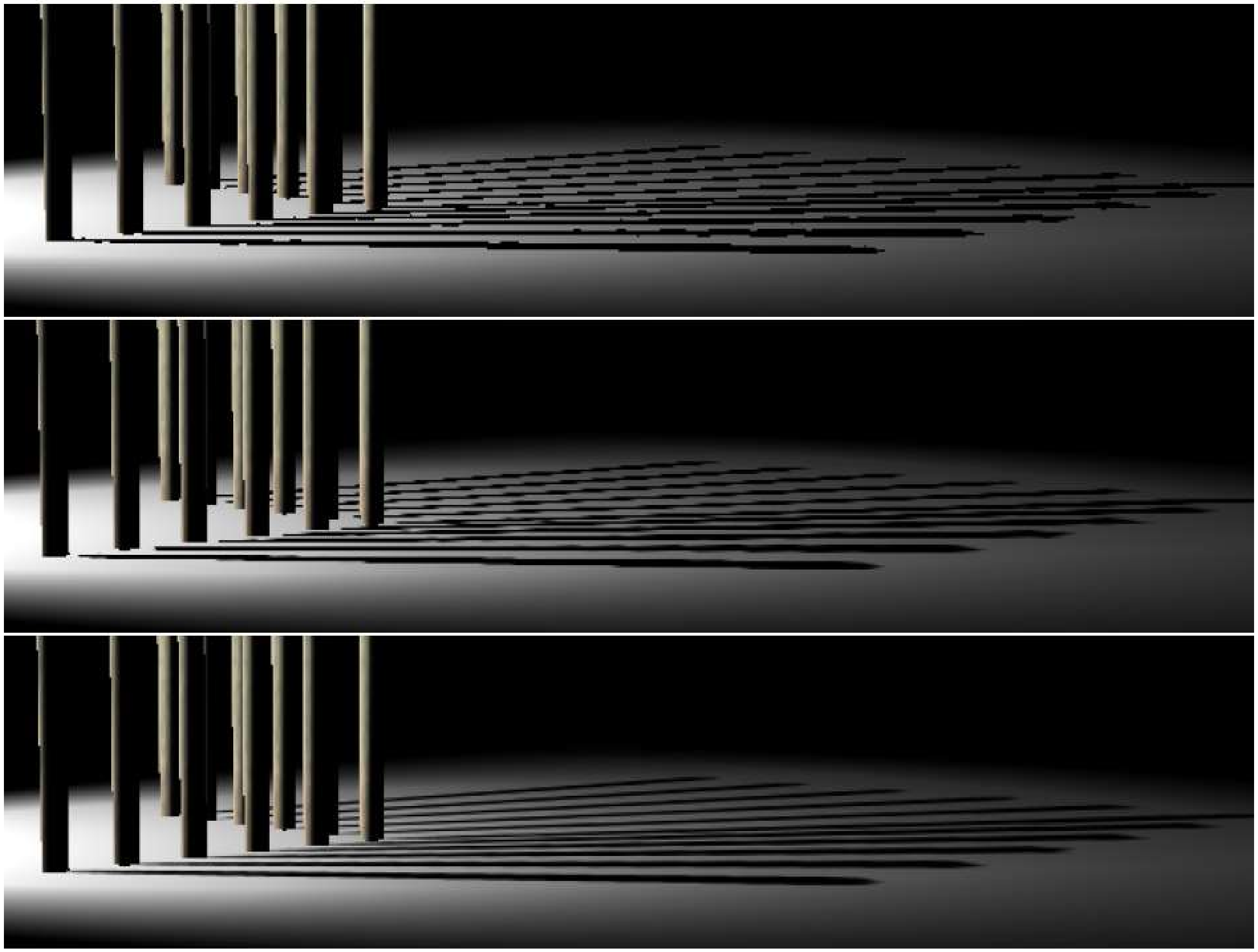


variance Shadow Map



Variance Shadow Maps

[Donnelly and Lauritzen 2006]



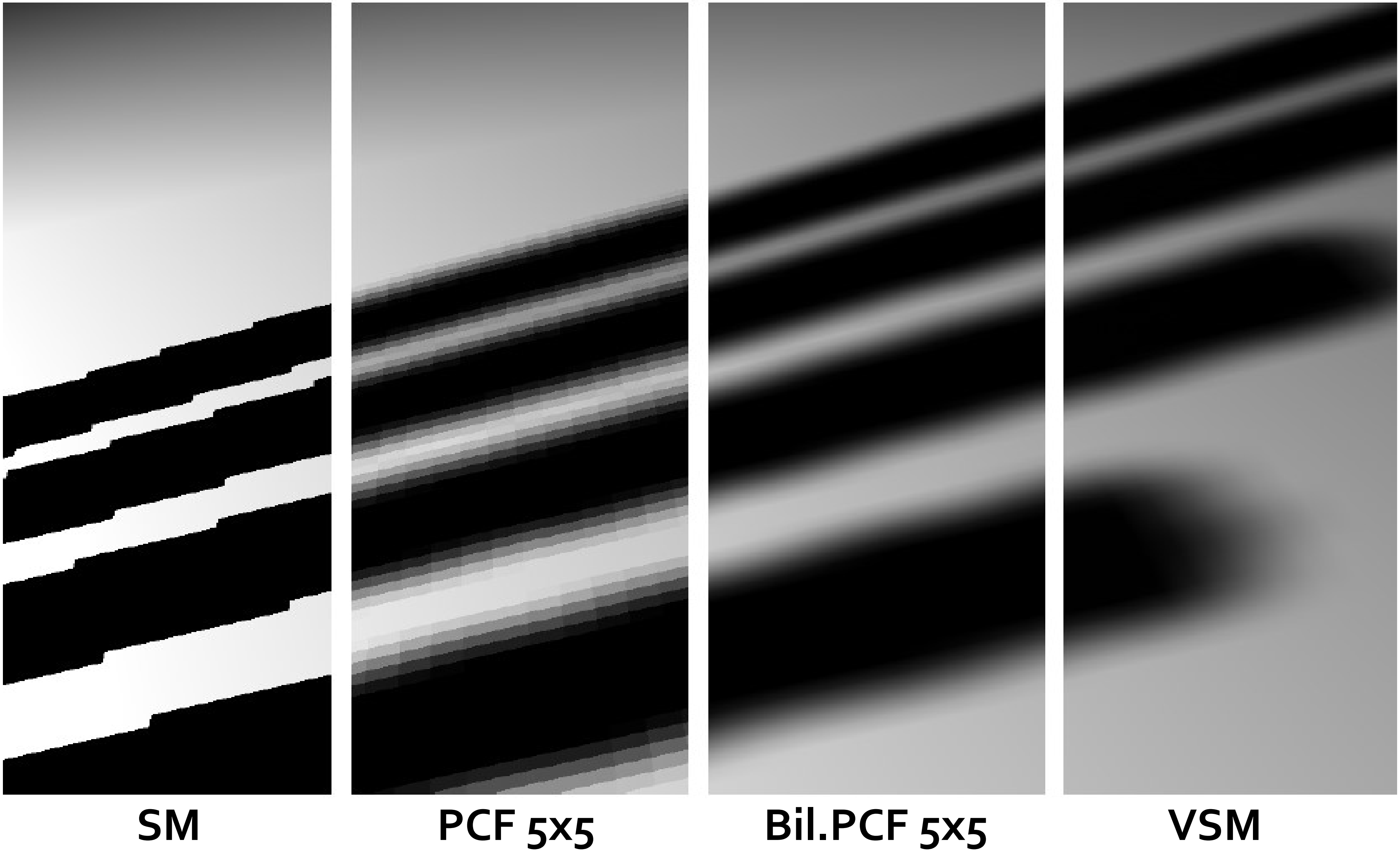
shadow Map

bilinear PCF

variance Shadow Map

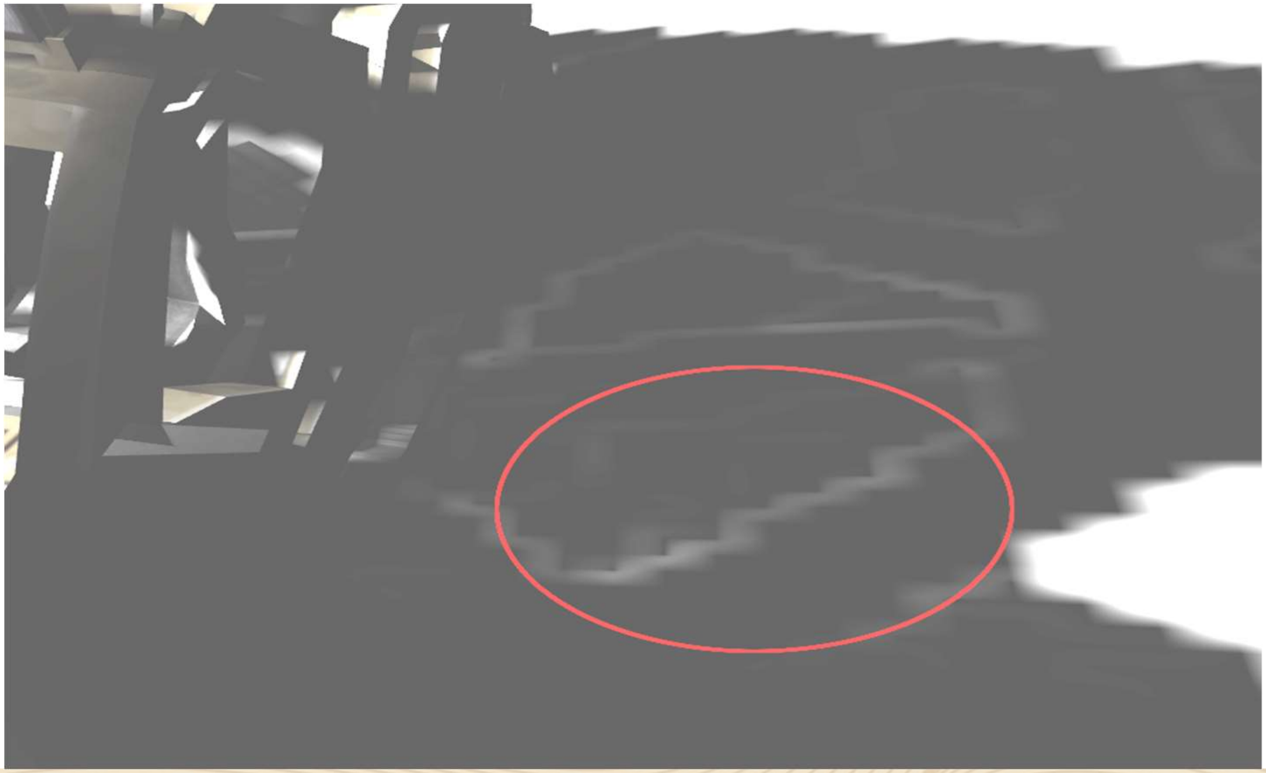
Variance Shadow Maps

[Donnelly and Lauritzen 2006]

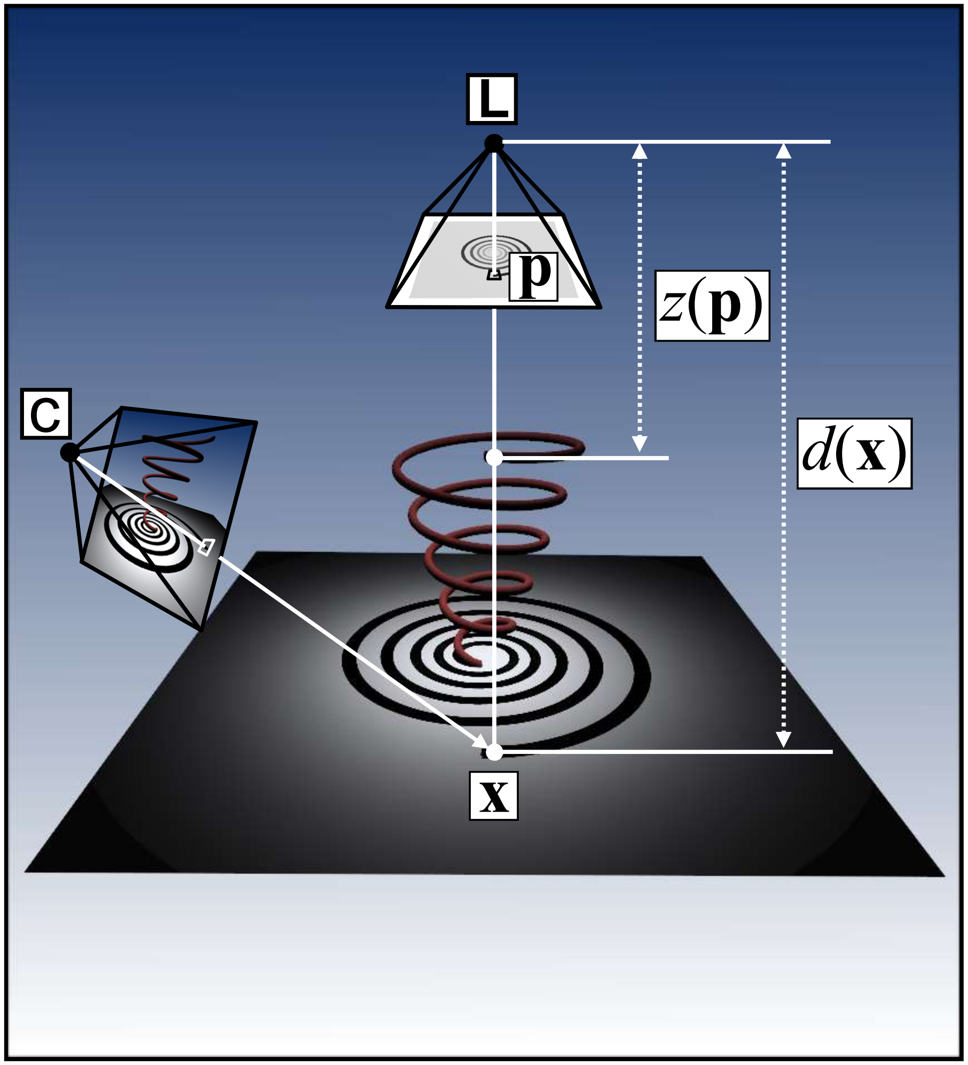


Variance Shadow Maps
[Donnelly and Lauritzen 2006]

- P(d) works in many situations
- When σ^2 is large "light bleeding"
  - "Layer" several VSM to alleviate these problems

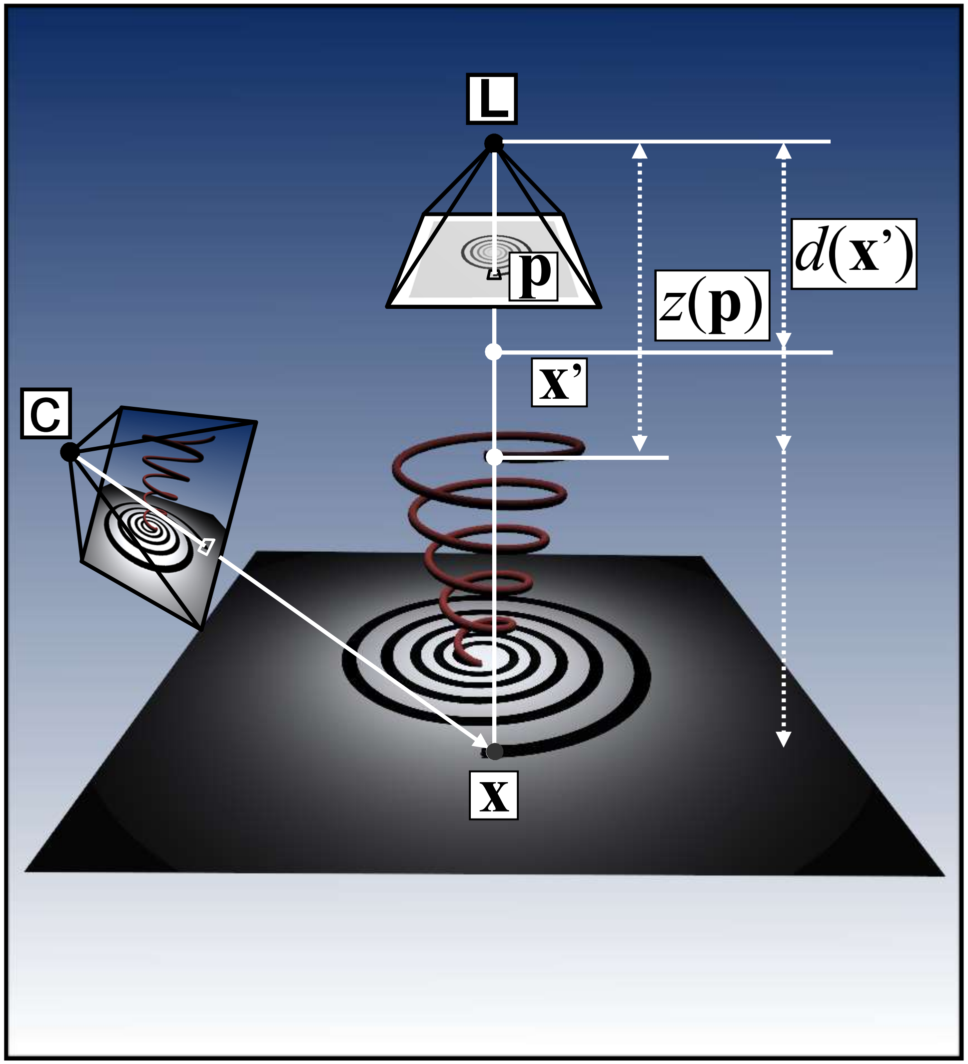


Revisiting the Shadow Map Test

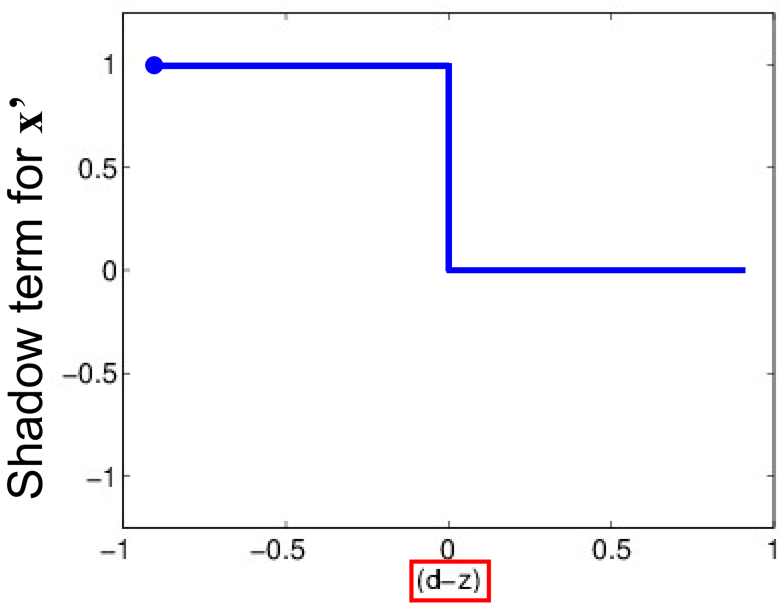


- $\mathbf{x} \in R^3$
- $\mathbf{p} \in R^2$
- $\mathbf{x}$ equals $\mathbf{p}$ just in different spaces
- Shadow function: $s(\mathbf{x}) := f(d(\mathbf{x}), z(\mathbf{p}))$
- Binary result:
  - 1 if $d(\mathbf{x}) <= z(\mathbf{p})$
  - 0 else (shadow)

Shadow Test Function: $s(\mathrm{x})$



- What kind of function is $s(\mathbf{x})$?

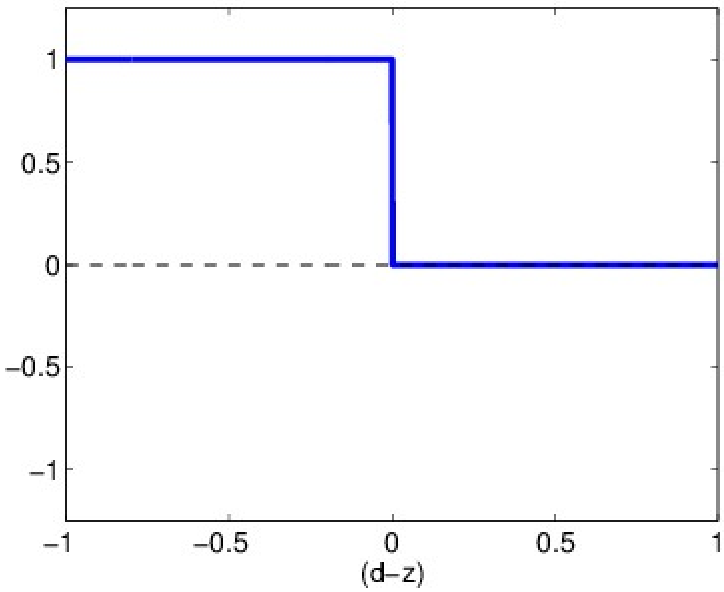


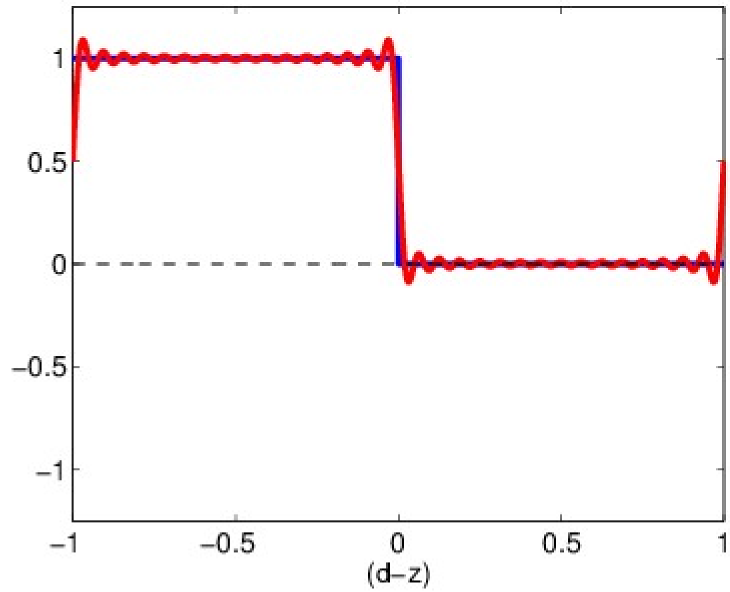
- Heaviside Step Function: $H(d\text{-}z)$

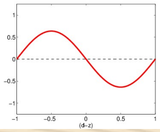
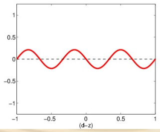
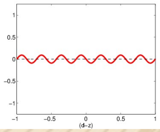
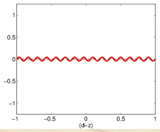
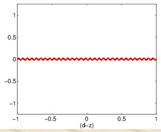
Convolution Shadow Maps

[Annen et al. 2007]

- Shadow test is a step function
- Idea: transform depth map such that we can write the shadow test as a sum
- Use convolution formula with Fourier expansion

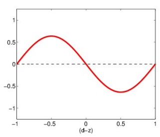
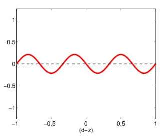
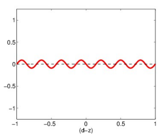
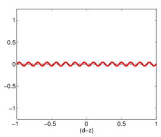
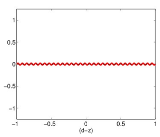




$H(t) \approx c_1$  $+c_2$  $+..+c_4$  $+..+c_8$  $+..+c_{16}$ 

Important Properties of a Fourier Series

- Step function becomes sum of weighted sin() and cos()

$H(t) \approx c_1$  $+c_2$  $+..+c_4$  $+..+c_8$  $+..+c_{16}$ 

- Series is separable!

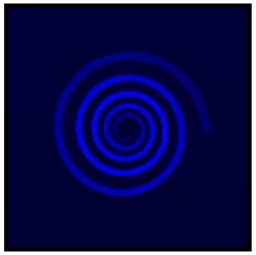
$$\sin(d-z) = \boxed{\sin(d)}\boxed{\cos(z)} - \boxed{\cos(d)}\boxed{\sin(z)}$$

  - One term depends on shadow map
  - One term depends on lookup value
  - Pre-filtering possible!

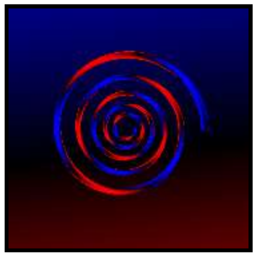
Filtering Example

Original

$s(\mathbf{x}) \approx$

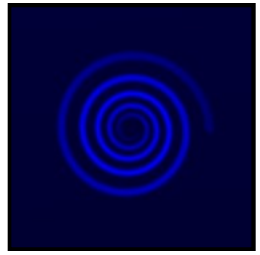
$a_1(d)$  $+..+a_4(d)$ 

$+\ldots+$

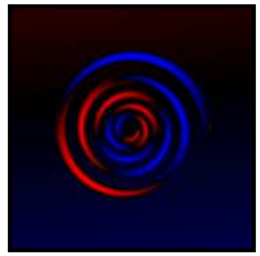
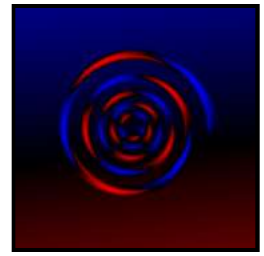
$a_8(d)$  $+..+a_{16}(d)$ 

After filtering

$s_f(\mathbf{x}) \approx$

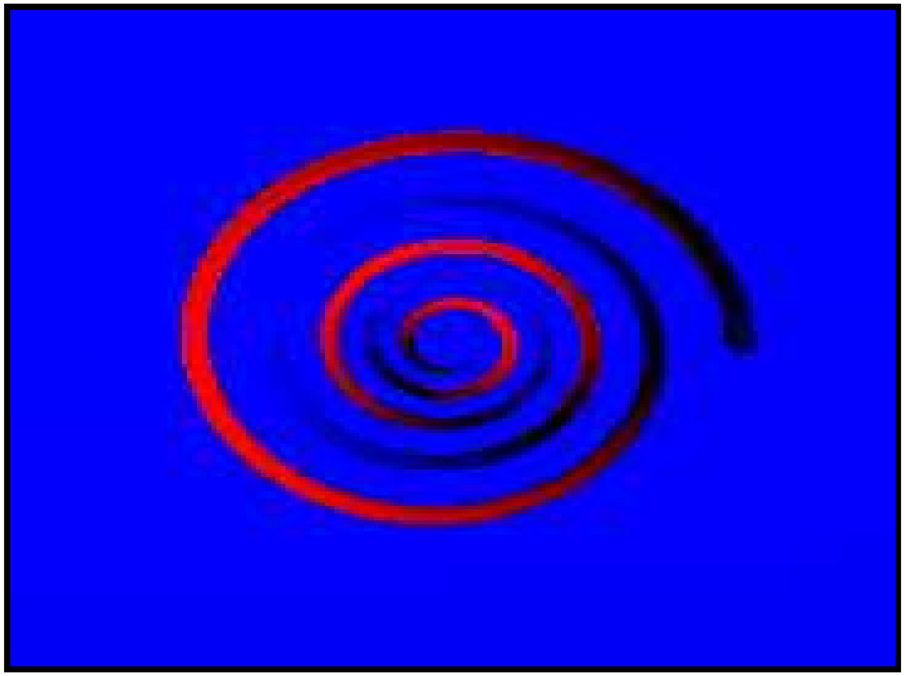
$a_1(d)$  $+..+a_4(d)$ 

$+\ldots+$

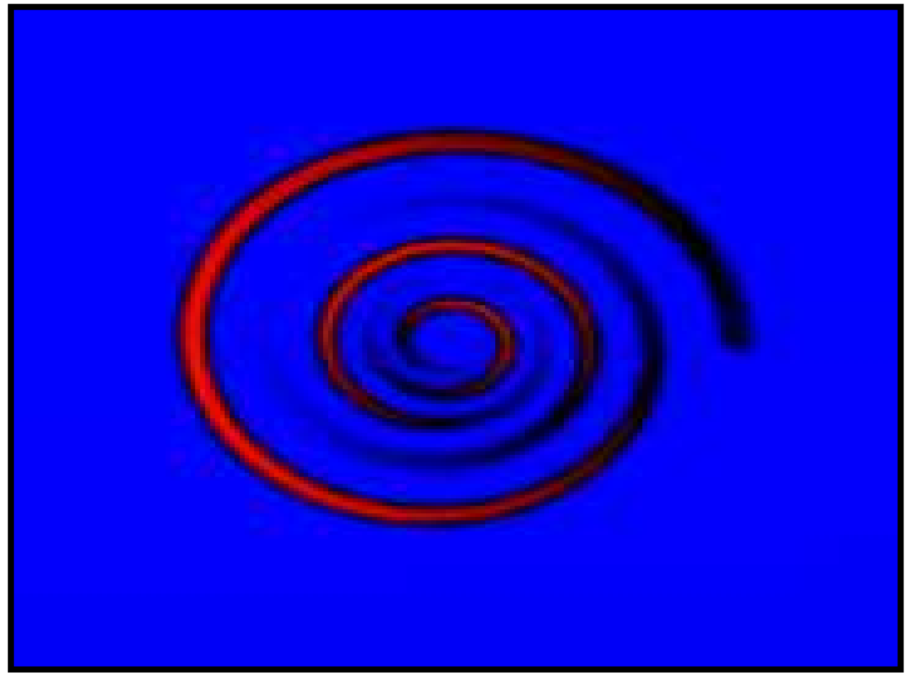
$a_8(d)$  $+..+a_{16}(d)$ 

Filtering Example

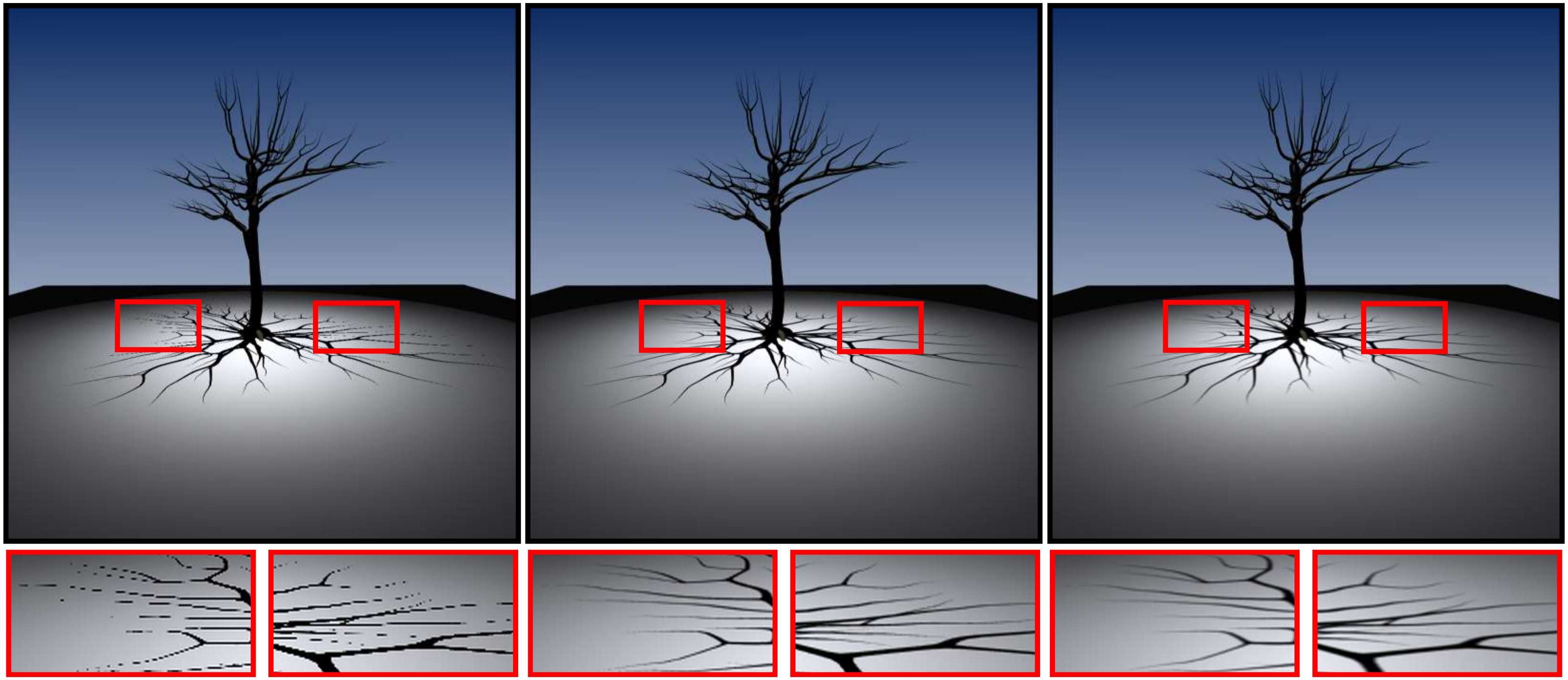
original



after filtering



Mipmapped CSM recovers fine details (SM: 2048^2)



PCF **(hardware)**

CSM

CSM – 7x7 Gauss

CSM Blurred Shadows

Filter size: 3X3

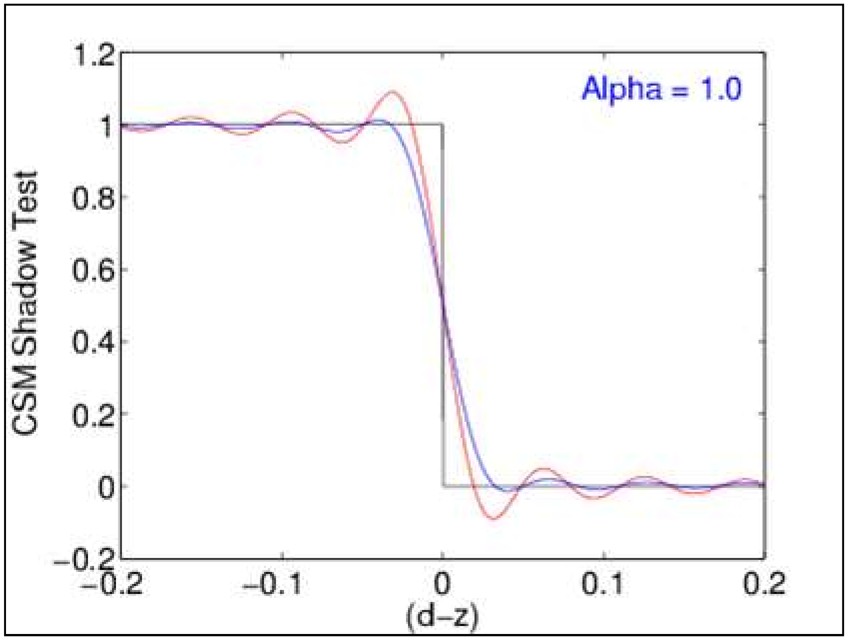
SM: 128^2 256^2 512^2 1024^2

Filter size: 7X7

SM: 128^2 256^2 512^2 1024^2

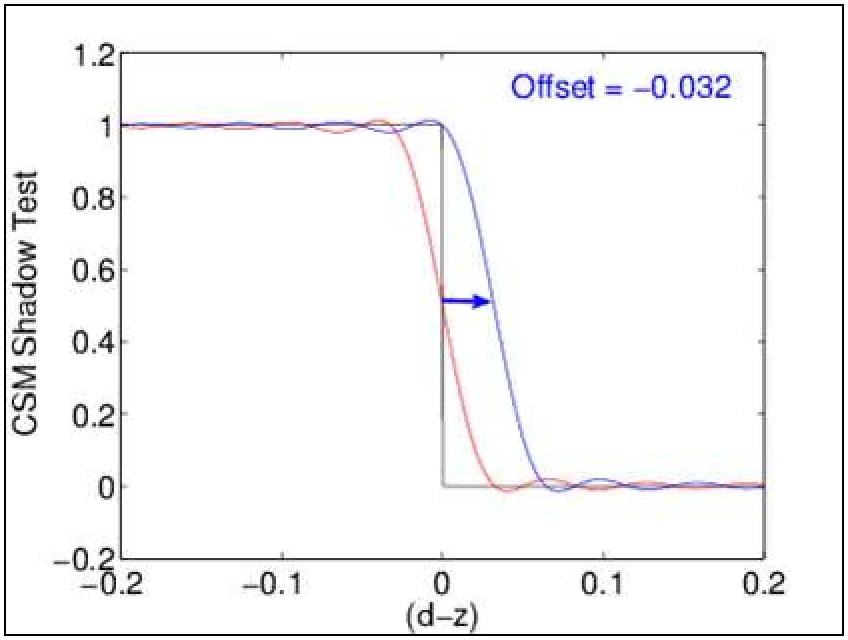
Issues with a Fourier series

- Ringing suppression
  - Reduce higher frequencies



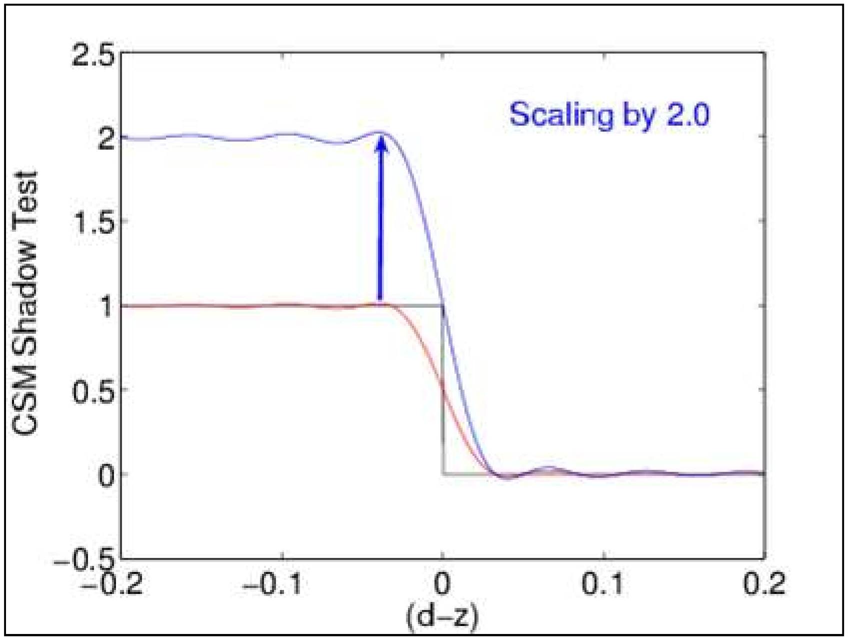
Issues with a Fourier series

- Ringing suppression
  - Reduce higher frequencies
- Steepness of "ramp"
  - Offset (transl. invariance!)
    - Shift shadow test
    - Increases lightness prob.



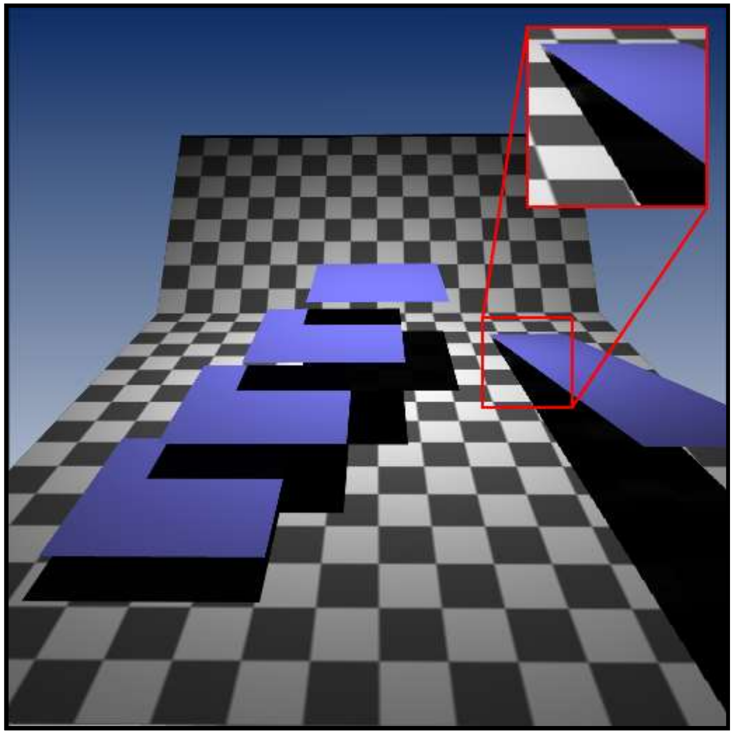
Issues with a Fourier series

- Ringing suppression
  - Reduce higher frequencies
- Steepness of "ramp"
  - Offset (transl. invariance!)
    - Shift shadow test
    - Increases lightness prob.
  - Scaling
    - Scale shadow test
    - Decreases filtering



Limitations and drawbacks

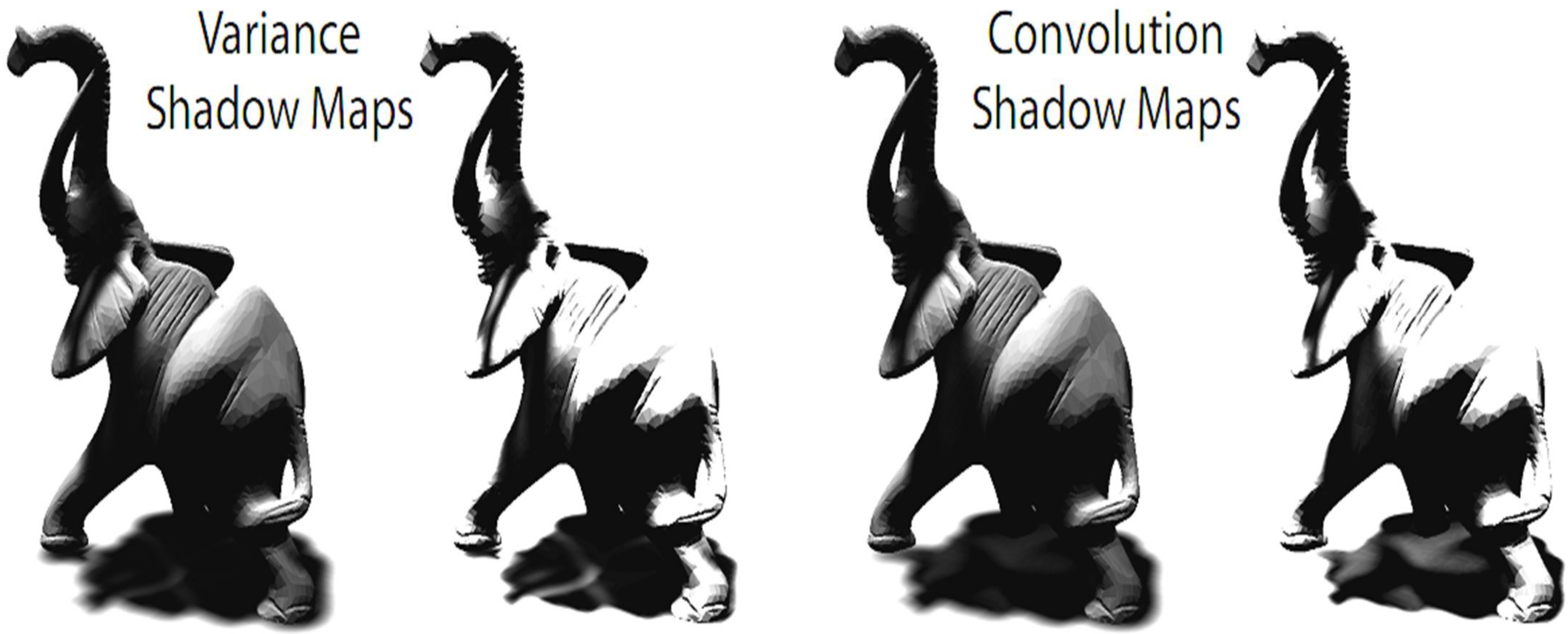
- Influence of reconstruction order M



$M = 16$

- Memory consumption increases as M grows
- Performance (filtering) decreases as M grows

VSM vs CSM



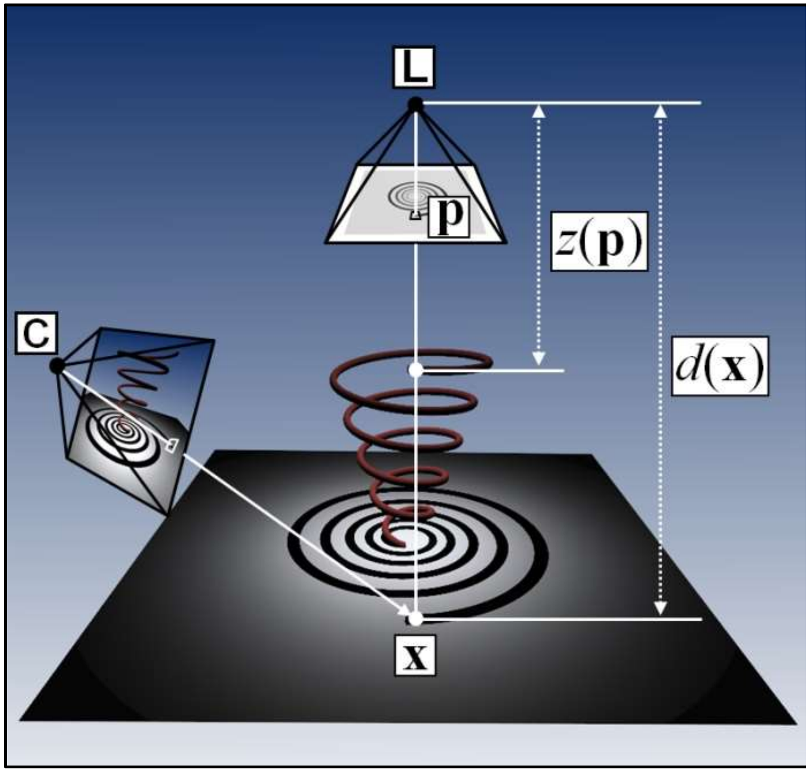
Exponential Shadow Maps

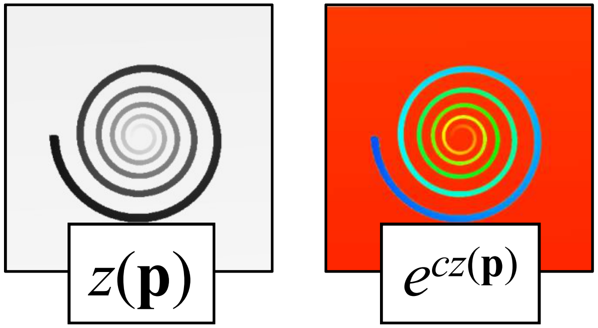
[Annen et al. 2008]

- Same general idea as CSM
  - "Linearize" shadow test
- Core idea:
  - Assume $(d(\mathbf{x}) - z(\mathbf{p})) >= 0$
  - → Assume shadow map represents visible front

→ Can use exponential

$$f(d(x), z(p)) \approx e^{-c(d(x) - z(p))}$$
$$= e^{-cd(x)} e^{cz(p)}$$

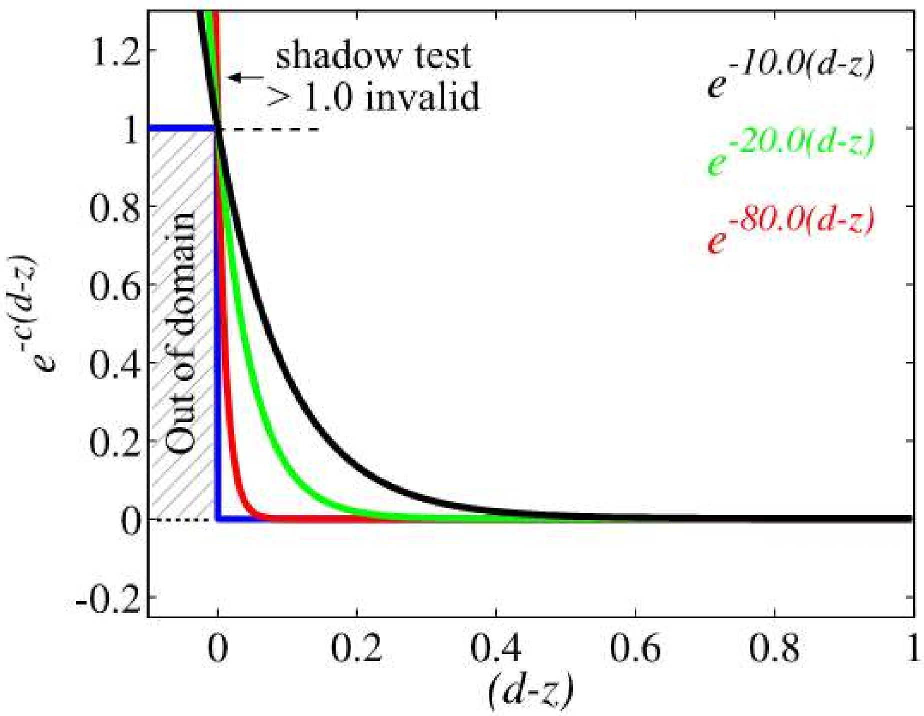


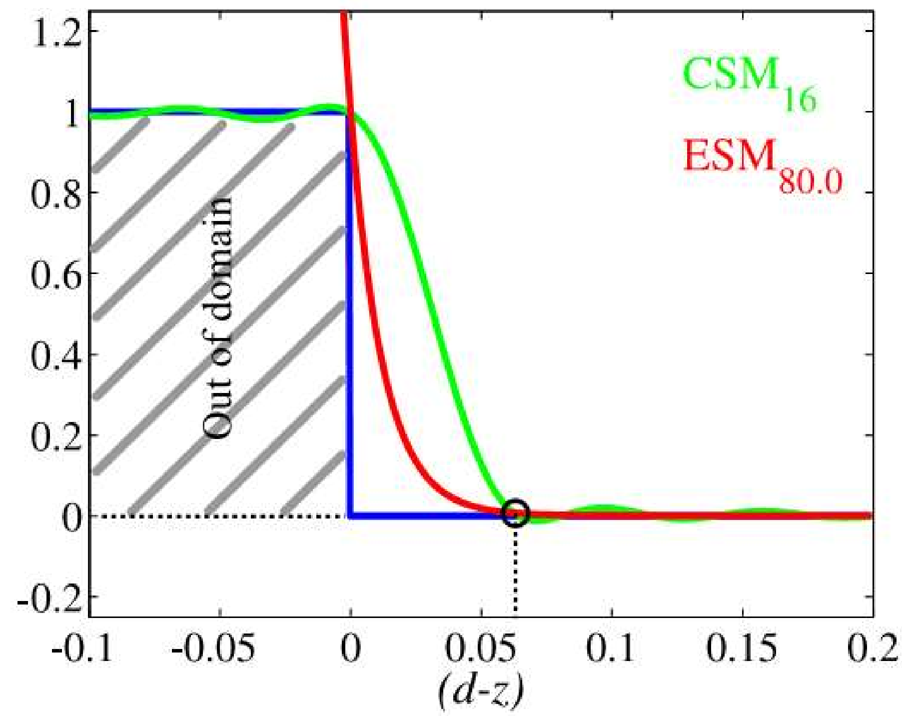


Exponential Shadow Maps

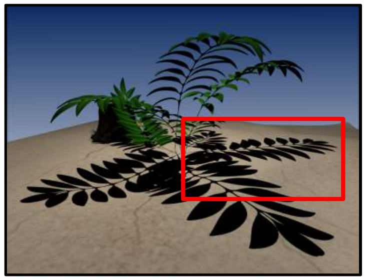
[Annen et al. 2008]

- Same approach as CSM, but uses exponential
- Exponential is separable
- Less memory and faster

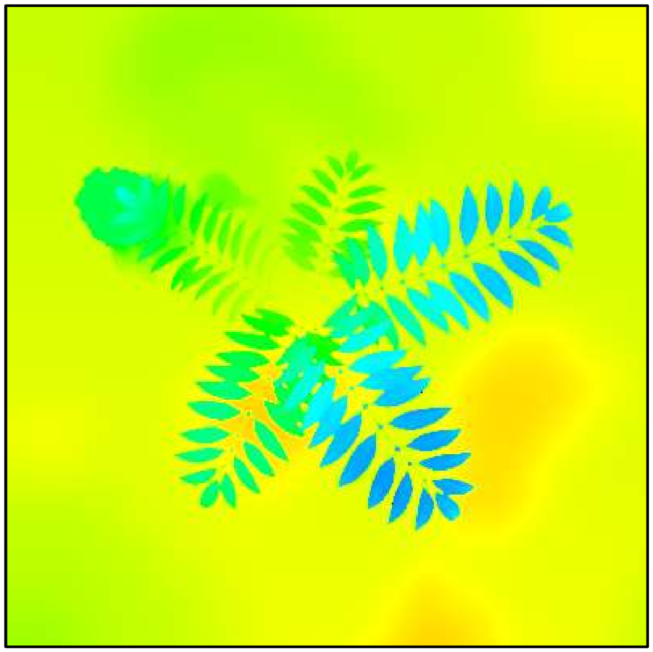




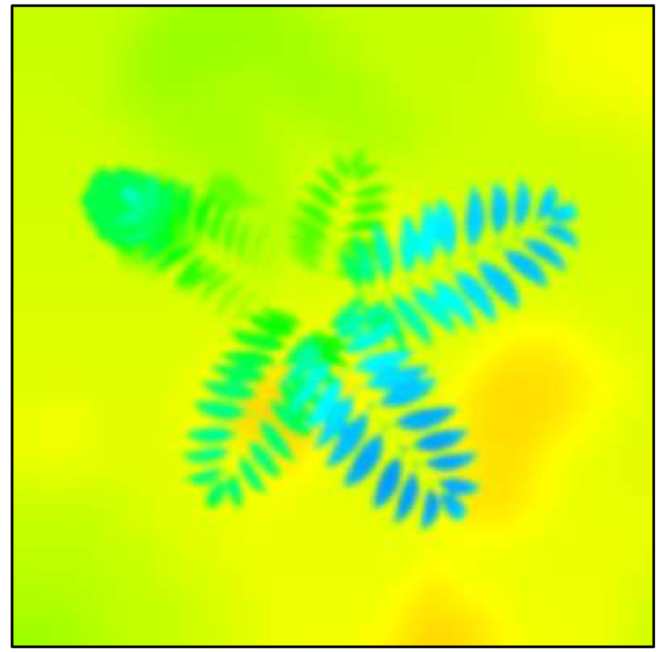
Filtering Example



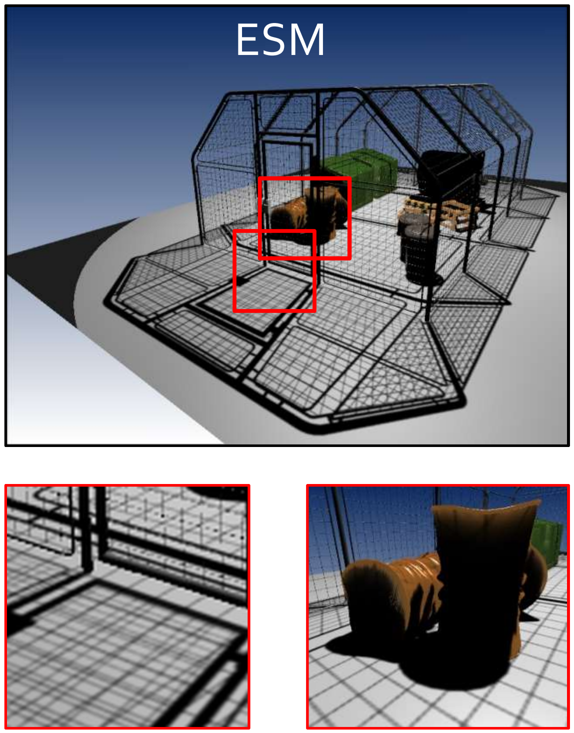
Original e^{cz}



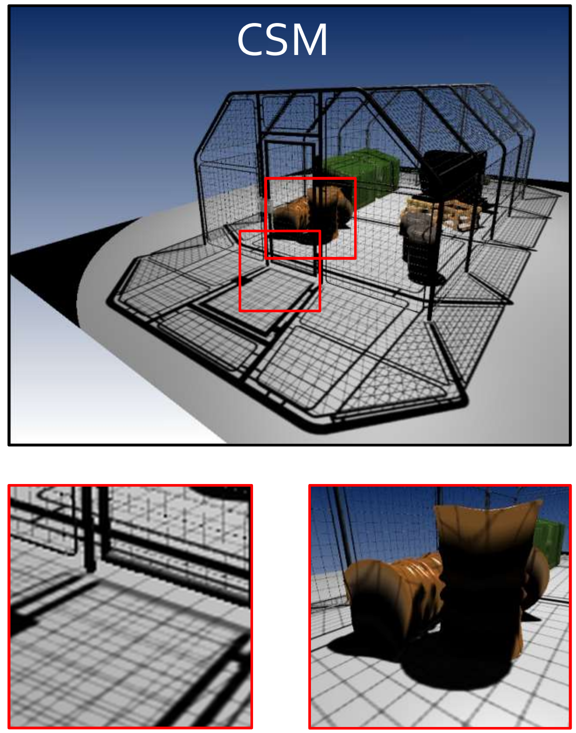
After filtering e^{cz}



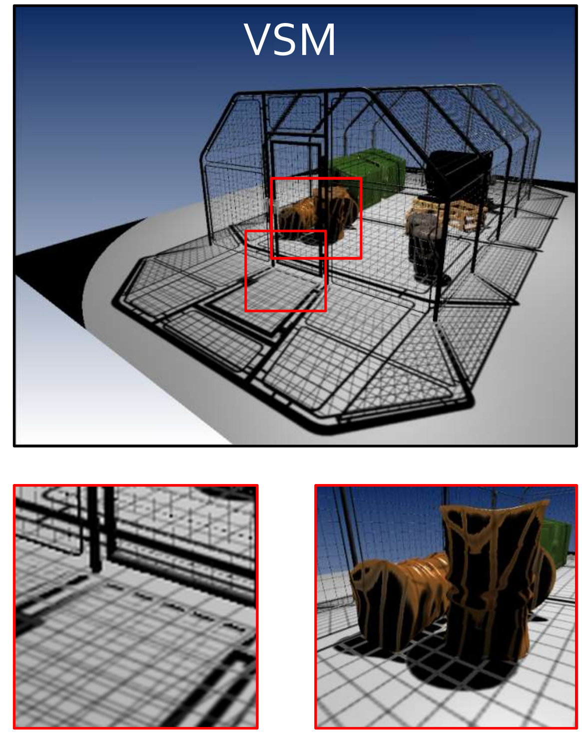
Results: Quality Comparison



66 FPS, 21 MB



21 FPS, 170 MB



60 FPS, 42 MB

Faces: 365K, SM resolution: 2Kx2K

Efficient Filtering with Summed Area Tables

- Build summed-area table from the shadow map
  - Can be done efficiently on the GPU
- Summing arbitrary rectangles is O(1)!

Data (t_{ij})

2	1	5
0	3	2
4	4	7

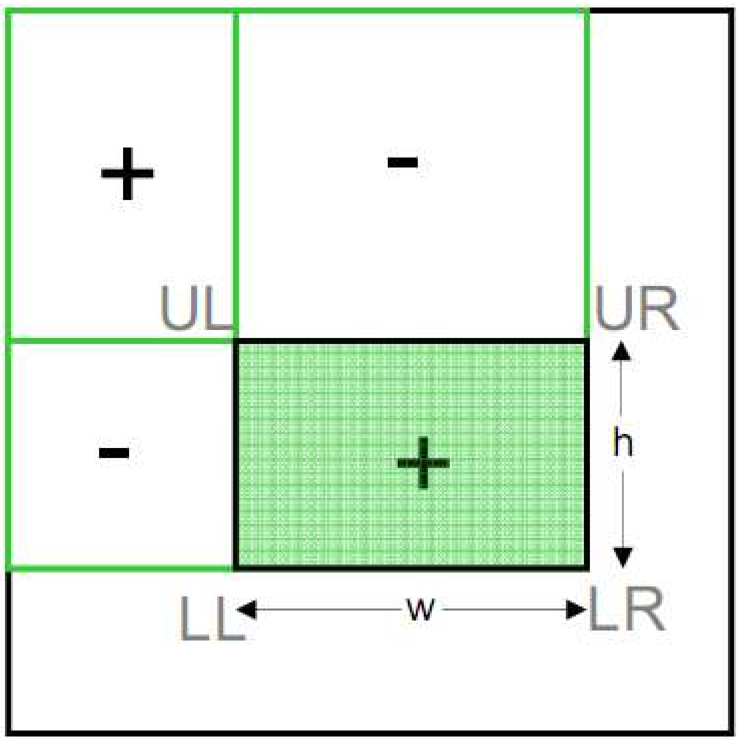
$$s_{mn} = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}$$

Summed-Area Table s_{mn}

2	3	8
2	6	13
6	14	28

Efficient Filtering with Summed Area Tables

- Summing arbitrary rectangles



$$average = \frac{LR - UR - LL + UL}{w * h}$$