3D Computer Graphics and Animation

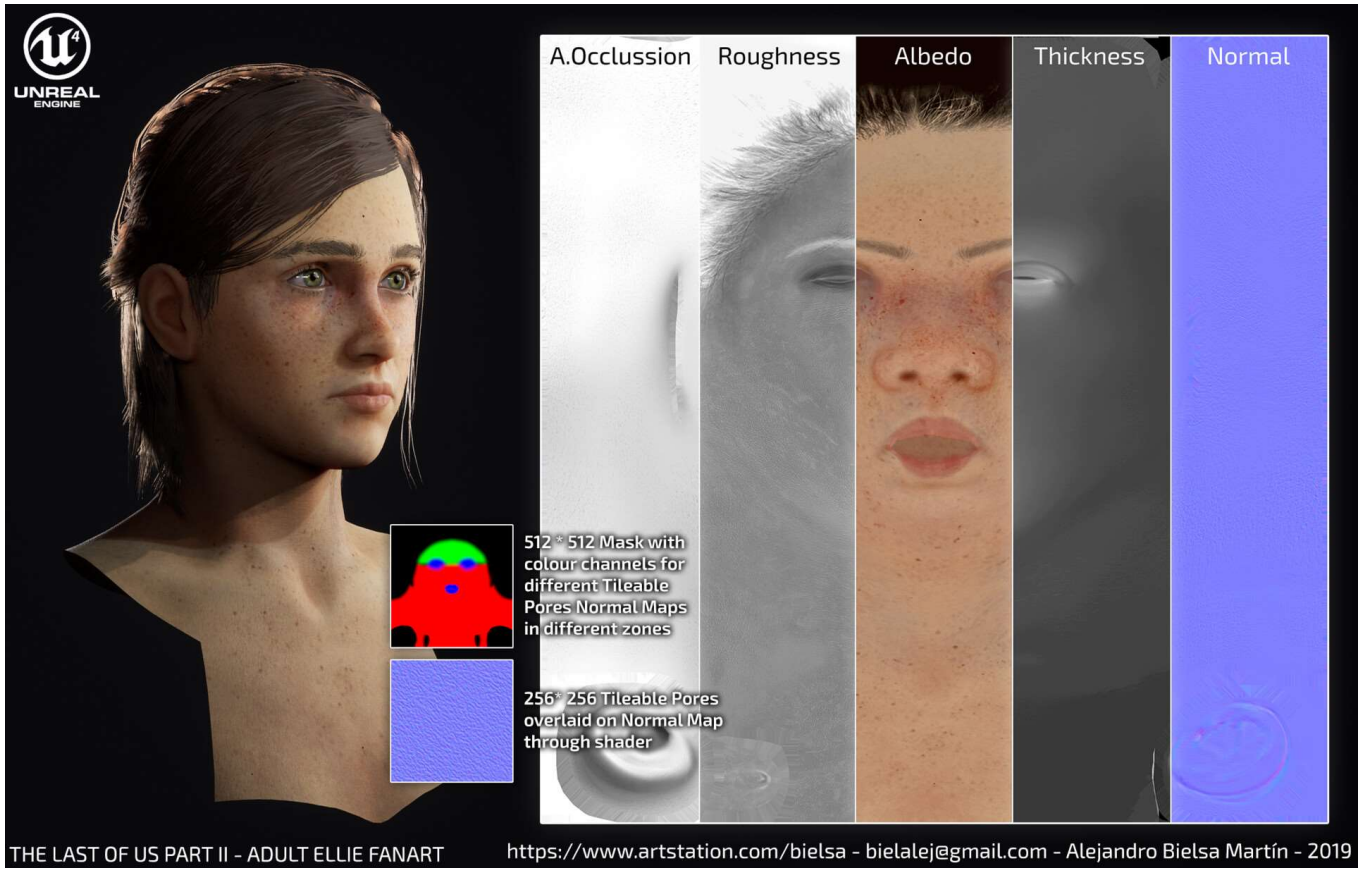
Filtered Shadows

The line begins to blur...

Elmar Eisemann

Delft University of Technology

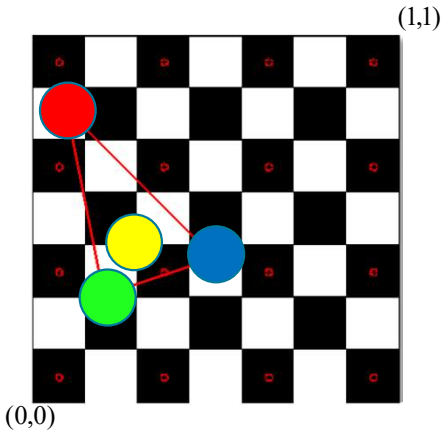
Professional Example



Texture Mapping: Special Software

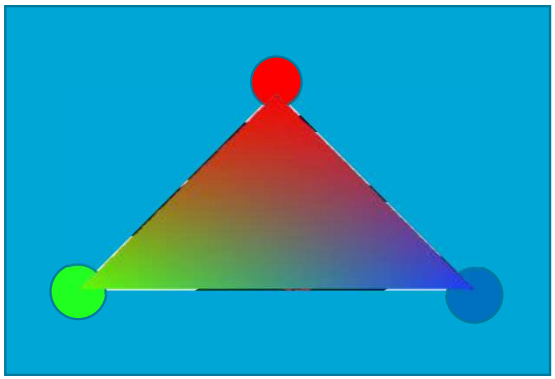
Textures

- Image mapped on the surface via texture coordinates
  - Specified at each vertex `glTexCoord{123}{fi}`
  - Interpolated over triangles



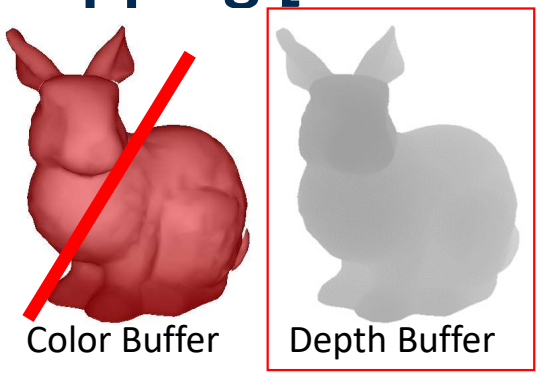
domain

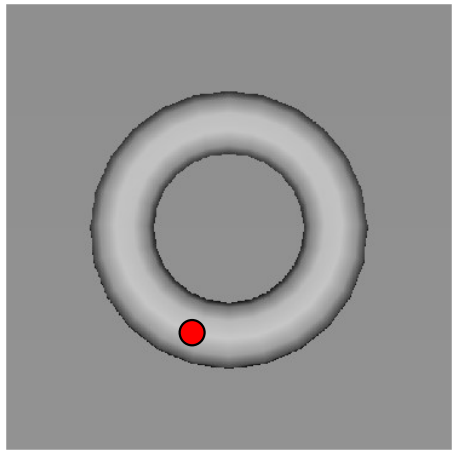
```
TextureCoords(0.2f,0.3f);
Vertex(-1.0f,0.0f);

TextureCoords(0.5f,0.4f);
Vertex(+1.0f,0.0f);

TextureCoords(0.1f,0.8f);
Vertex( 0.0f,1.0f);
```



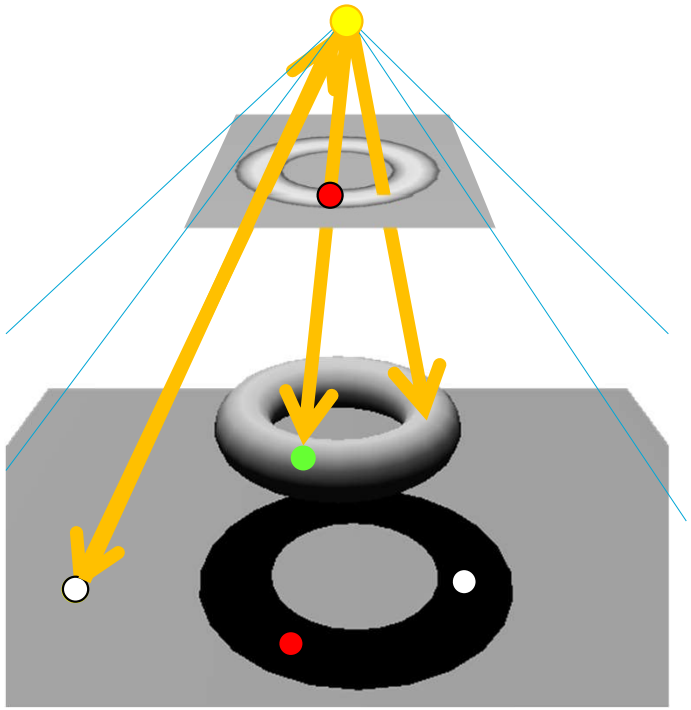
3D View

Shadow Mapping [Williams78]



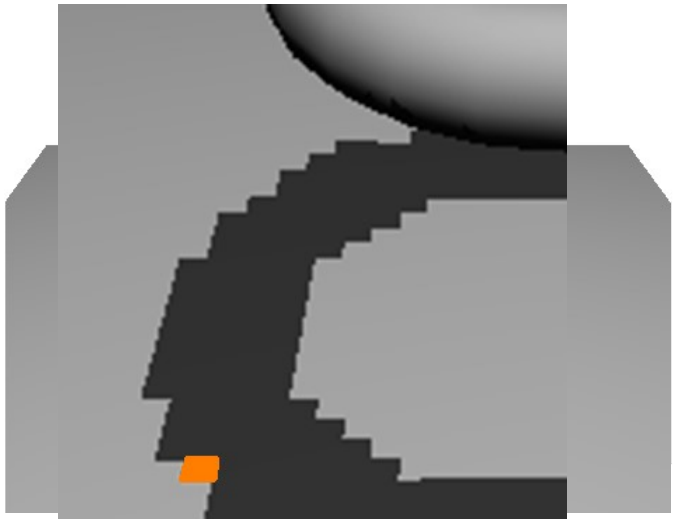


1. render view from light
use depth buffer

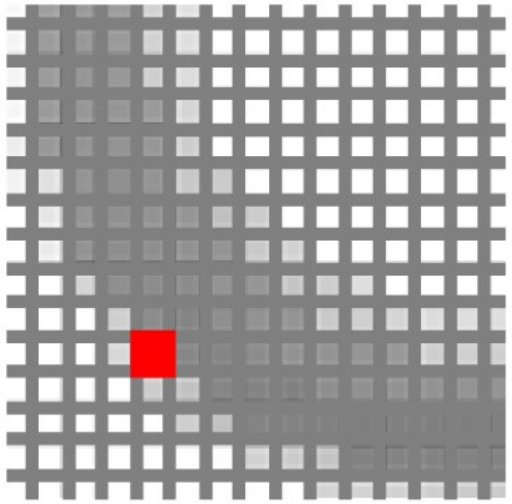


2. render from viewpoint

Shadows and Rasterization



from viewpoint

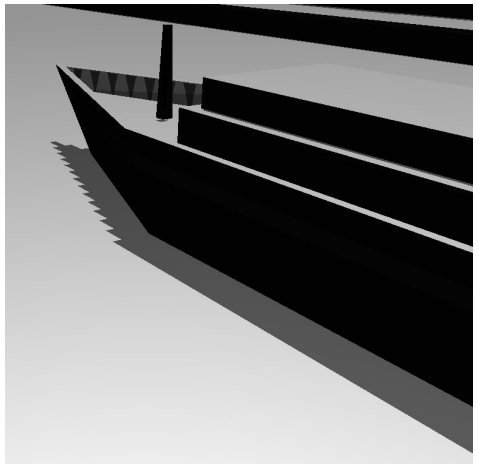


from light

Texture Problems for Shadow Mapping

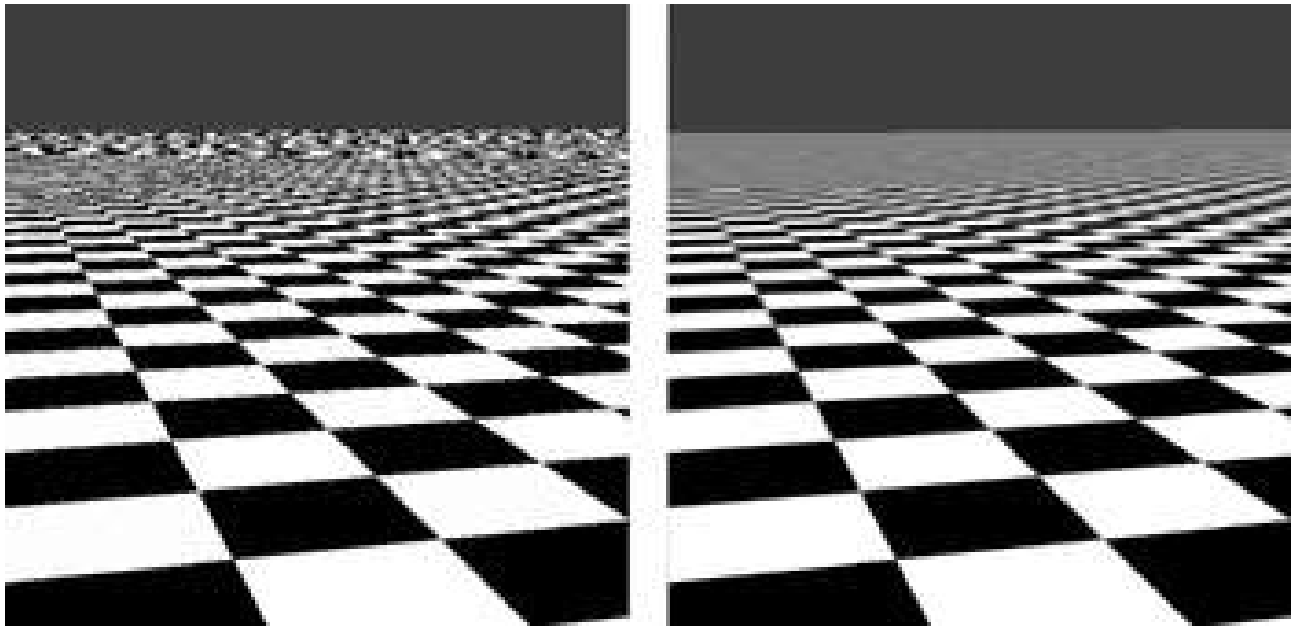
Reconstruction

- Staircase artifacts



Texture Oversampling

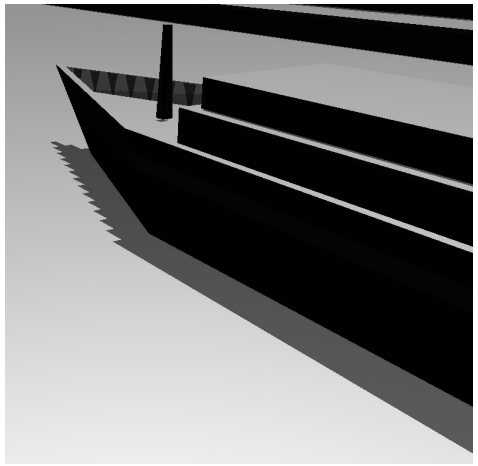
Texture Undersampling



Texture Problems for Shadow Mapping

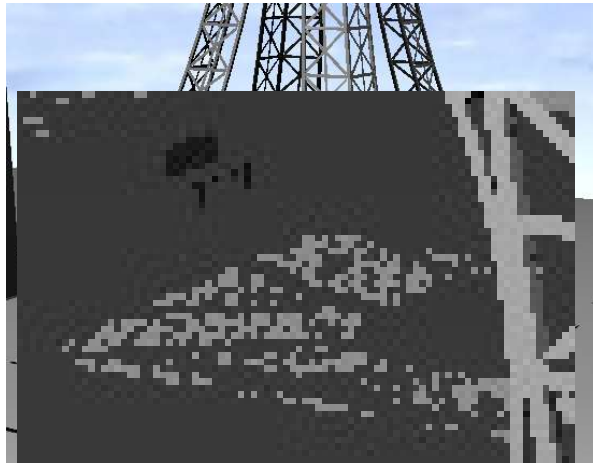
Reconstruction

- Staircase artifacts

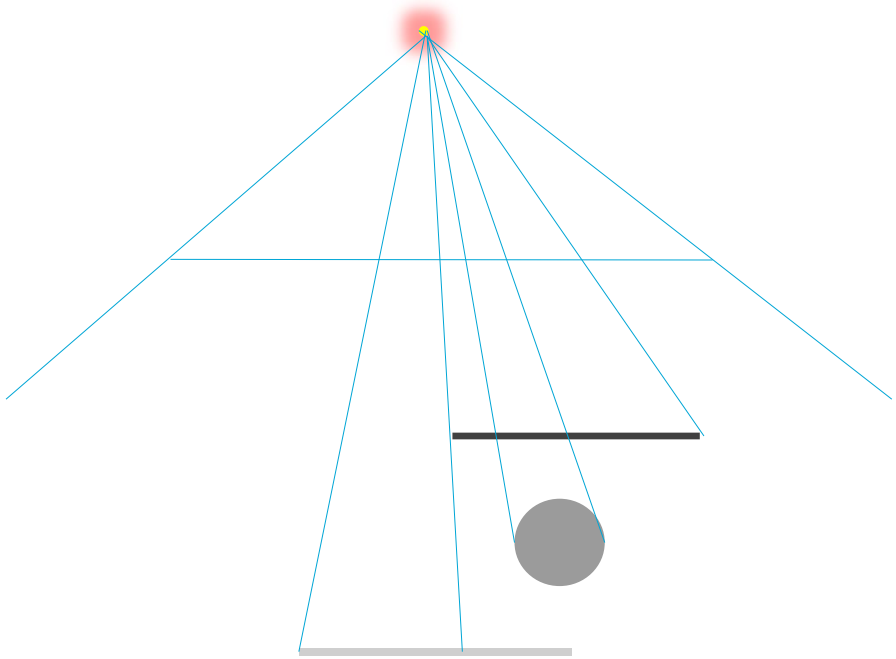


Undersampling

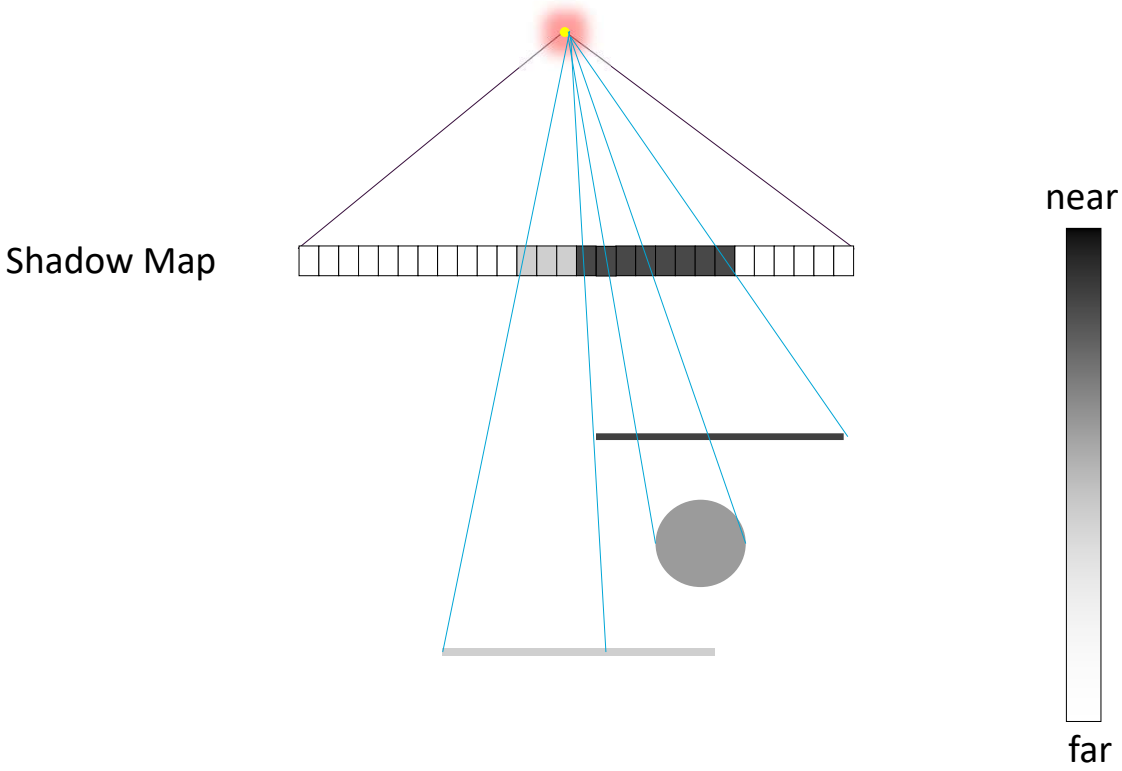
- Noise



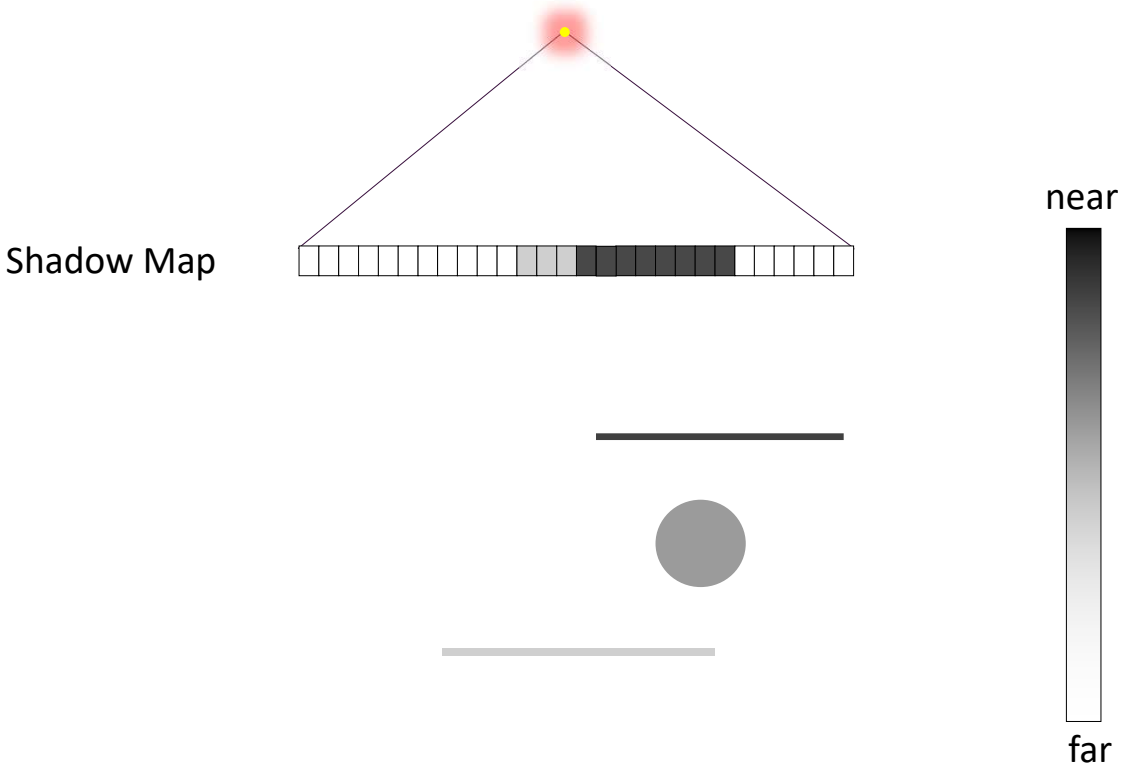
Shadow Mapping



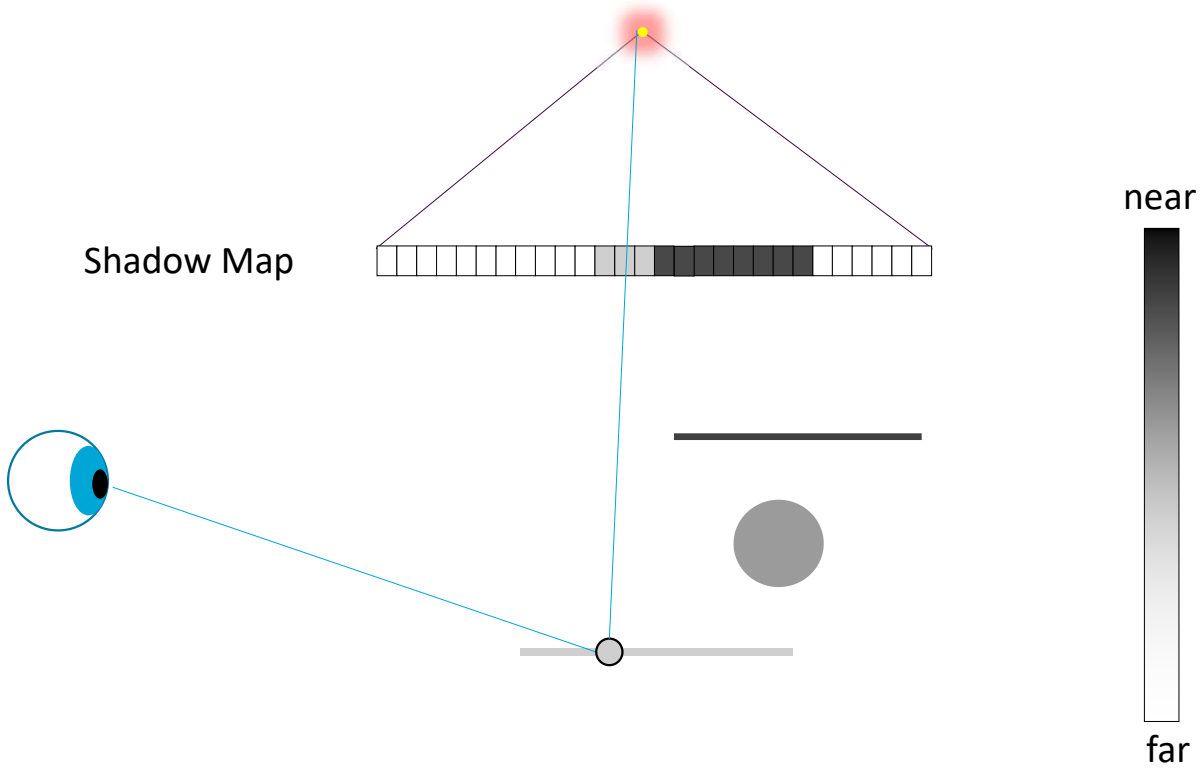
Shadow Mapping



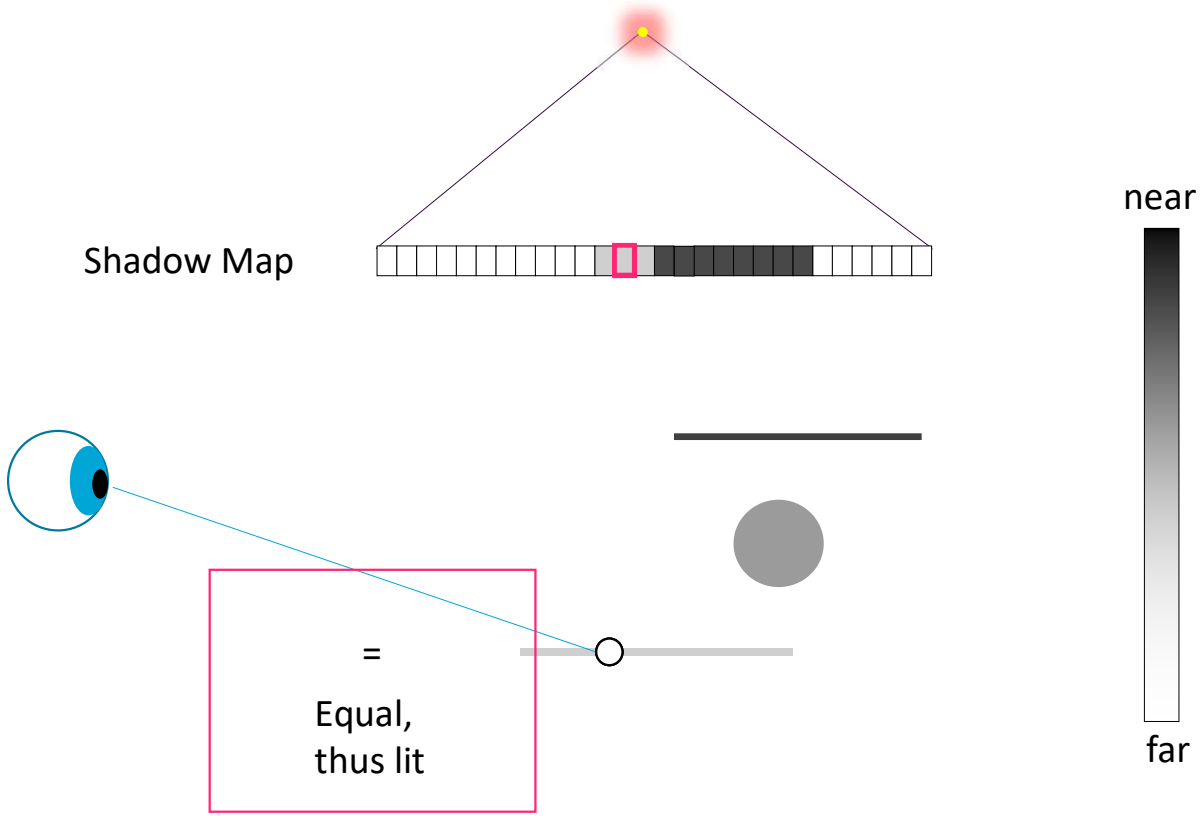
Shadow Mapping



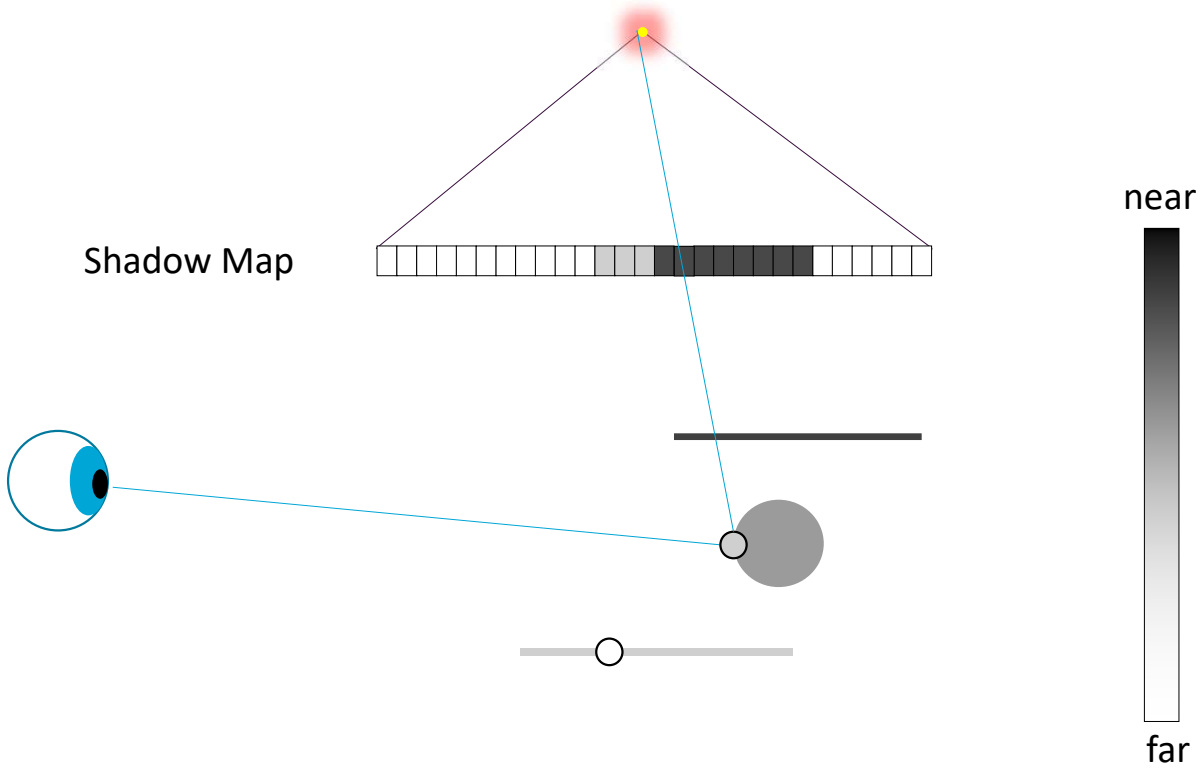
Shadow Mapping



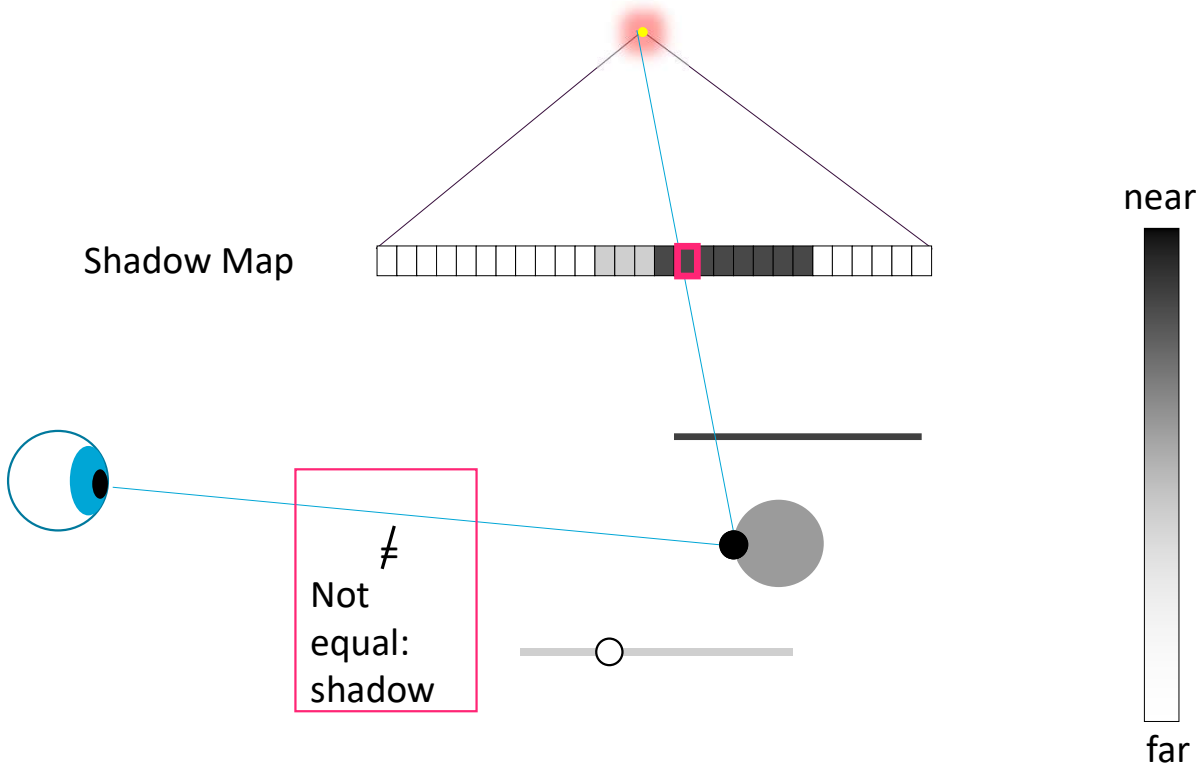
Shadow Mapping



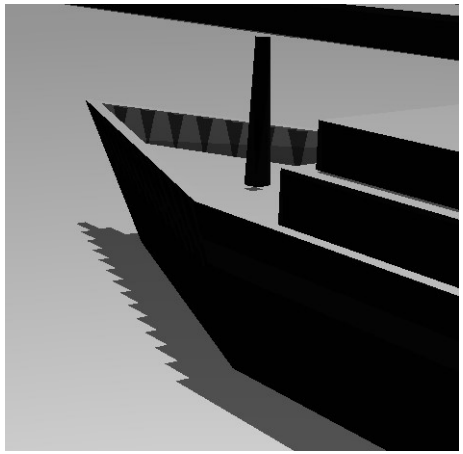
Shadow Mapping

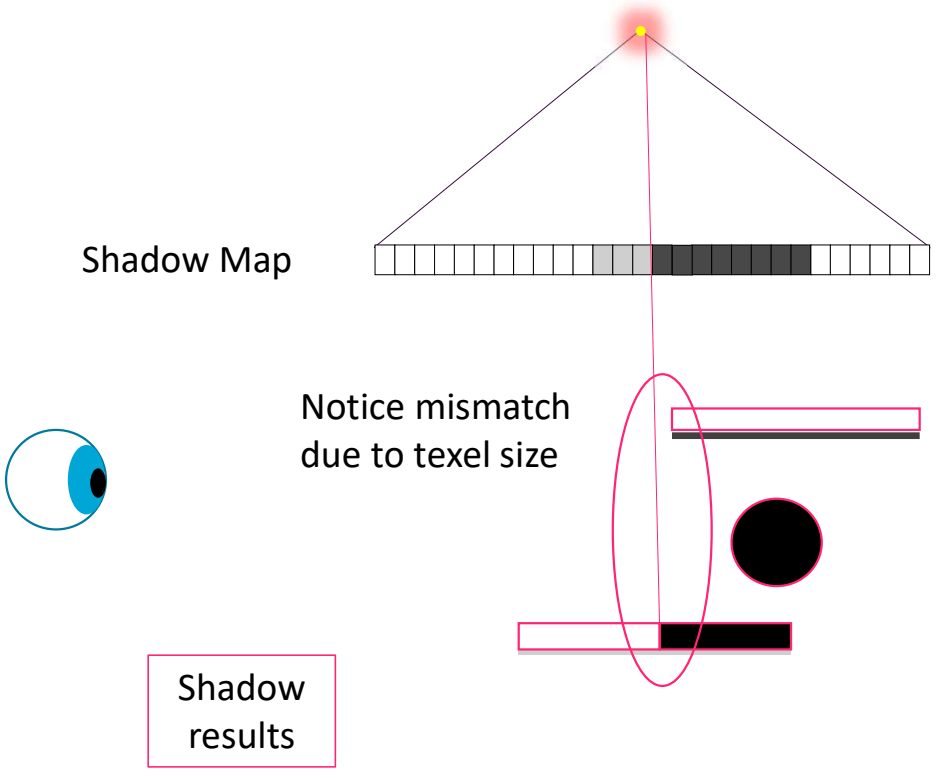


Shadow Mapping

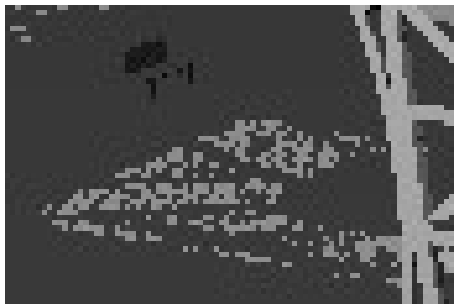


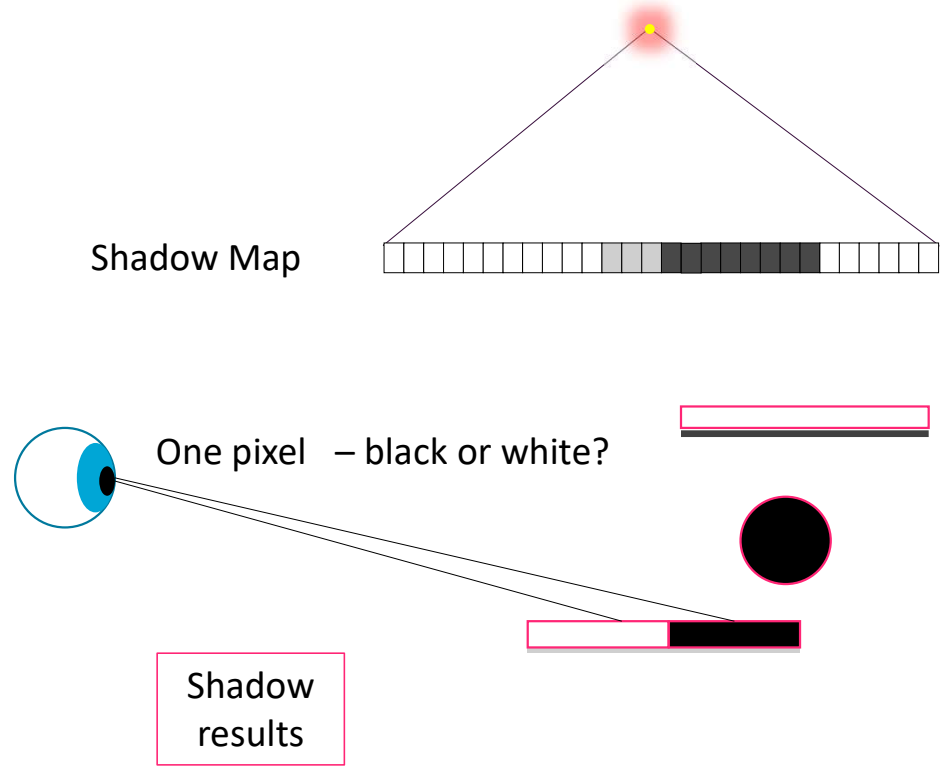
Shadow Mapping

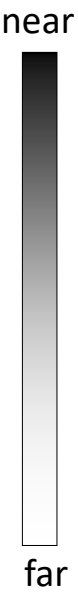




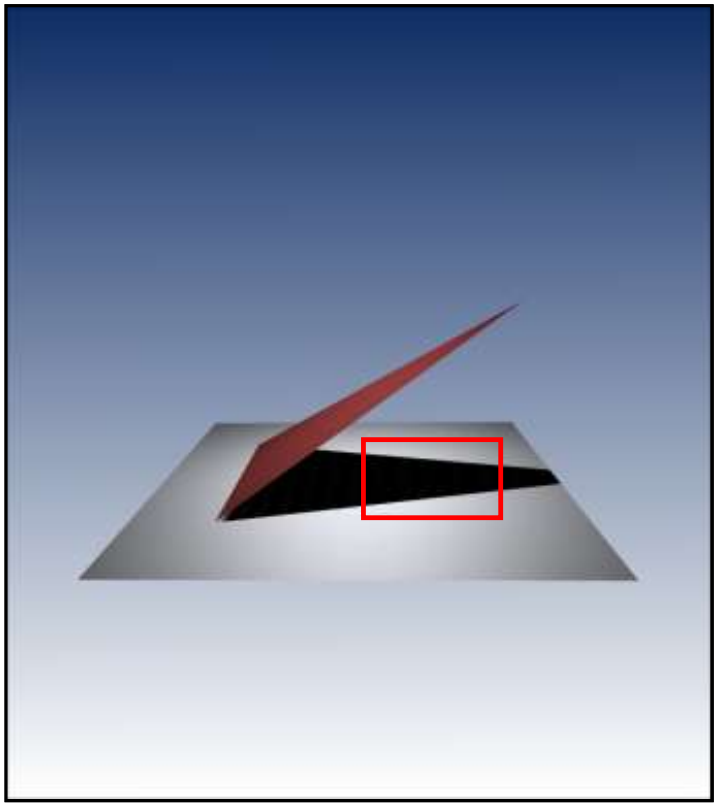
Shadow Mapping





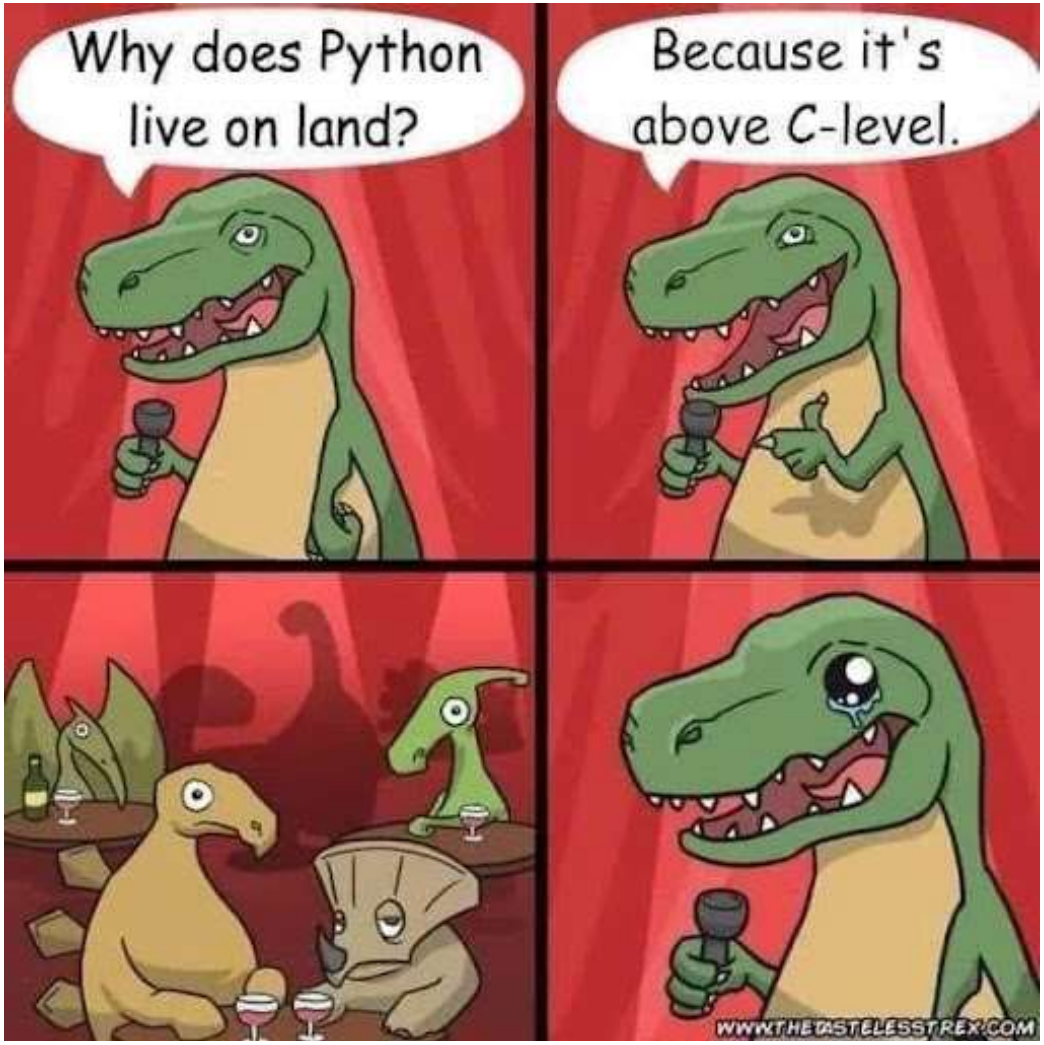


Filtered Hard Shadows

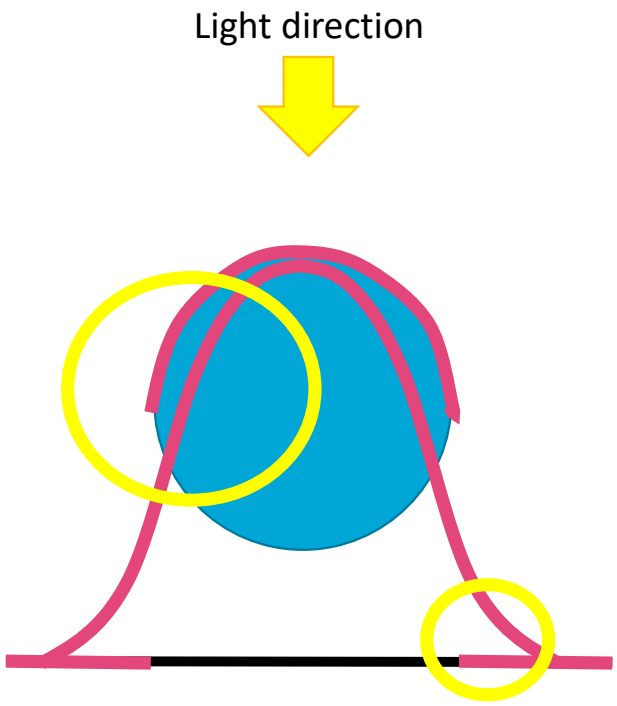


Filtered Shadow Mapping

Questions?

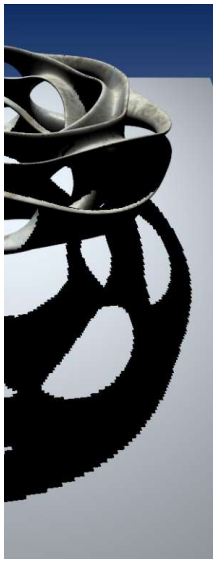

Why does Python live on land?
Because it's above C-level.
WWW.THEBASTELESSTREX.COM

Can we just filter the Depth Map?

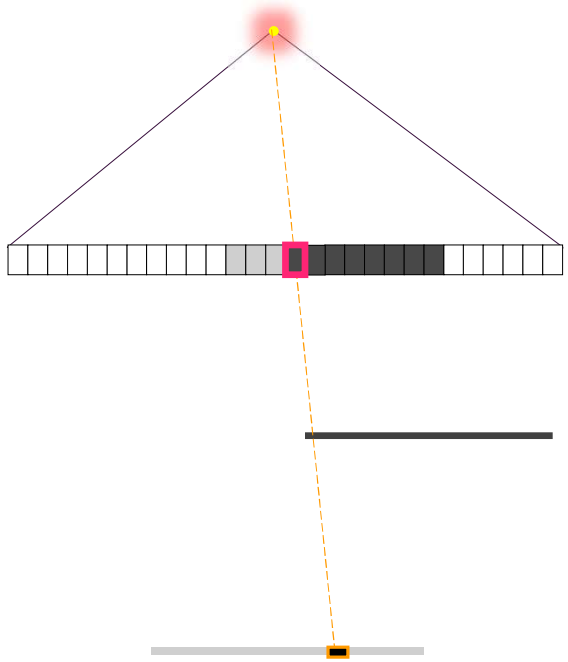


Filtered Depth Map

Percentage-Closer Filtering

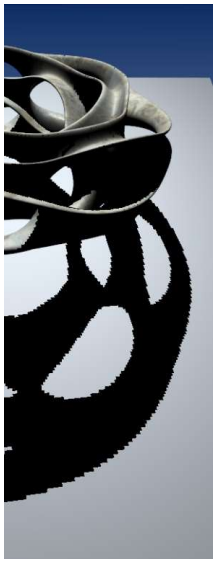


unfiltered

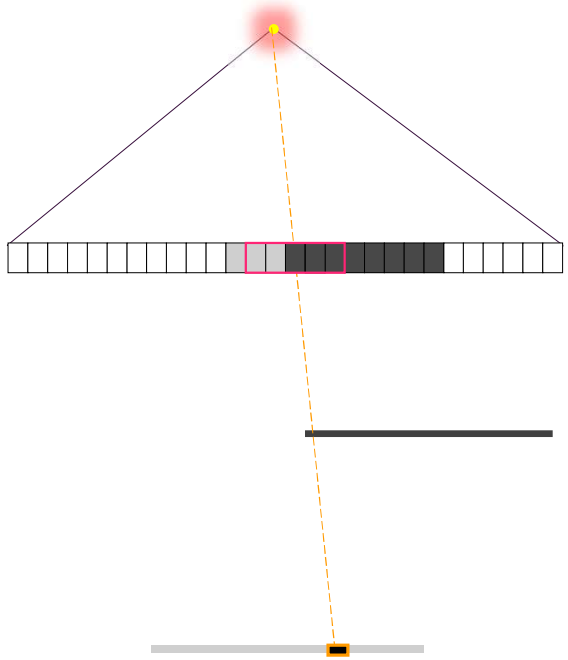


[Reeves et al. ‘87]

Percentage-Closer Filtering

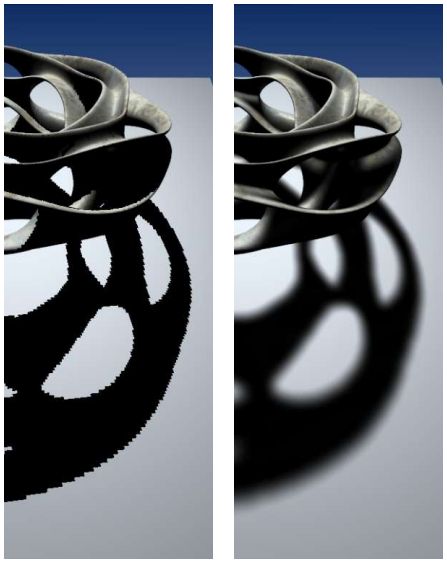


unfiltered

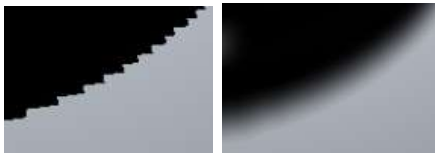


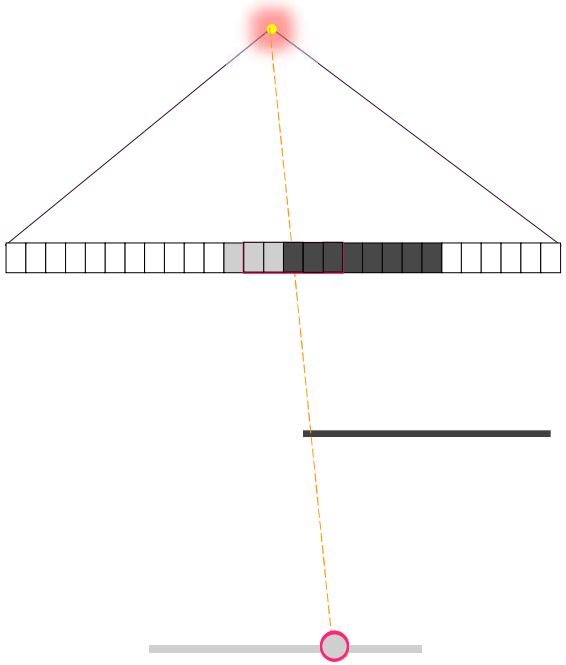
[Reeves et al. ‘87]

Percentage-Closer Filtering



unfiltered filtered





[Reeves et al. ‘87]

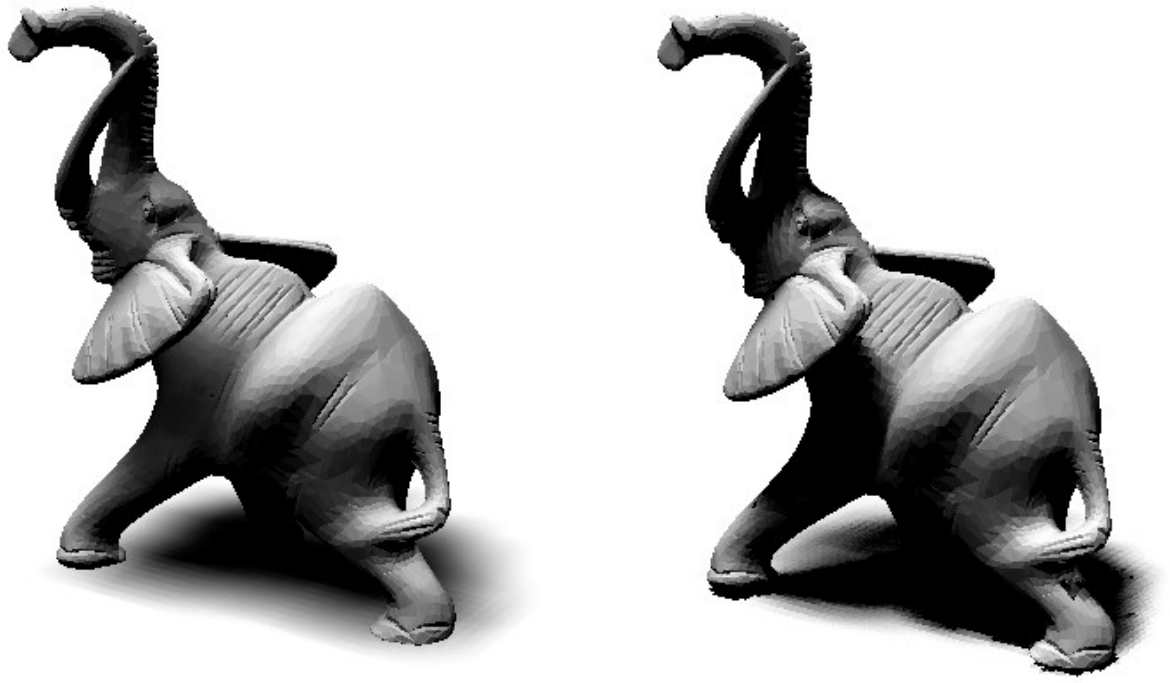
40 % lit, 60% shadow

≠	shadow
≠	shadow
≠	shadow
=	lit
=	lit

compare first, then average!

Attention

- Percentage-Closer Filtering is NOT computing soft shadows

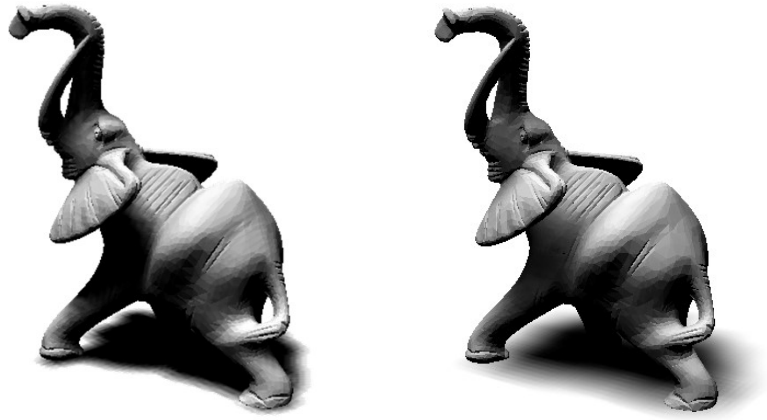


Pixar

Shadows computed with Percentage-Closer Filtering (PCF)

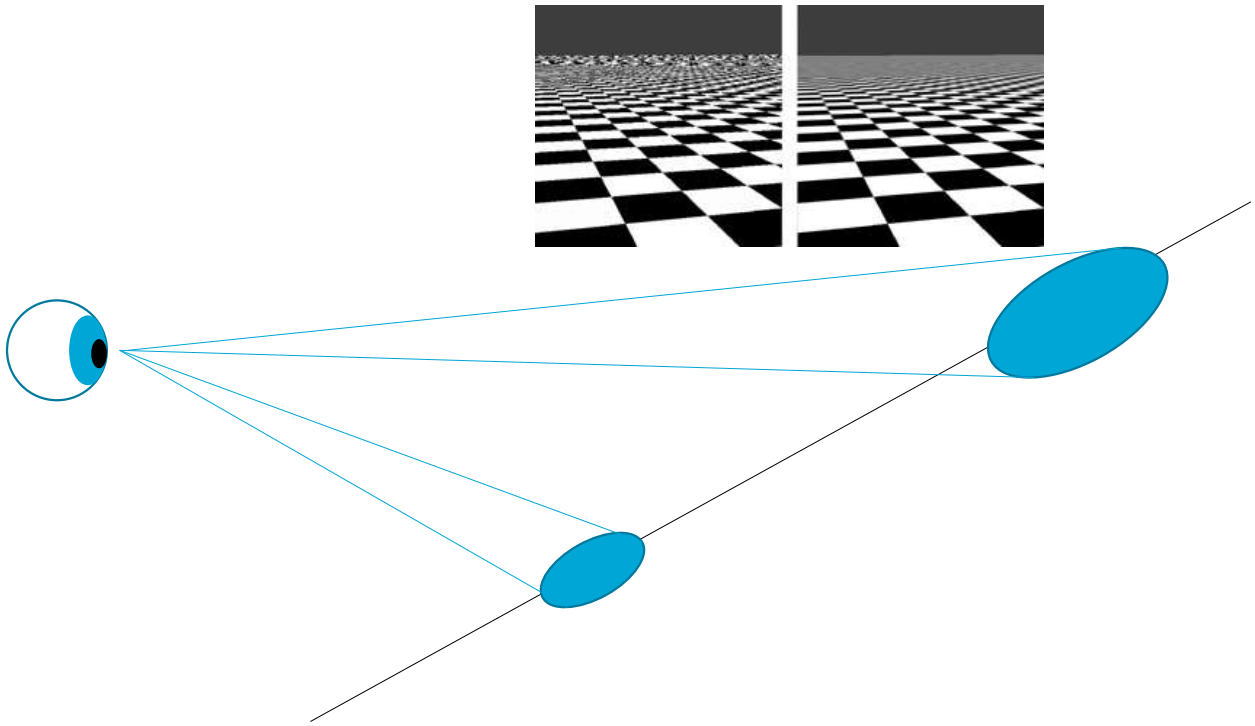
Percentage-Closer-like Filtering

+ Standard (simple to implement)
+ Hides discretization



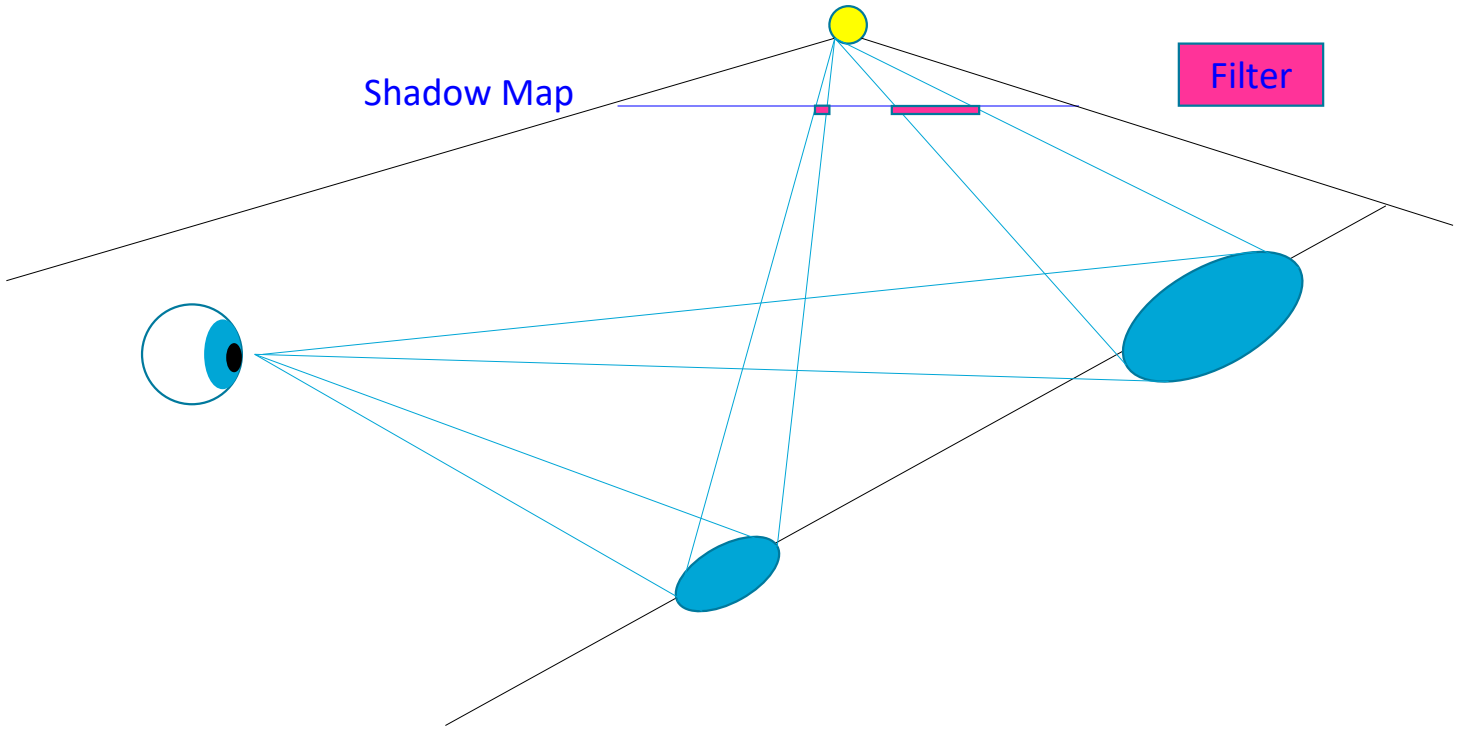
How to choose the PCF Filter size?

- Choose filter according to viewing distance



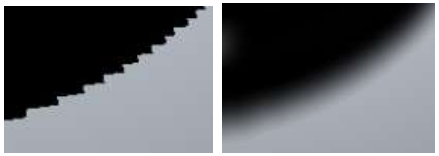
How to choose the PCF Filter size?

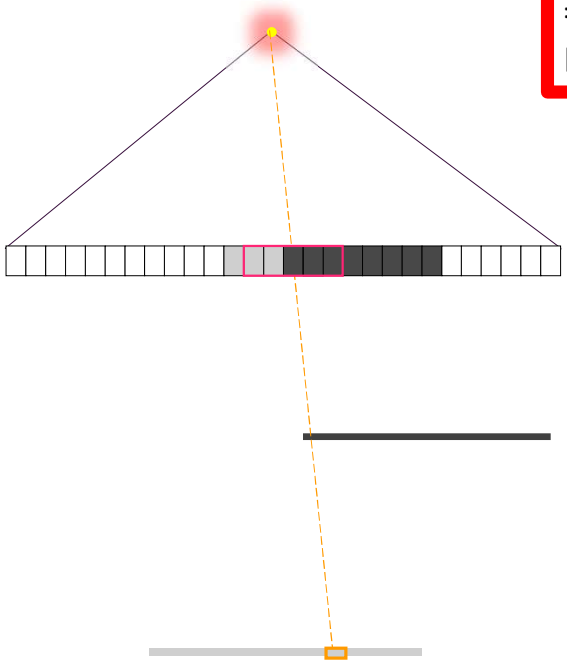
- Choose filter according to viewing distance
- PCF Filter can become very large



Percentage-Closer Filtering

unfiltered filtered

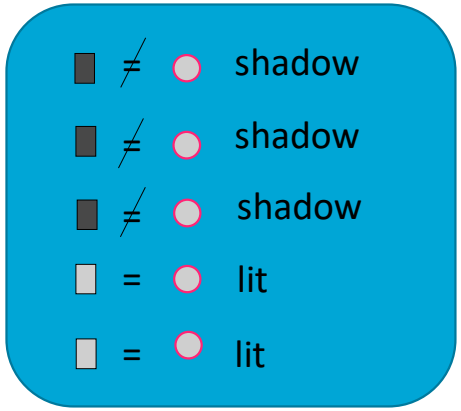




[Reeves et al. ‘87]

Attention:
Large neighborhood
= costly evaluation!
E.g., 5x5 implies 25 lookups

40 % lit, 60% shadow



compare first, then average!

Adapted PCF Filter Kernel

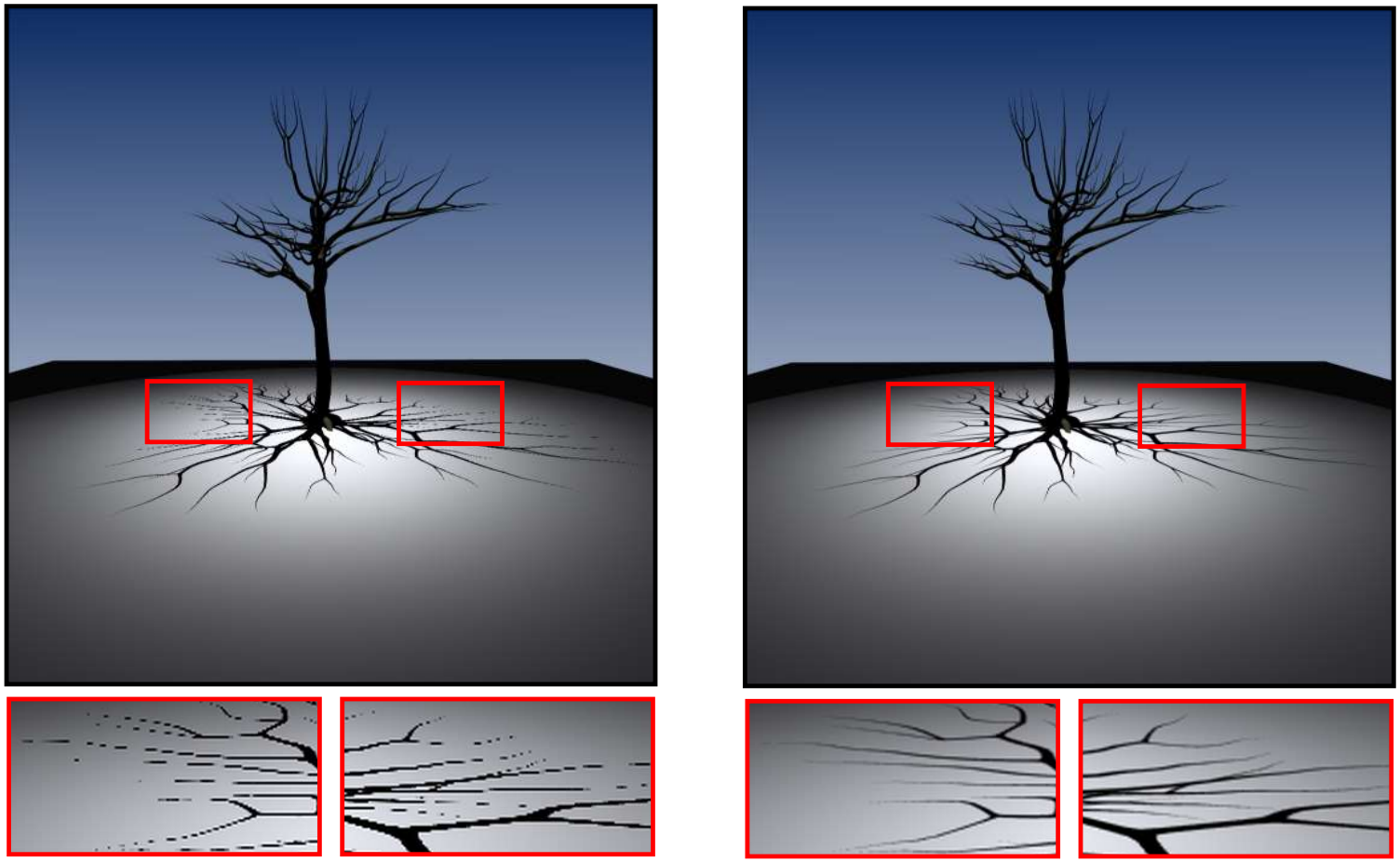
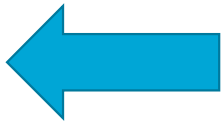


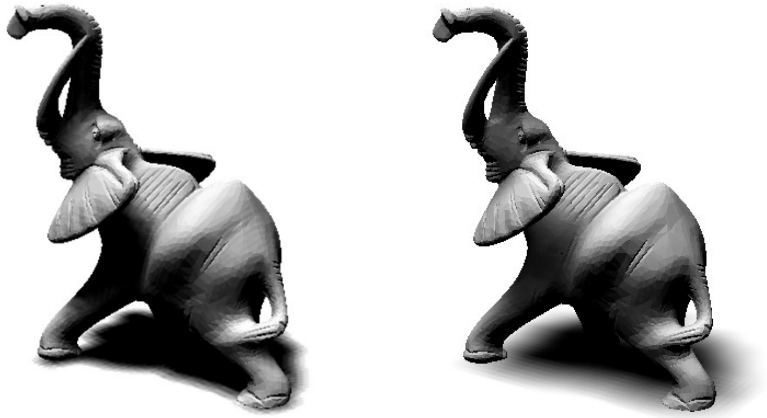
Illustration courtesy of Annen et al.

Percentage-Closer-like Filtering

+ Standard (simple to implement)

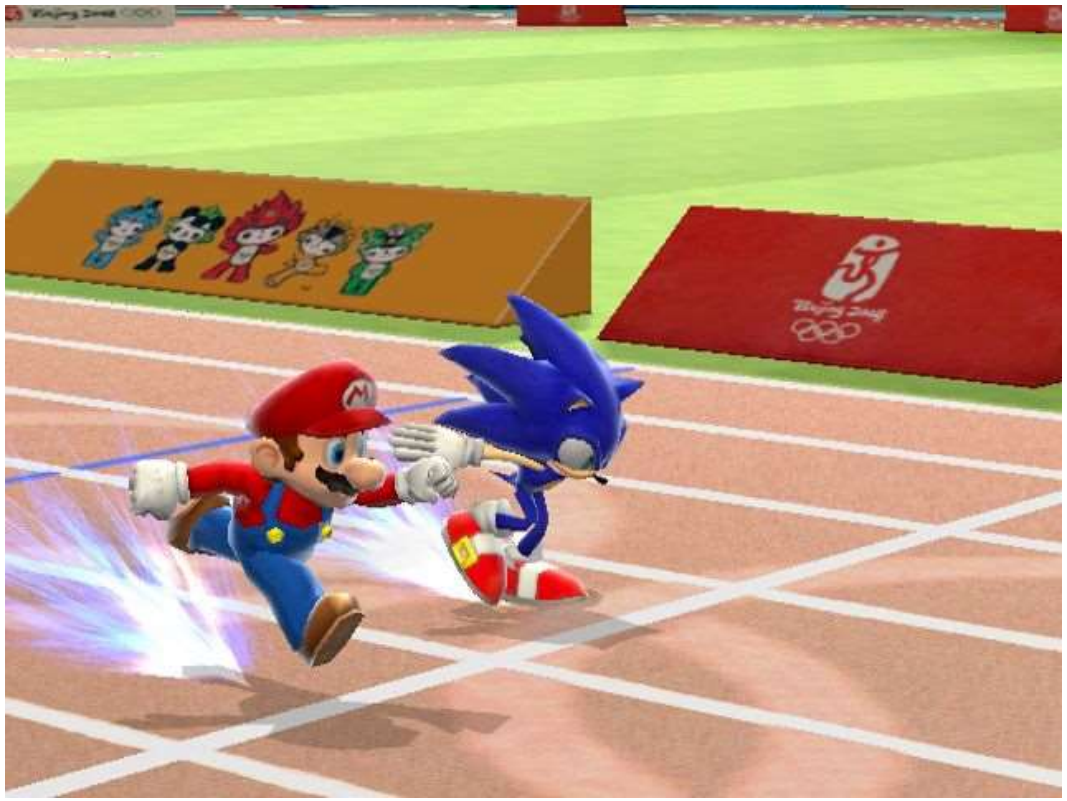
+ Hides discretization

- Costly for large kernels  Topic of today!



Any Questions...

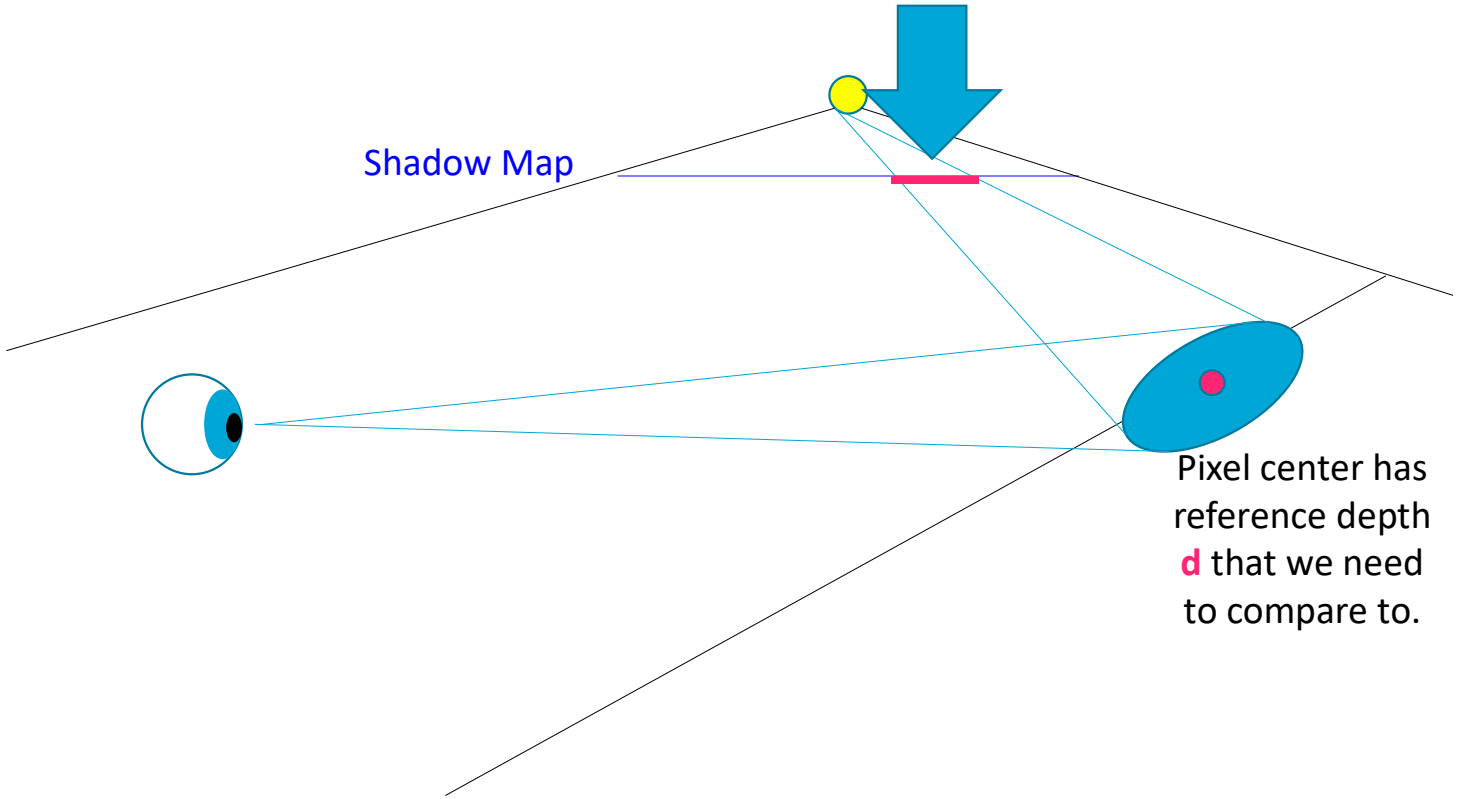
besides: How can we accelerate PCF?



Improvements

- Variance Shadow maps [Lauritzen&Donnelly06]
- Convolution Shadow Maps [Annen et al. 07]
- Exponential Shadow Maps [Annen et al. 08, Salvi08]

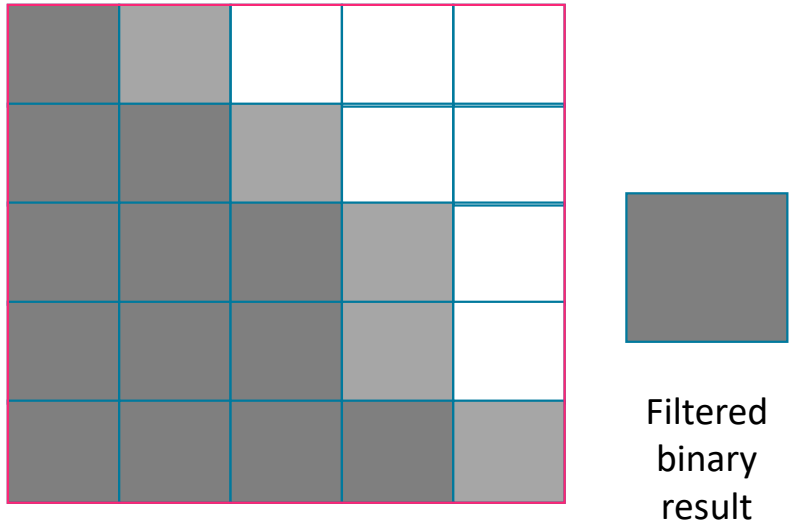
How evaluate PCF in this region quickly?



Variance SMs [Lauritzen & Donnelly 2006]

- Avoid many lookups and comparisons

- Solution:
  - FIRST filter
  - SECOND one lookup



Filtered binary result

Reference d

Wait a second… didn't we just say:
We cannot first filter then compare???

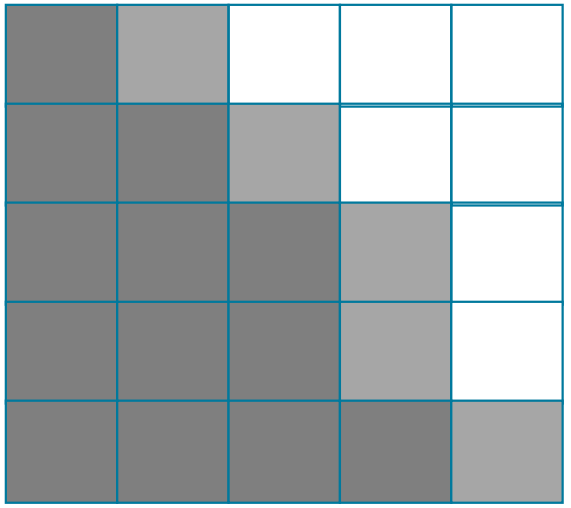
Variance SMs [Lauritzen & Donnelly 2006]

- Reformulation:

- Given depth d
- What is probability

  that d < x

  for all x in window?



Probabililty

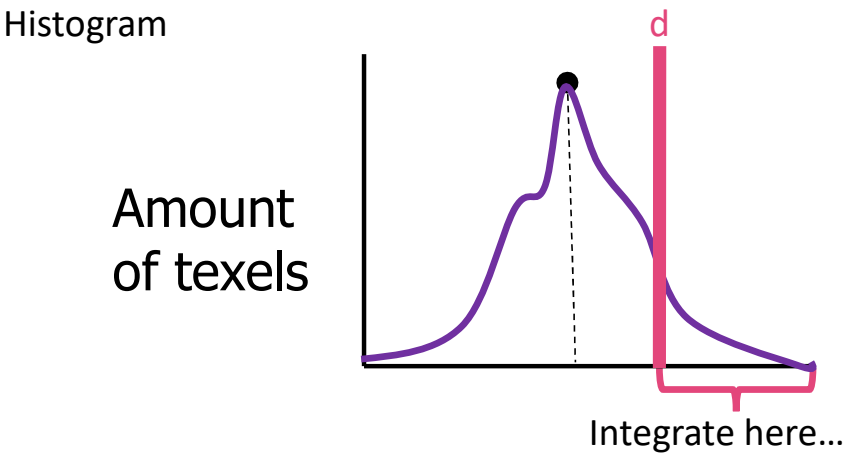
reference

It is the same computation!

➡ Idea: Use statistics to “guess” the outcome

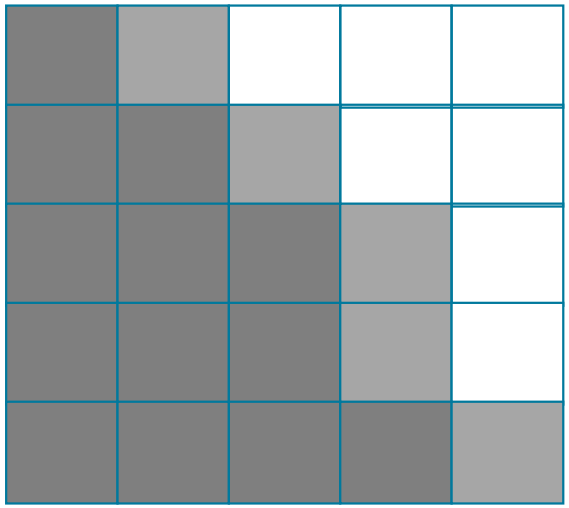
Theoretical Approach

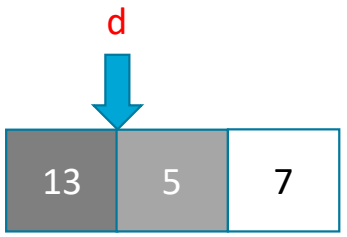
- Take the depth distribution (like a histogram) and compare to d



The integrated part tells you how many depth values are larger than d and would therefore not occlude it.

But building up the actual distribution requires us to do again lots of lookups…





13 shadow
12 lit thus
12/25

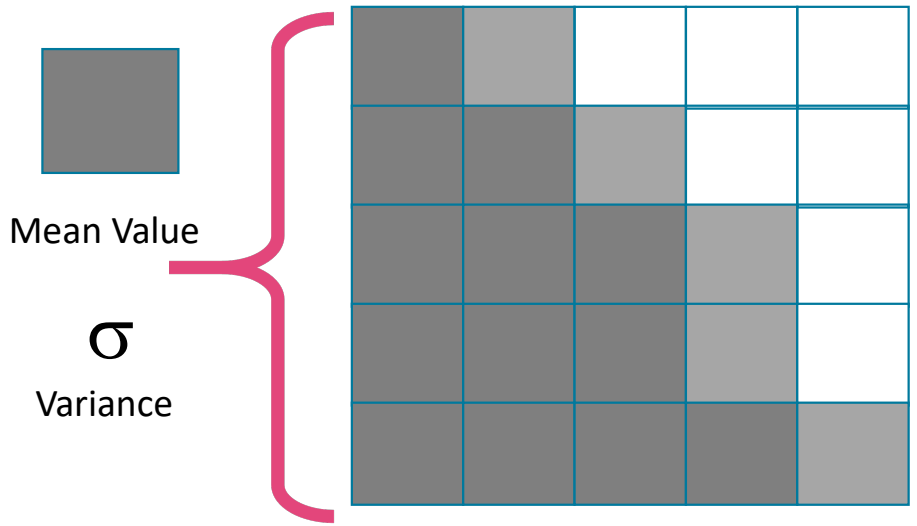
Approximate Distribution

- Instead of looking at individual depth values,

  consider an approximate distribution :

Use mean and variance

Approximate Distribution

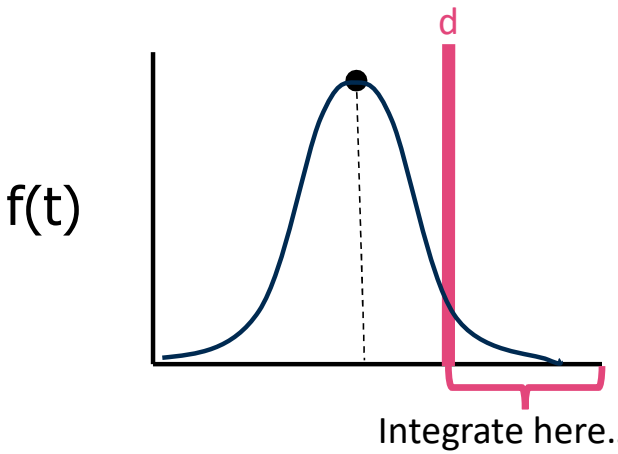


Suppose a “Magic Transformation” for the moment

What can we do with Distribution?

For a depth-value distribution

the test result can be bounded



bounded by

$$\frac{\sigma^2}{\sigma^2 + (d-\mu)^2}$$

In other words:

Bound on the shadow result

Lauritzen and Donnelly'06:

"Just assume equality and you have a fast and good-looking method!"

Where does this bound come from?

- Chebyshev Inequality:

  Probality(d < D) <= $\frac{\sigma^2}{\sigma^2 + (d-\mu)^2}$

  (for d<=μ, for d>=μ assume 1)

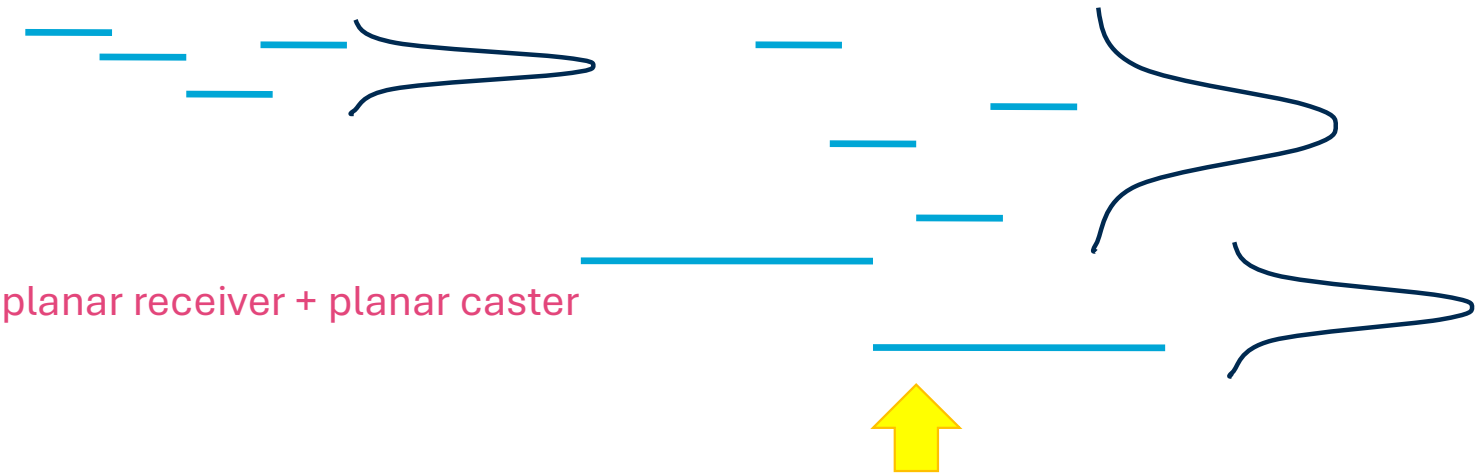
where D is random Variable from an arbitrary depth distribution (mean μ, variance σ)

Estimating shadows

- Using:

$$\frac{\sigma^2}{\sigma^2 + (d - \mu)^2}$$

- Small sigma leads to quick fall-off
- Large sigma leads to slow fall-off



- Accurate for planar receiver + planar caster

Estimating shadows

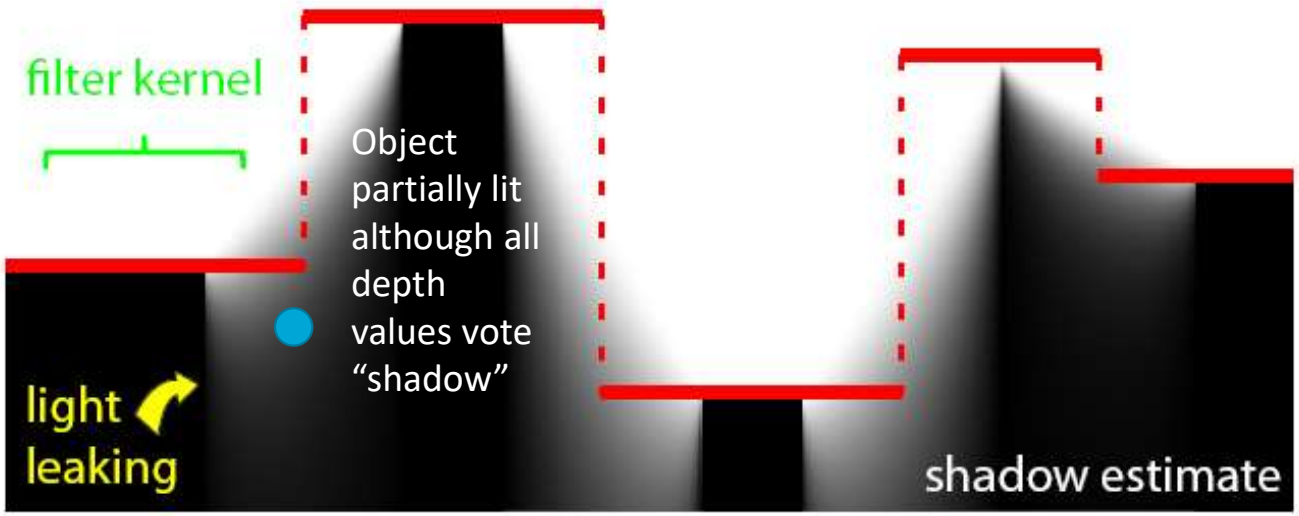
Light direction

- What is the problem?

Estimating shadows

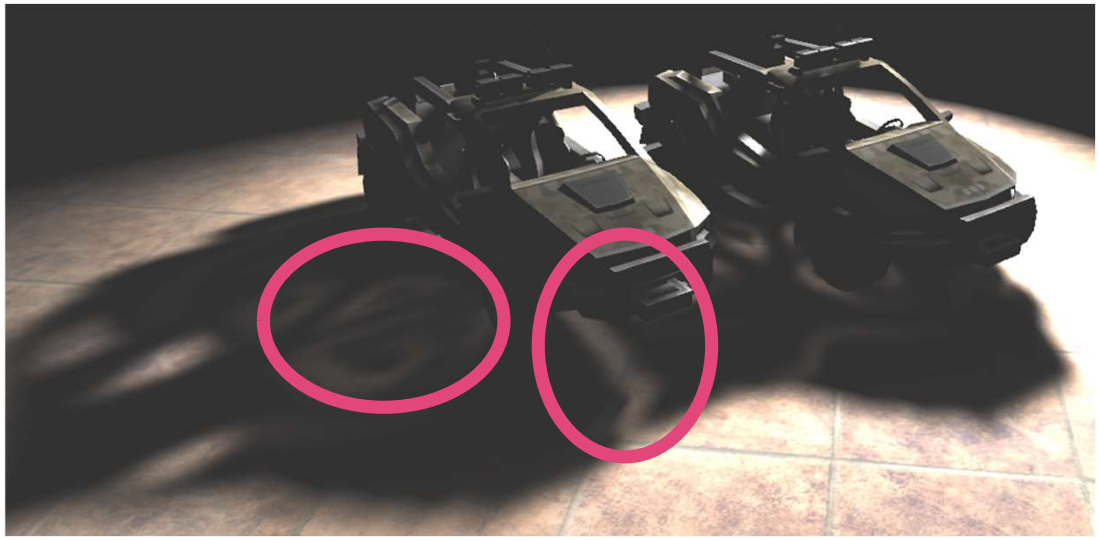
Light direction

- What is the problem?
- Solution is an upper bound, not accurate in general!



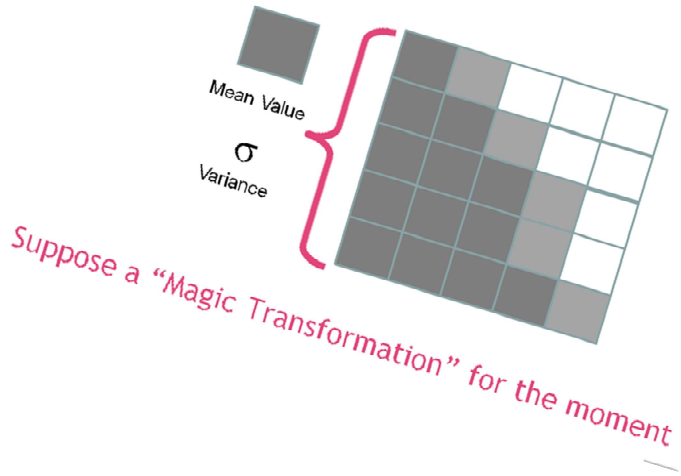
Variance SMs [Lauritzen & Donnelly 2006]

- Light leaks



But there was still some detail…

- “Magic Transformation”



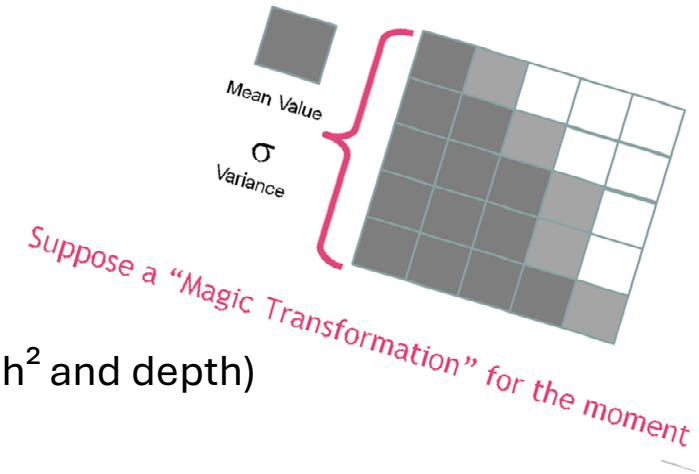
Transforming Depth Values to Distribution

- Mean Value: $\mu = E(d)$

  an average of depth values

- Variance: $\sigma^2 = E(d^2) - E(d)^2$

  easy to compute given two average values (depth^2 and depth)



➡ Need to quickly average values in a window

Mmm… I heard that one before…

Applied to Variance Shadow Maps

- All techniques compatible with

  Variance Shadow Maps:

1. Create Shadow Map and Squared Shadow Map (d, d^2)

2. Create filtered maps

   $(E(d), E(d^2))$

3. Use maps for shadow computation

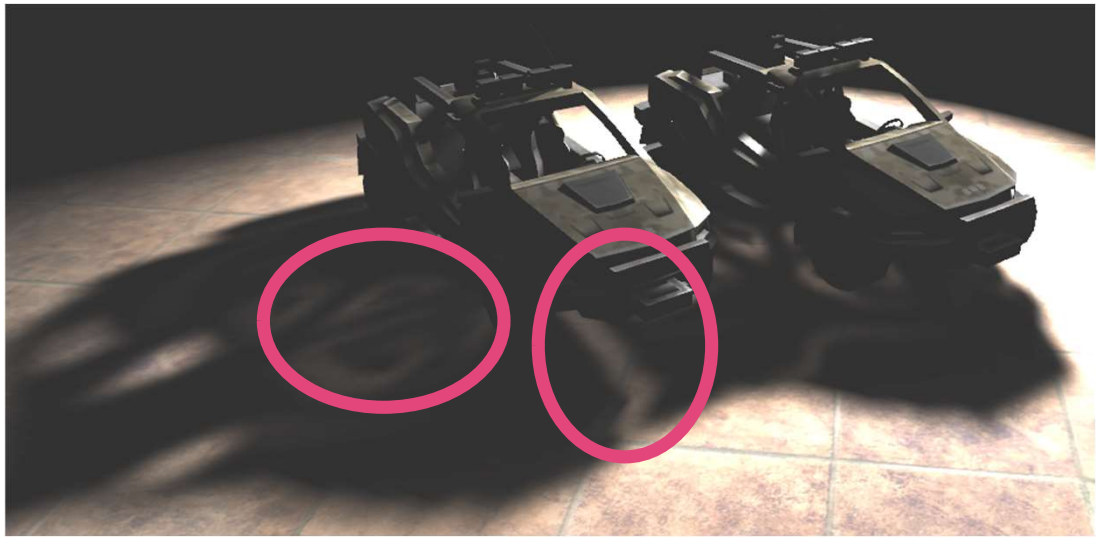
   $(\mu = E(d), \sigma^2 = E(d^2) - E(d)^2)$

Variance SMs [Lauritzen & Donnelly 2006]

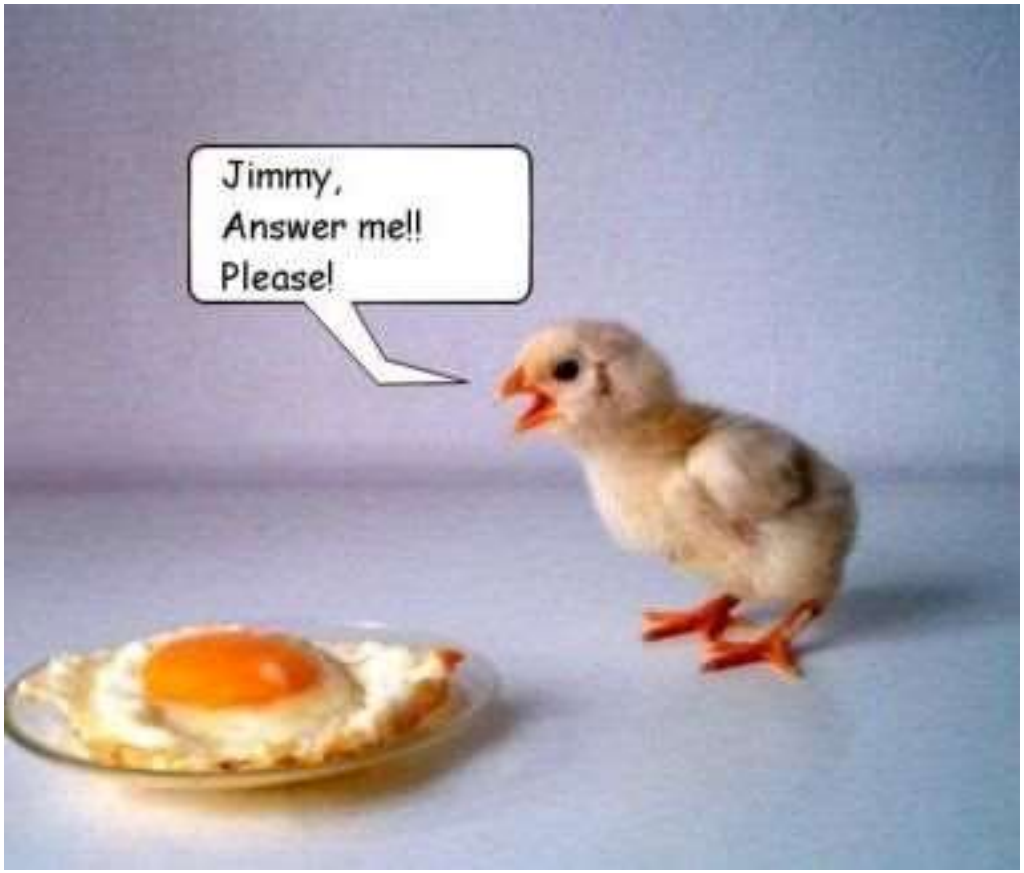
+ Very efficient

+ Independent of Filter size

- Light leaks



Questions?

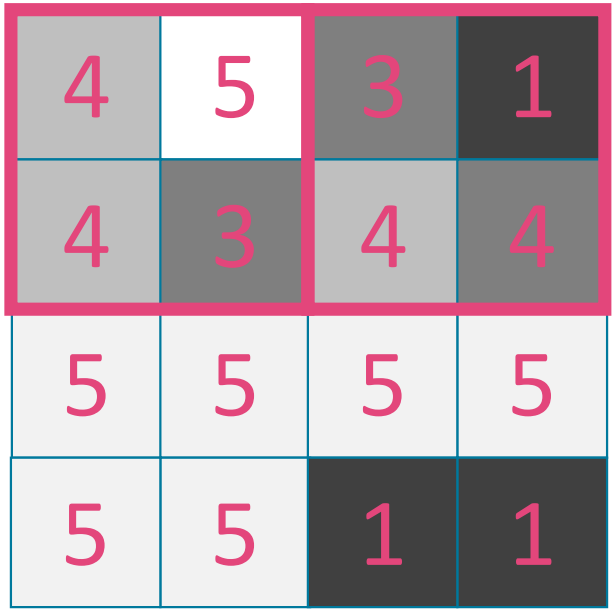


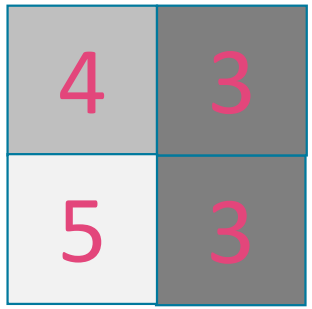
Sidenote: Computing Averages on the GPU

- MipMapping
- N-Buffer
- Summed-Area Tables

Computing Averages: MipMaps [Williams1983]

- Power of 2 hierarchy
- Average 4 pixels together for next level

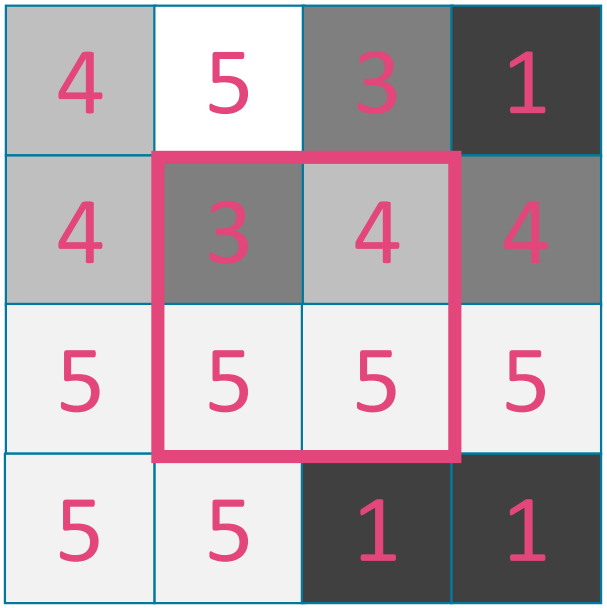


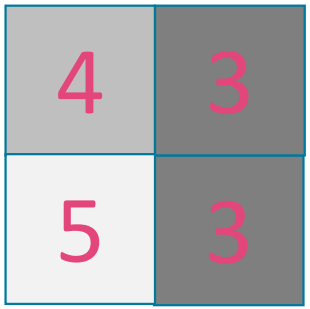


Power-of-2 kernels

Computing Averages: MipMaps [Williams1983]

- Power of 2 hierarchy
- Average 4 pixels together for next level

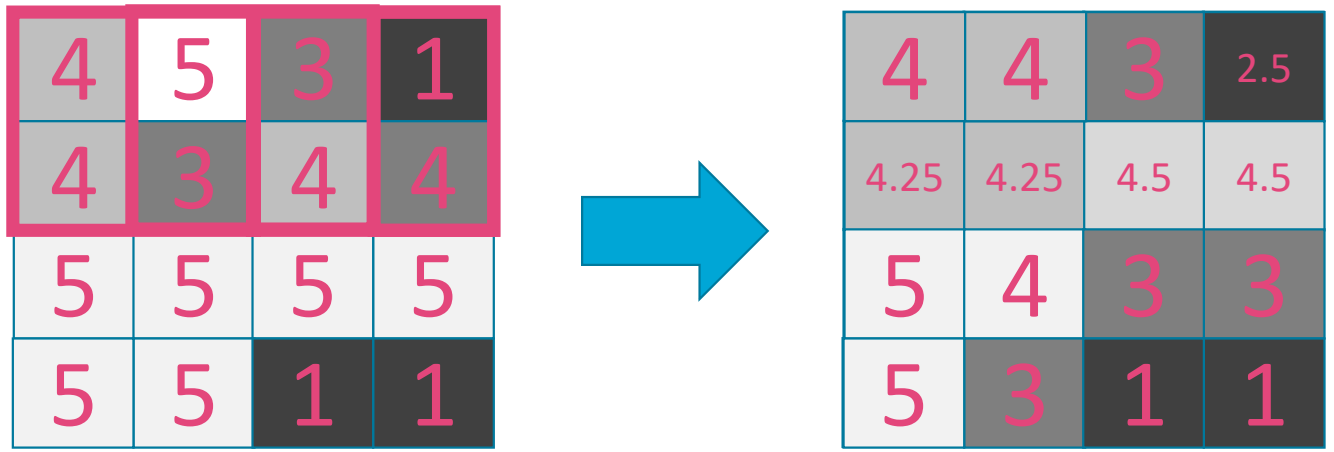




Some regions are not well represented!

Computing Averages: N-Buffers [Décoret2005]

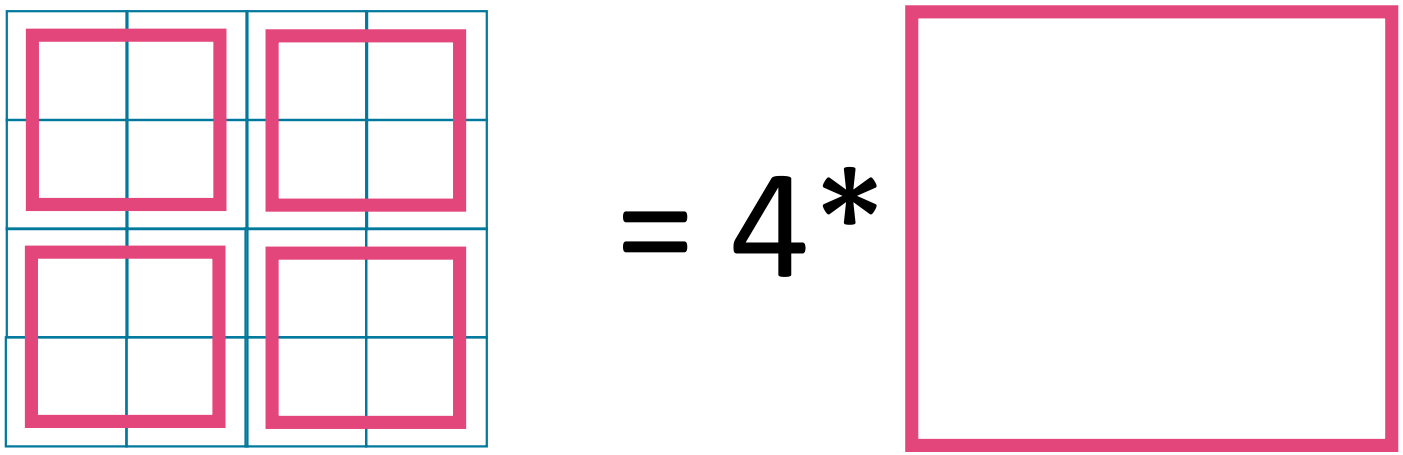
- Multi-Level Representation
- Each Level same resolution
- Level i: Pixel contains average of 2^i window



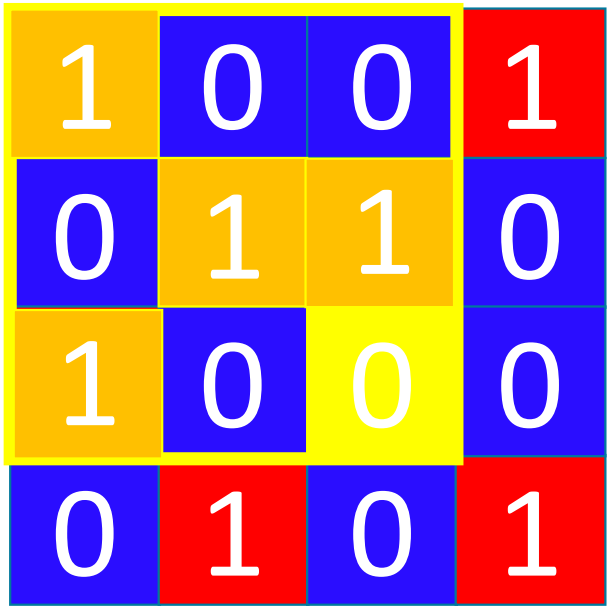
Level 0 Level 1

Computing Averages: N-Buffers [Décoret2005]

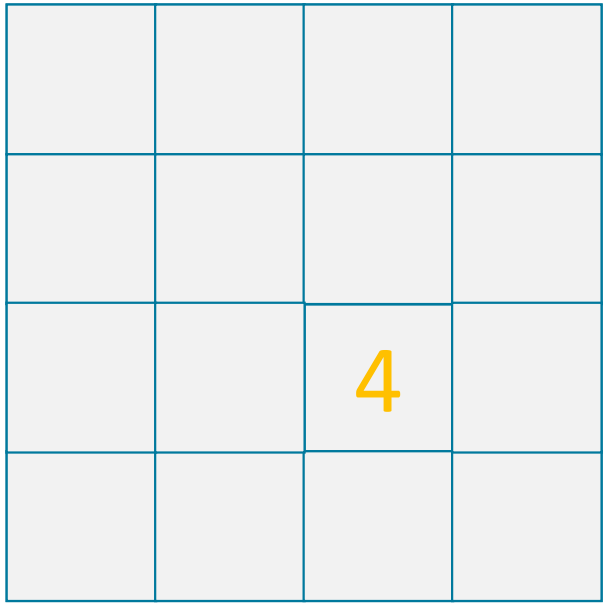
- Efficient Recursive Construction:
  - Use level i for level i+1 (4 LU per texel)



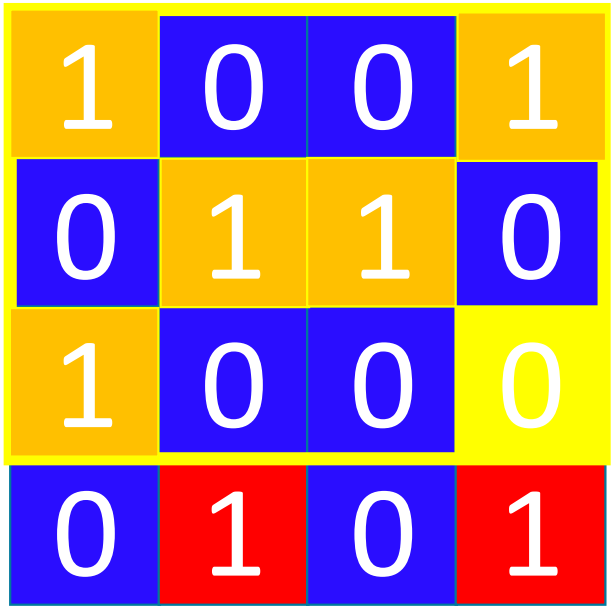
Computing Averages: Fast Summed-Area Tables [Crow1984]



FSAT



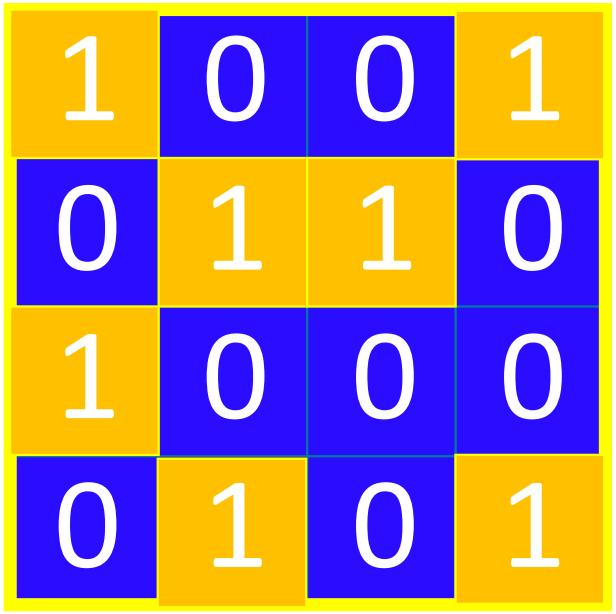
Computing Averages: Fast Summed-Area Tables [Crow1984]



FSAT

		4	5

Computing Averages: Fast Summed-Area Tables [Crow1984]



FSAT

		4	5
			7

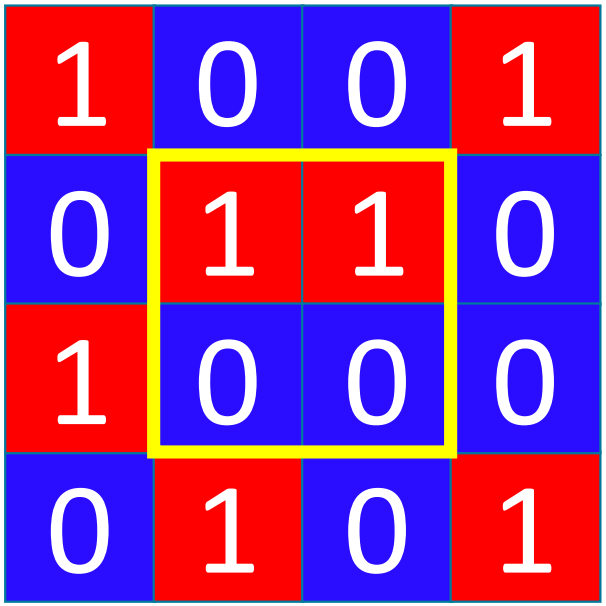
Computing Averages: Fast Summed-Area Tables [Crow1984]

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

1	1	1	2
1	2	3	4
2	3	4	5
2	4	5	7

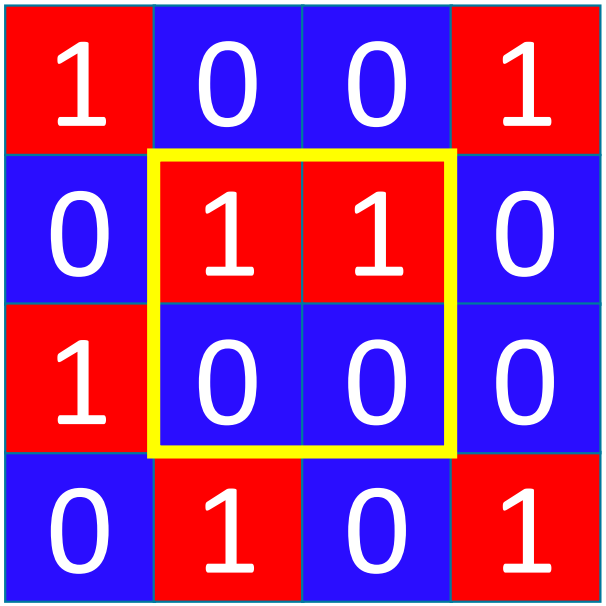
Computing Averages: Fast Summed-Area Tables [Crow1984]



FSAT

1	1	1	2
1	2	3	4
2	3	4	5
2	4	5	7

Computing Averages: Fast Summed-Area Tables [Crow1984]



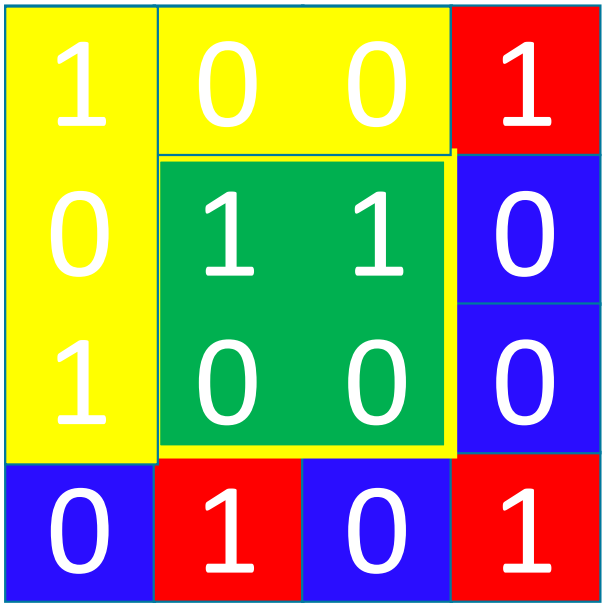
FSAT

1	1	1	2
1	2	3	4
2	3	4	5
2	4	5	7

4 -1 -2 +1 = 2

Computing Averages: Fast Summed-Area Tables [Crow1984]

Get average of any window with just 4 value lookups!



FSAT

1	1	1	2
1	2	3	4
2	3	4	5
2	4	5	7

Fast Summed-Area Tables

- Efficient GPU Construction [Hensley et al.2005]

Comparison

Filtering Quality

- MipMaps : fastest, but lowest quality

- N-Buffer : average construction time, medium quality

  (higher memory consumption)

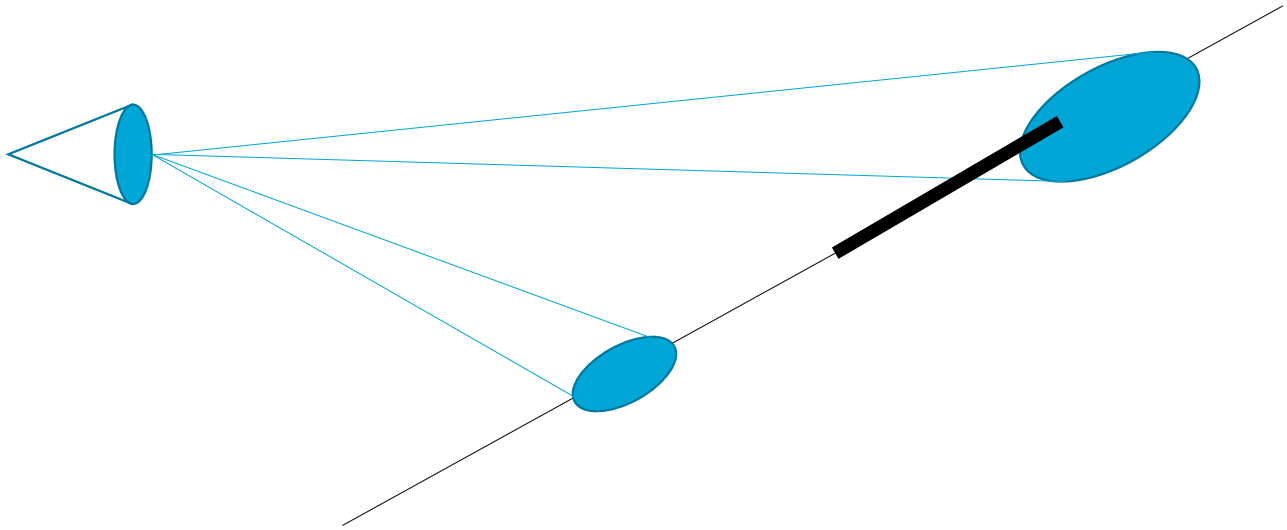
- FSAT: slower construction time, highest quality

  (slightly more costly lookups)

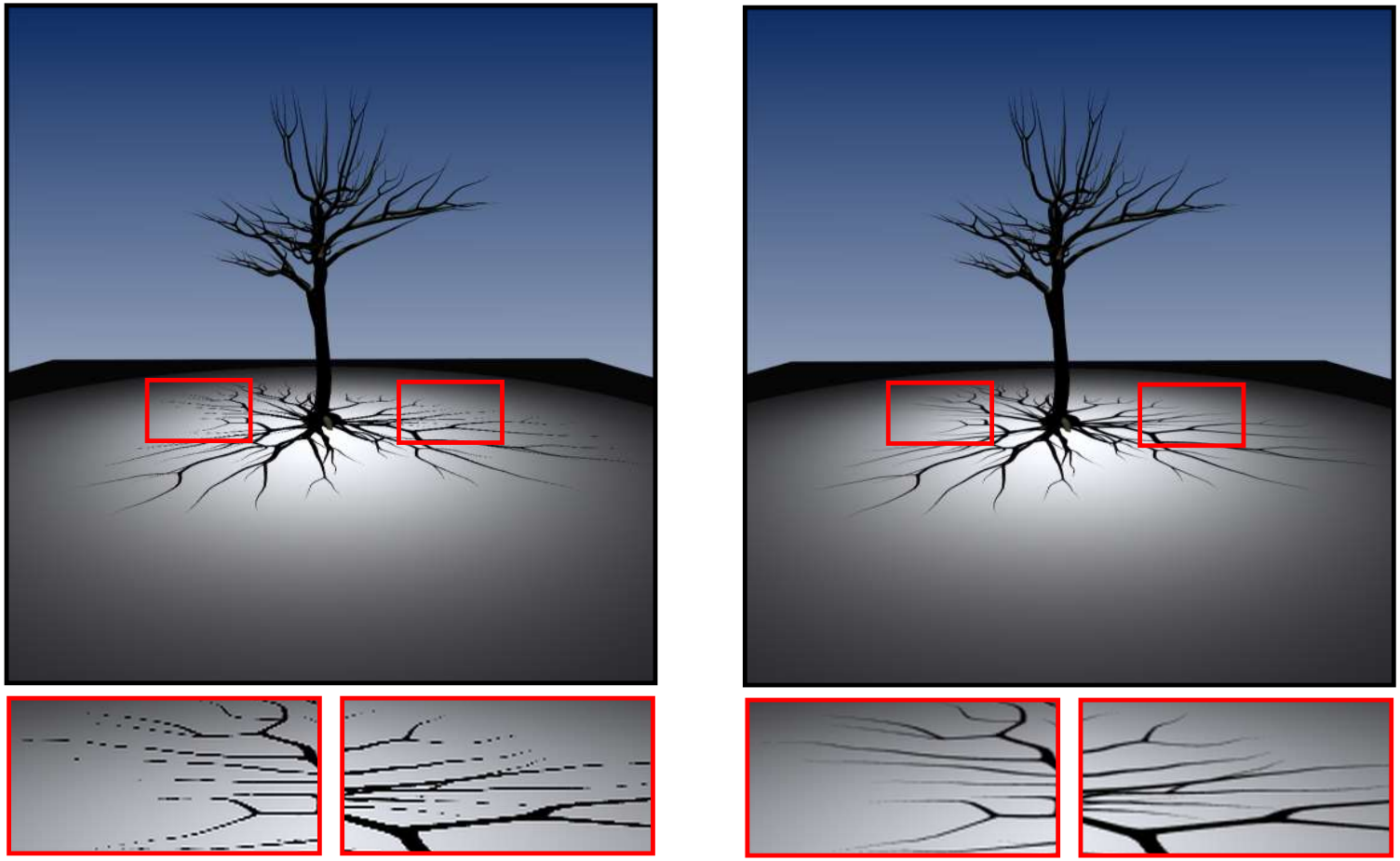
- Please note:
  All methods allow to choose the region size on the fly (i.e., varying filter size)

Perspective Foreshortening

- Choose kernel according to viewing distance



Perspective foreshortening



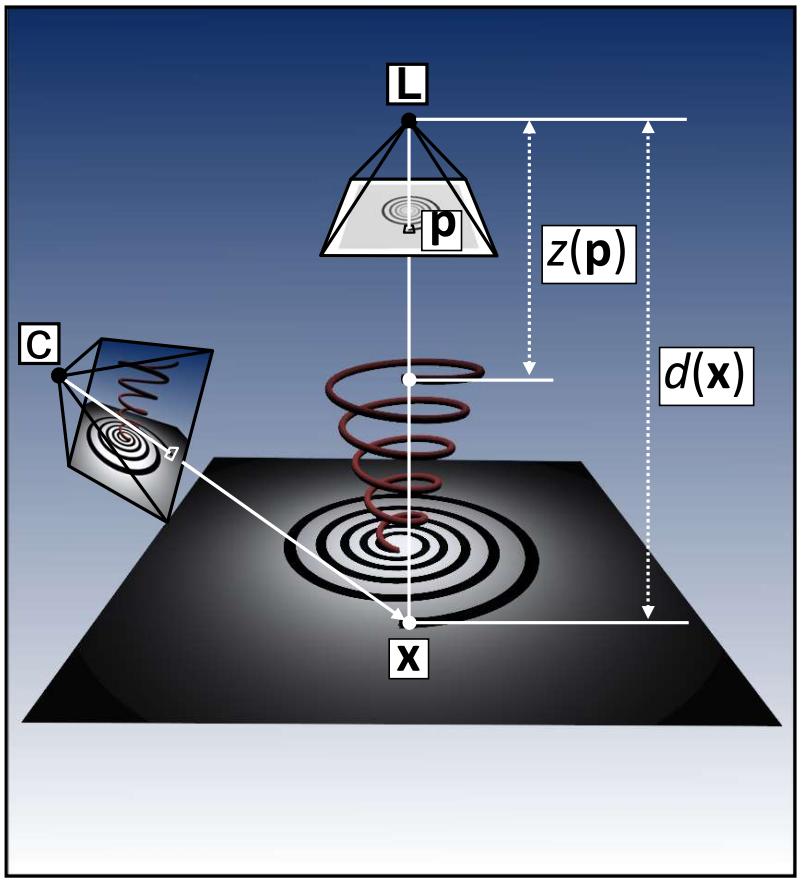
Questions?

Approximate the Shadow Function

- Convolution Shadow Maps [Annen et al. 2007]

- Exponential Shadow Maps [Annen et al. 2008, Salvi 2008]

Shadow Mapping [Williams 1978]



Animation courtesy of Annen et al.

- $\mathbf{x} \in \mathbb{R}^3$
- $\mathbf{p} \in \mathbb{R}^2$

Shadow Test:

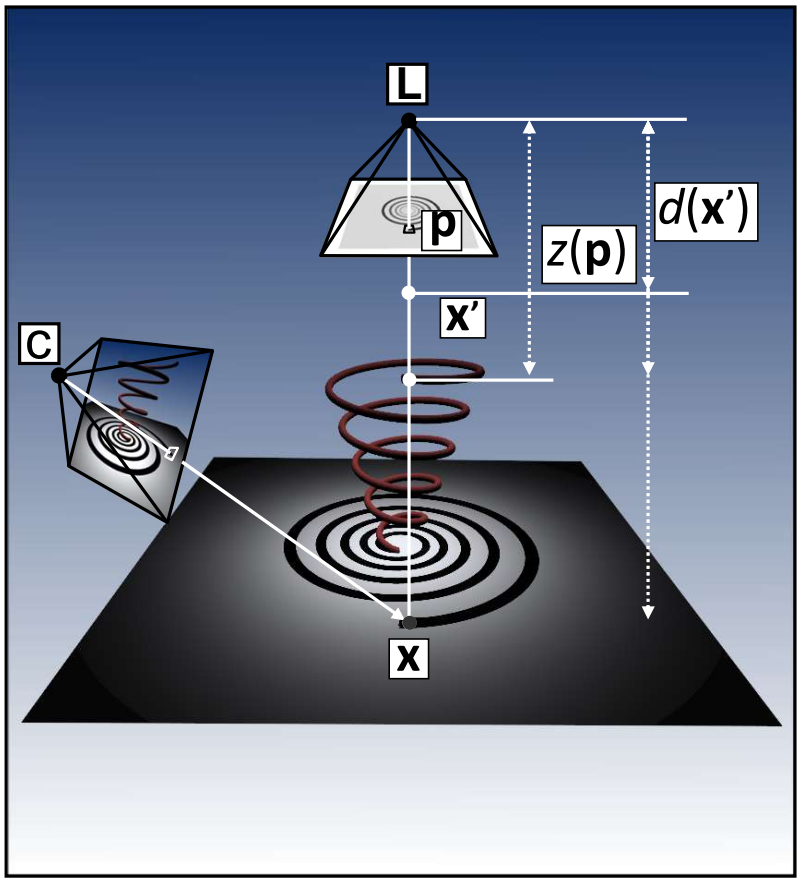
$$t(d(\mathbf{x}),z(\mathbf{p})):=$$

$$s(d(\mathbf{x})-z(\mathbf{p})),$$

where

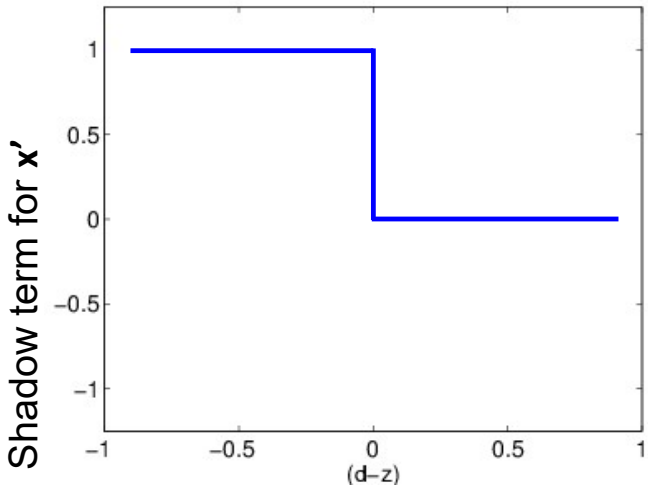
$$s(\mathbf{x}):=\begin{cases}1 \text{ if } \mathbf{x}<=\mathbf{0}\\ 0 \text{ else}\end{cases}$$

Shadow Test: $s(\mathbf{x})$



Animation courtesy of Annen et al.

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



- Heaviside Step Function: $H(t)$

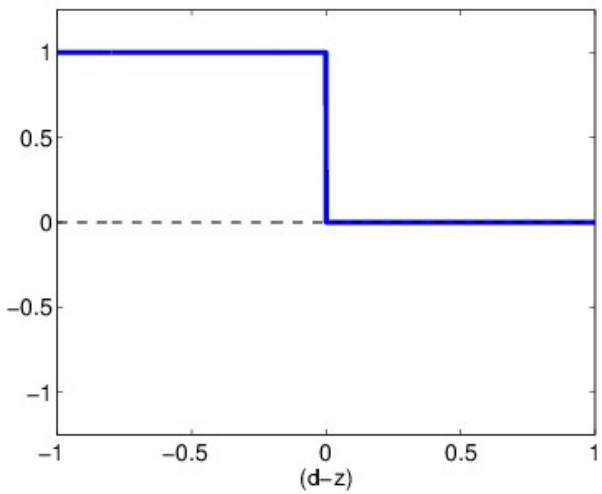
Fourier Analysis

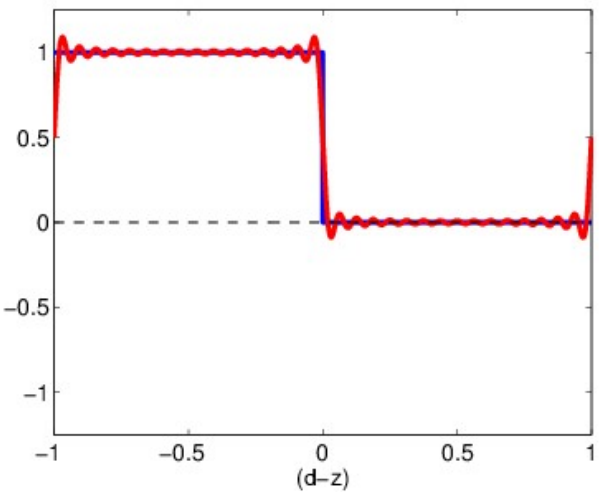
- Check out:
- https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/

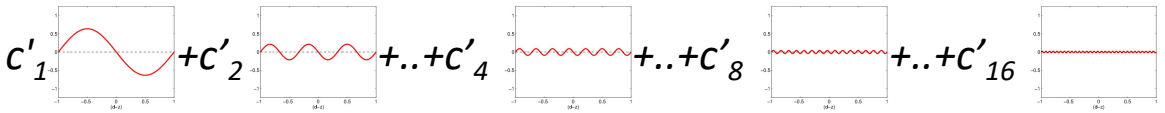
$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \frac{k2\pi x}{T} + b_k \sin \frac{k2\pi x}{T} \right)$$

Fourier Analysis

- Approximate shadow test with Fourier series





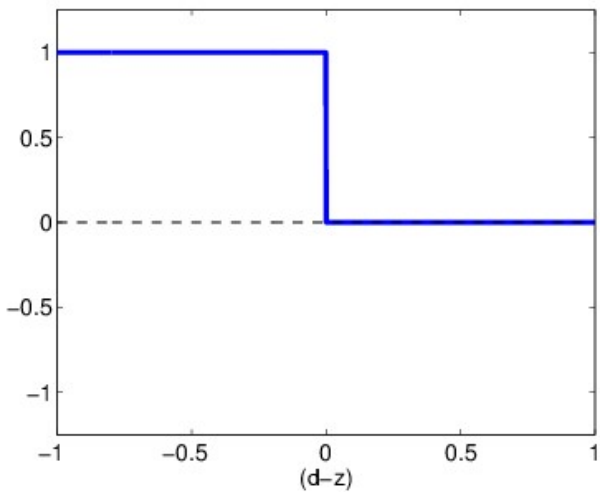


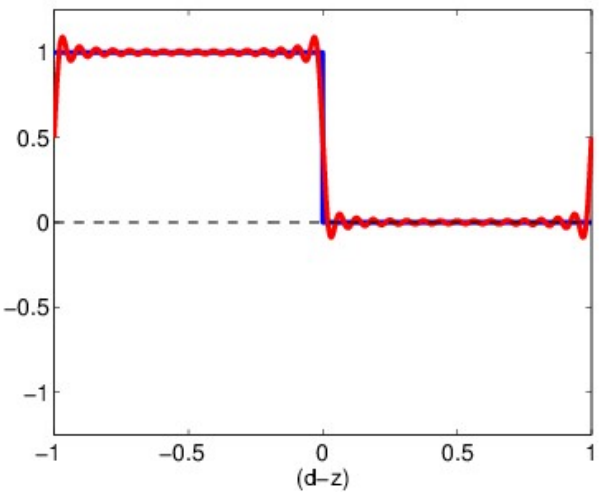
We can write the shadow test as:

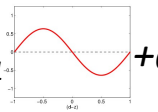
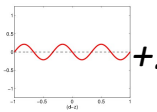
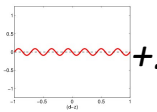
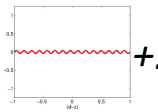
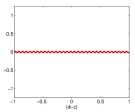
$$f(d\text{-}z) = \sum_{i=1}^{\infty} c_i(d)\, B_i(z)$$

Fourier Analysis

- Approximate shadow test with Fourier series





c'_1  $+c'_2$  $+..+c'_4$  $+..+c'_8$  $+..+c'_{16}$ 

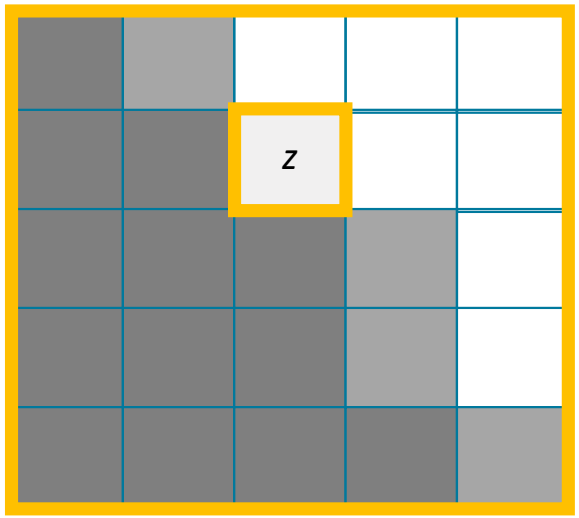
We can write the shadow test as:

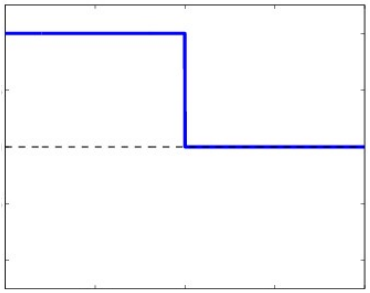
$$f(d\text{-}z) = \sum_{i=1}^{\infty} c_i(d)\, B_i(z) = s(d\text{-}z)$$

Fourier Analysis

- Why is this useful???

$$\sum_{s=1}^{N} f(d-z_s) = \sum_{s=1}^{N} \sum_{i=1}^{\infty} c_i(d)\, B_i(z_s)$$



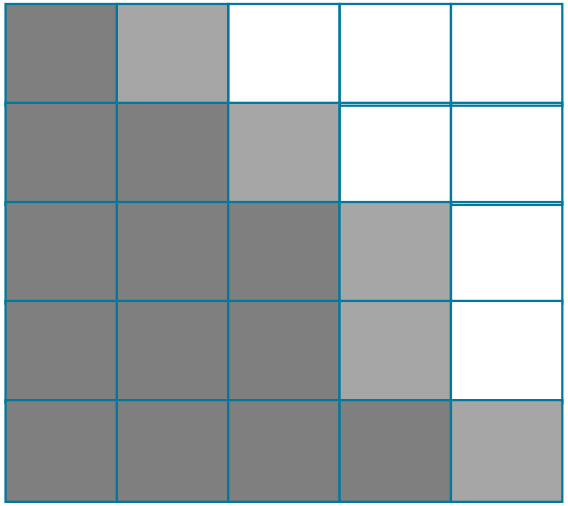


Fourier Analysis

- Why is this useful???

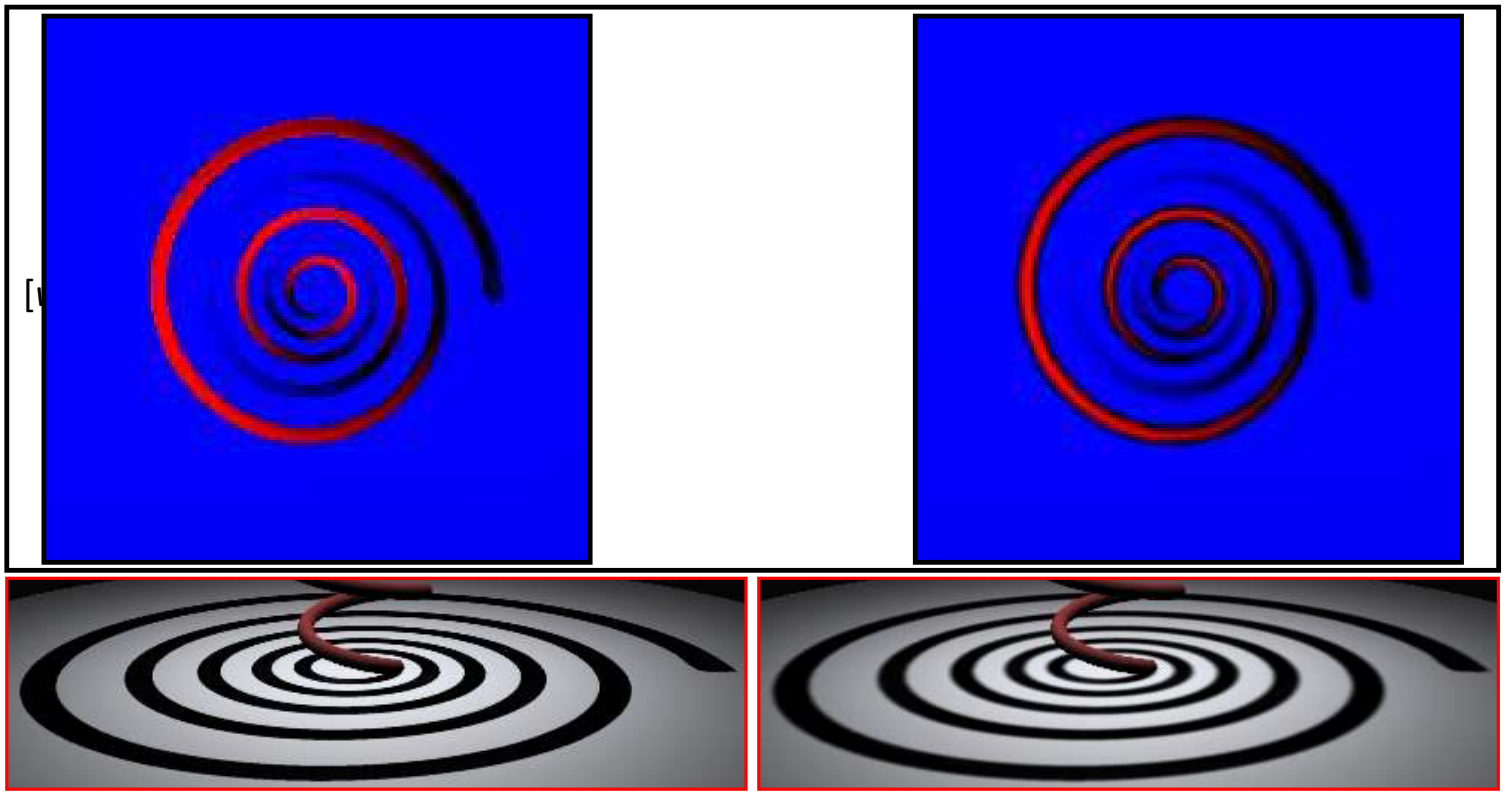
$$\sum_{s=1}^{N} f(d - z_s) = \sum_{i=1}^{\infty} c_i(d) \quad \sum_{s=1}^{N} B_i(z_s)$$

Average of result of Bi's applied to SM



Filtering Example

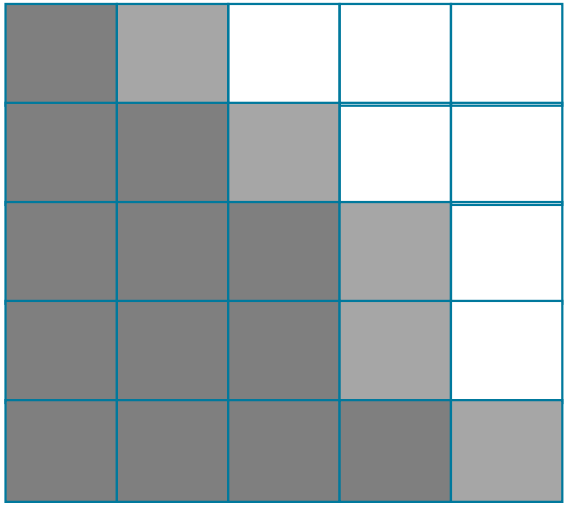
Original $B_i(z)$ After filtering $B_i(z)$



- We can use any fast averaging method

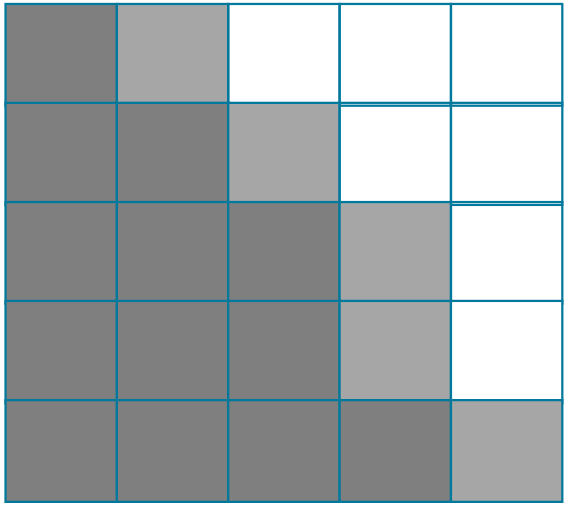
Reality strikes back...

$$\sum_{s=1}^{N} f(d-z_s) = \sum_{i=1}^{\infty} c_i(d) \sum_{s=1}^{N} B_i(z_s)$$



Reality strikes back...

$$\sum_{s=1}^{N} f(d-z_s) \approx \sum_{i=1}^{M} c_i(d) \sum_{s=1}^{N} B_i(z_s)$$



Truncation can lead to ringing

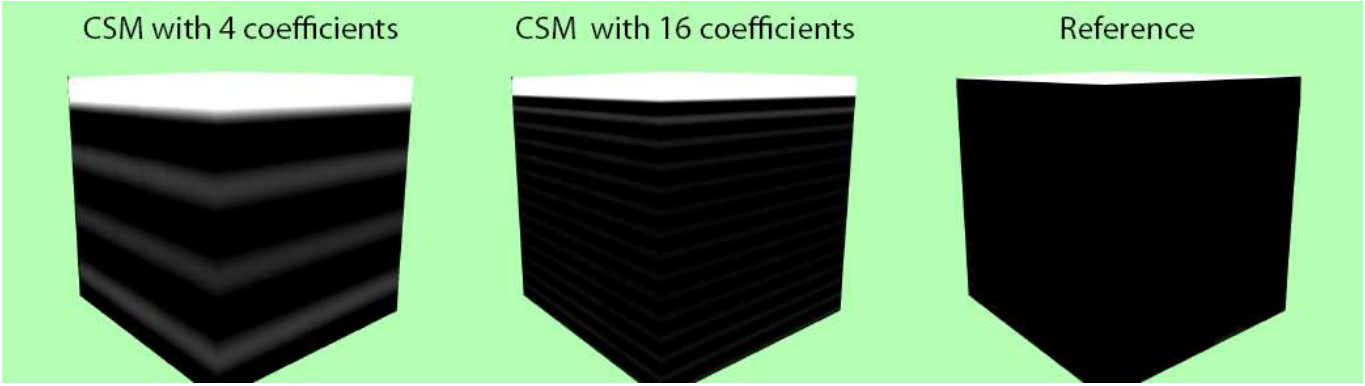
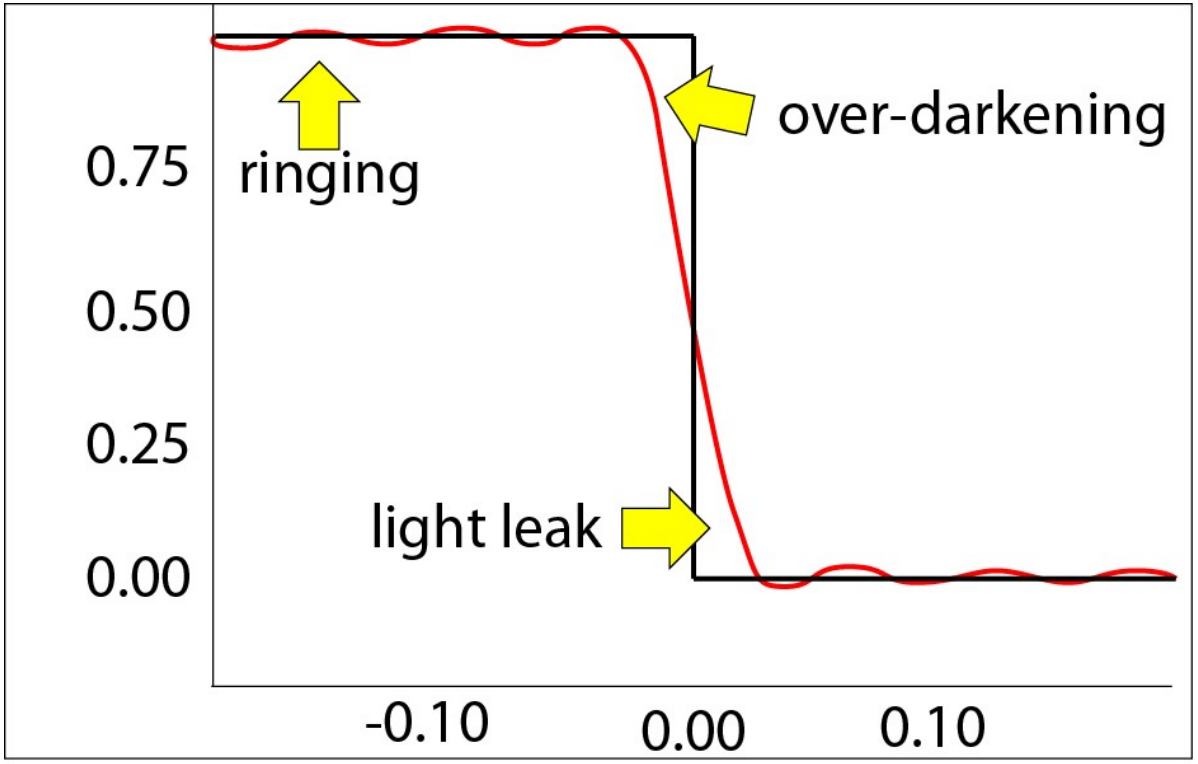


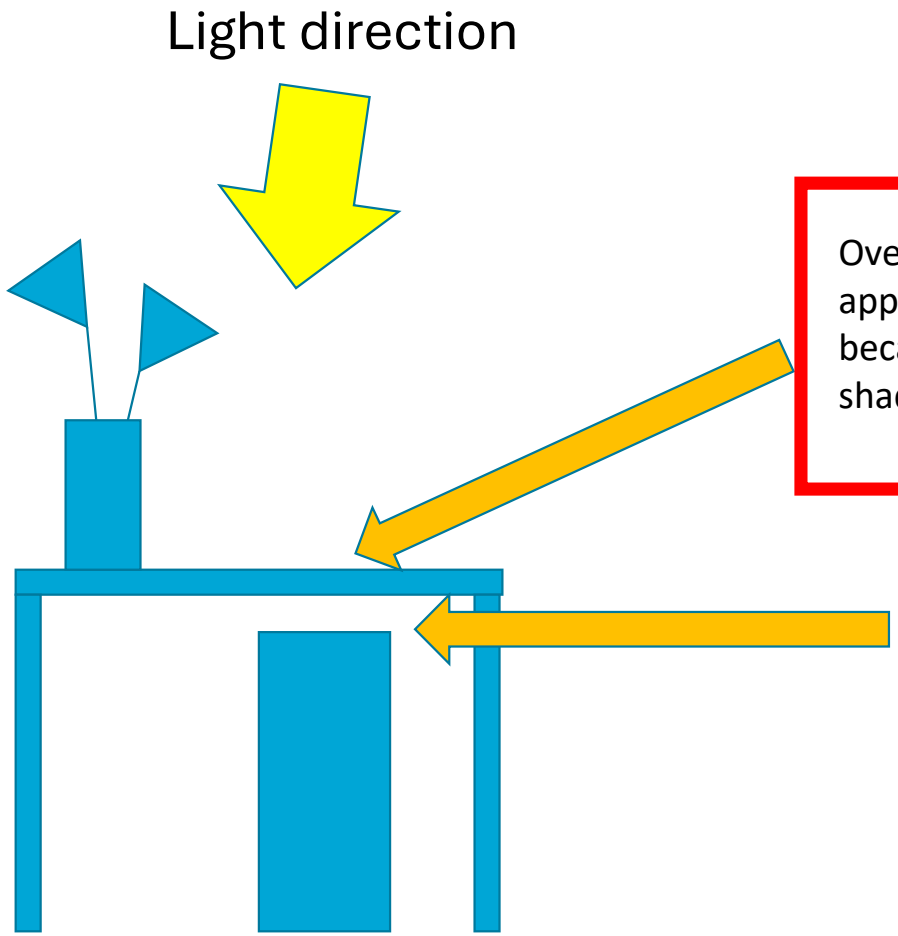
Illustration shows the shadow calculation not considering diffuse lighting

Reduce Ringing



What is worse? Light leaks or over-darkening?

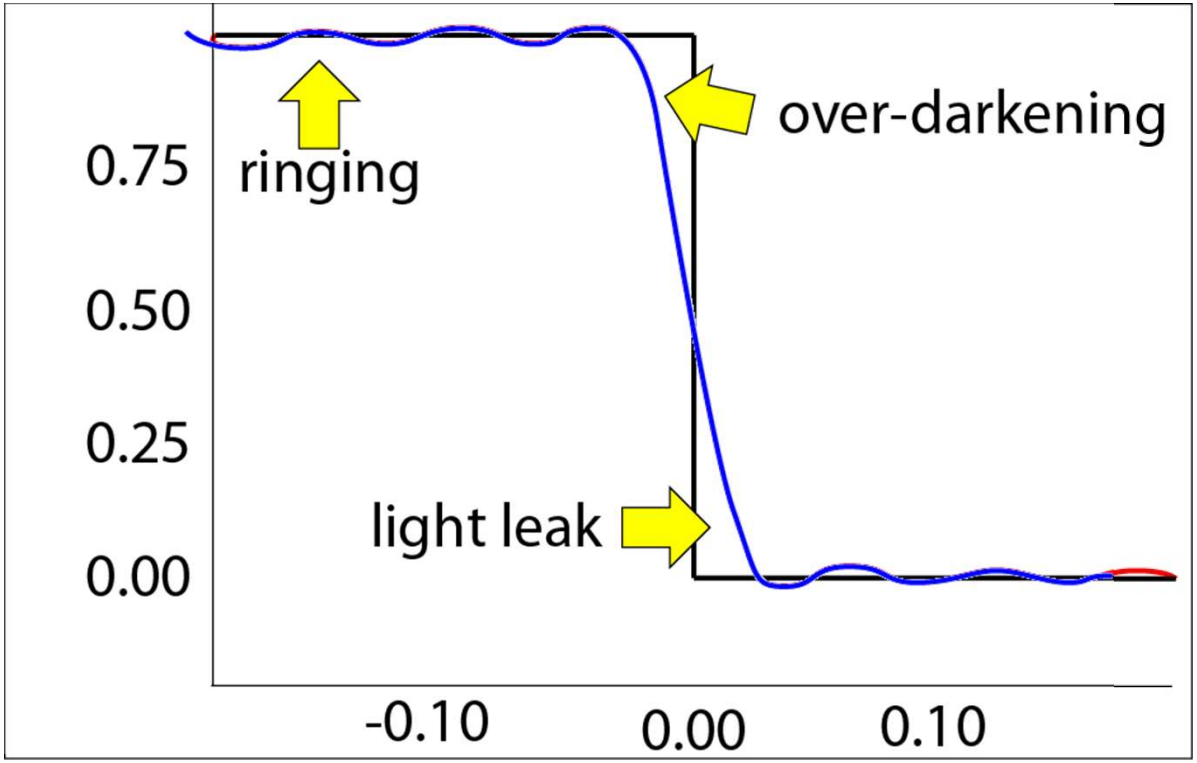
Light Leaks vs. Over-darkening



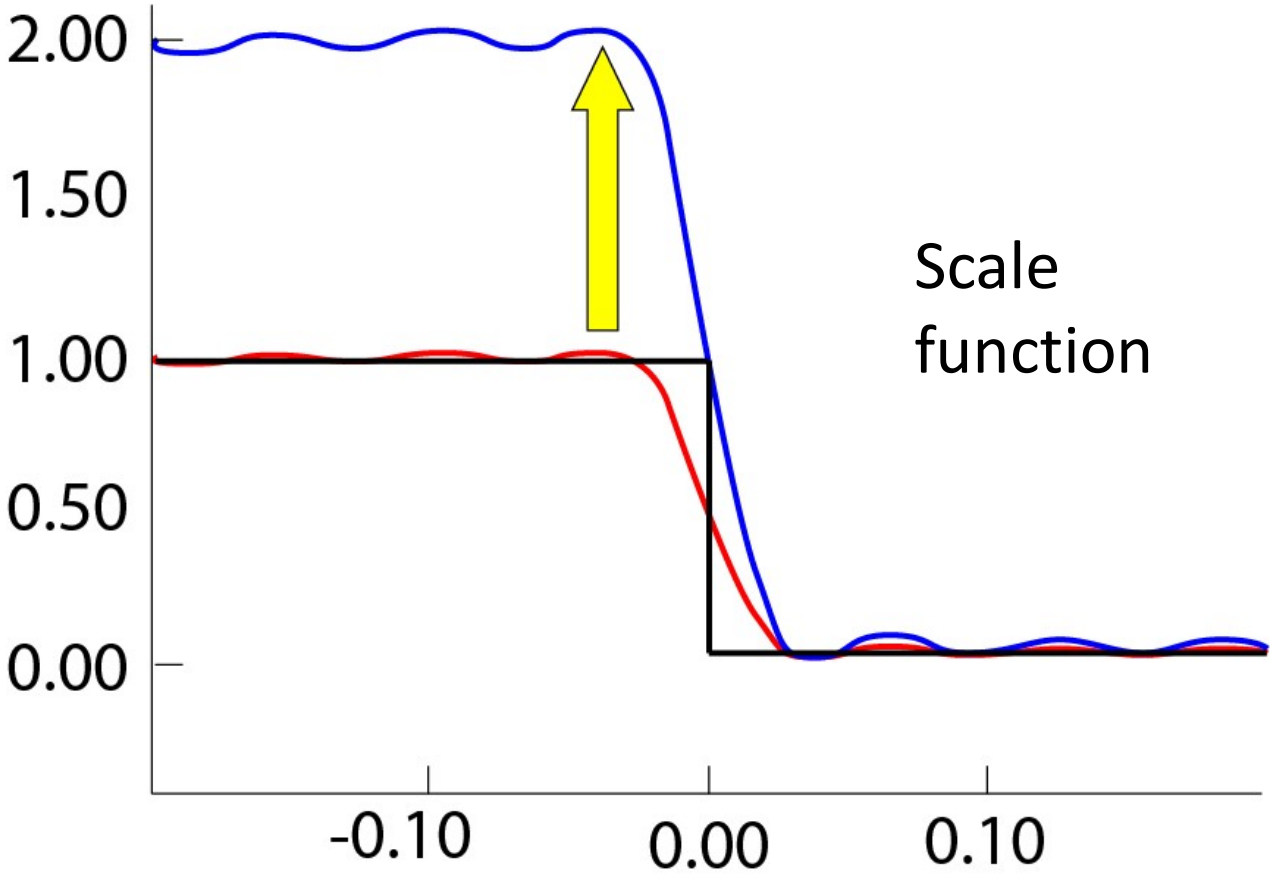
Over-darkening would appear on the table because it could falsely shadow itself

Light leaks would appear on the box below as its depth is near to the table depth

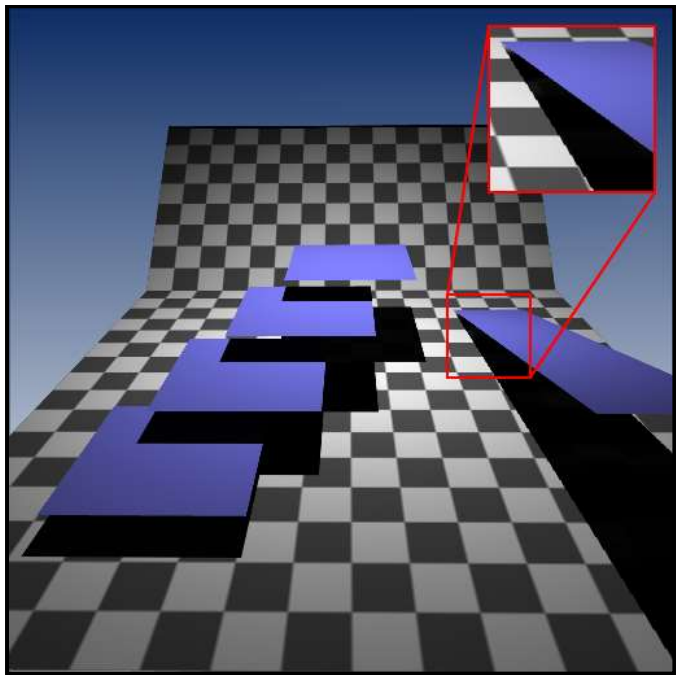
Reduce Ringing



Reduce Ringing



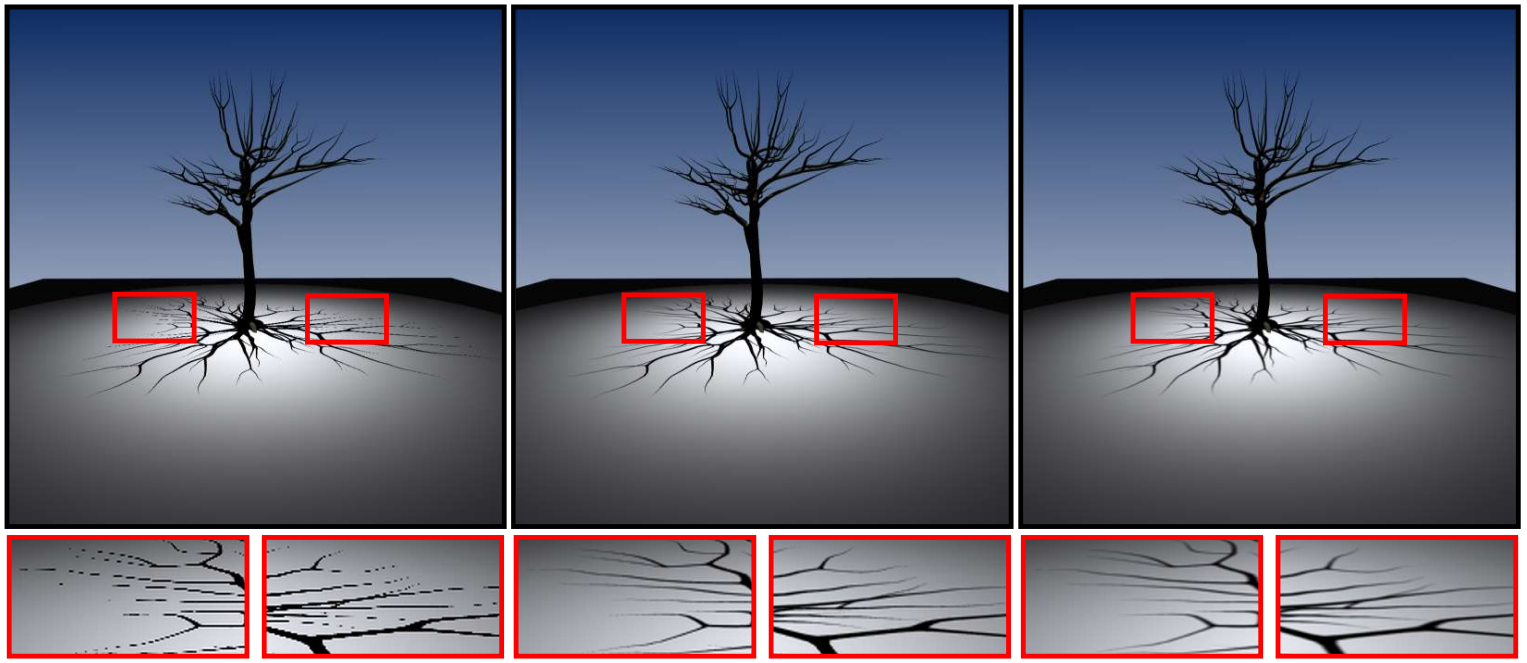
Influence of reconstruction order M



M = 2 6

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

Results



PCF (NVIDIA) CSM CSM – 7x7 Gauss

CSM [Annen et al. 2007]

+ Beautiful theoretical result

+ high-quality smoothing
  (includes anisotropic blur)

~ relatively fast (for M=8)

- Relatively high memory consumption

Questions


DAD! Can i ask you a question?
But son, you just did.
Can i ask two?
You just did...
LOL0.COM

Exponential Shadow Maps

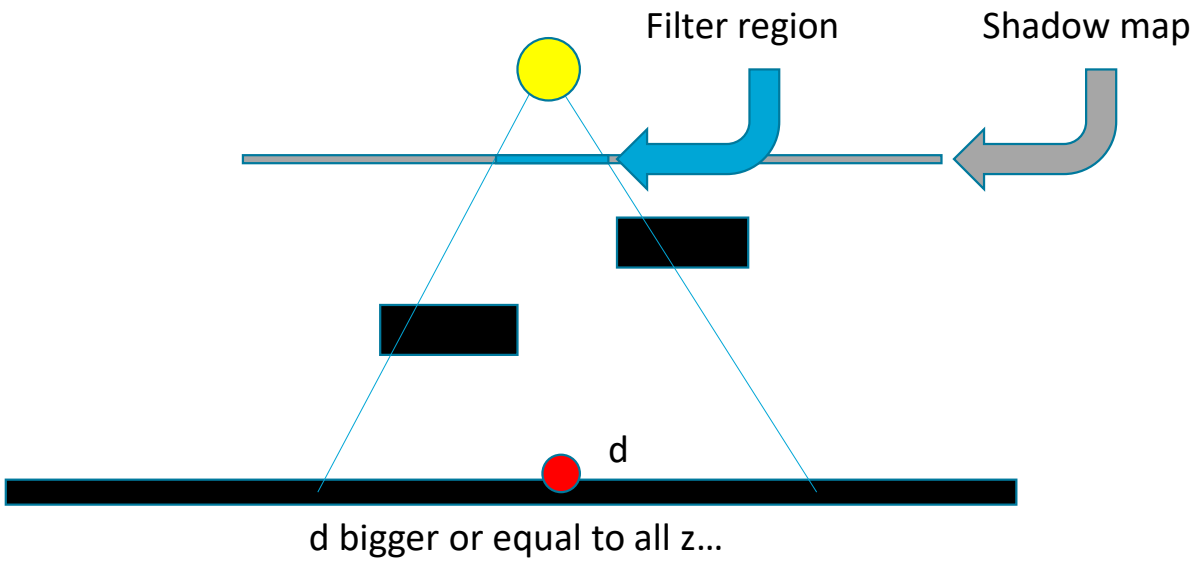
[Annen et al. 2008, Salvi2008]

- Are their other ways so that the shadow test could be decomposed into a filtered part depending only on SM values?

Exponential SM [Annen et al. 2008, Salvi2008]

- Assumption:

  Tested pixel is the farthest from the light
  in the filter region of the shadow map



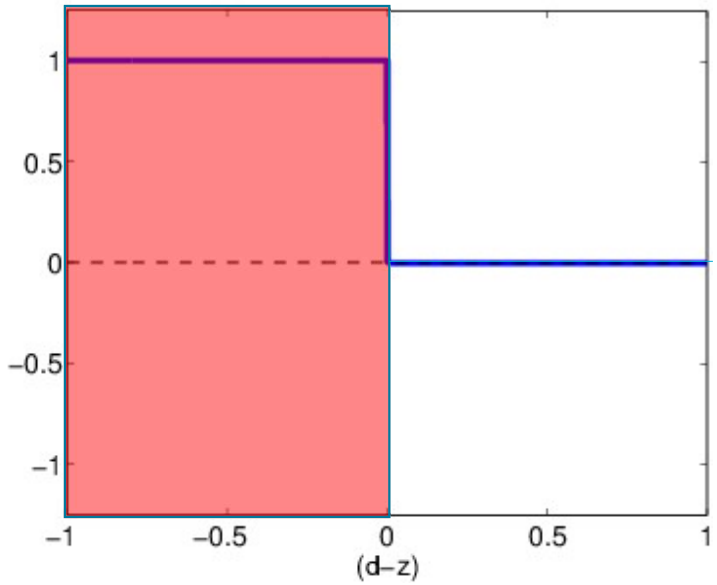
Exponential SM [Annen et al. 2008, Salvi2008]

- Assumption:

  Tested pixel is the farthest from the light

  in the filter region of the shadow map

  -> others are either equal or closer



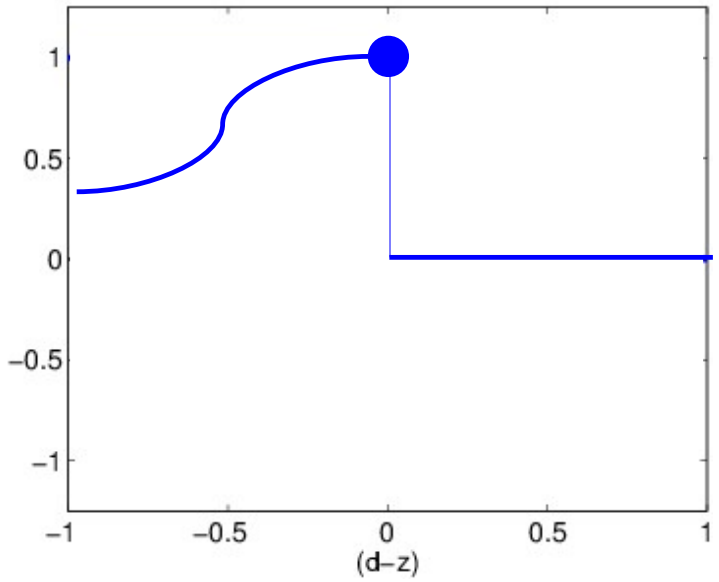
Exponential SM [Annen et al. 2008, Salvi2008]

- Assumption:

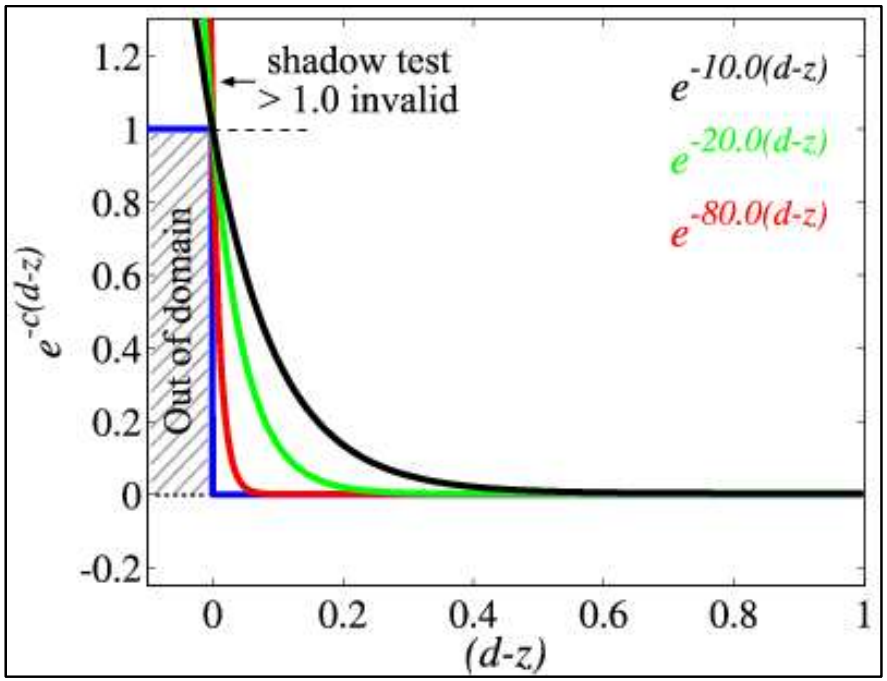
  Tested pixel is the farthest from the light

  in the filter region of the shadow map

  -> others are either equal or closer



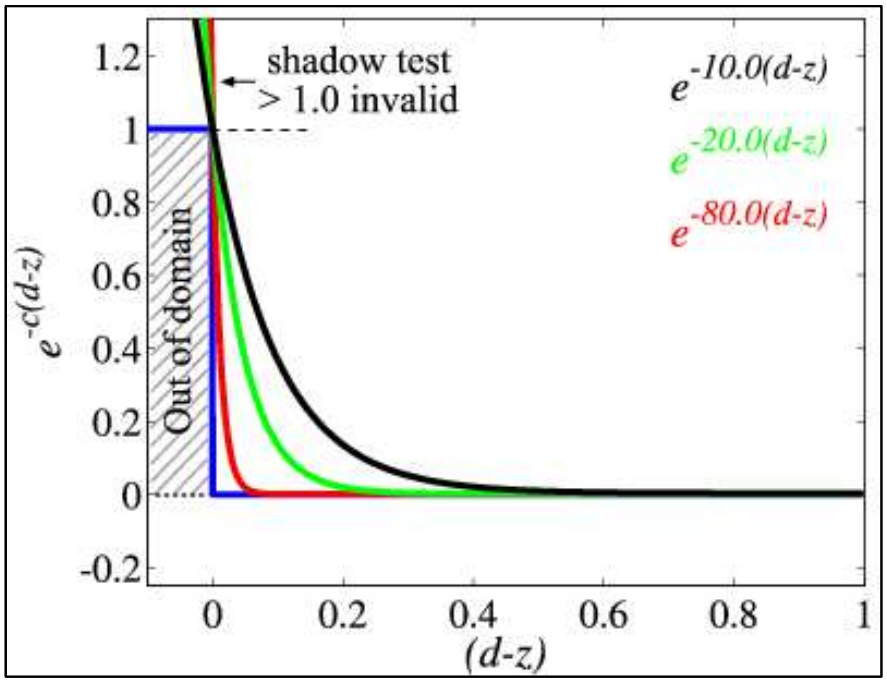
Exponential SM [Annen et al. 2008, Salvi2008]



Exponential SM [Annen et al. 2008, Salvi2008]

- Shadow function assumed to be:

$$s(d\text{-}z) \approx e^{-c(d(\mathbf{x}) - z(\mathbf{p}))}$$

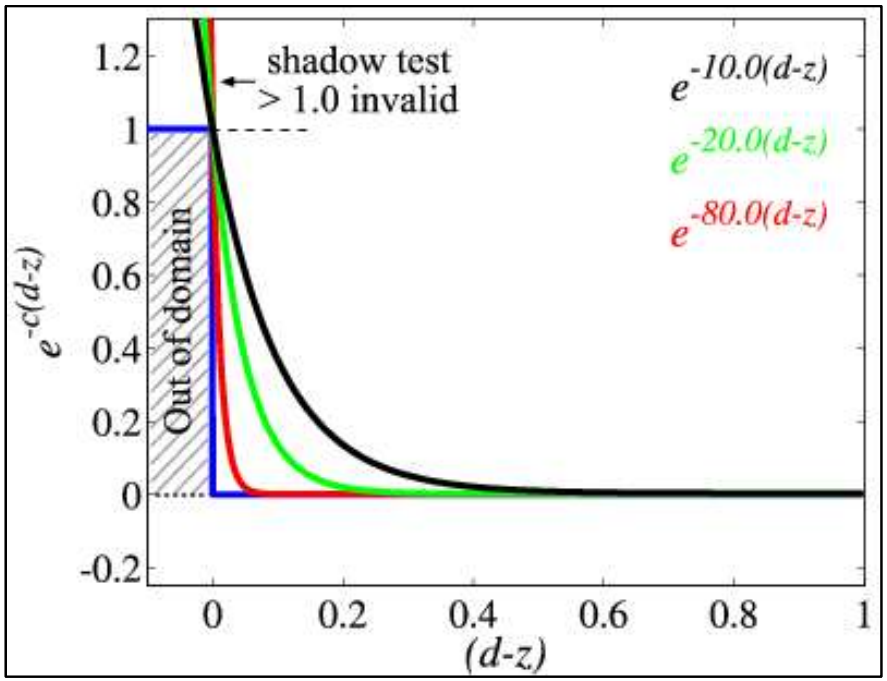


Exponential SM [Annen et al. 2008, Salvi2008]

- Shadow function assumed to be:

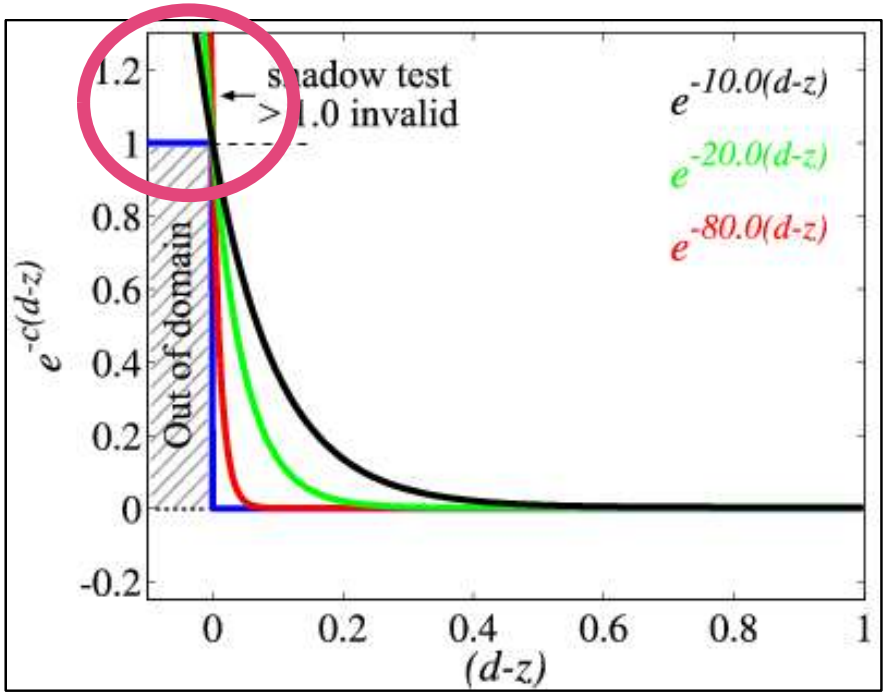
$$s(d\text{-}z) \approx e^{-c(d(\mathbf{x}) - z(\mathbf{p}))} = e^{-c(d(\mathbf{x}))} e^{cz(\mathbf{p})}$$

decomposition



Problem: Overshooting

- Remember the assumption:
- We need: d > z ! Otherwise, the approximation can be very wrong!



Detection of Erroneous Pixels

Hacky fix:

Detect the locations where the assumption fails
and those pixels use PCF computation

ESM [Annen et al. 2008, Salvi2008]

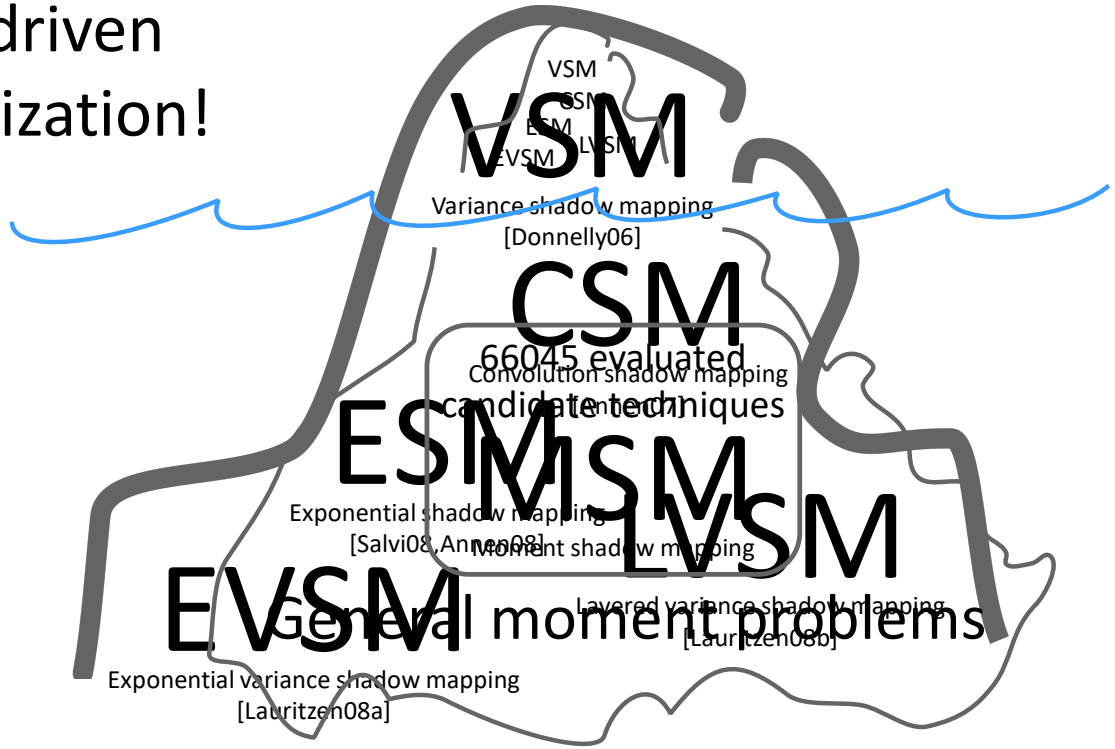
+ Faster than CSM some scenes
  (depends on correction)
+ Same quality as CSM
+ Memory consumption lower than CSM
+ Creation of ESM is faster

- Correction cost unpredictable
- Should be used with smaller kernels

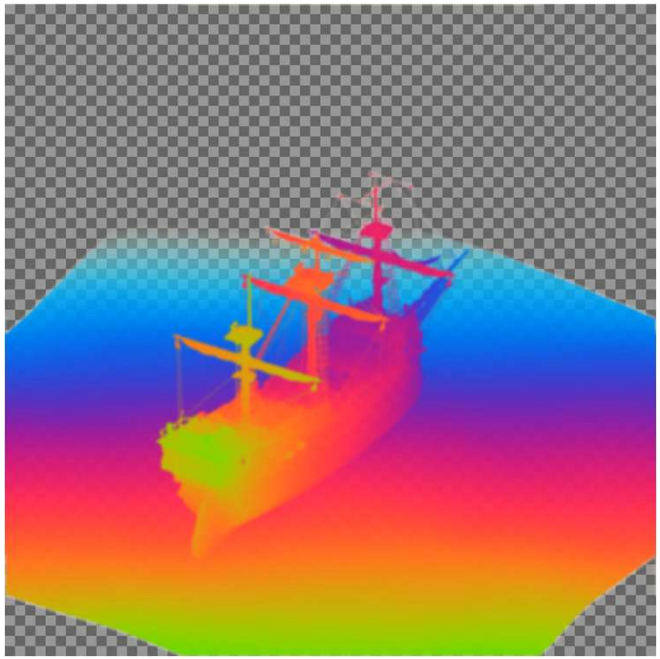
What is the best?

- Moment Shadow Maps! [Peters & Klein i3D05]

Data-driven optimization!



Moment Shadow Mapping



Summary

- Combat Aliasing for Shadow mapping
- Discussed main algorithms and limitations
- Filtering methods
  - MipMapping
  - NBuffers
  - Summed Area Tables

Overview

- Really fast:

  Variance SM

- Best tradeoff Quality/Performance:

  Exponential Shadow Maps
  Moment Shadow Maps

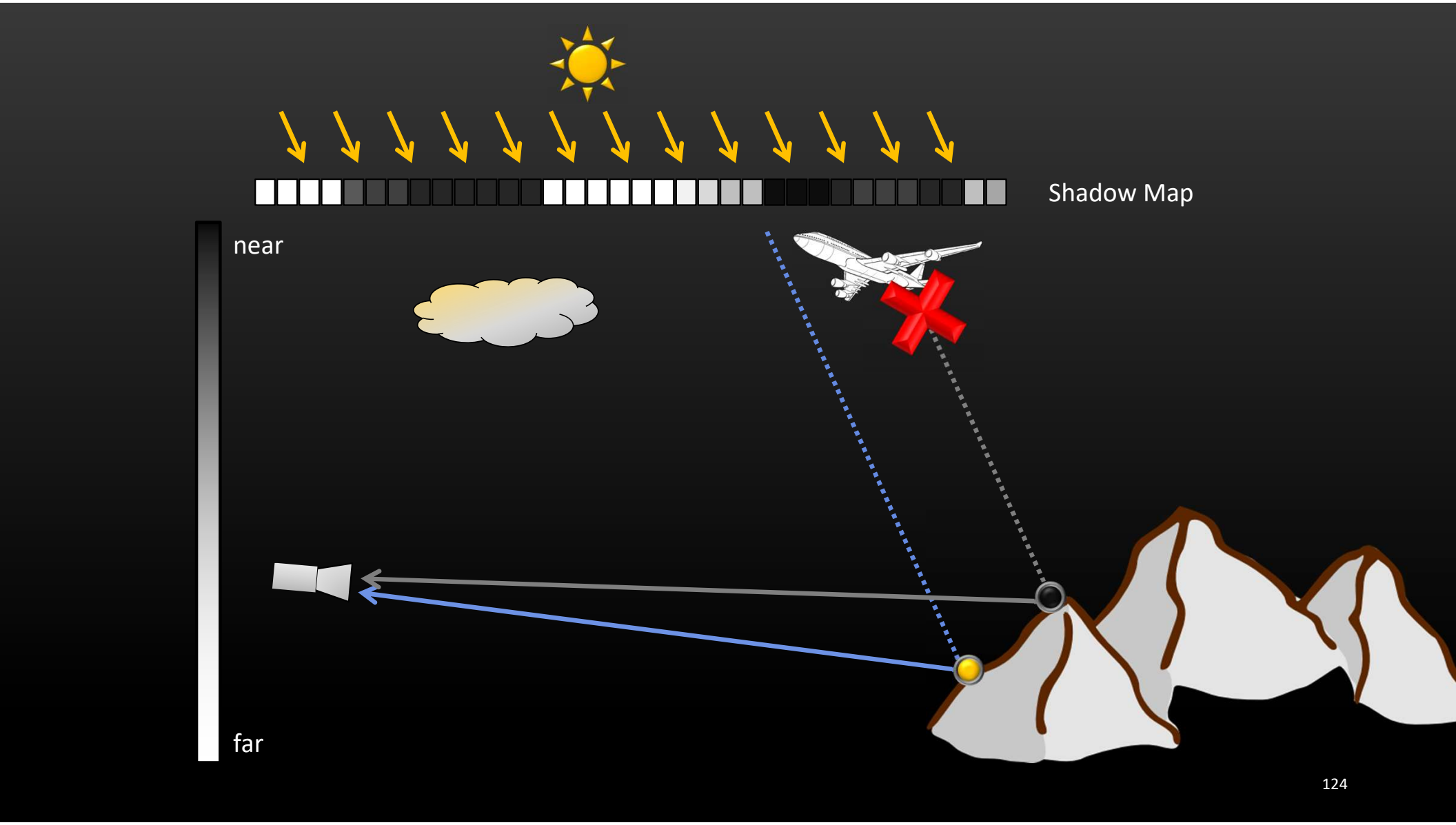
BUT all these methods filter hard shadows.

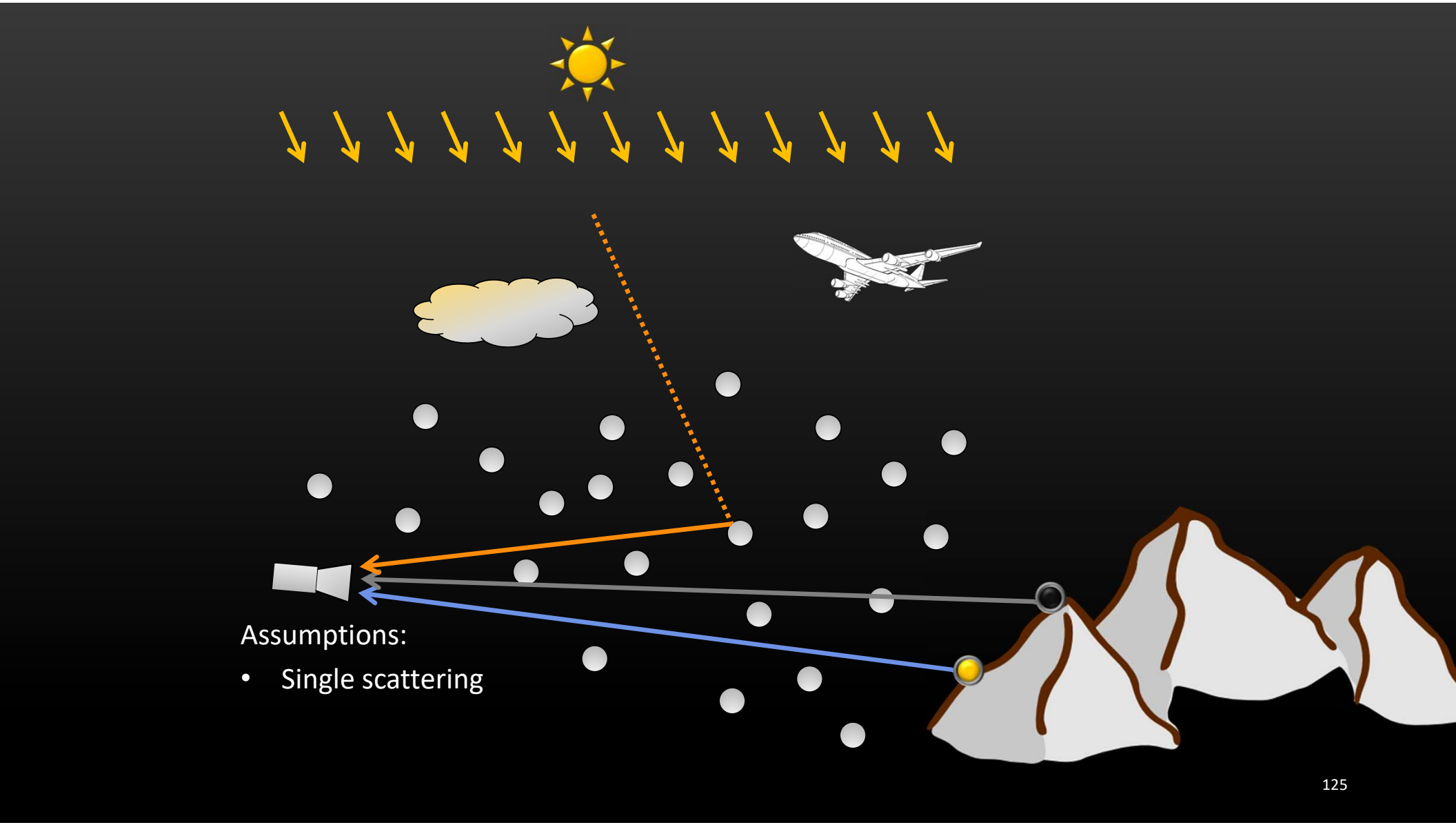
There is no real penumbra computation...

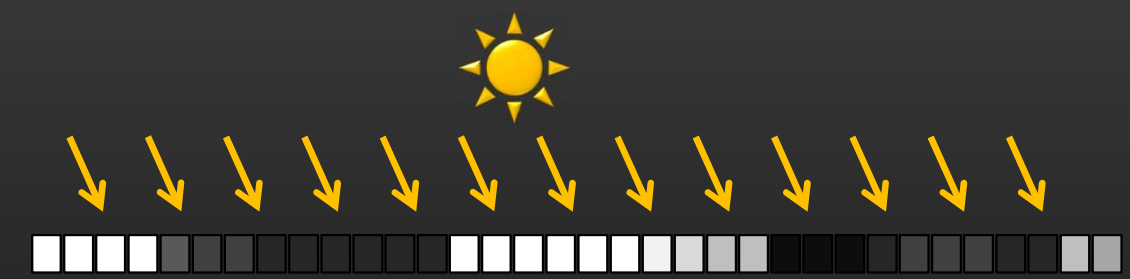
Unreal Engine 5

Unreal Engine 5

Additional Application of Soft Shadows

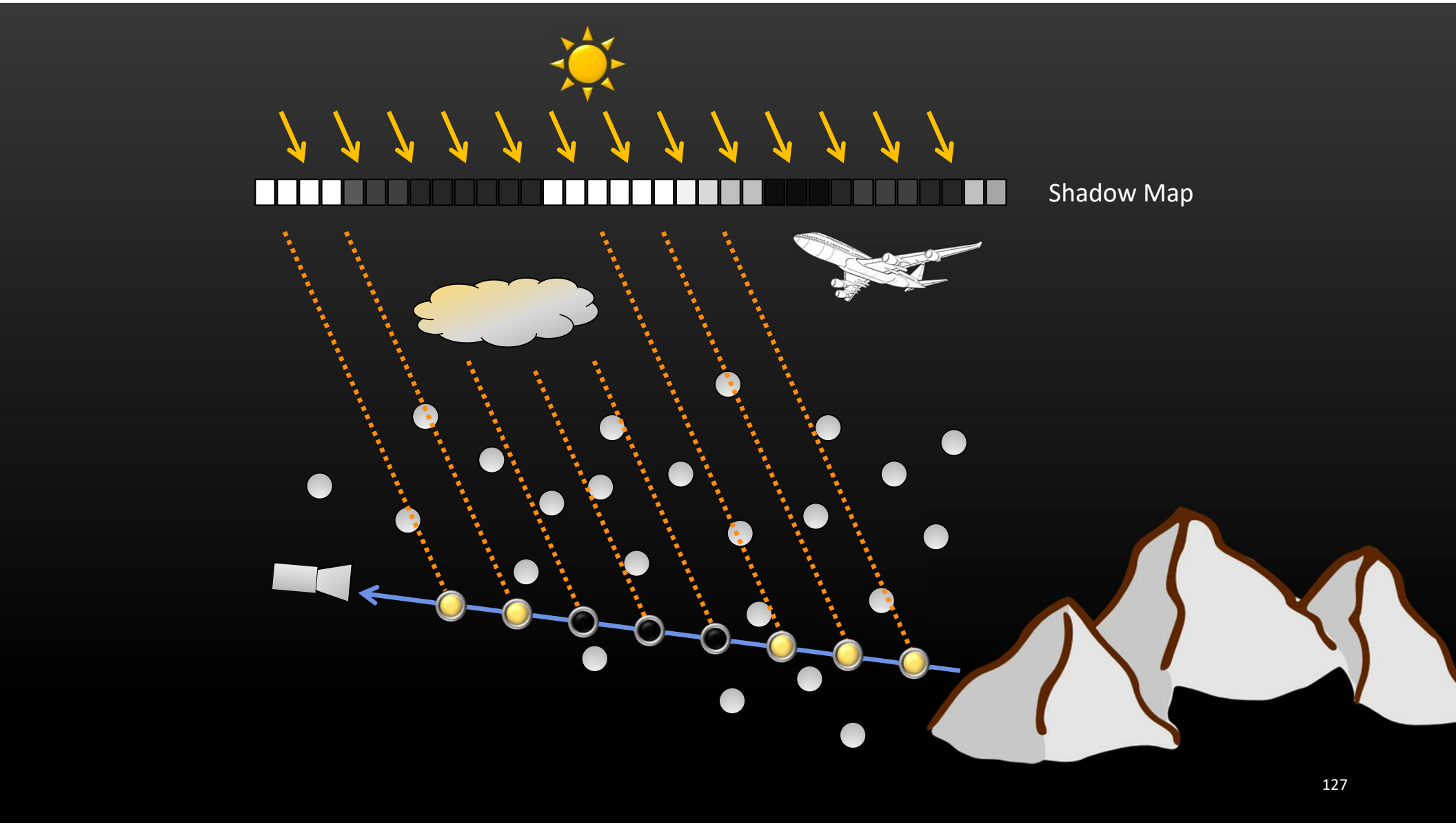

Shadow Map
near
far

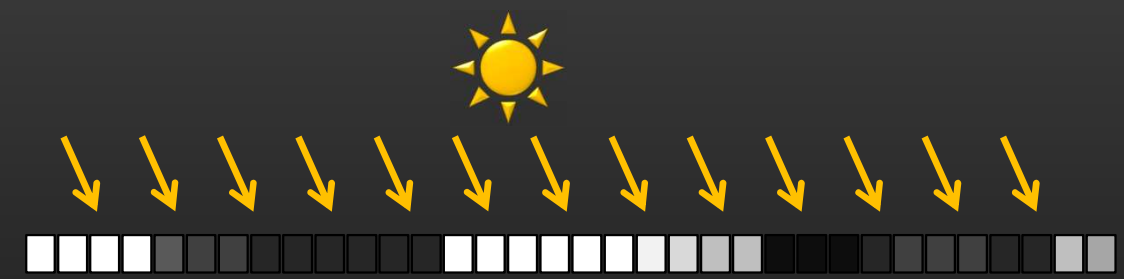

Assumptions:
• Single scattering



Assumptions:

- Single scattering
- Homogeneous medium


Shadow Map



How to do this efficiently?

Naïve: O(w*h * d)

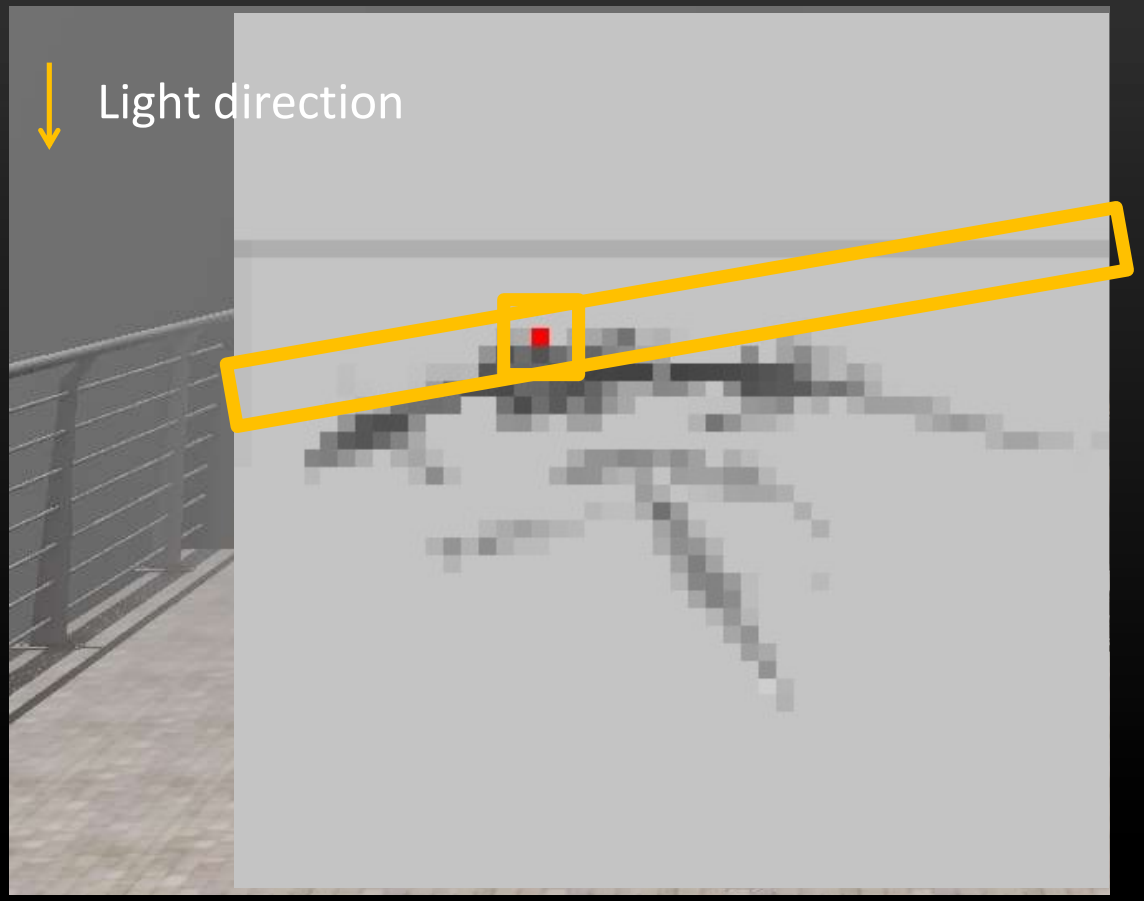
w*h pixels, d integration steps

Link to Percentage Closer Filtering

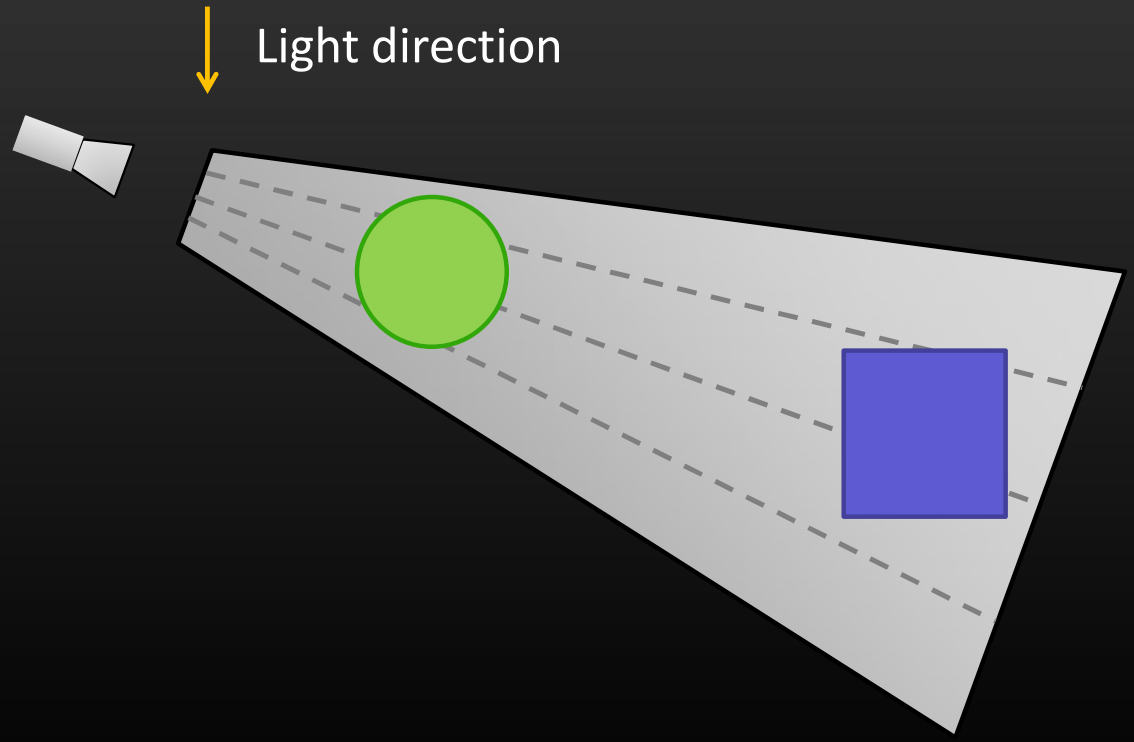
Percentage Closer Filtering

Visibility function *V*

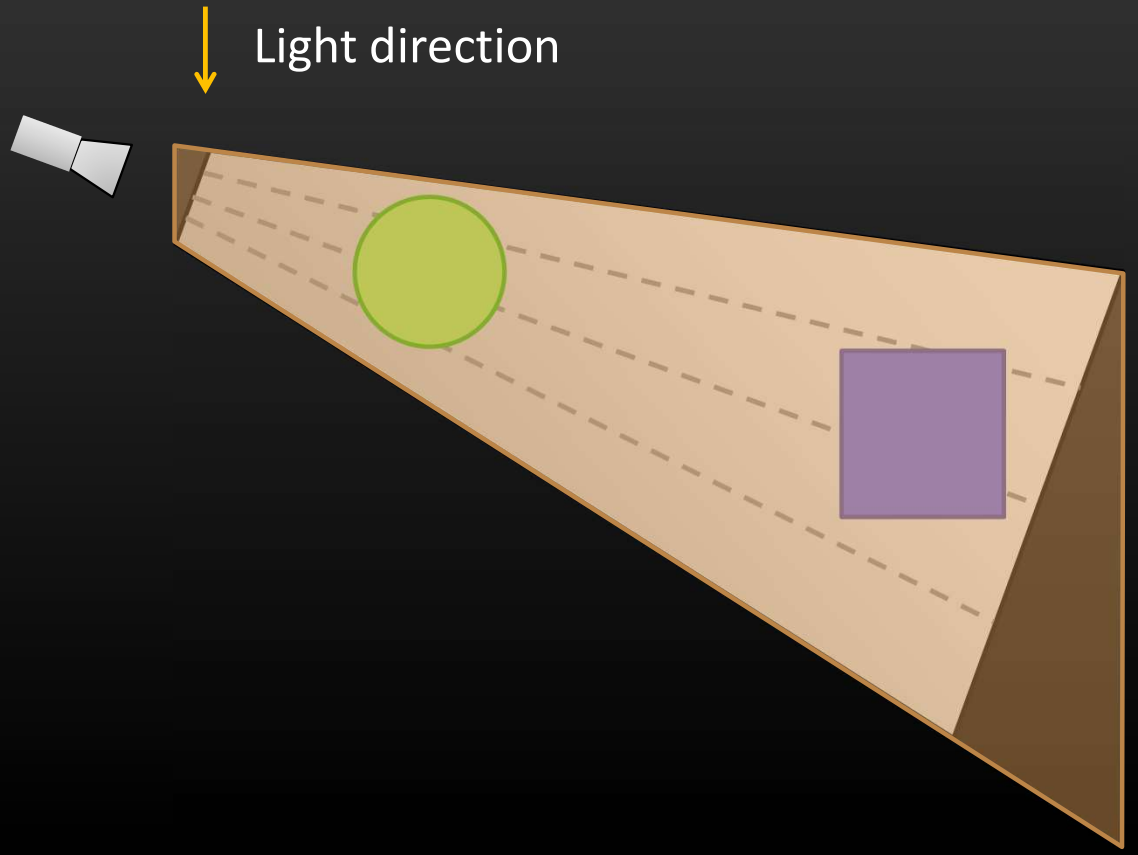
$$\sum_{s=1}^{N} V(d, z_s)$$



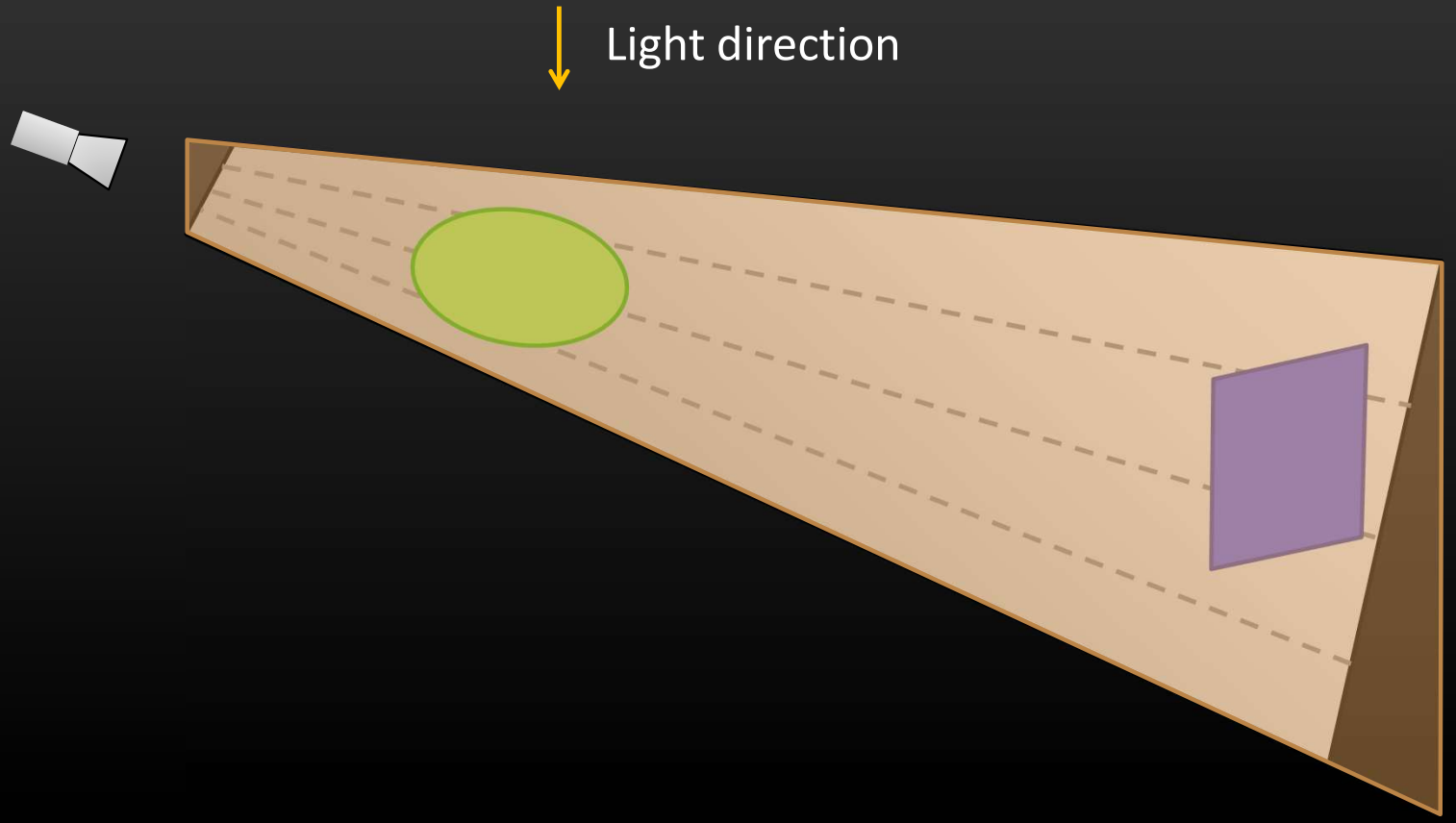
Rectified Shadow Map



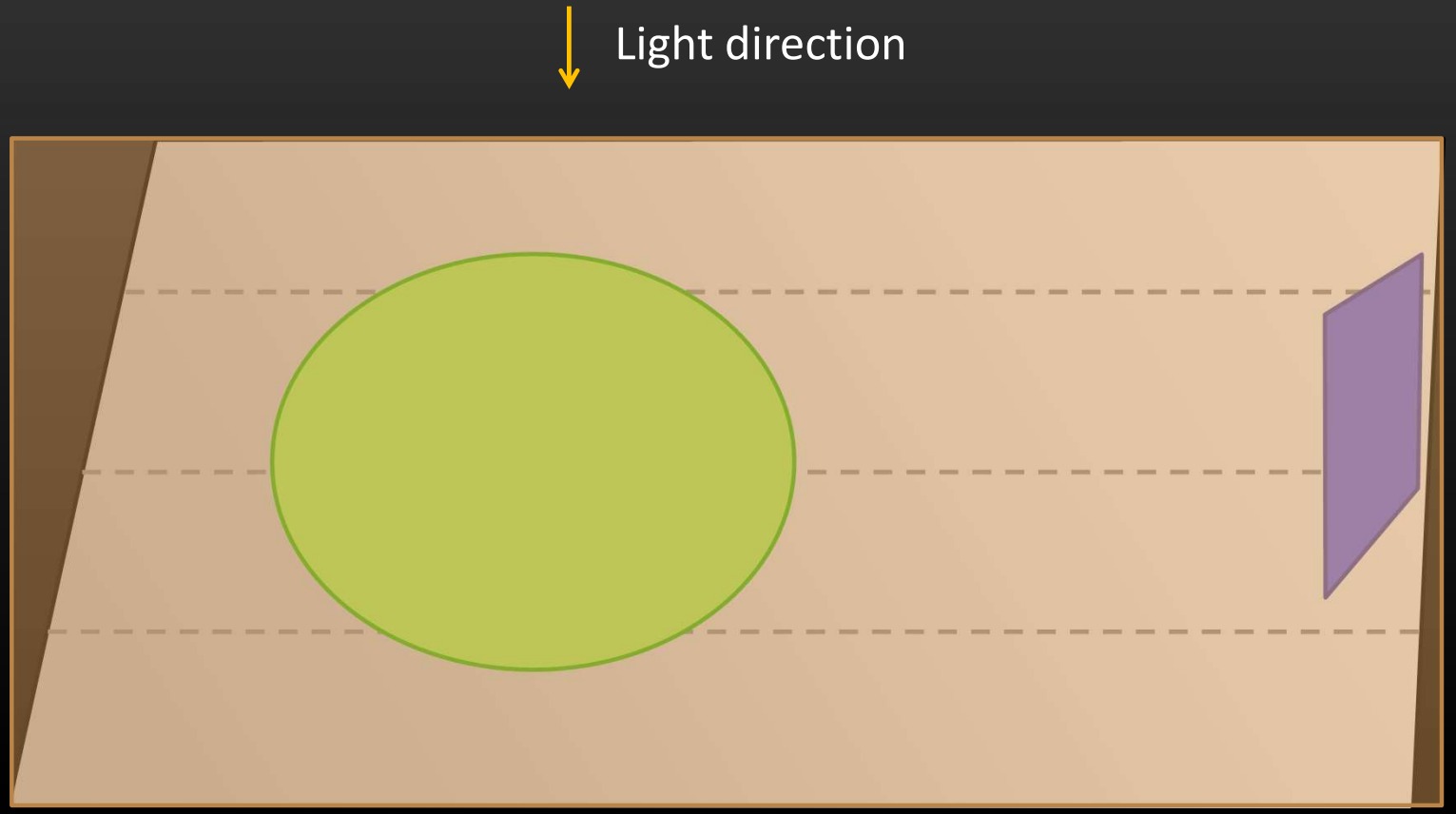
Rectified Shadow Map



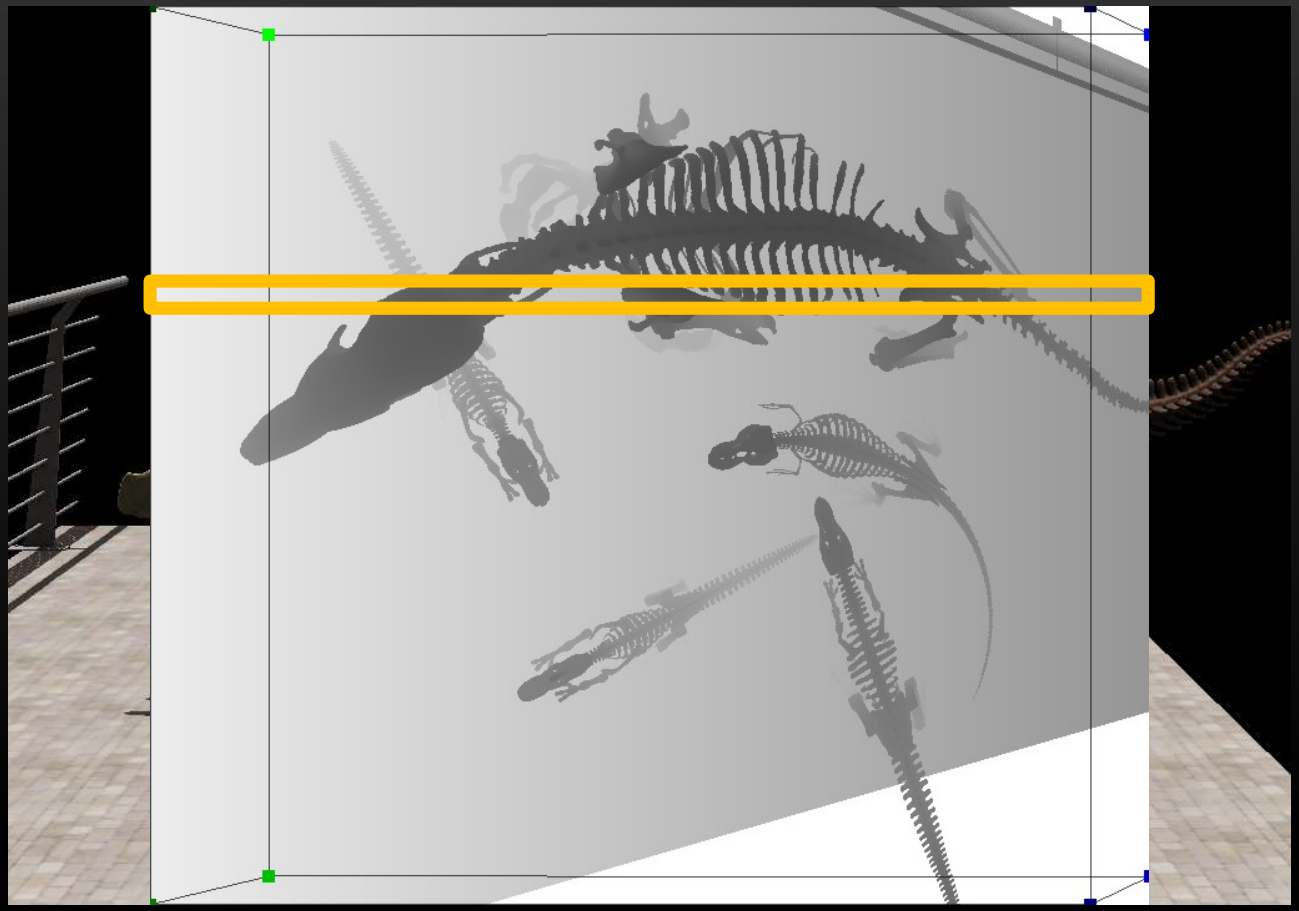
Rectified Shadow Map



Rectified Shadow Map

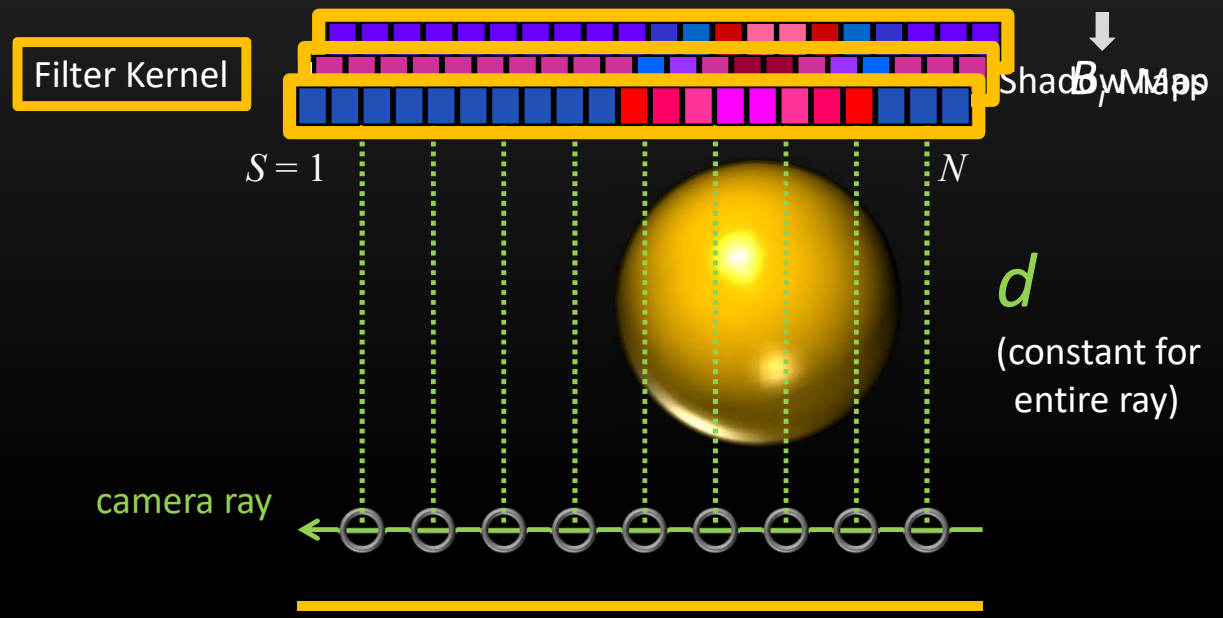


Rectified Shadow Map



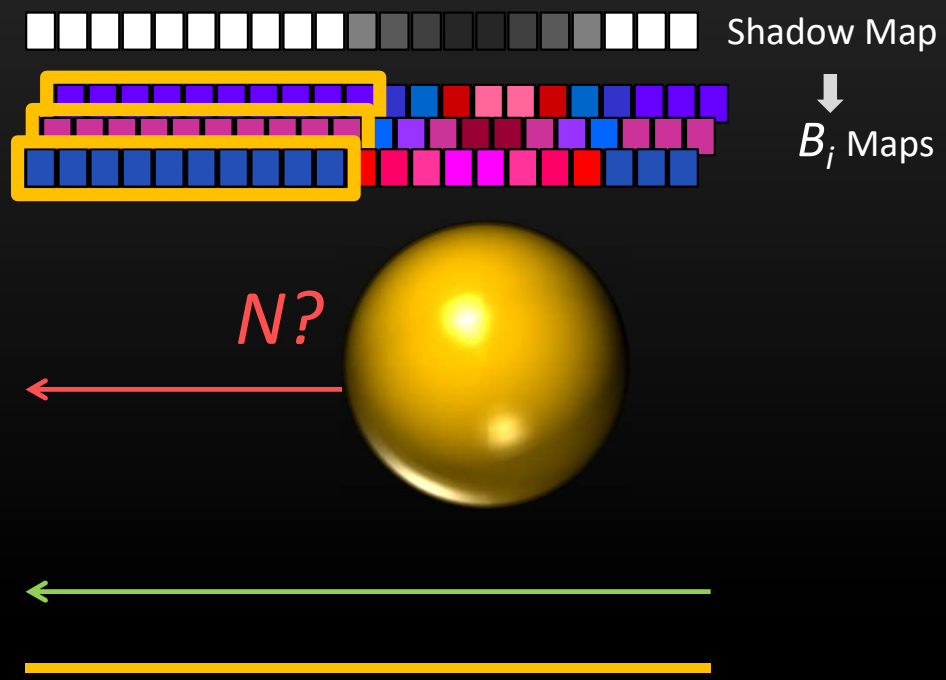
Filtering for camera rays

$$\sum_{s=1}^{N} V(d,z_s) = \sum_{i=1}^{\infty} a_i(d) \sum_{s=1}^{N} B_i(z_s)$$

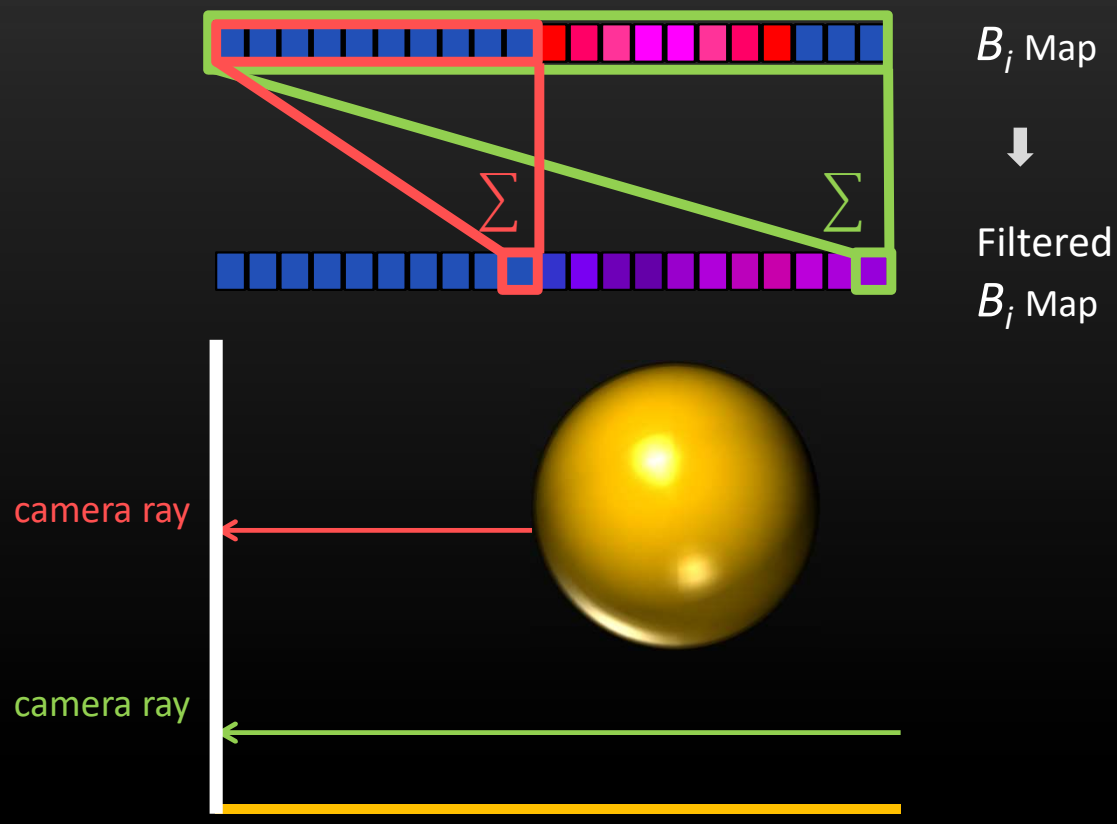


Filtering for camera rays

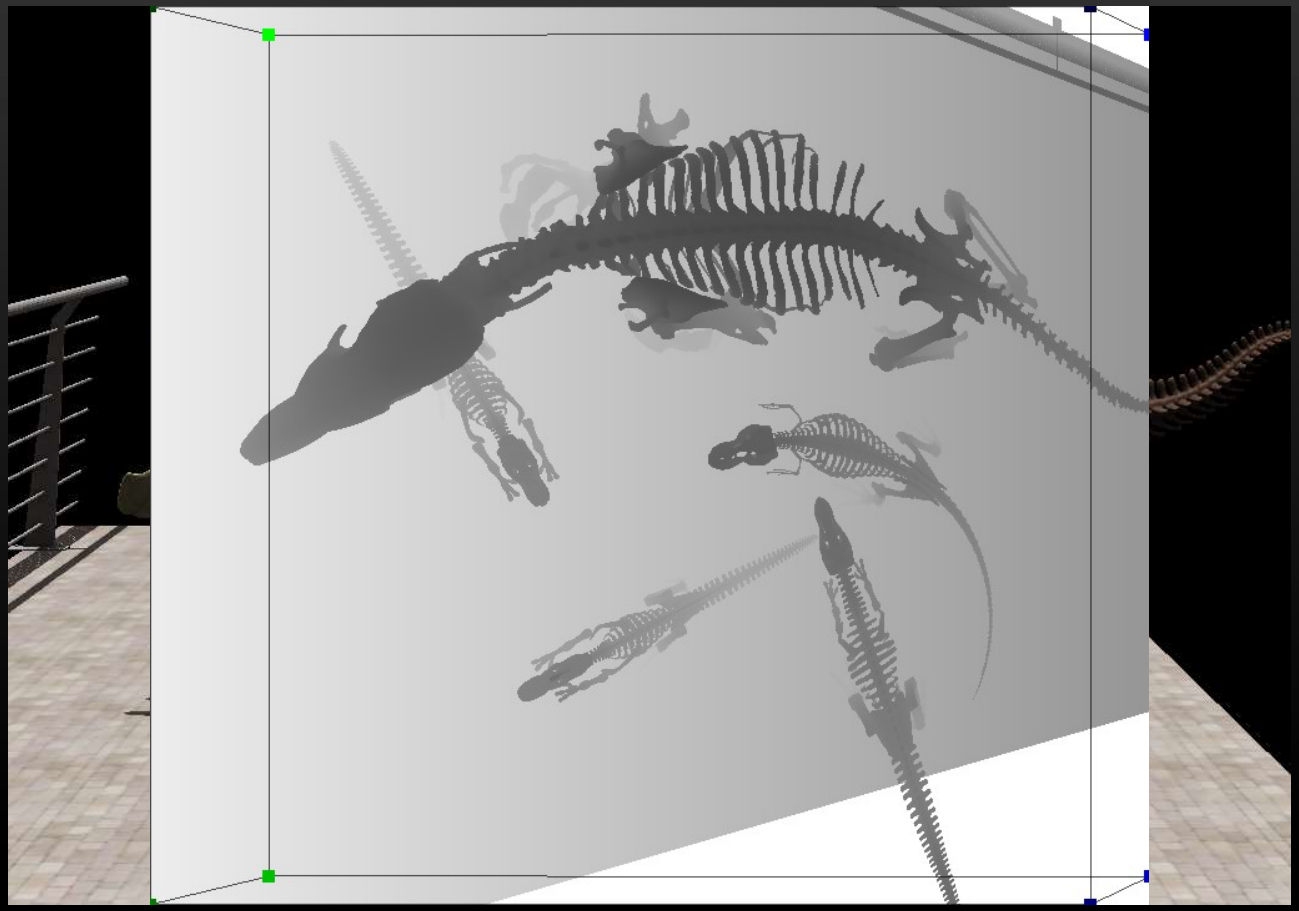
$$\sum_{s=1}^{N} V(d, z_s) = \sum_{i=1}^{\infty} a_i(d) \sum_{s=1}^{N} B_i(z_s)$$



Like Summed Area Tables but for a single row!



Rectified Shadow Map



Wrap up

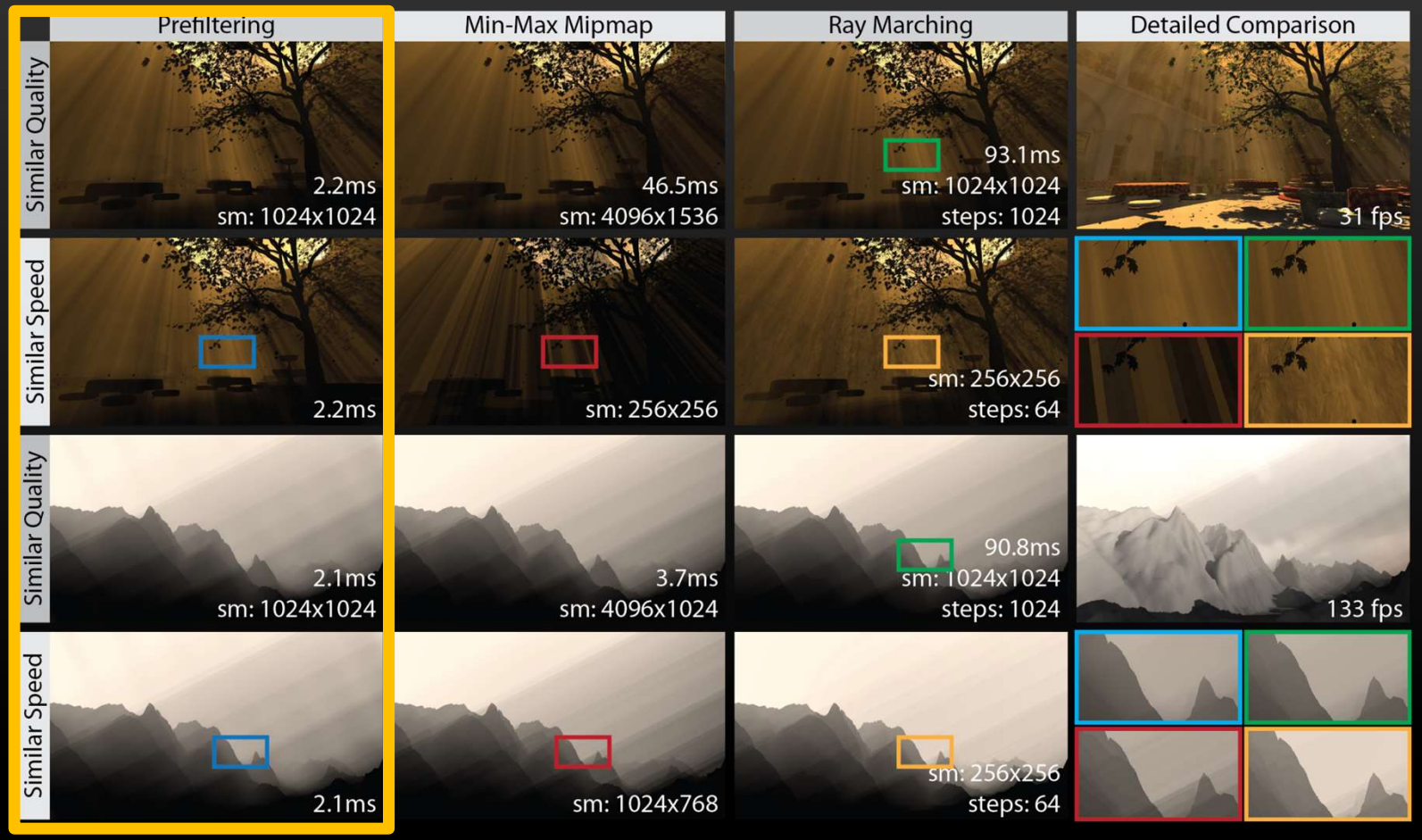
Prefiltered Single Scattering

Oliver Klehm[1] Hans-Peter Seidel[1] Elmar Eisemann[2]

[1]MPI Informatik [2]Delft University of Technology

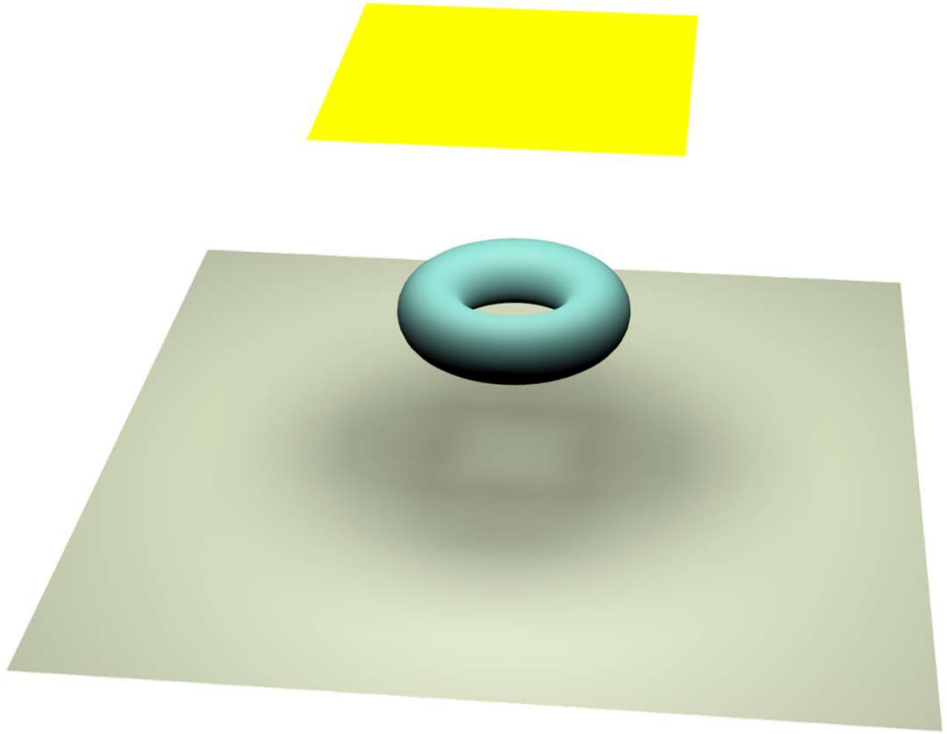
ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games

Comparison


Prefiltering
Min-Max Mipmap
Ray Marching
Detailed Comparison
Similar Quality
Similar Speed
Similar Quality
Similar Speed
2.2ms
sm: 1024x1024
46.5ms
sm: 4096x1536
93.1ms
sm: 1024x1024
steps: 1024
31 fps
2.2ms
sm: 256x256
sm: 256x256
steps: 64
2.1ms
sm: 1024x1024
3.7ms
sm: 4096x1024
90.8ms
sm: 1024x1024
steps: 1024
133 fps
2.1ms
sm: 1024x768
sm: 256x256
steps: 64

Outlook

- We are not here yet:



... but we will get there...

Thank you very much for your attention!