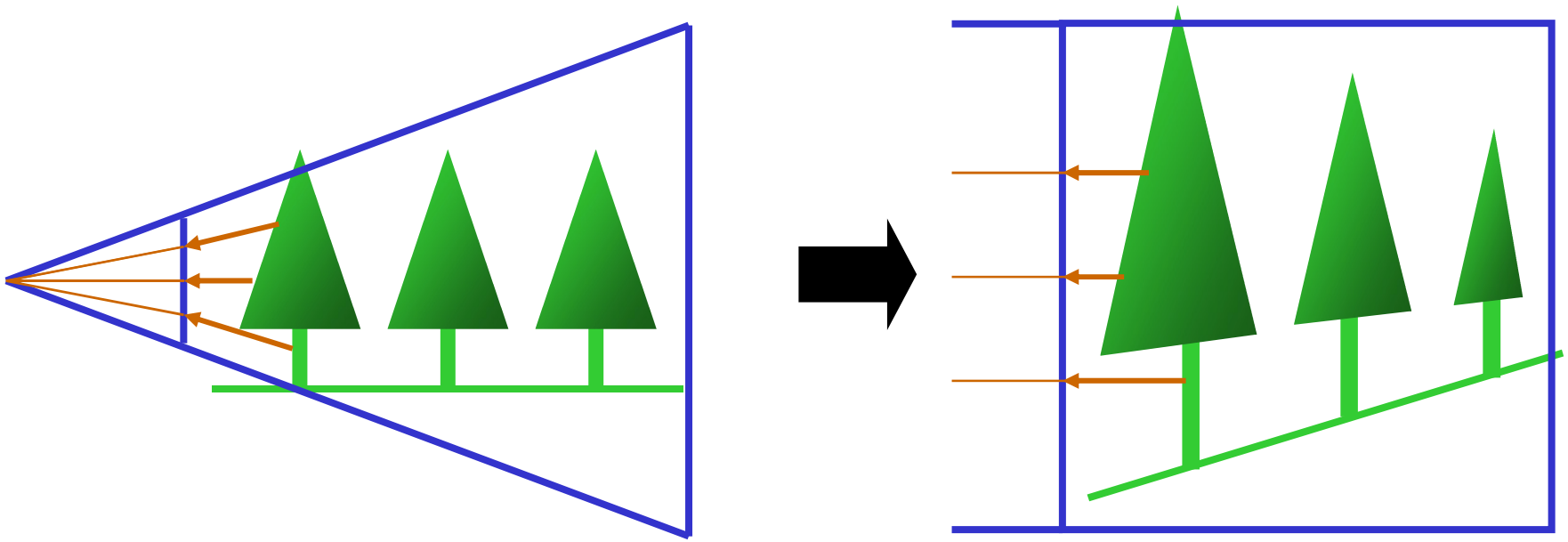
Perspective Shadow Maps

[Stamminger & Drettakis 2002]

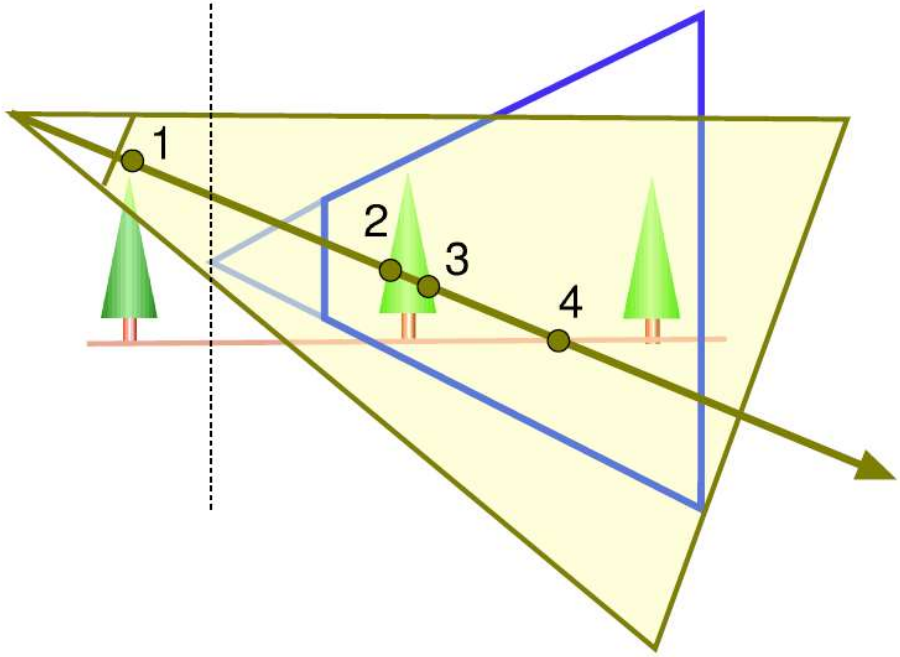
- Want to account for perspective warping of eye
- Shadow map in **post-perspective** eye space
- Reduce perspective aliasing



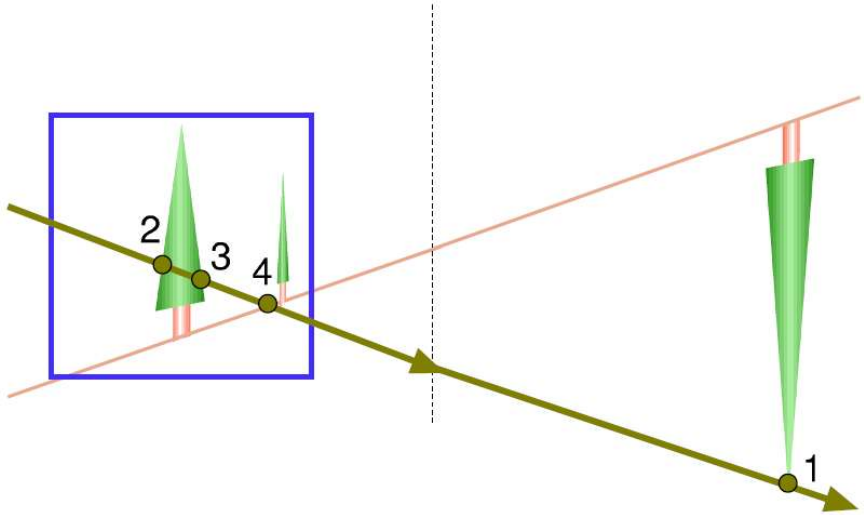
world space **post-perspective space**

PSM Problems

- Shadows from behind



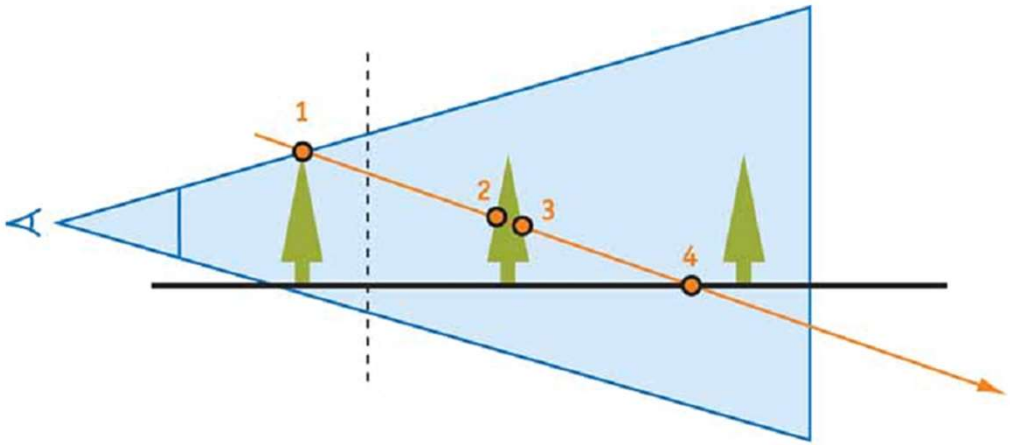
world space



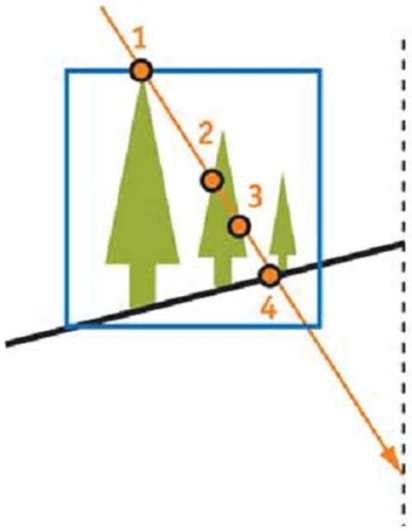
post-perspective space

PSM Problems

- Shadows from behind
  - Require move back due to singularity
  - Changes resulting quality of shadow

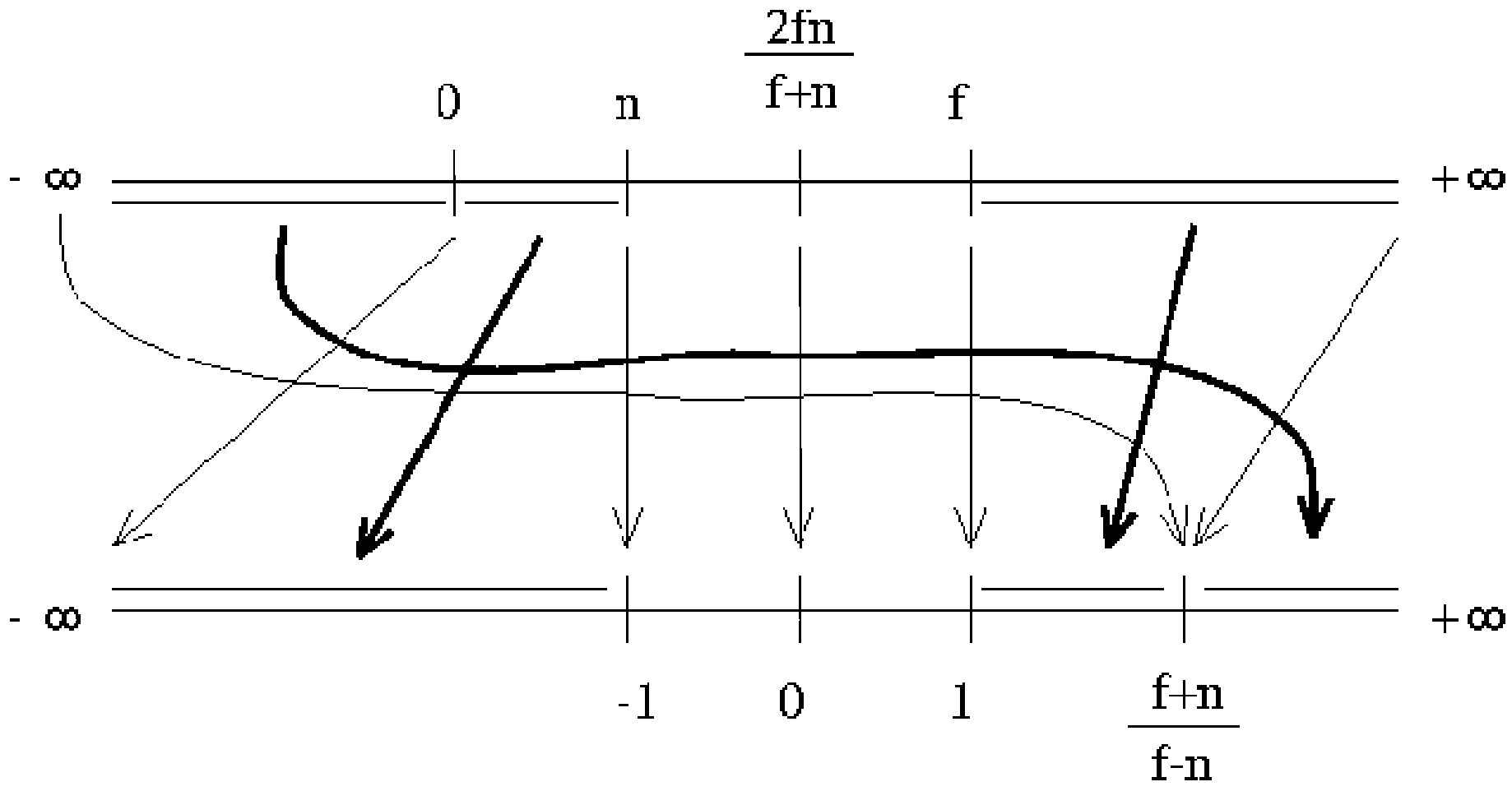


world space

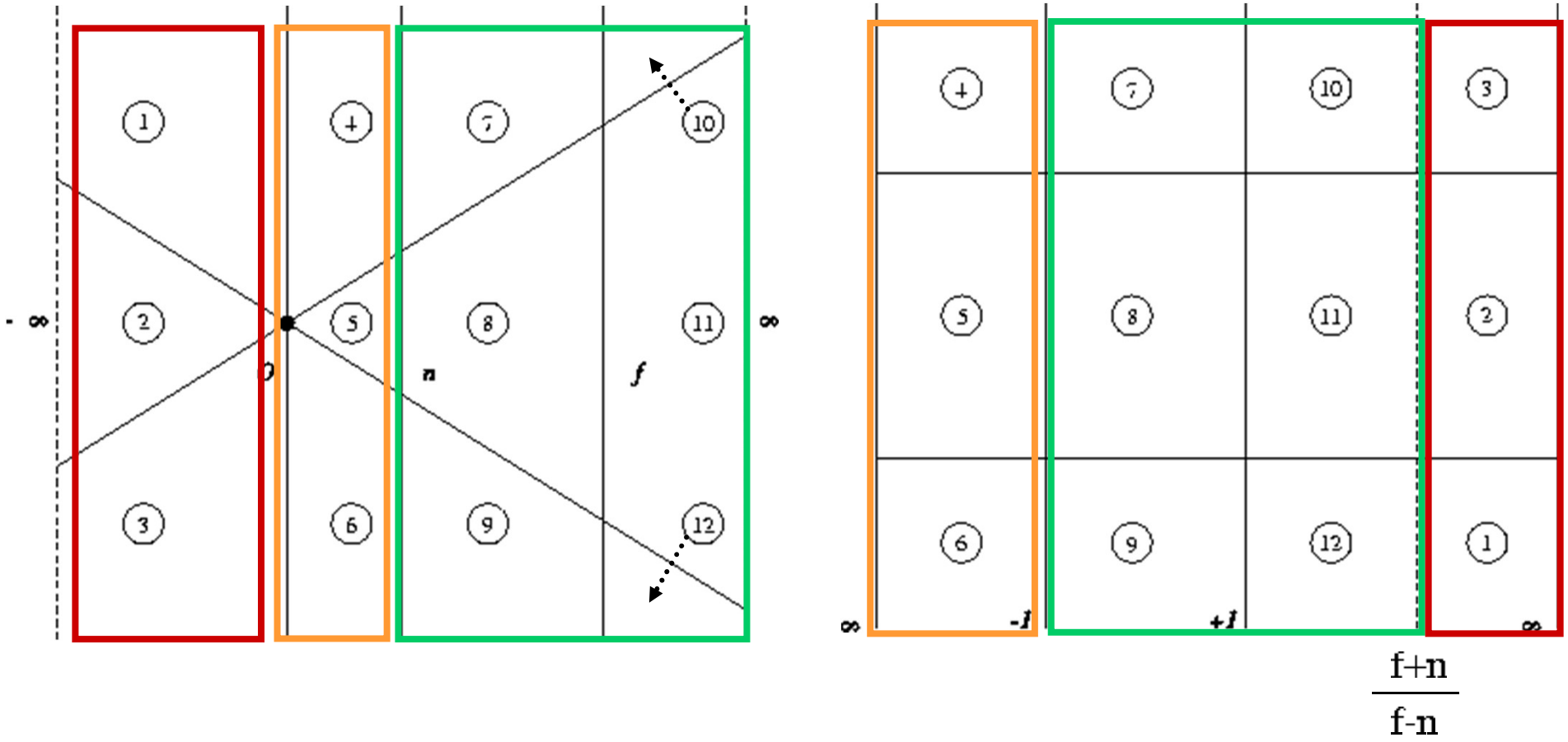


post-perspective space

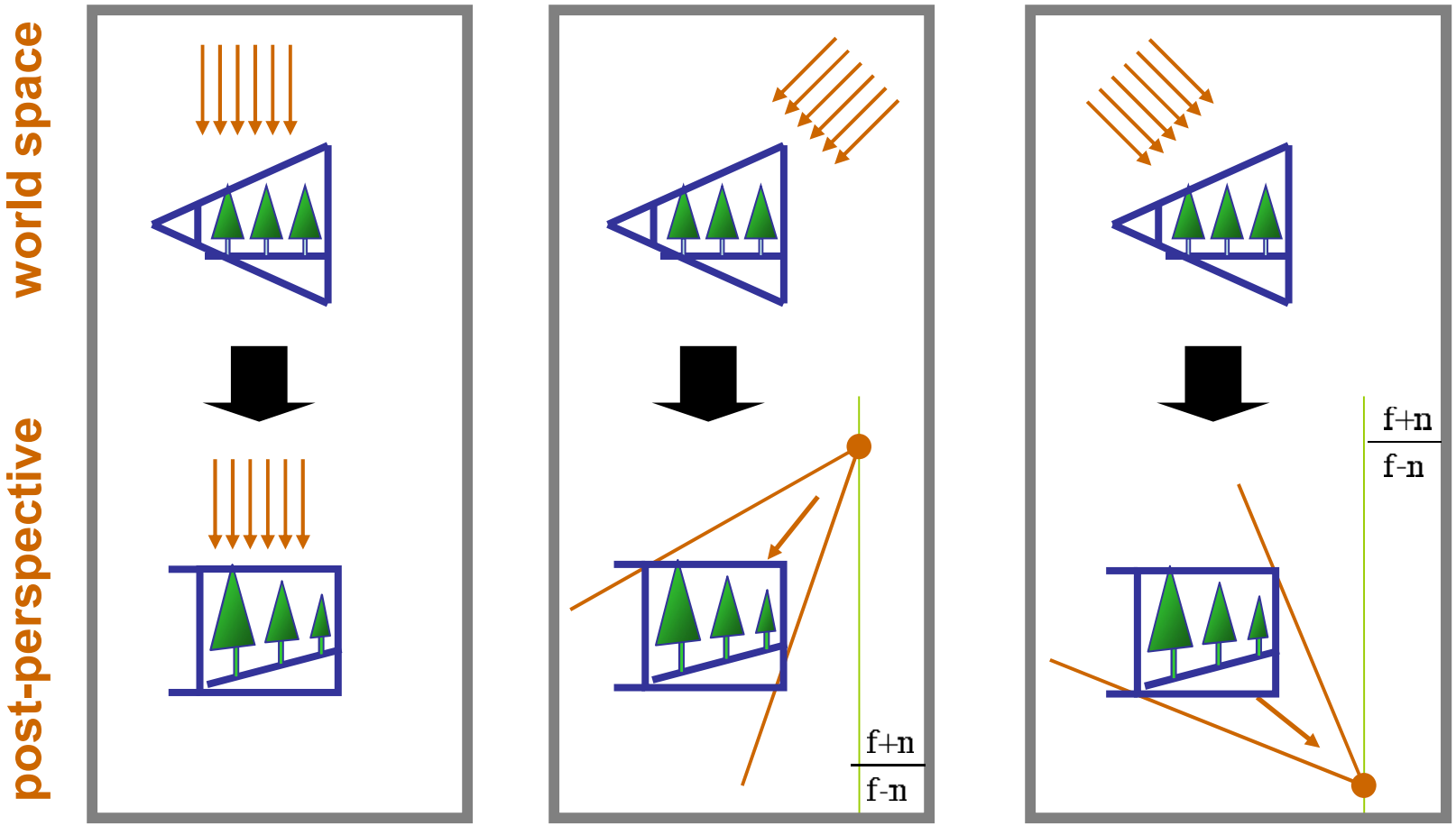
Post Perspective Space - Depth



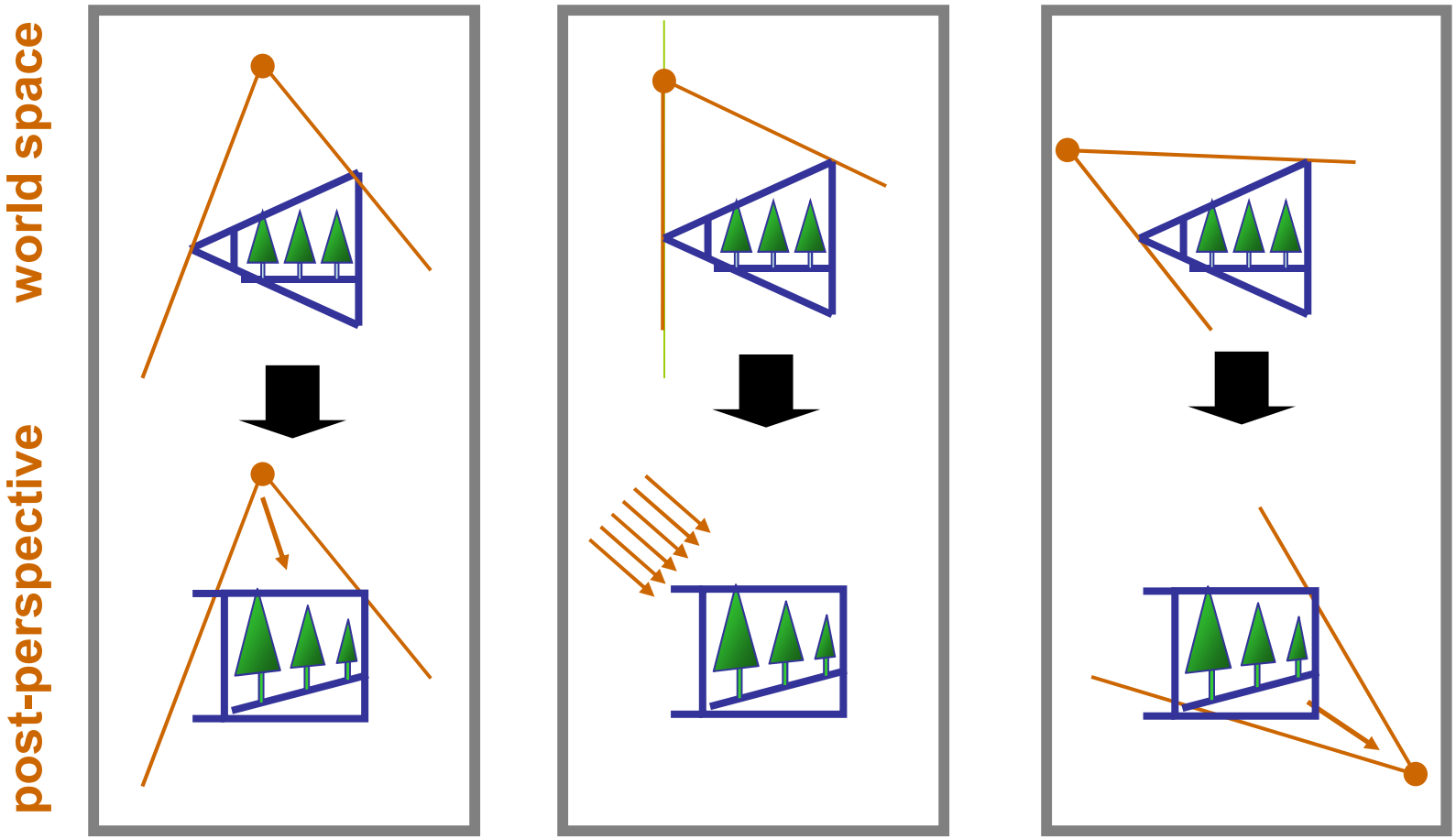
Post Perspective Space – x or y



Parallel Light Transformation



Point Light Transformation

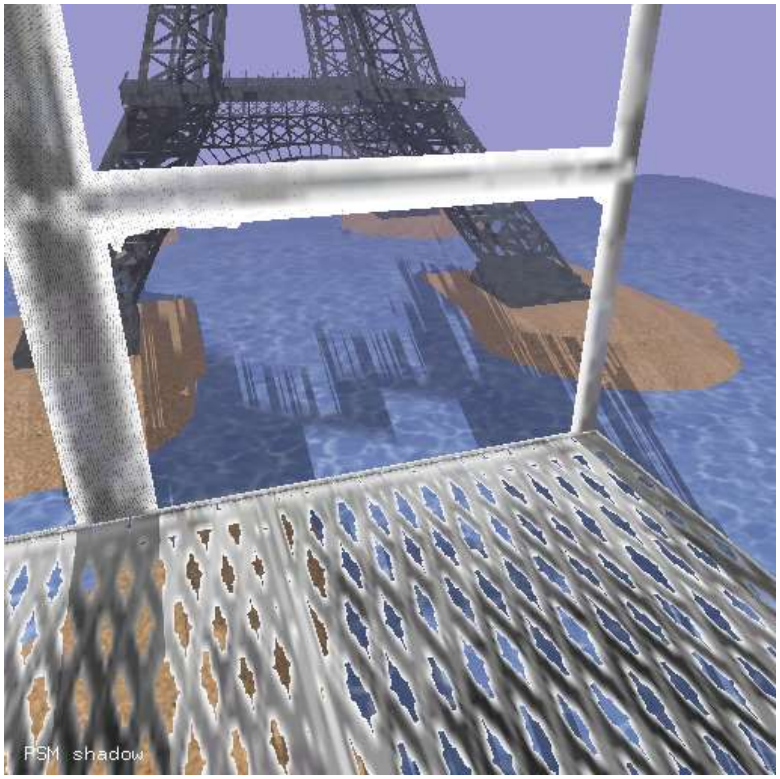


PSM Problems

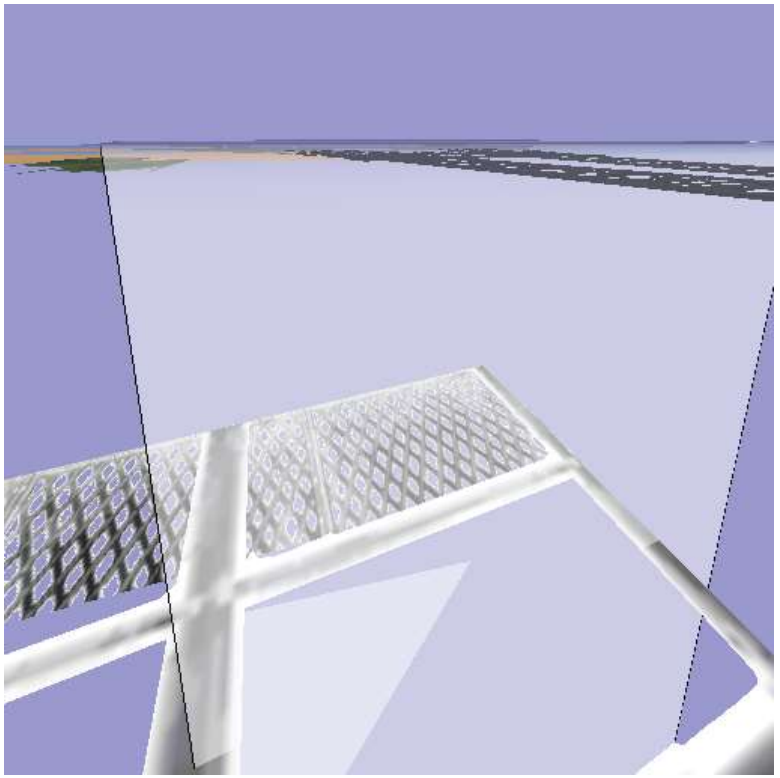
- Shadows from behind
  - Require move back due to singularity
  - Changes resulting quality of shadow
- Lights change their type (point/directional)
  - Many special cases
  - Non-intuitive post-perspective space
- View-dependent shadow quality

PSM Problems

- Most severe: uneven z-distribution
  - Good near viewer, very bad far away!
  - Can be reduced by pushing near plane away



world space



post-perspective space

Observation

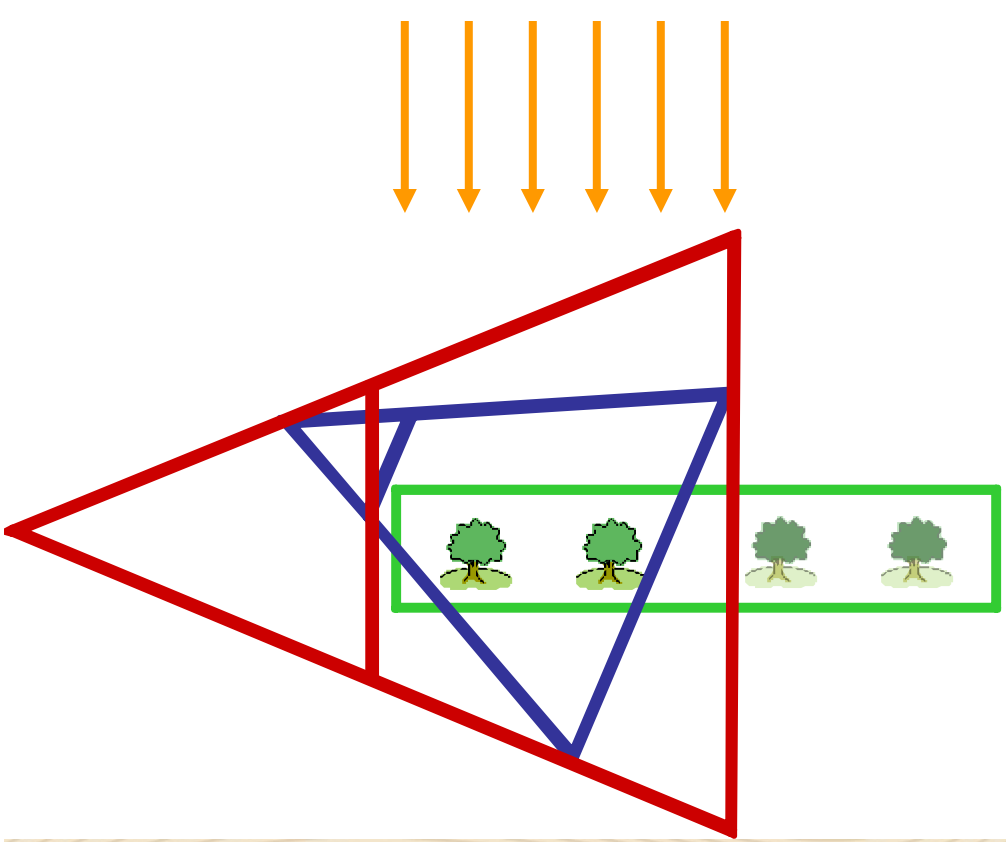
- Want to redistribute samples in shadow map
- Perspective transform is a good option
  - Supported by hardware

- BUT: why choose perspective based on observer transform???

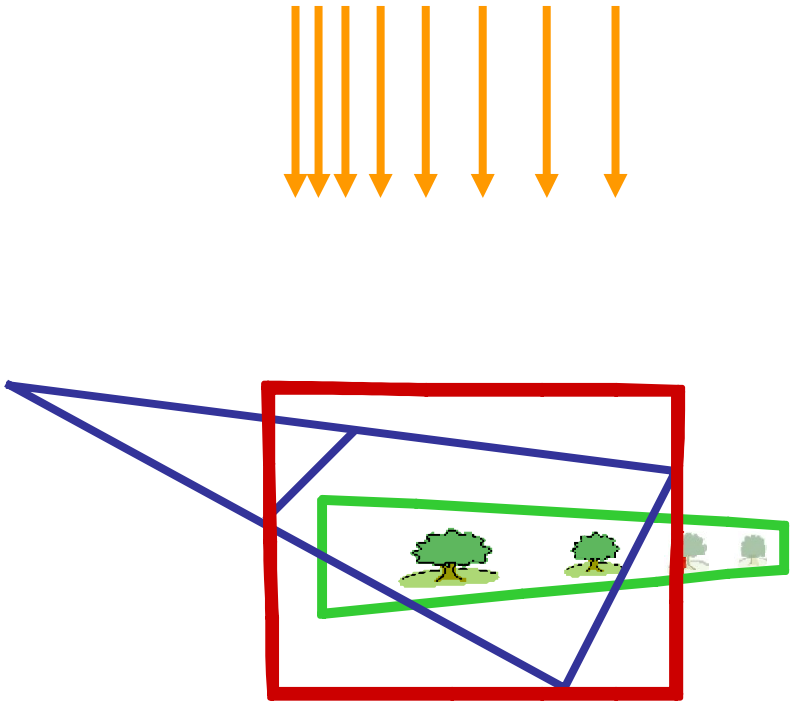
Light Space Perspective Shadow Maps

- Use an **additional perspective frustum** in **light space**

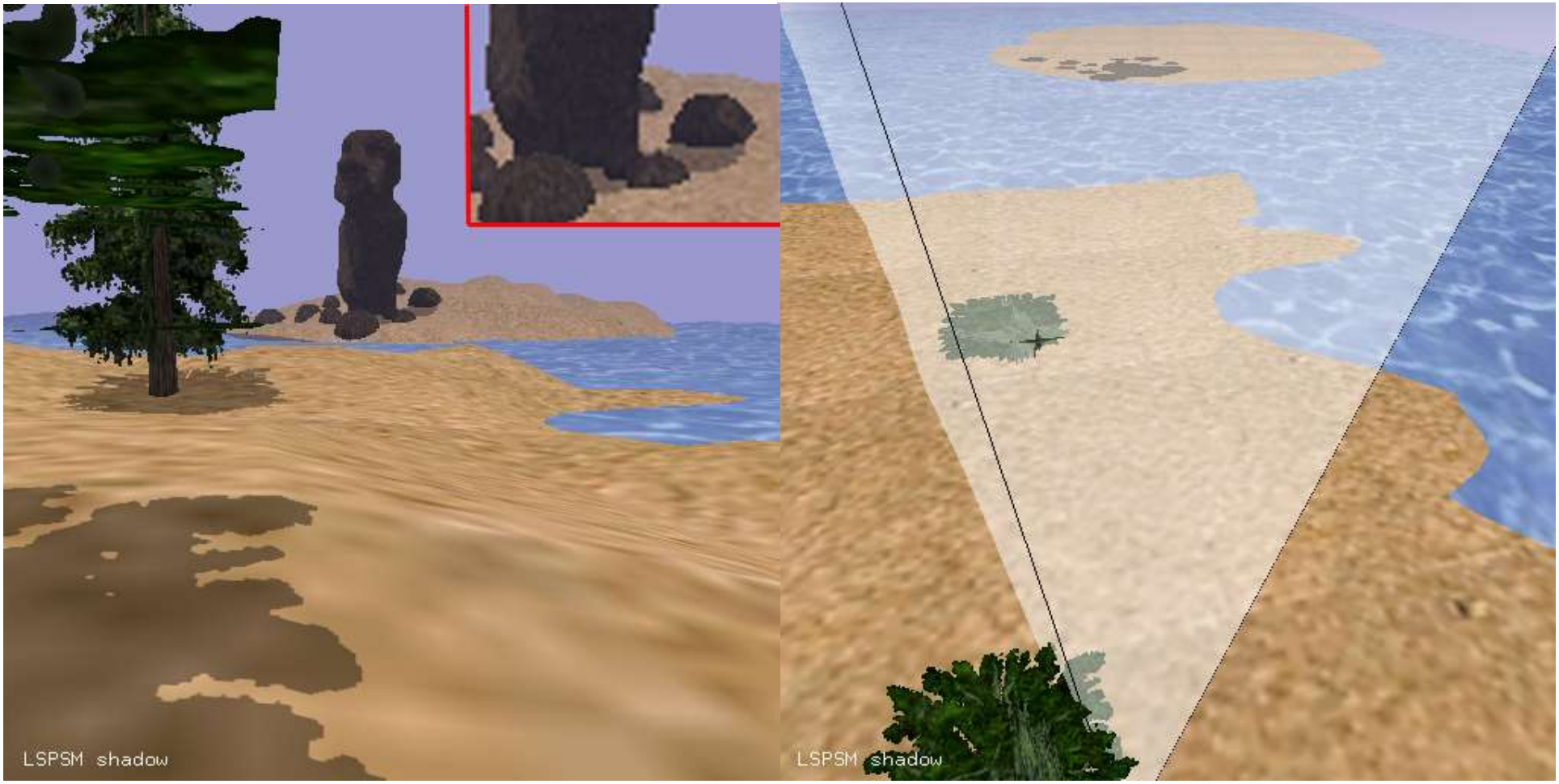
uniform shadow map



LiSPSM shadow map



Light Space



light space

Advantages of LiSPSM

- No singularities in relevant part of the scene
  - … by construction
- Directional lights remain directional lights
  - … and direction remains the same
- Points lights are converted to directional lights
  - … perspective point light transformation already done in light space
- Easy construction, general and robust
- Smooth quality changes

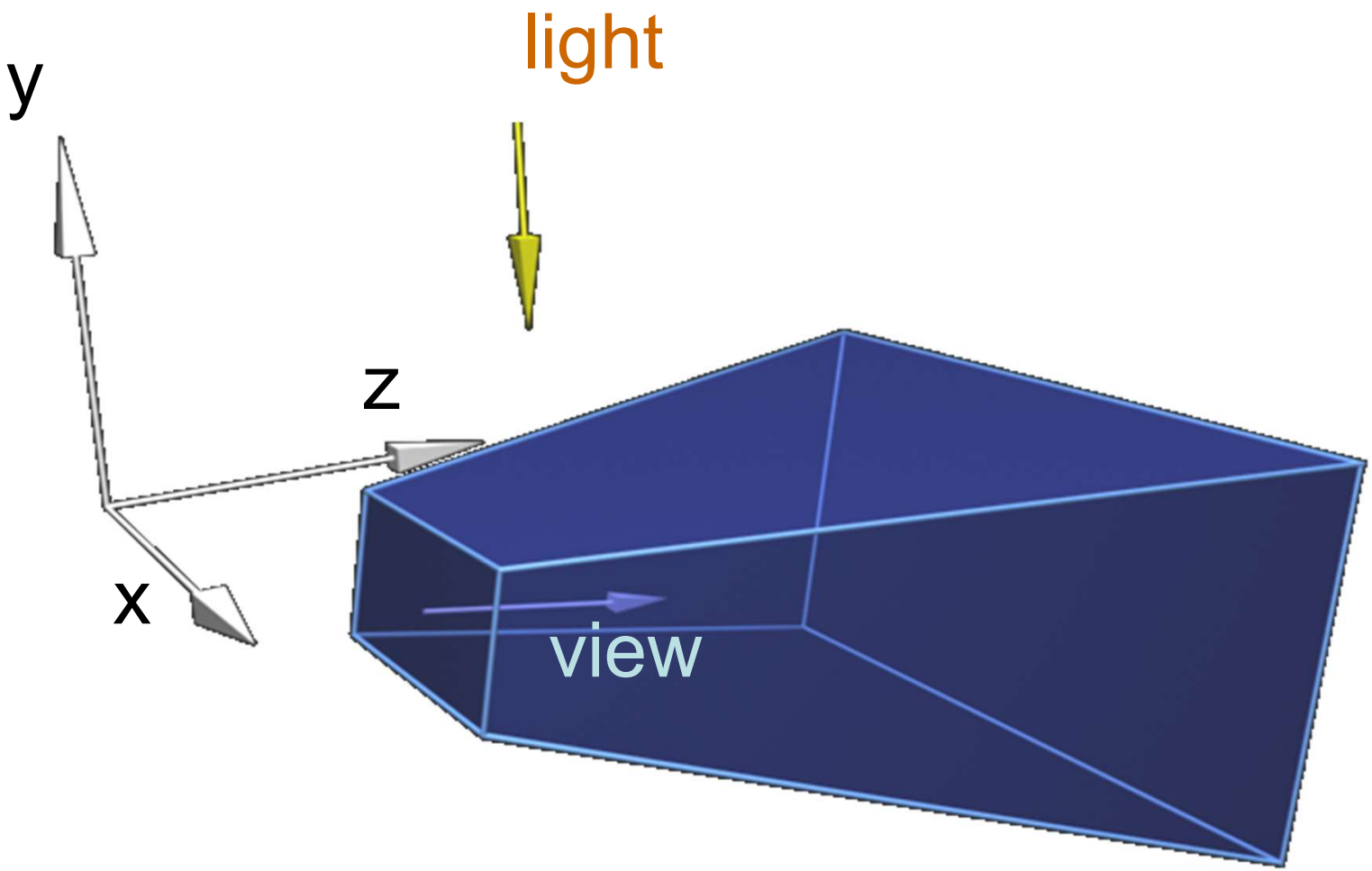
- Most important: **control over shadow quality**
  - … by choice of free parameter

LiSPSM Overview

- Focus shadow map
- Construct light space
- Create perspective transform
- Apply

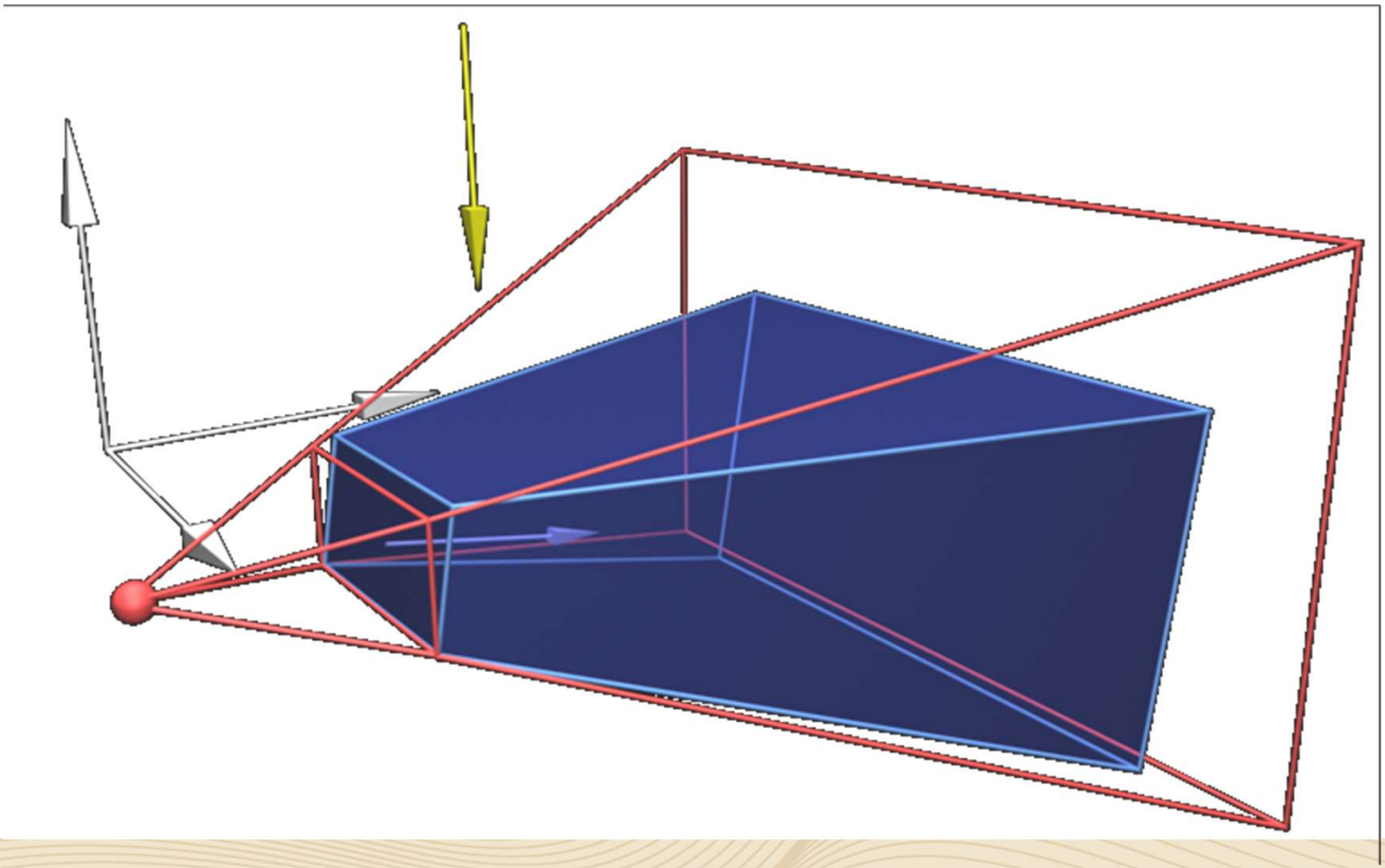
LiSPSM Fitting and Warp Construction

- Light and view vector define yz-plane

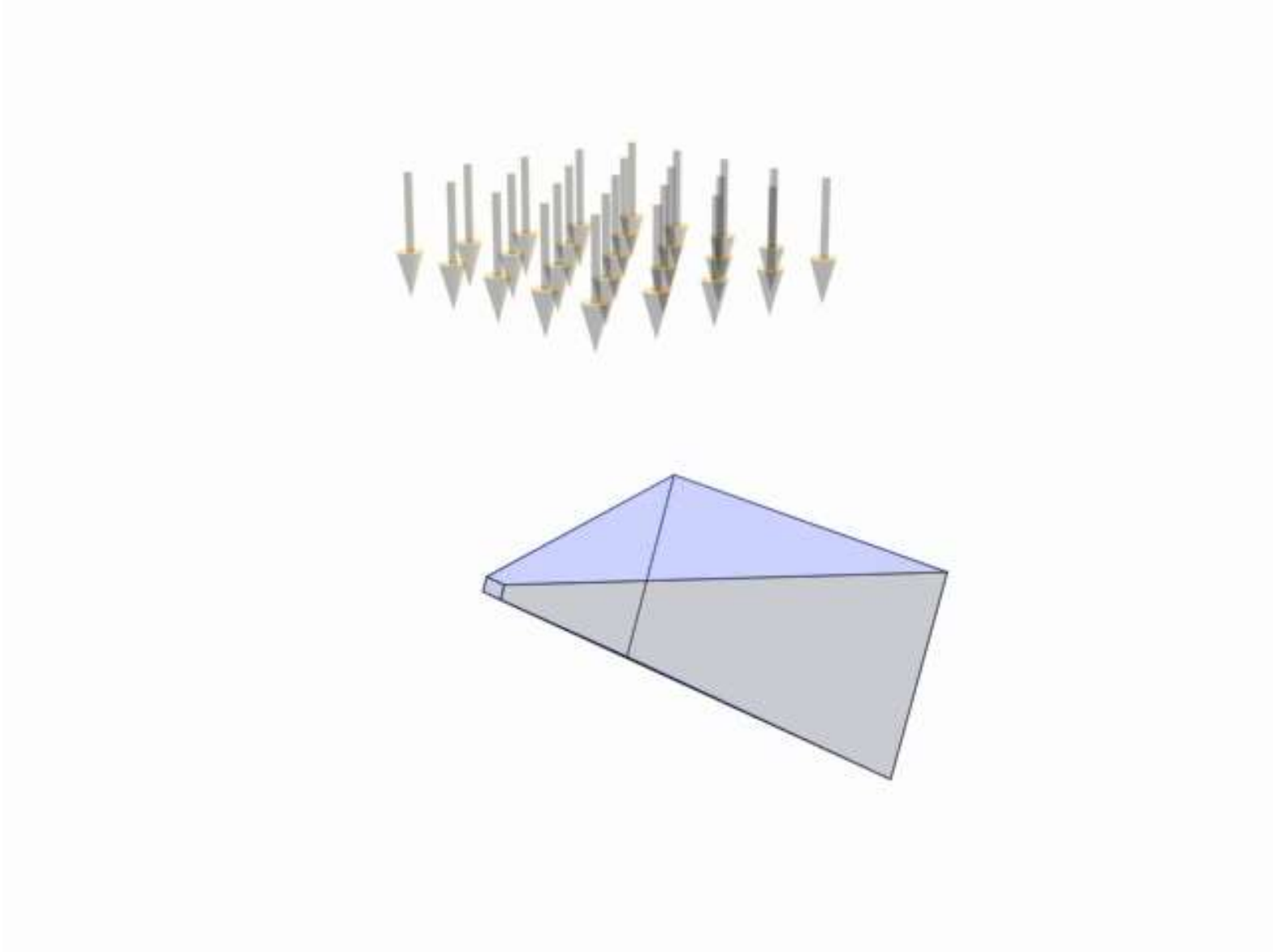


LiSPSM Fitting and Warp Construction

- Find a tight perspective frustum on focused region
  - In light space!

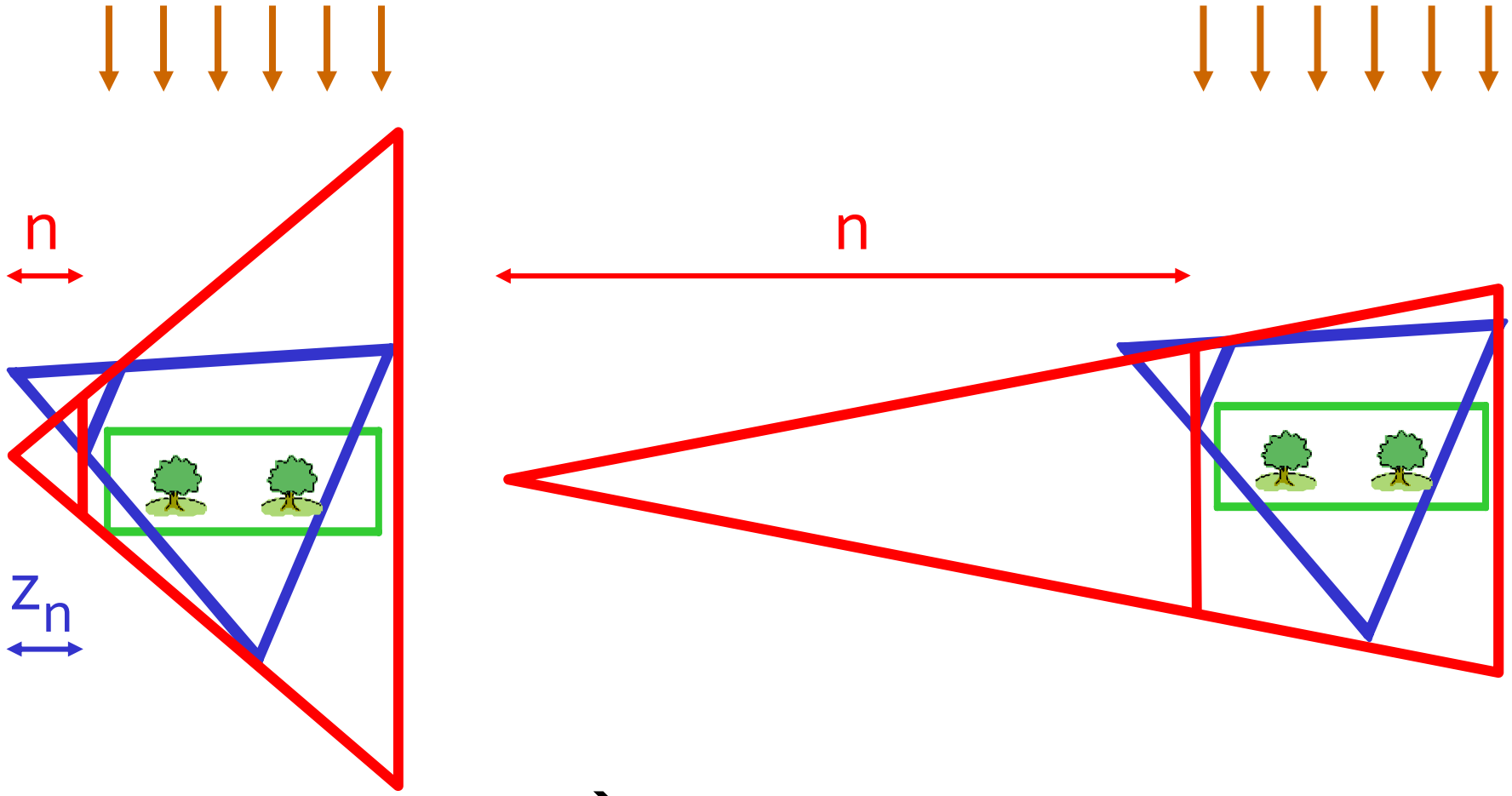


LiSPSM Construction



Free Parameter n

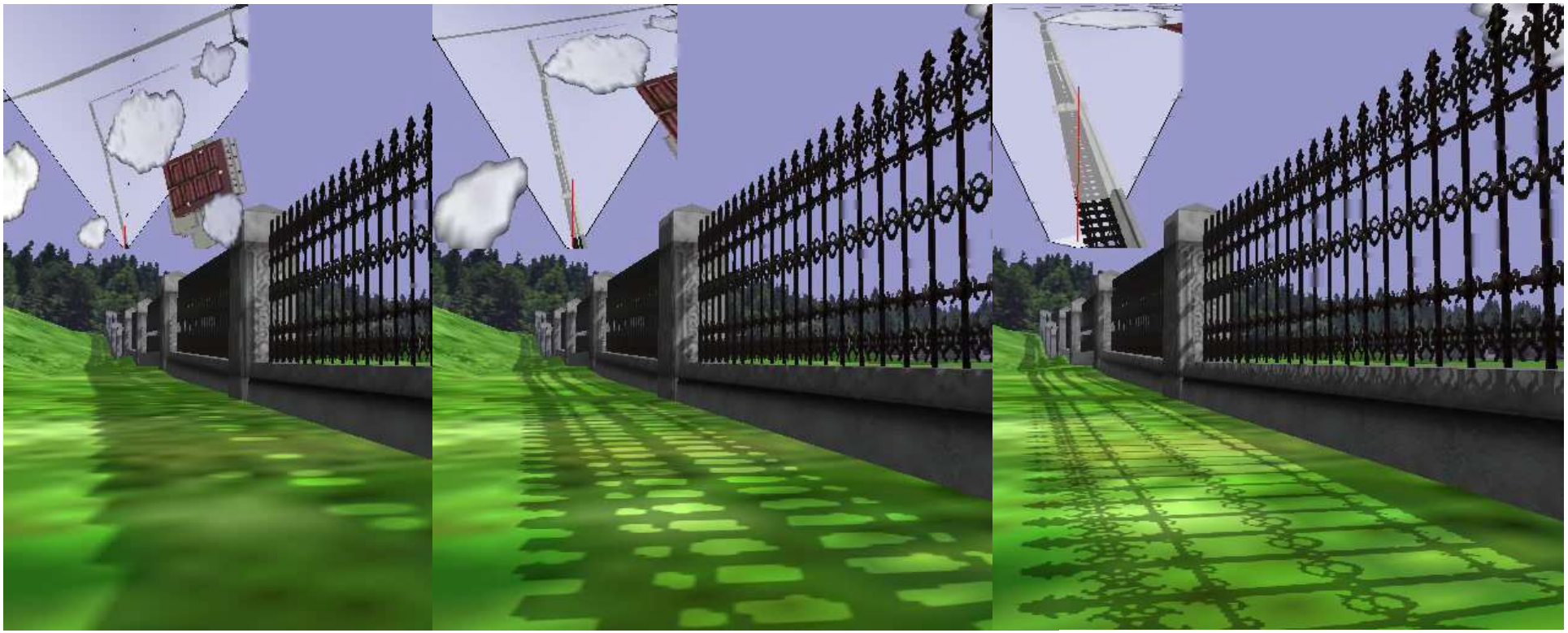
- Controls warping effect



n = z_n : PSM

n → infinity : uniform shadow maps

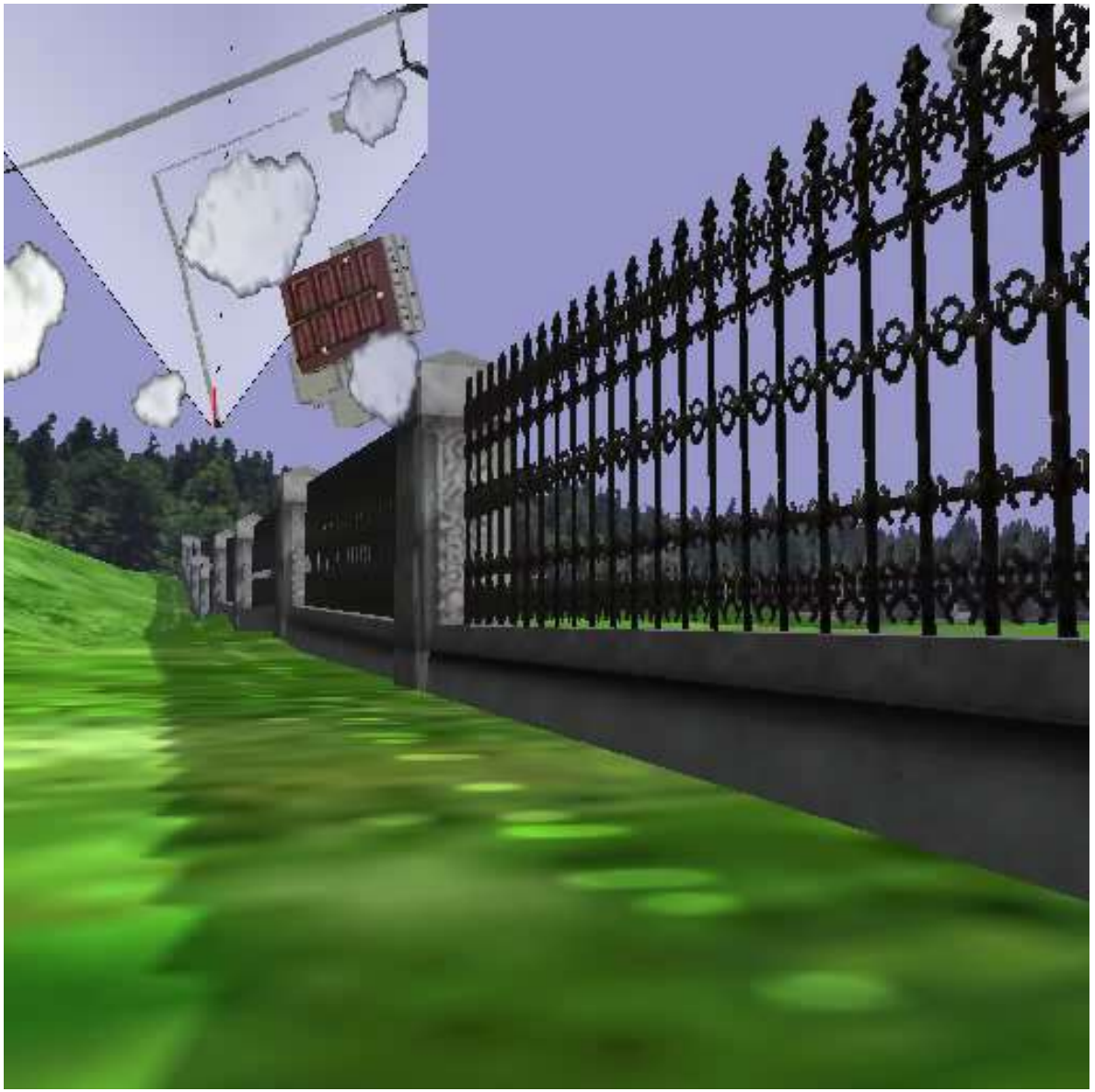
Free Parameter n



very big n

very small n

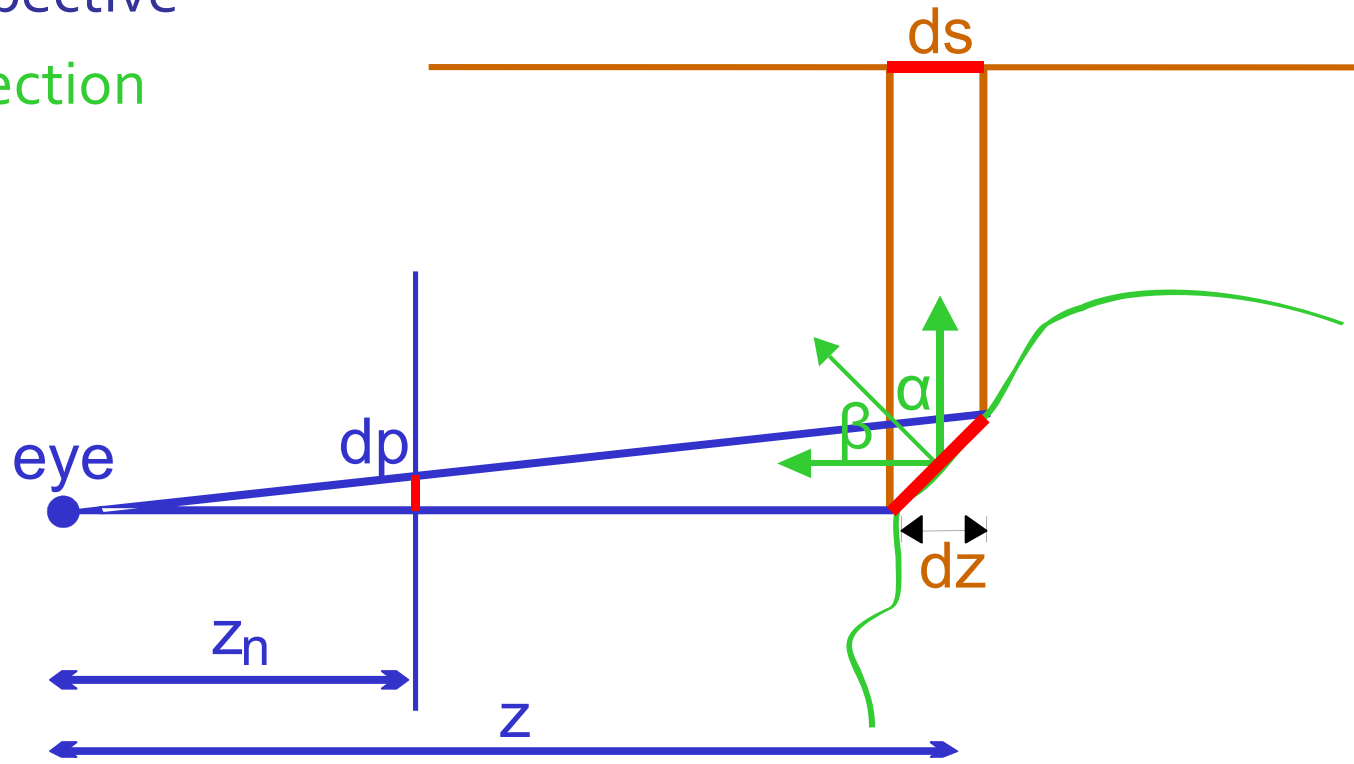
Free Parameter n



Sampling Error Analysis

- Aliasing error:
  - Parameterization
  - Perspective
  - Projection

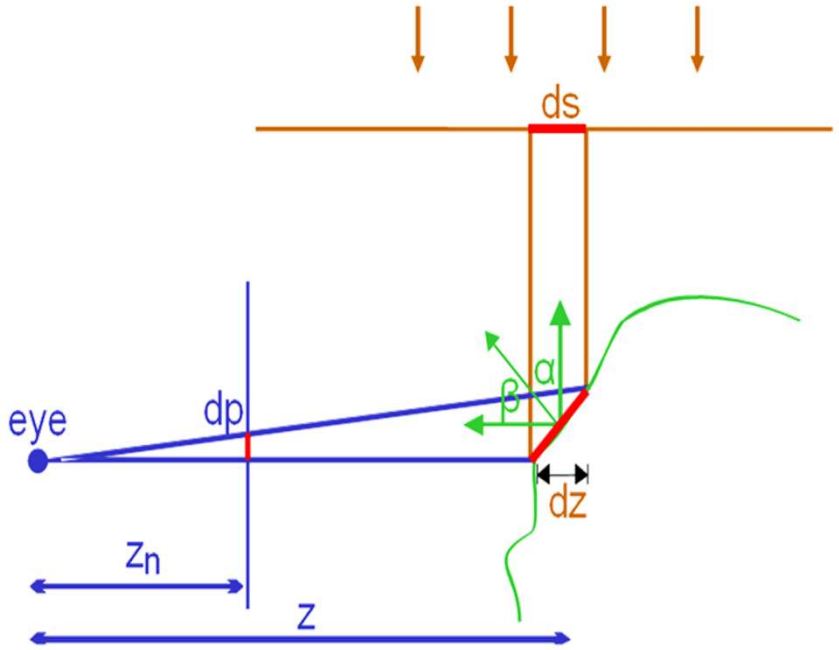
$$\frac{dp}{ds} = \frac{z_n}{z}\frac{dz}{ds}\frac{\cos\alpha}{\cos\beta}$$



How to Choose the Free Parameter n?

- Recall error analysis
- $\frac{dp}{ds} > 1$ → shadow map undersampling
- **Projection aliasing** cannot be changed
- Counter **perspective aliasing** with new shadow map **parameterization** s
- Goal: $\frac{dp}{ds} \sim 1$

$$\frac{dp}{ds} = \frac{z_n}{z} \frac{dz}{ds} \frac{\cos\alpha}{\cos\beta}$$



Error Analysis

$$\frac{dp}{ds} = \frac{1}{z}\frac{dz}{ds}$$

- Uniform shadow maps

$$s \sim z \quad \overset{s'}{\Rightarrow} \quad \frac{ds}{dz} \sim 1 \quad \Rightarrow \quad \frac{dp}{ds} \sim \frac{1}{z}$$

- Perspective shadow maps

$$s \sim \frac{1}{z} \quad \overset{s'}{\Rightarrow} \quad \frac{ds}{dz} \sim \frac{1}{z^2} \quad \Rightarrow \quad \frac{dp}{ds} \sim z$$

  - Linear increase in error!

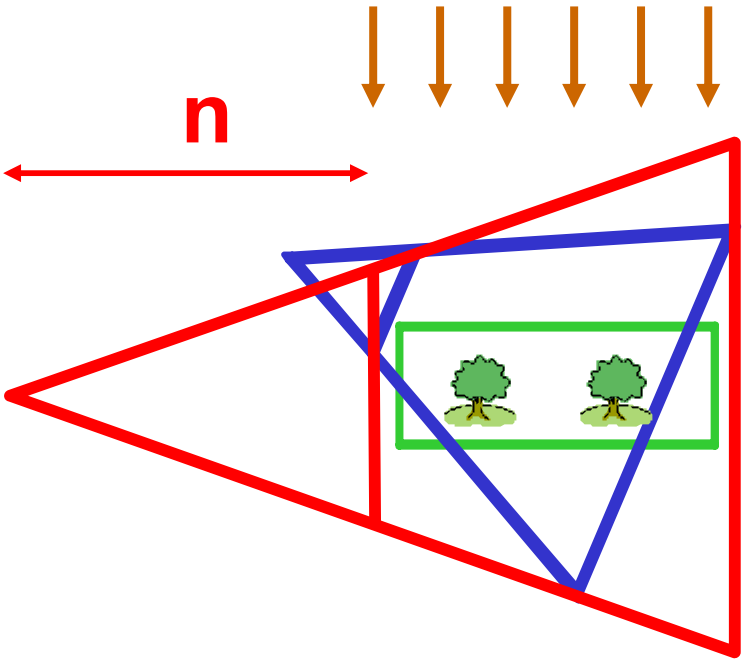
- Perfect: logarithmic re-parameterization

$$s \sim \log z \quad \overset{s'}{\Rightarrow} \quad \frac{ds}{dz} \sim \frac{1}{z} \quad \Rightarrow \quad \frac{dp}{ds} \sim 1$$

  - Hardware support? [Lloyd 2007, 2008]

Error Analysis: LiSPSM Optimal Choice

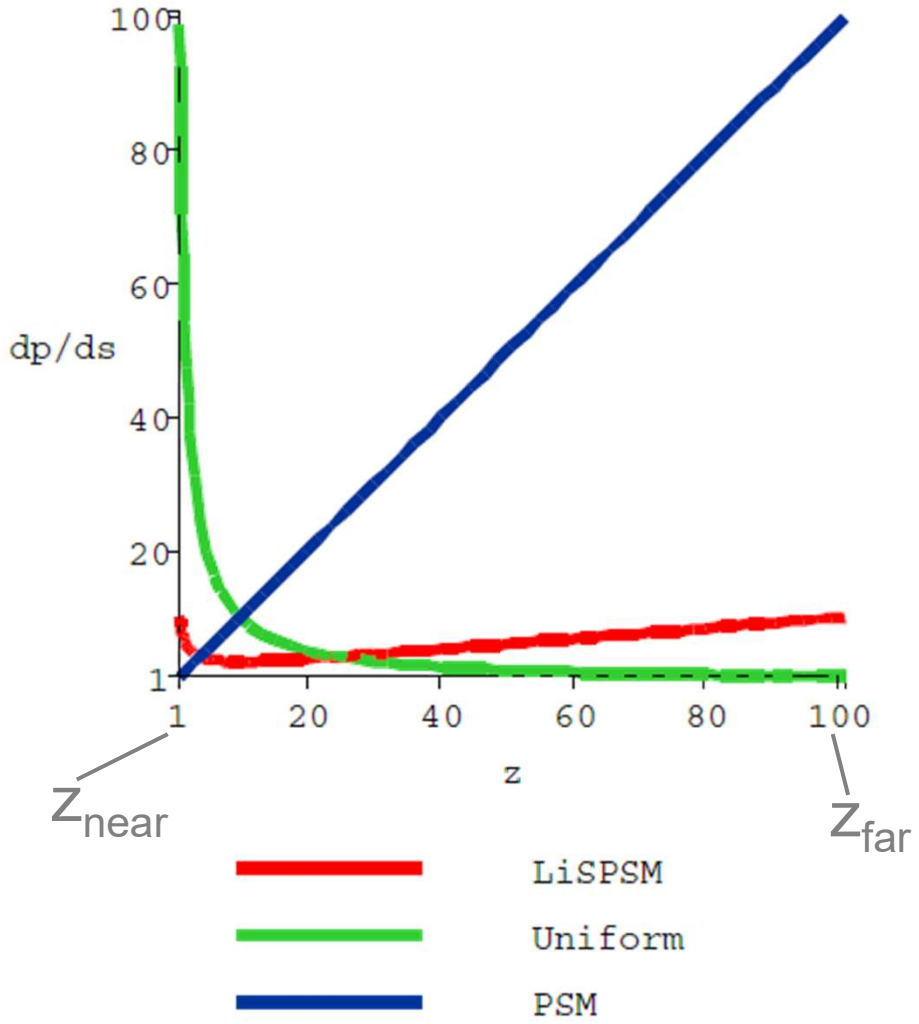
- For LiSPSM, $\frac{dp}{ds}$ depends on **n**
  - Gives $\frac{dp}{ds}$ between uniform and perspective
- Optimal choice:



$$n_{opt} = z_n + \sqrt{z_f z_n}$$

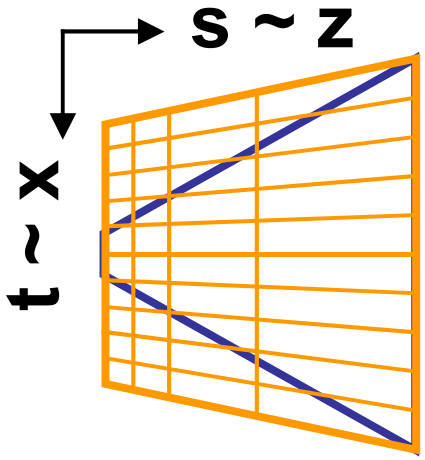
Error Comparison

- LiSPSM optimal choice
- Measured along view dir
- LiSPSM vs PSM
  → for same depth range, LiSPSM error much lower
- More advanced analysis was done in [Lloyd 2006]



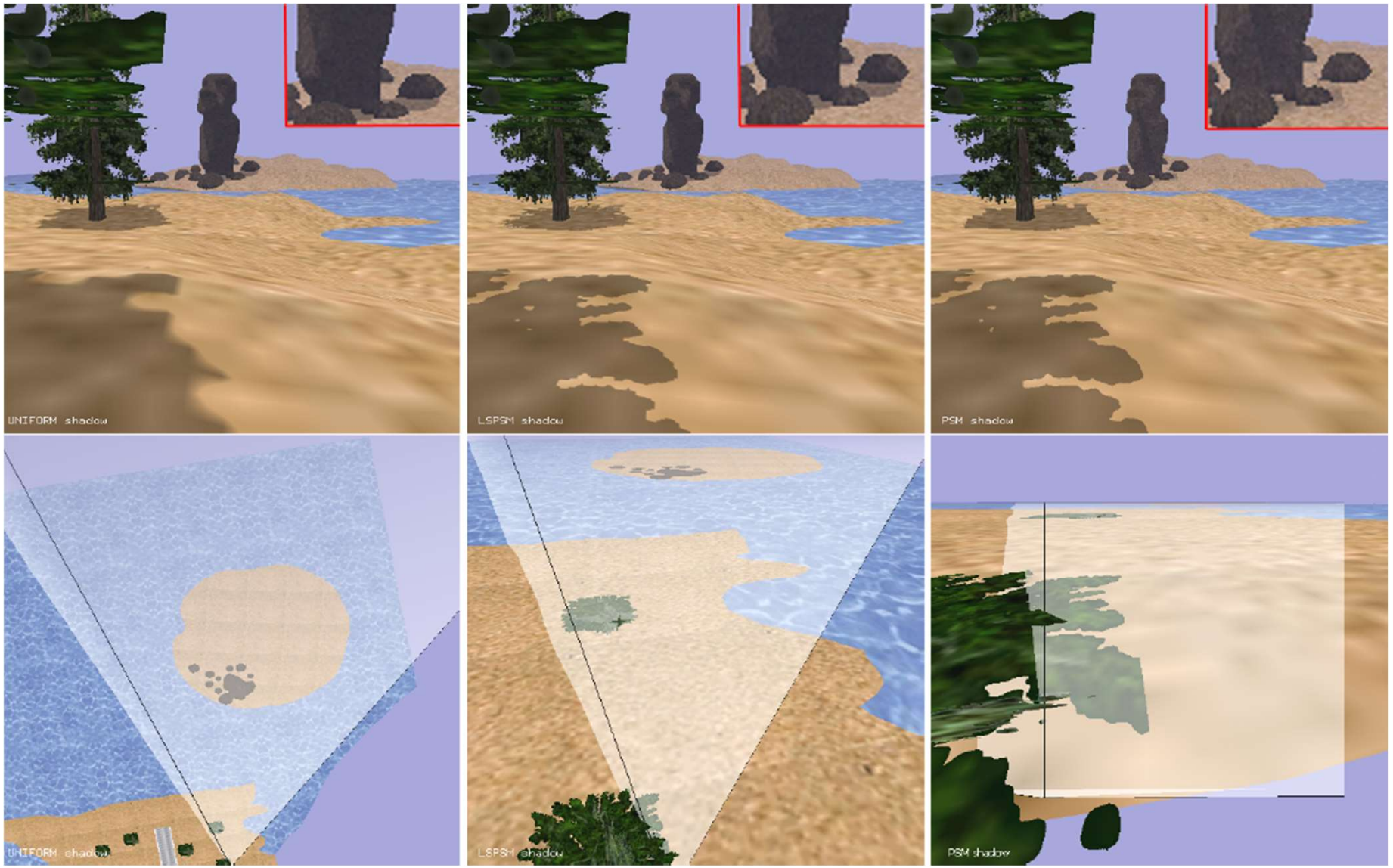
Error Comparison

- Caveat: only measured along view direction
- What about $\frac{dp}{dt}$?
  - $\frac{dp}{dt} \sim 1$ for PSM, slightly worse for LiSPSM



- More advanced analysis was done in [Lloyd 2006]
  - Result: "storage factor" constant for **n** in [1, n_{opt}]
  - But: best error distribution for n_{opt}

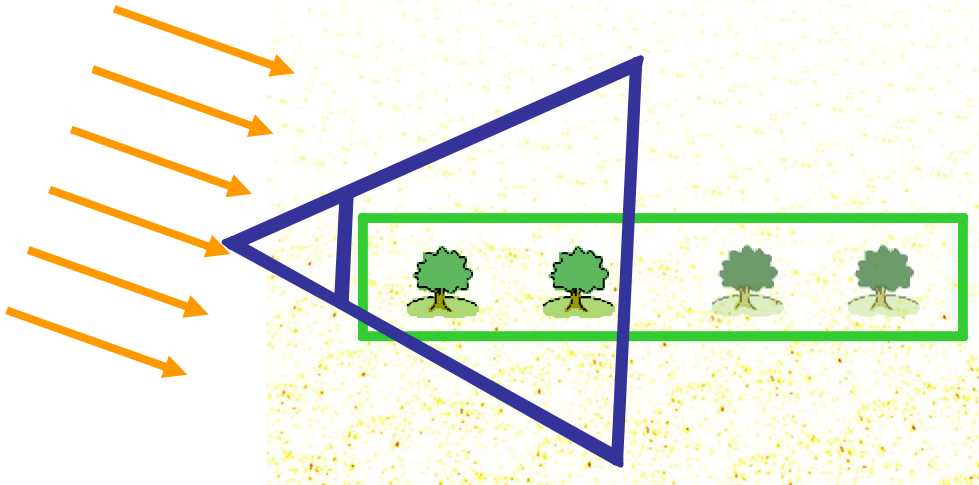
Comparison

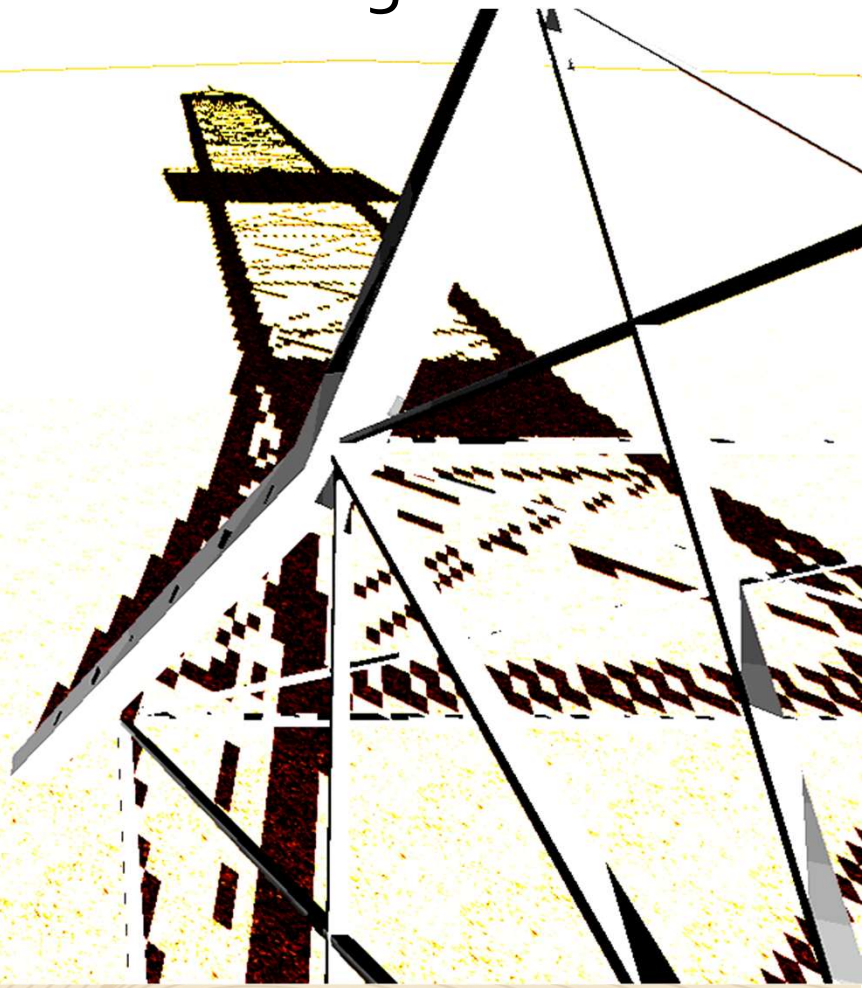


Uniform LiSPSM PSM

Warping: Problems

- Only works if large z-range visible from light
  - Outdoor scenes
- Bad: Dueling frusta case
- But: Fitting often alleviates this



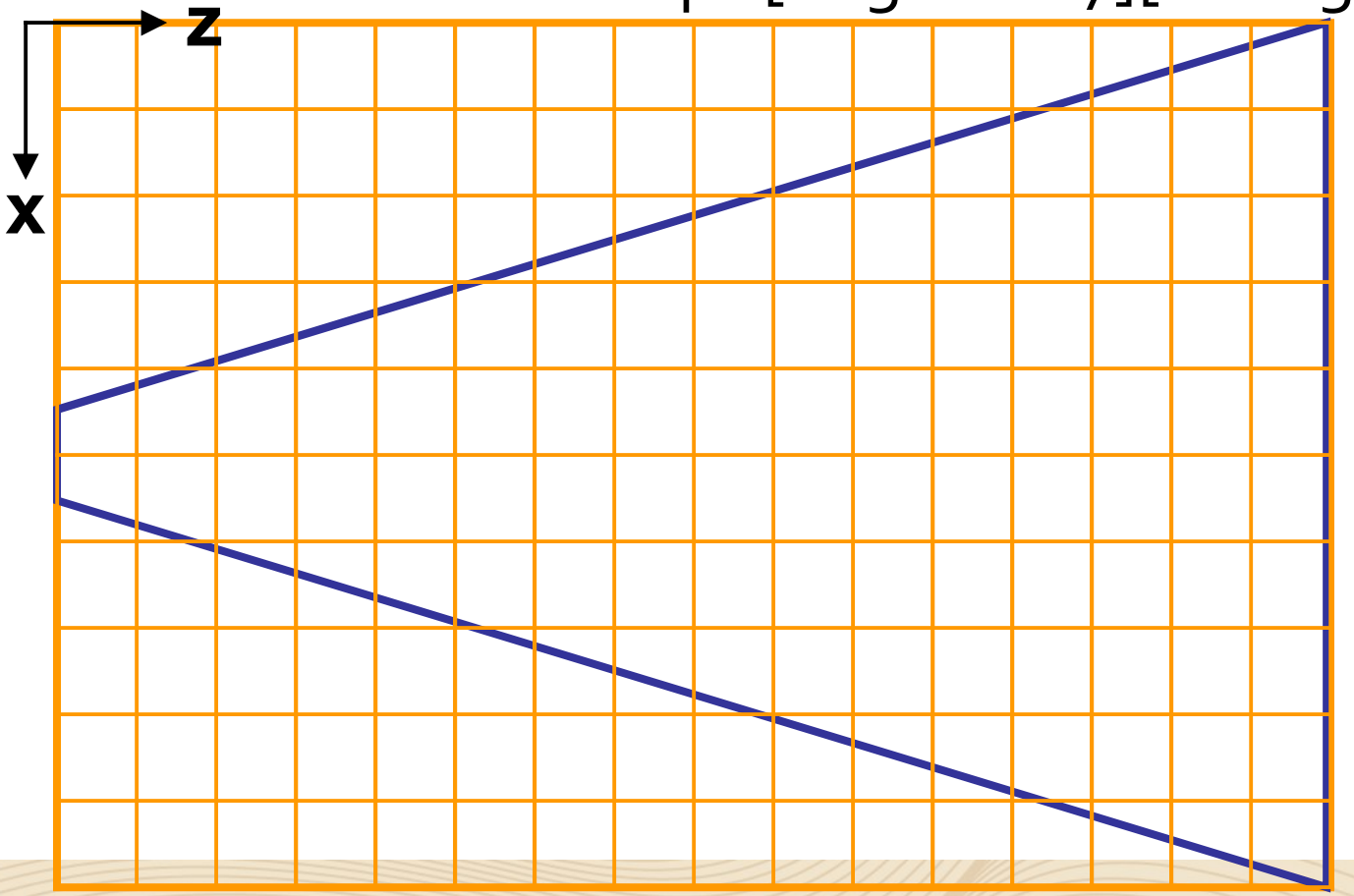


Hard Shadows

Fighting Undersampling - Partitioning

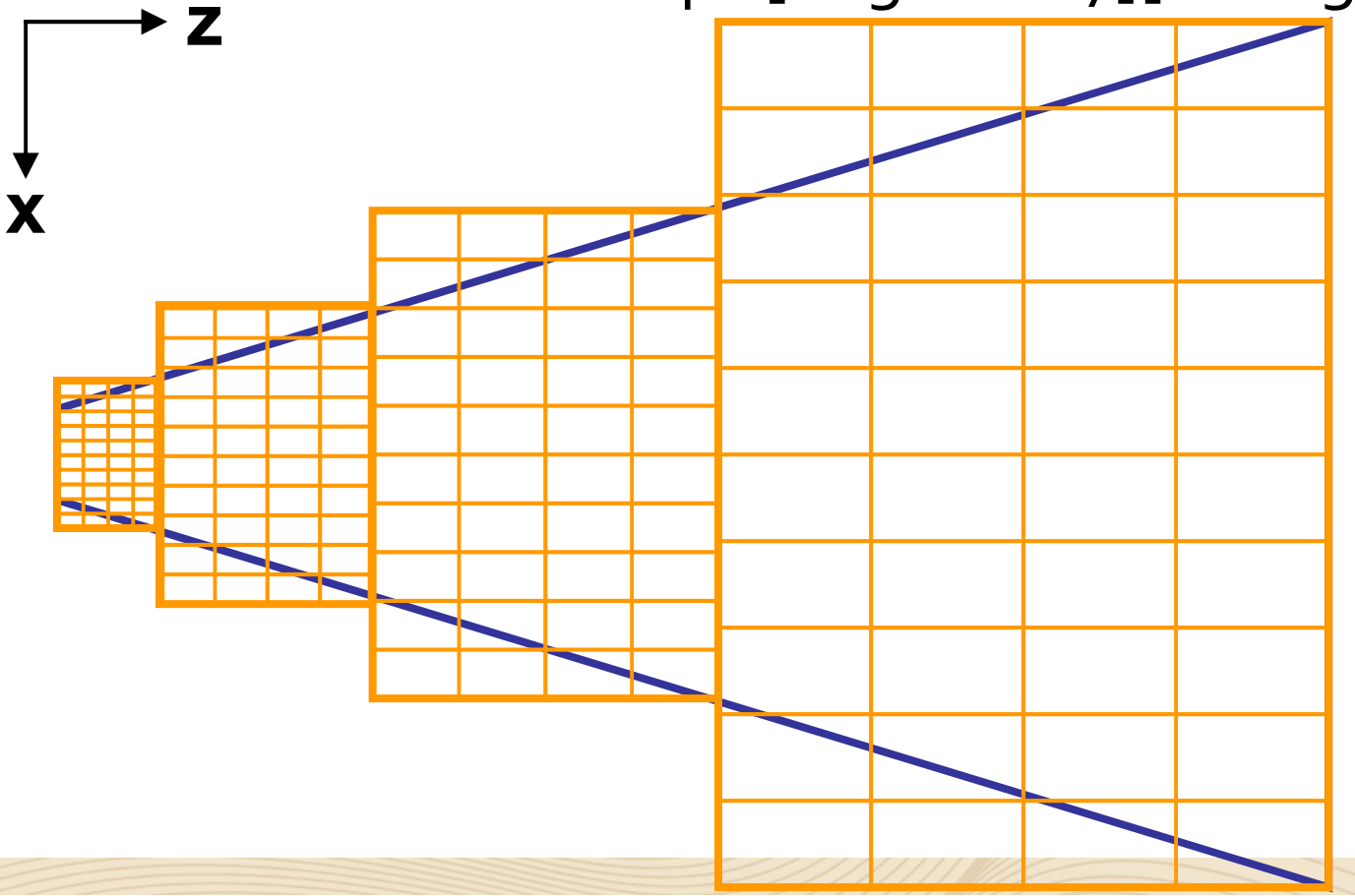
Z-Partitioning: Idea

- Parallel Split Shadow Maps [Zhang 2007]
- Cascaded Shadow Maps [Engel 2007][Zhang2009]



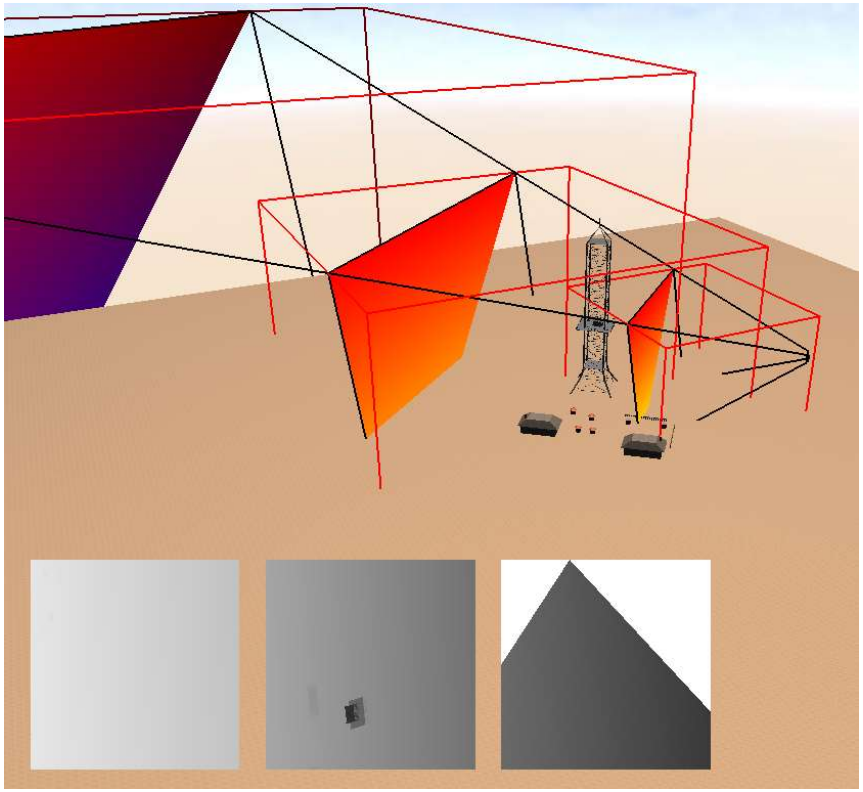
Z-Partitioning: Idea

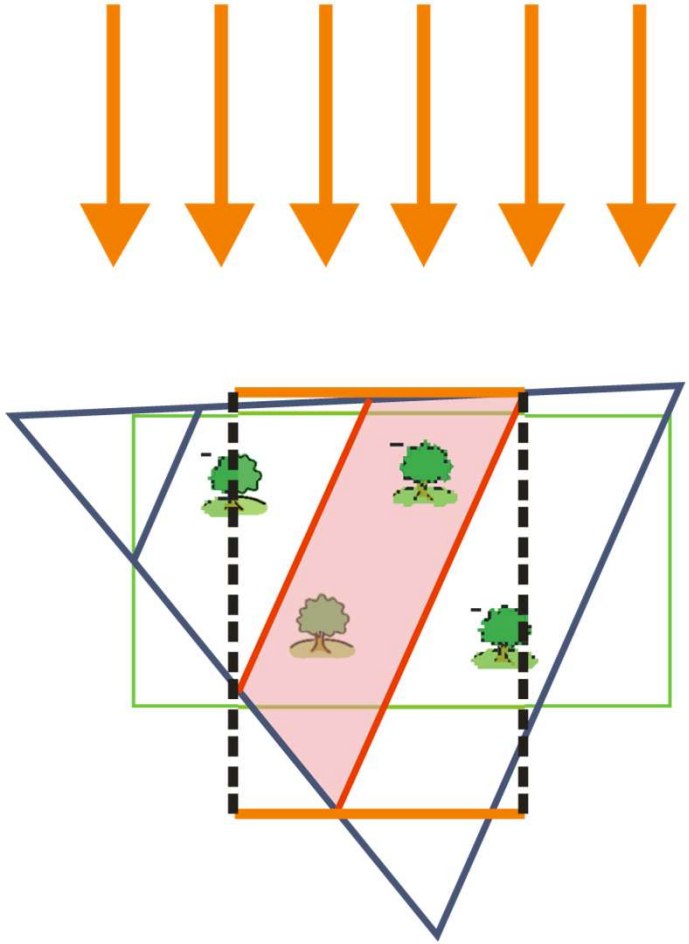
- Parallel Split Shadow Maps [Zhang 2007]
- Cascaded Shadow Maps [Engel 2007][Zhang2009]



Z-Partitioning

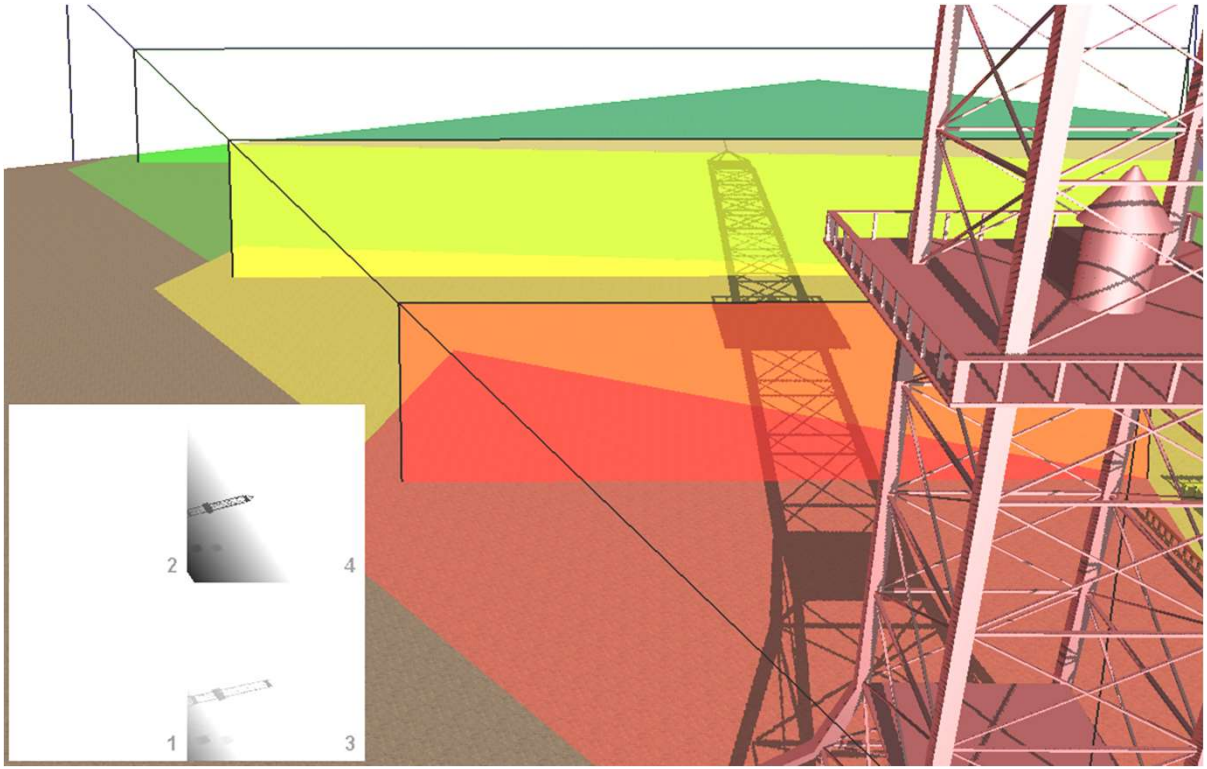
- Partition view frustum into n sub-frusta
- Calculate separate shadow map for each

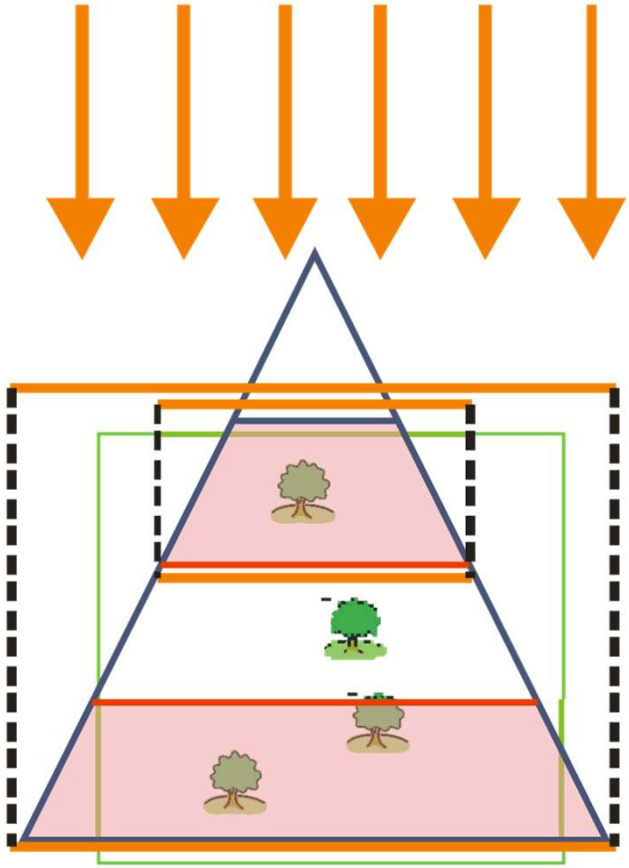


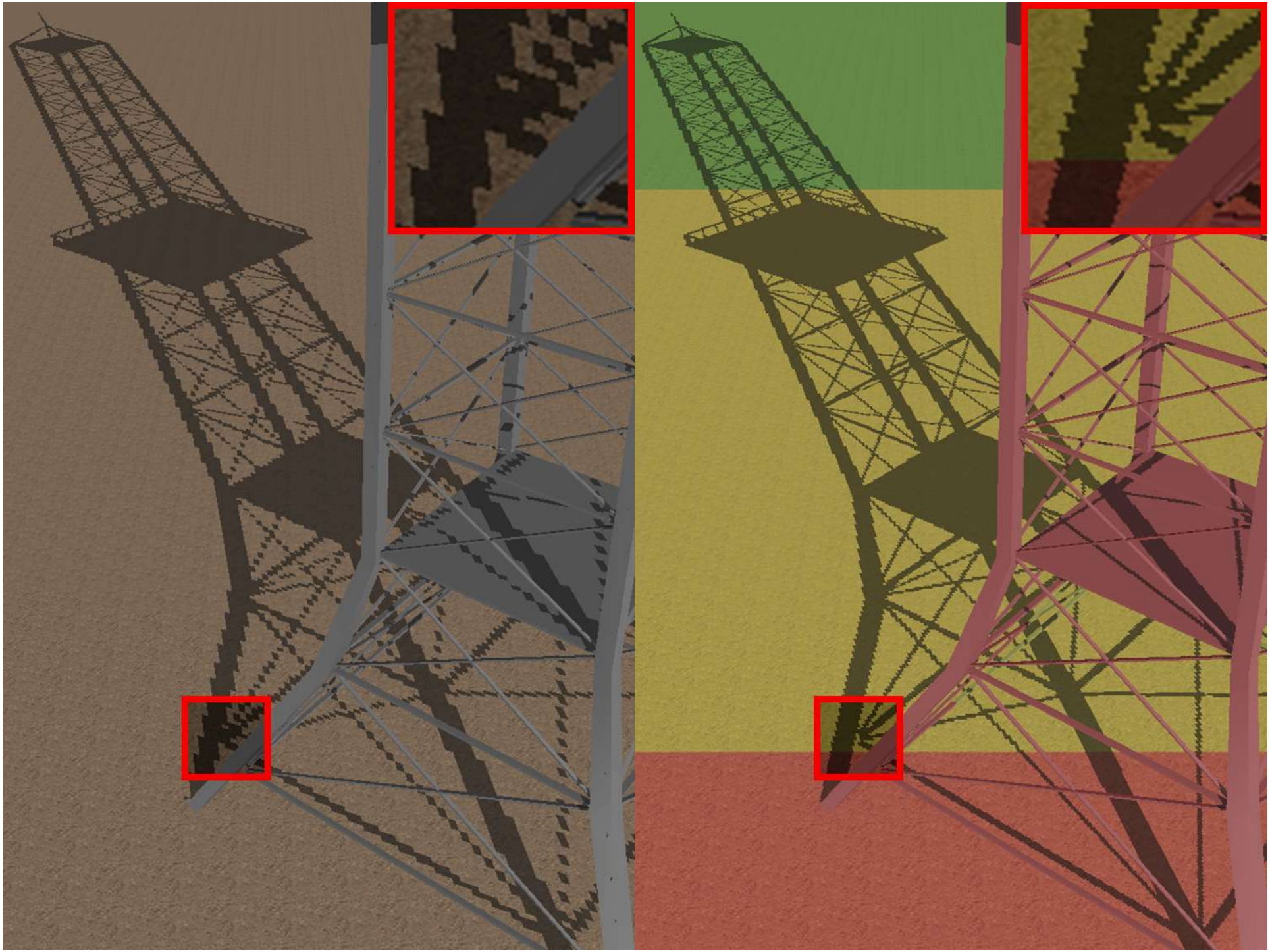


Z-Partitioning

- Works even in cases where warping fails
  - Light from behind, dueling frusta

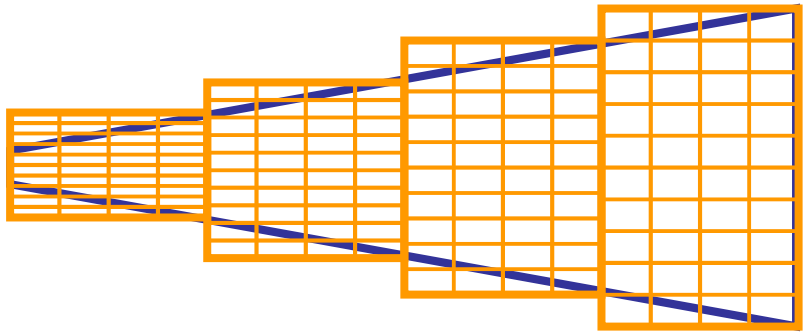






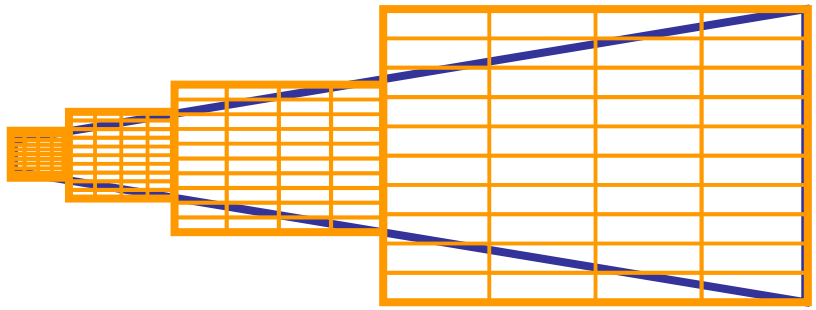
Z-Partitioning

- How to choose partition sizes?
  - Uniform

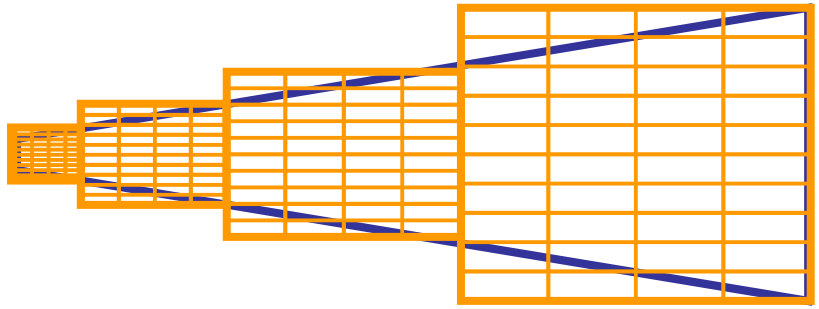


  - Logarithmic/self-similar

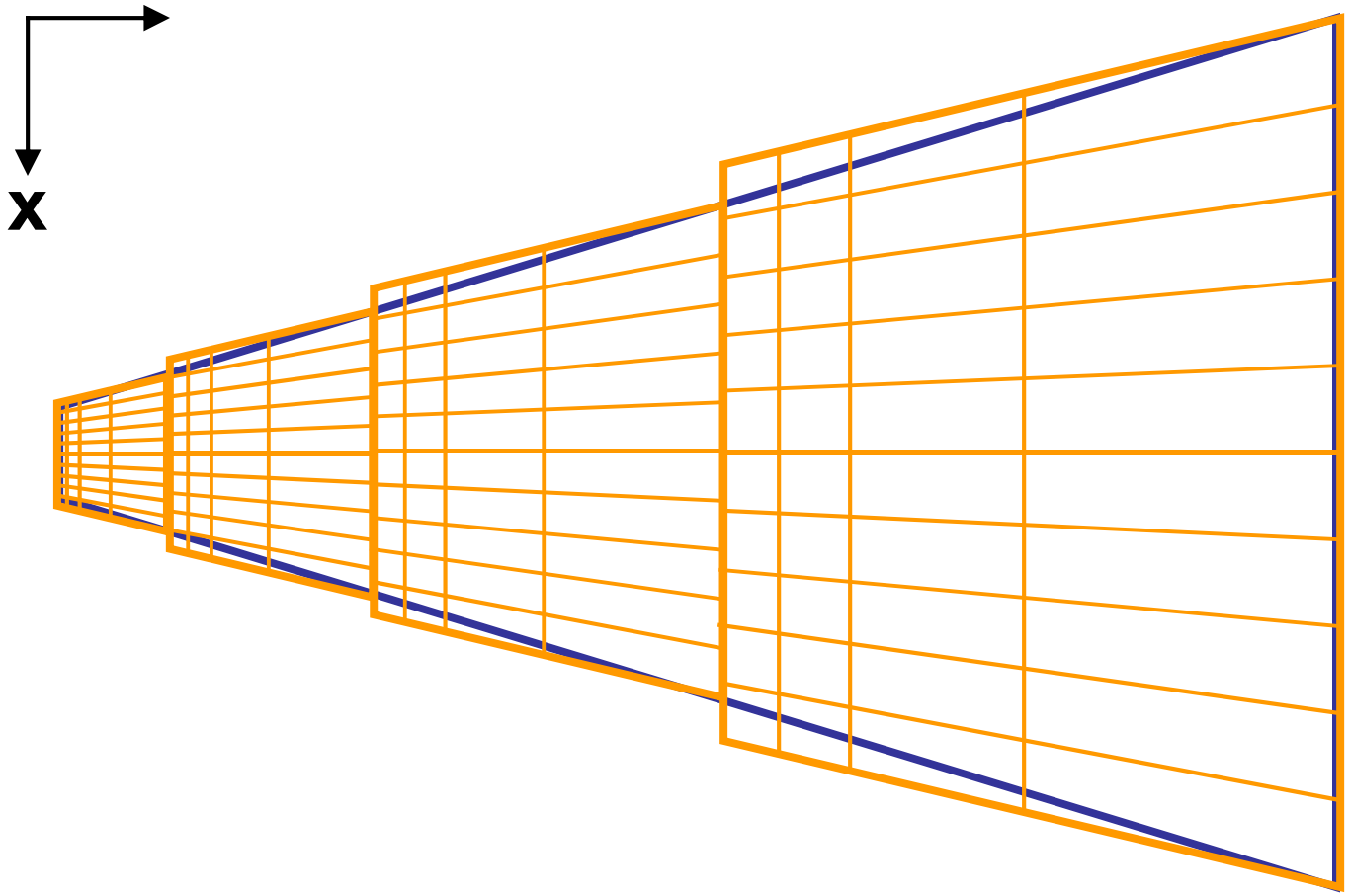
$$C_i = z_n \left(\frac{z_f}{z_n} \right)^{\frac{i}{m}}$$



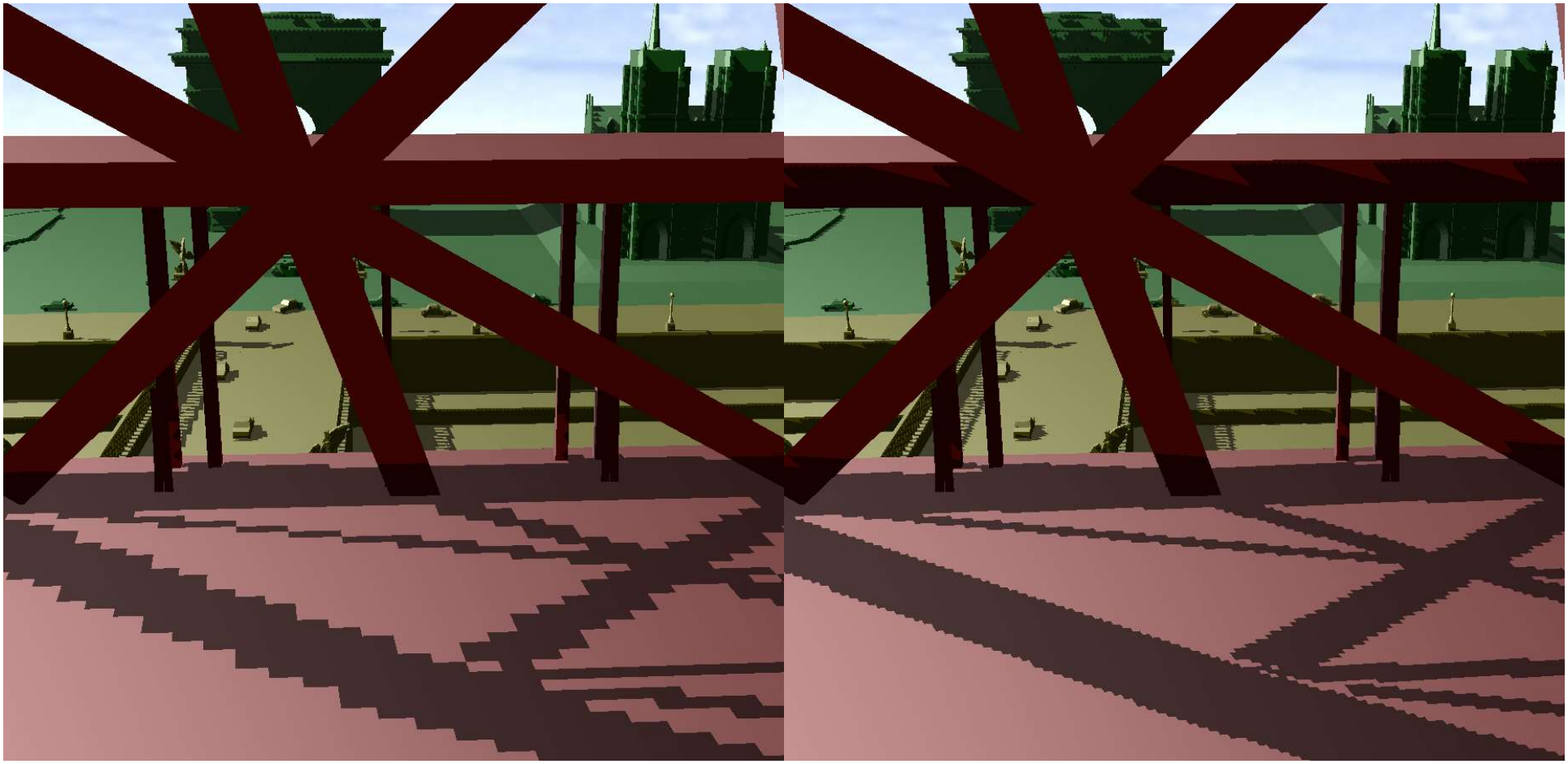
  - Linear blend between the two



Z-Partitioning and Warping



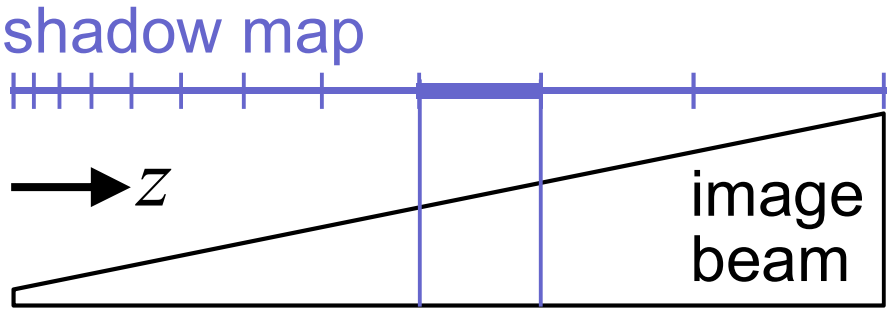
Z-Partitioning and Warping



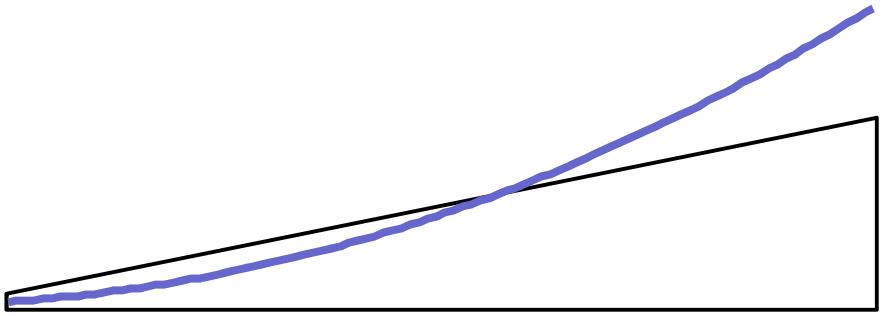
Partitioning **Partitioning + warping**

Z-Partitioning

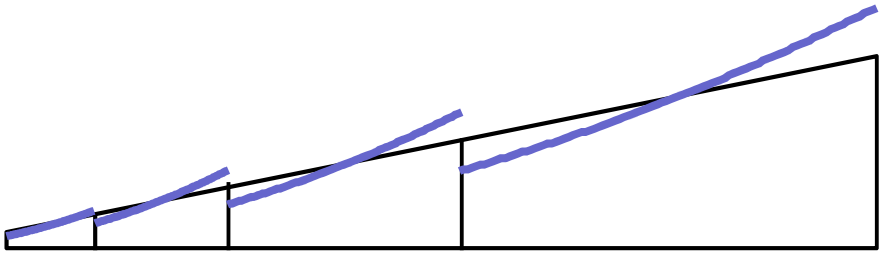
- Optimal shadow map texel spacing



- Spacing from a 4x4 projective transform

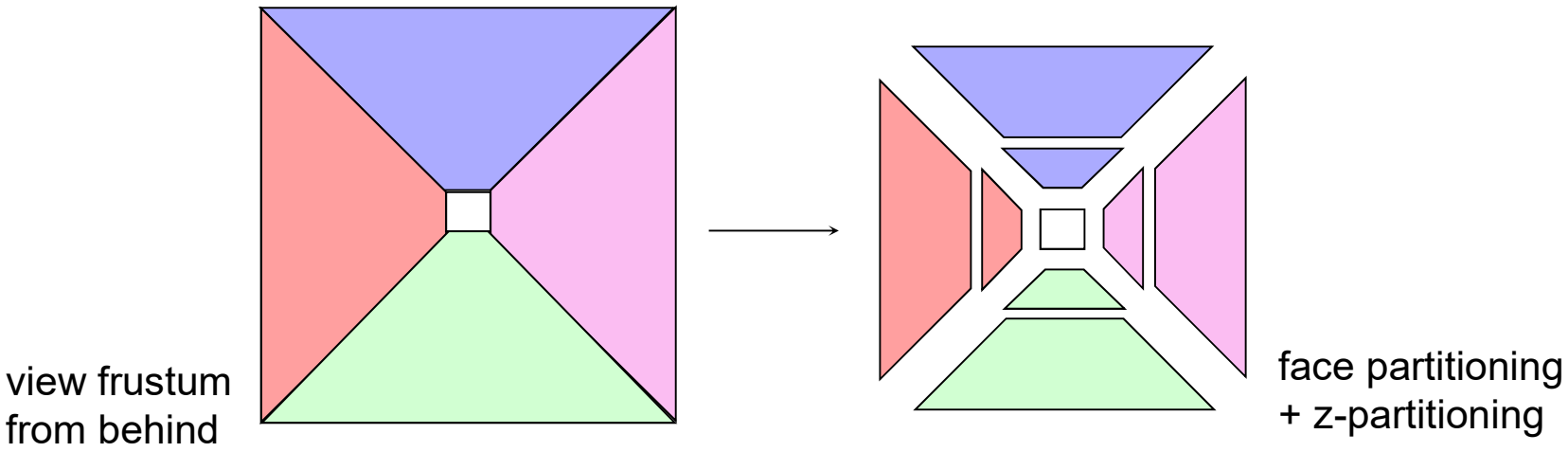


- Partitioning gives a better fit

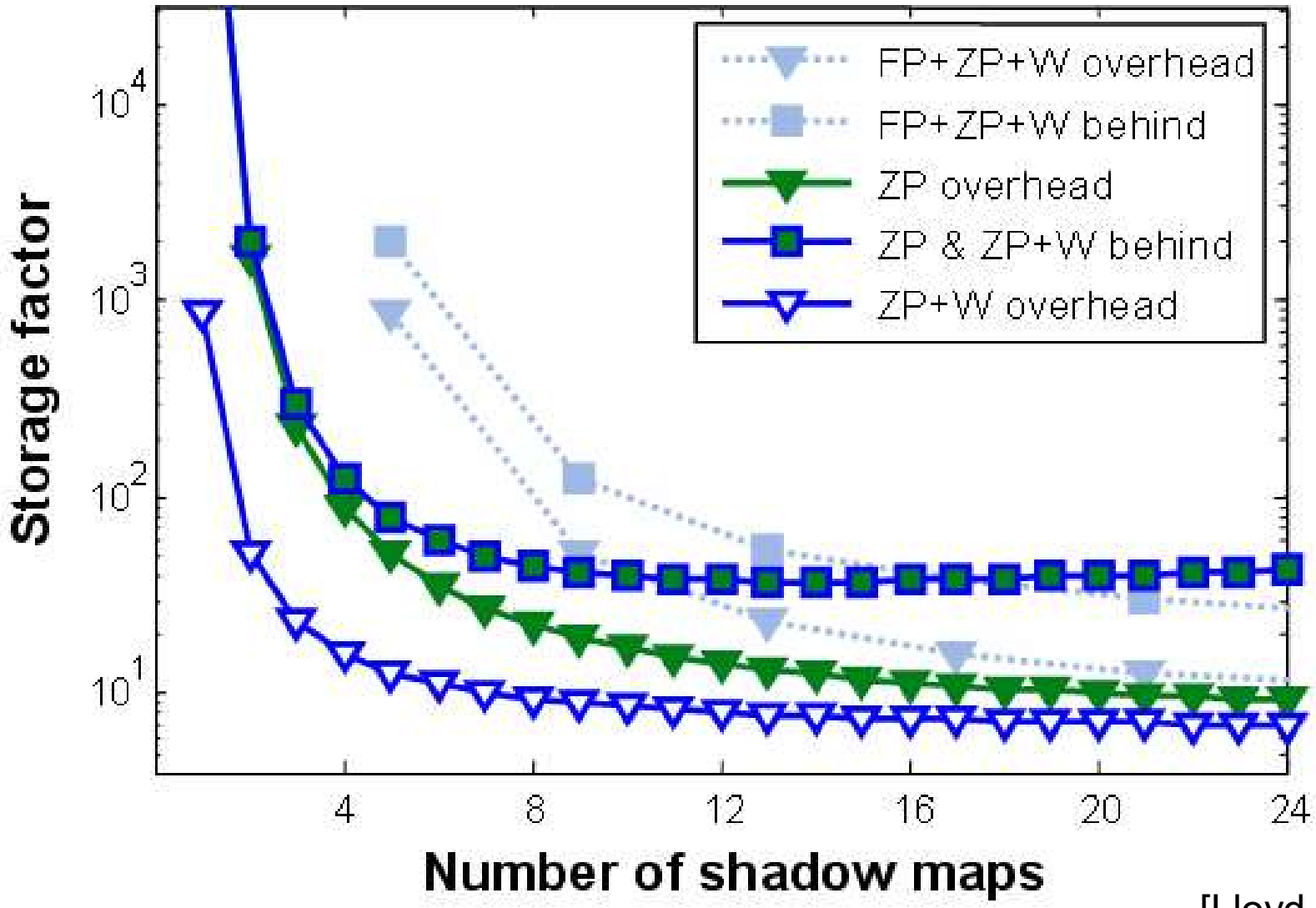


Face partitioning

- Problem: warping not good for all light directions
  - Especially from behind
- Solution: Partition frustum according to faces
- Can be combined with z-partitioning



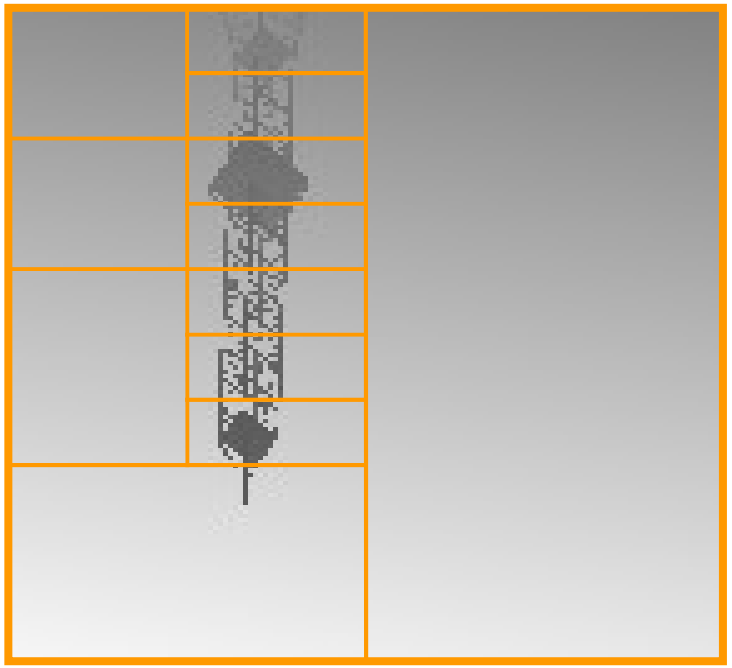
Full Error Analysis



[Lloyd 2006]

Adaptive Partitioning

- Warping and z-partitioning are **global** resampling schemes
  - Deal with perspective aliasing
  - Projection aliasing needs local scene adaptive resampling!
- Adaptive partitioning adaptively splits shadow map
  - Usually quad-tree subdivision
- Algorithms mainly differ in termination criteria

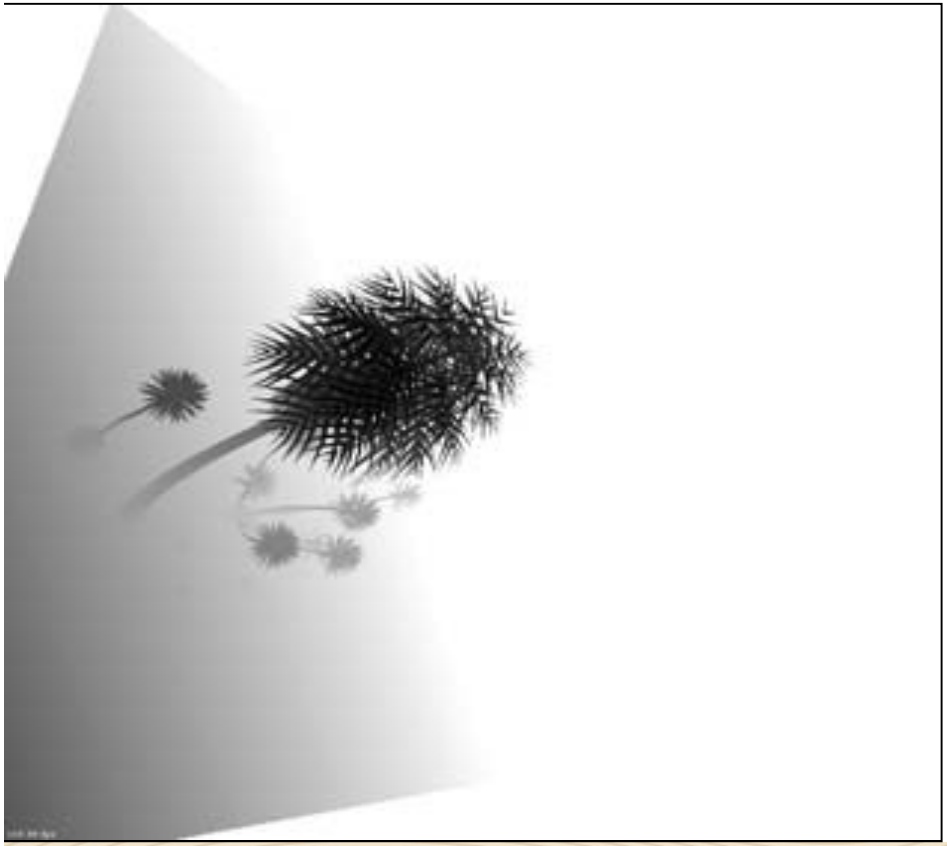


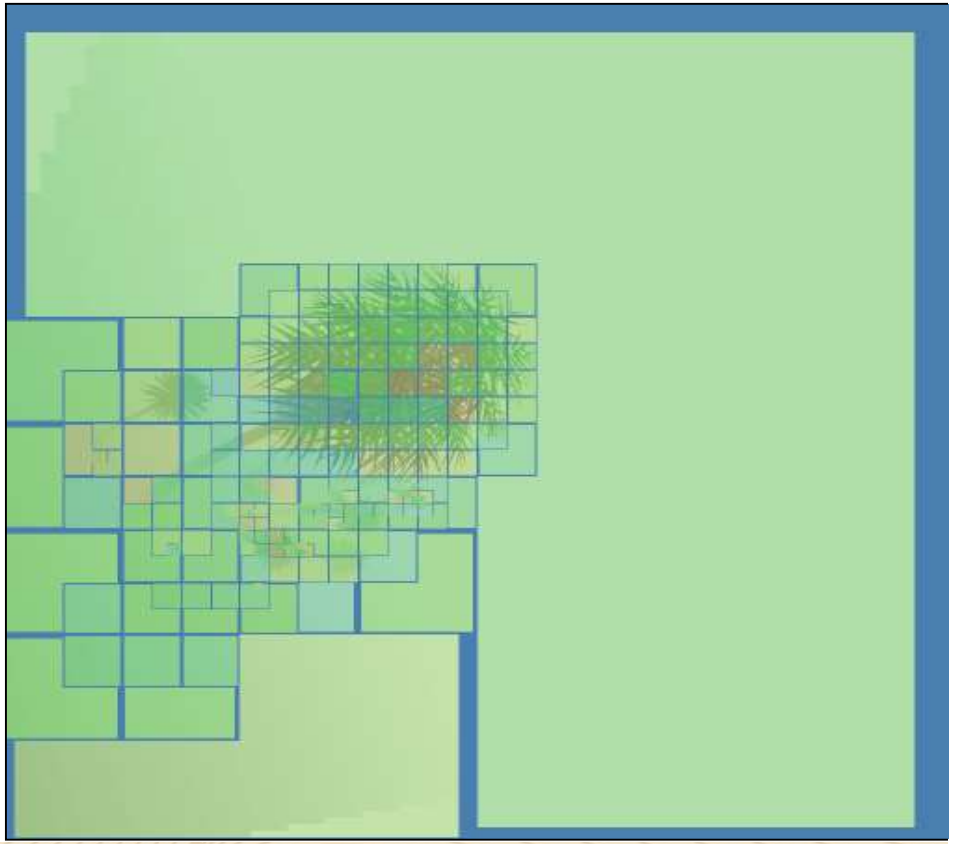
Tiled shadow maps
[Arvo 2004]

- Split according to heuristics
- Based on depth discontinuities, distances, ….
- Allows to trade speed against quality

Adaptive shadow maps
[Fernando et al. 2001; Lefohn et al. 2006]

- High resolution only needed at edges
- Search for edge (slow)
  - If edge split



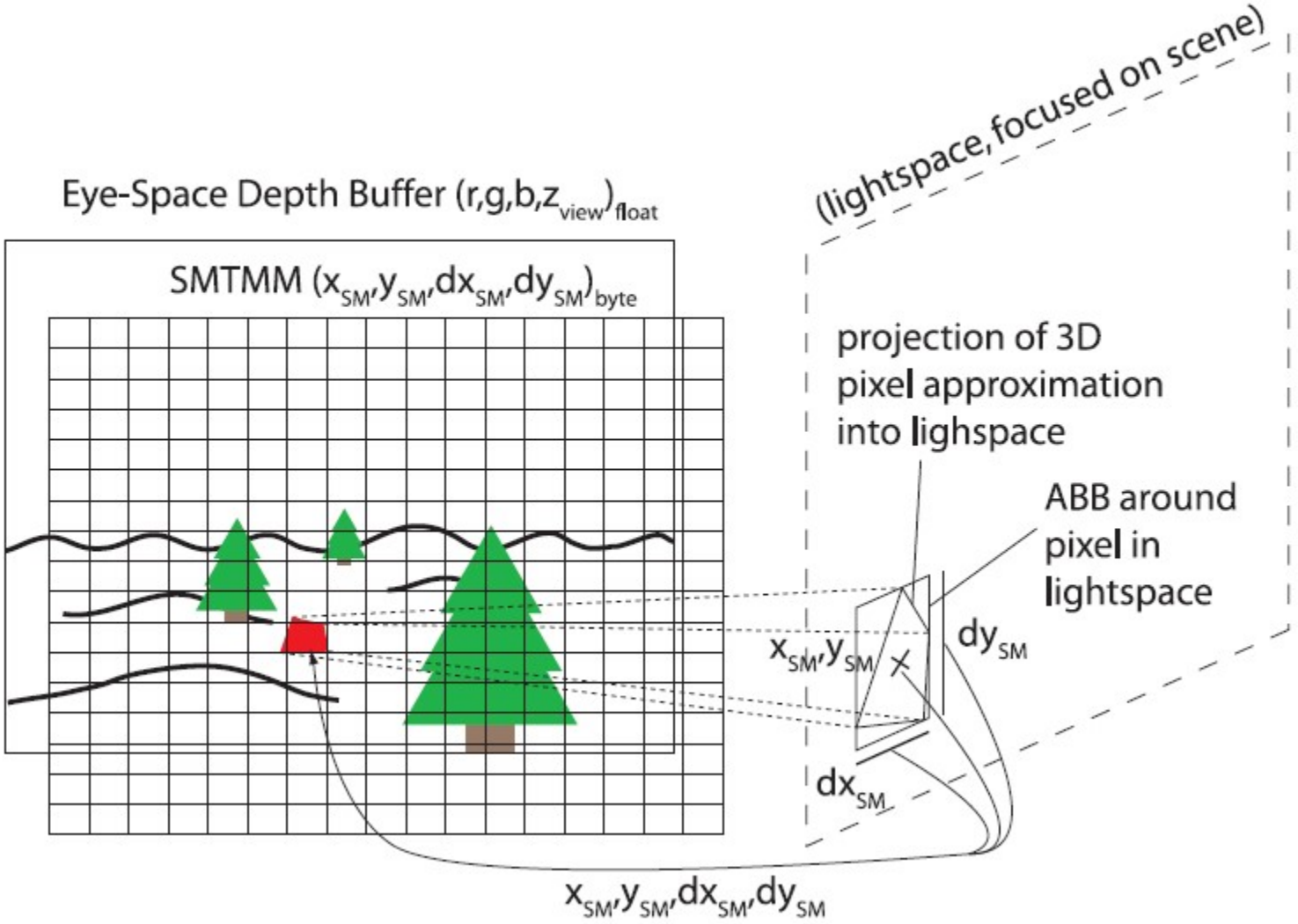


Queried/Fitted Virtual Shadow Maps

- Queried Virtual Shadow Maps [Giegl 2006]
  - Render 4 shadow maps of a quad-tree node
  - Check if this was necessary (occlusion query)
- Fitted Virtual Shadow Maps [Giegl 2007]
  Resolution Matched Shadow Maps [Lefohn 2007]
  - Project pixel footprints into shadow map
  - Generate quad-tree
  - FVSM: CPU, RMSM: GPU

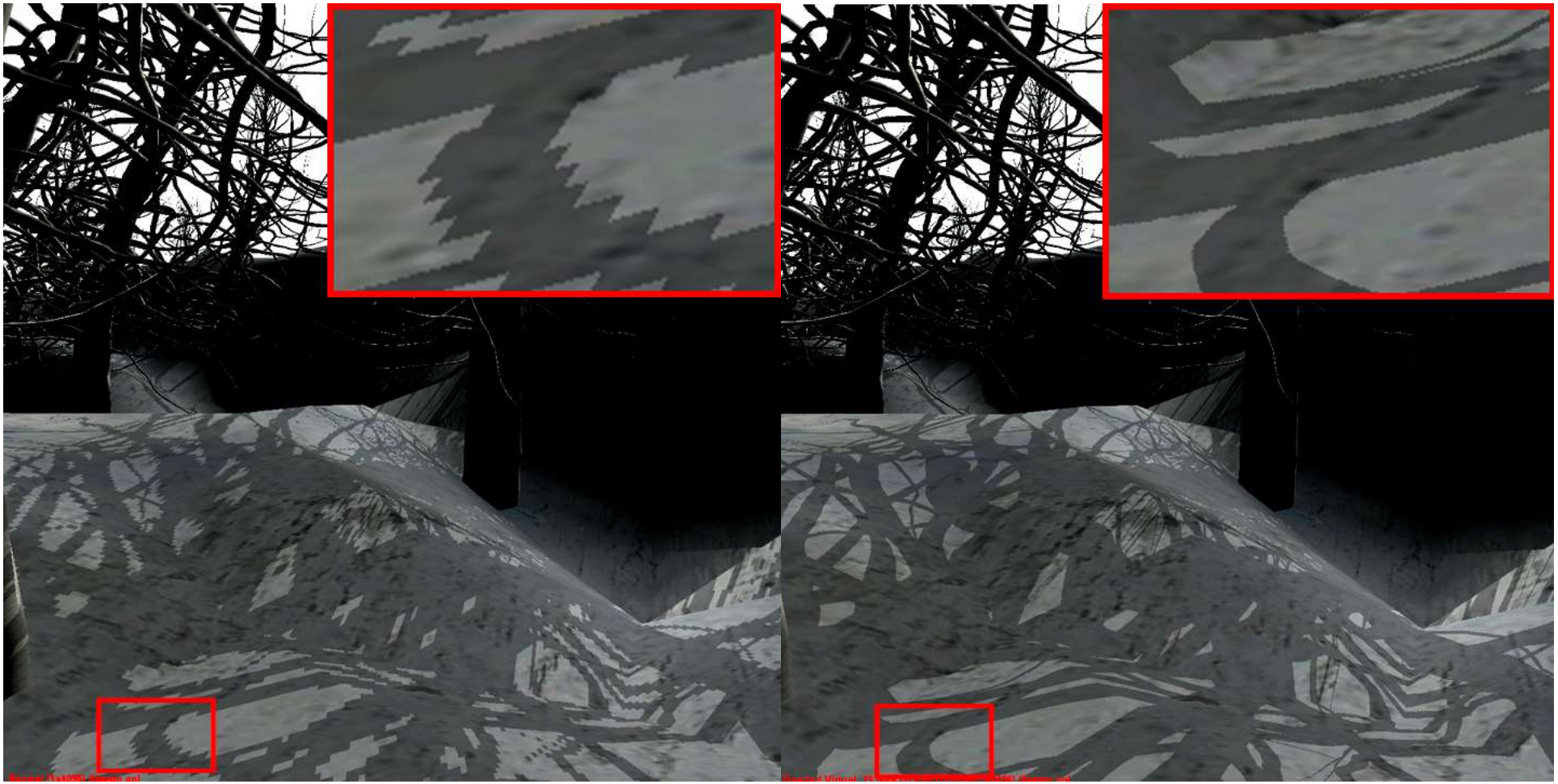
Fitted Virtual Shadow Maps

[Giegl & Wimmer 2007]



Fitted Virtual Shadow Maps

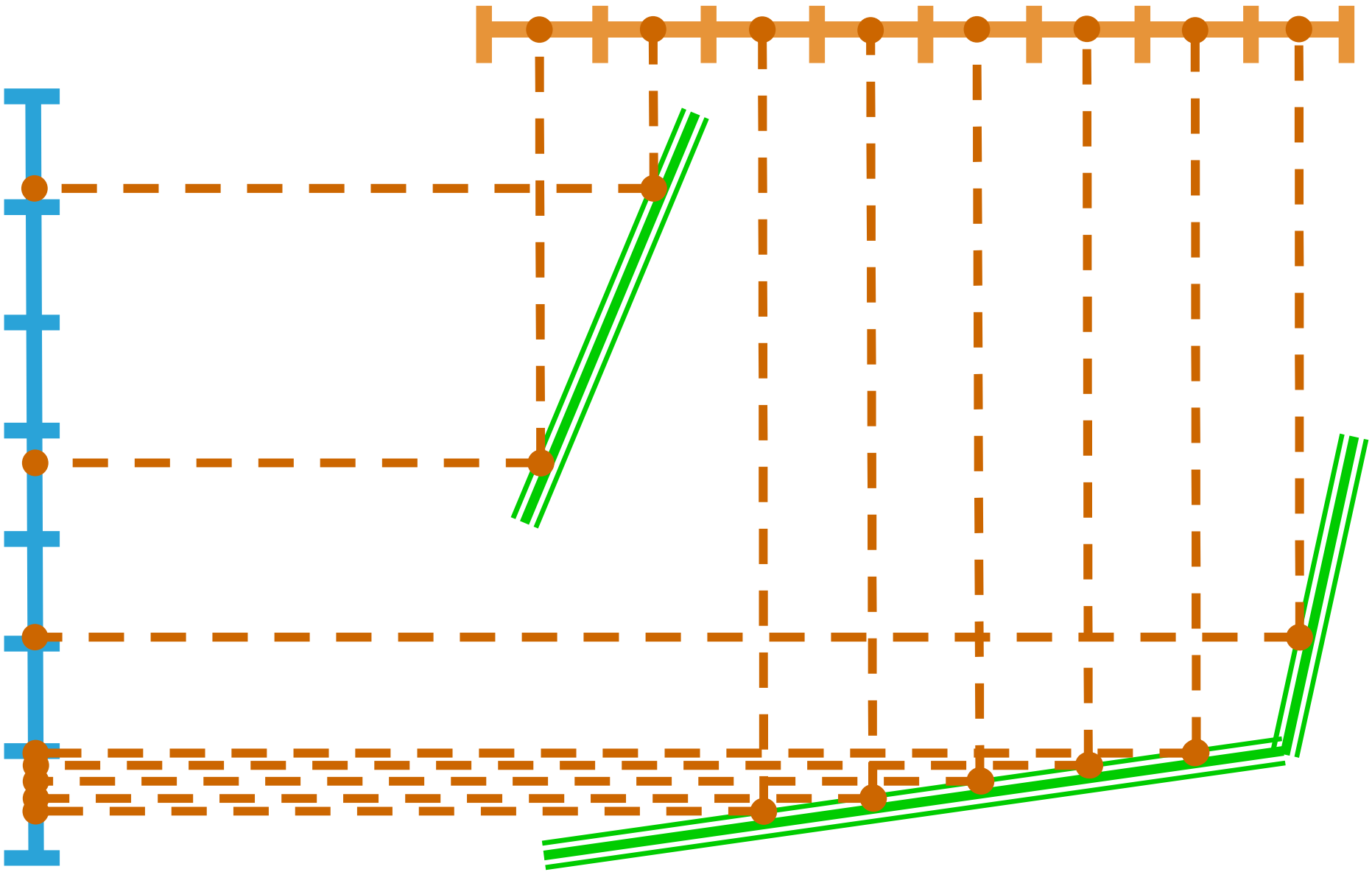
[Giegl & Wimmer 2007]



Hard Shadows

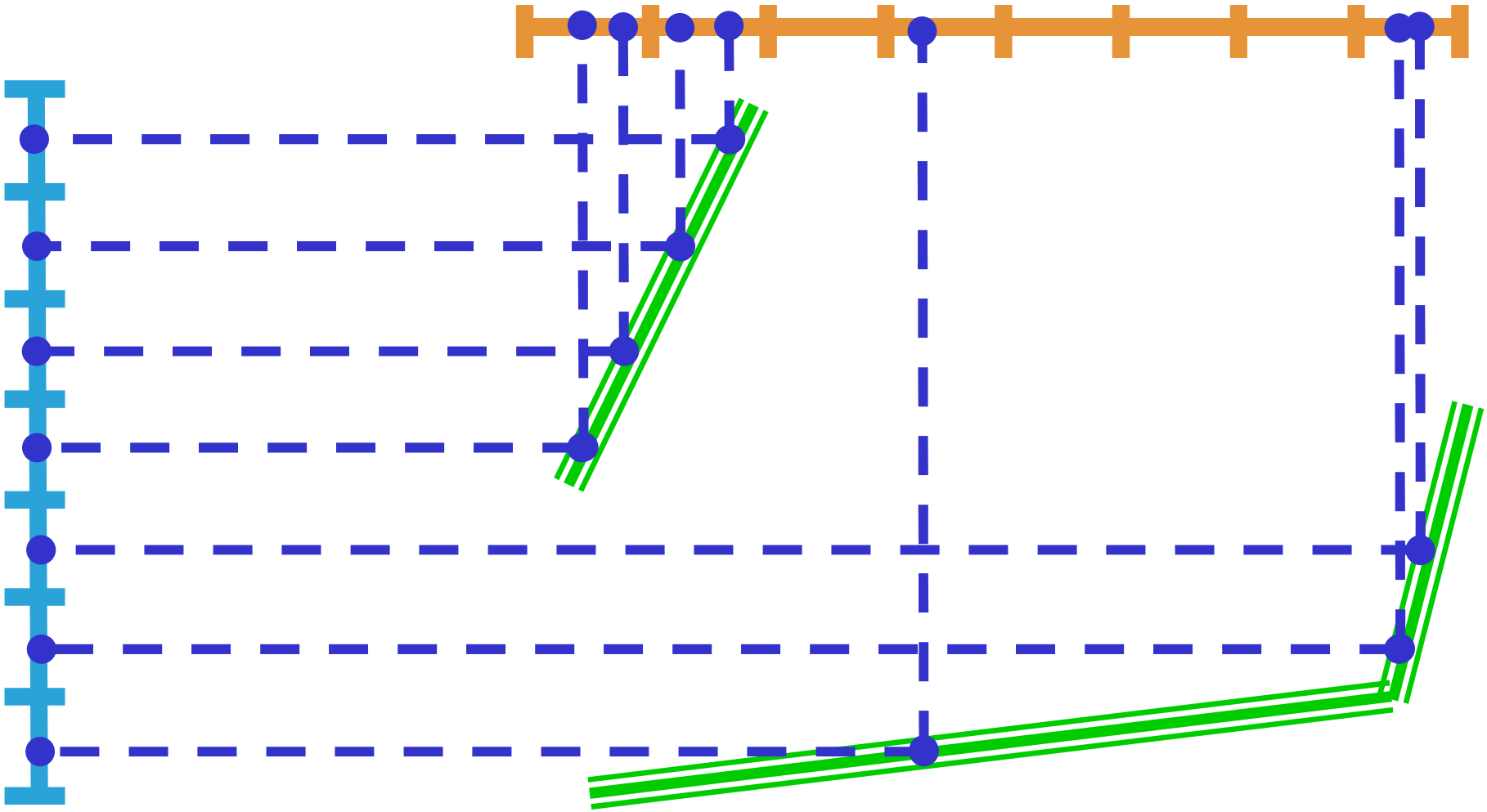
Fighting Undersampling - Irregular Sampling

Shadow Mapping Sampling



What Samples Do We Want?

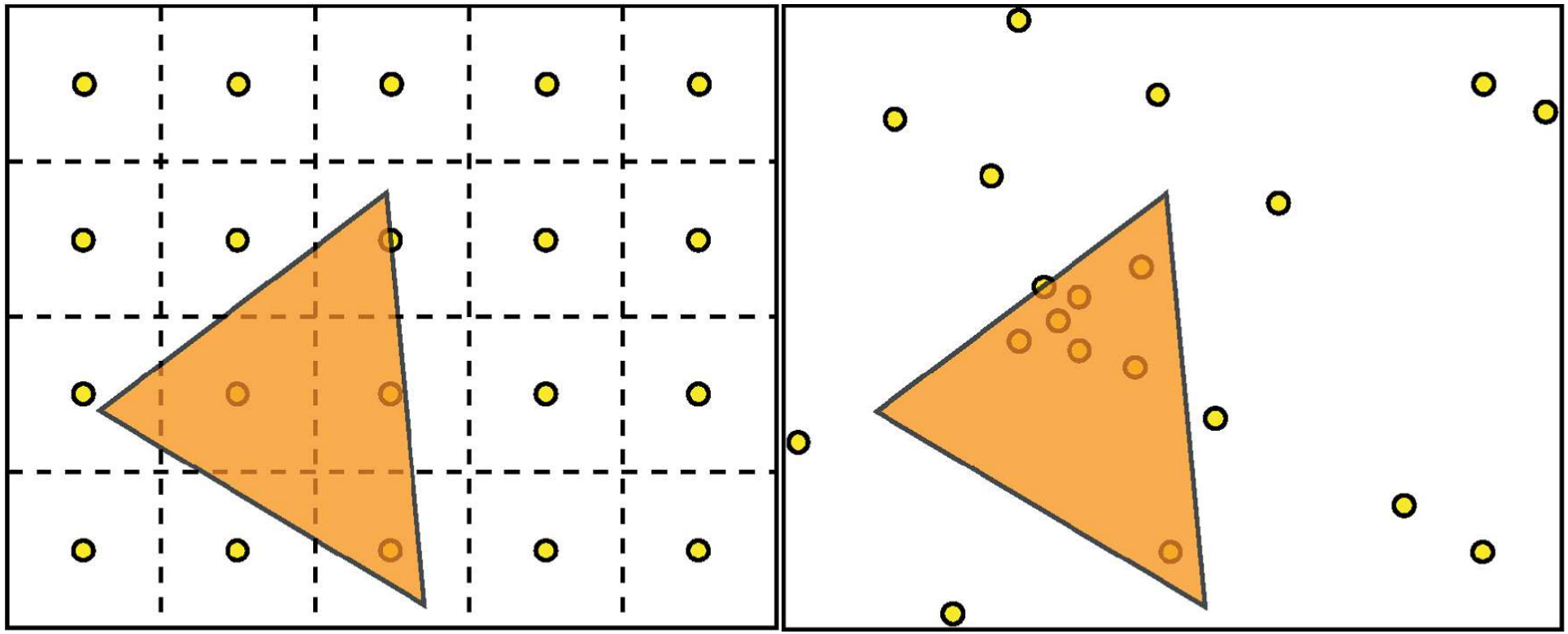
- Idea: use eye space samples to generate shadow map samples



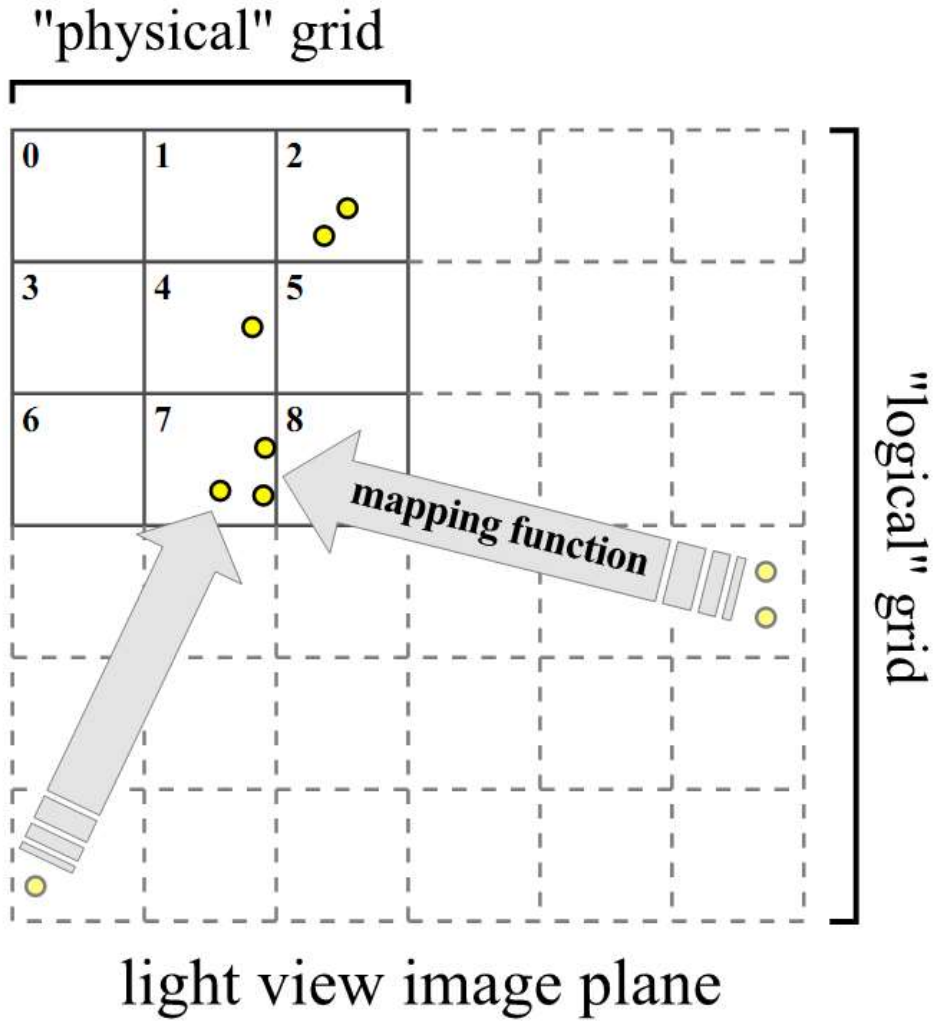
Regular vs Irregular Sampling

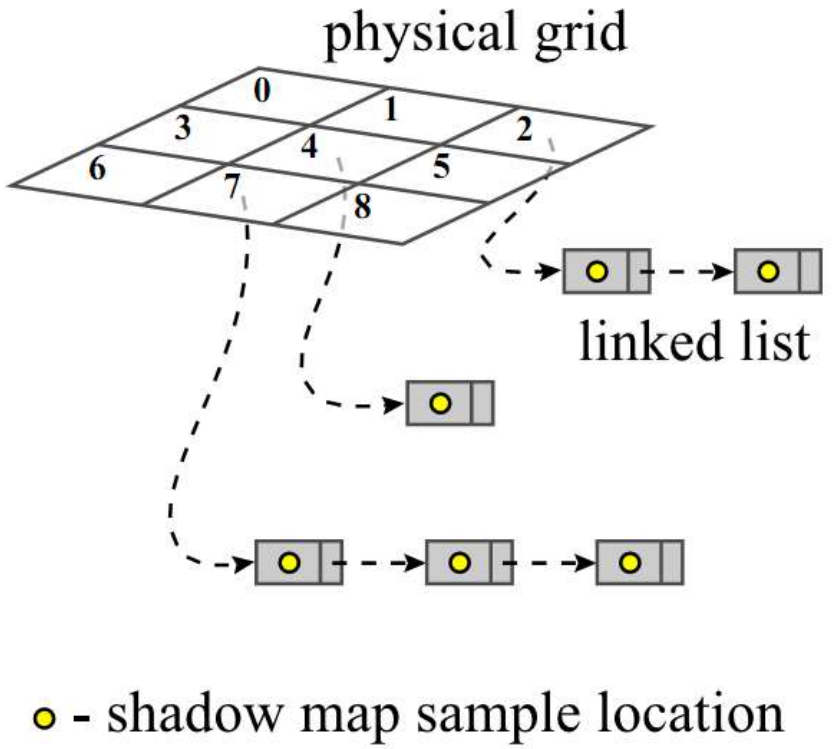
Irregular z-Buffer
[Johnson et al. 2004]

- Proposes hardware extension



Irregular z-Buffer





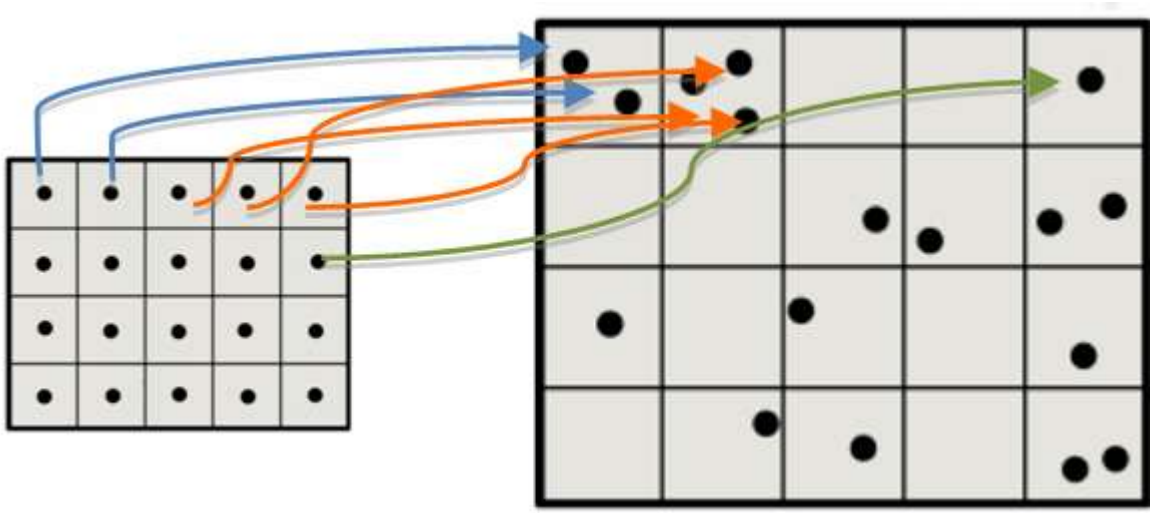
Alias Free Shadow Maps
[Aila and Laine 2004]

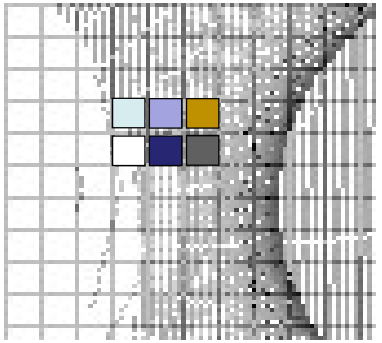
- = Irregular Z-Buffer [Johnson 2004] in software
- Transform and project view-samples into light-space
- Rasterize blocker geometry using them as sampling points
  - Test if sampling point covered
  - Hierarchical processing of sampling points (axis-aligned 2D BSP)
  - In software

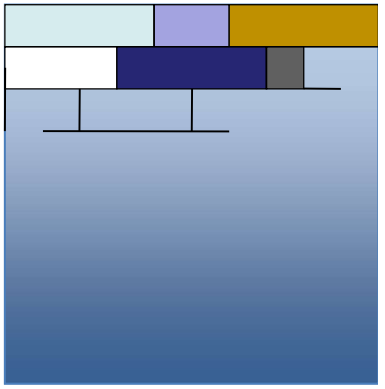
Alias Free Hard Shadows

[Sintorn et al. 2008]

- Transform and project view-samples into light-space
- Store in a compact data structure with a list per light-space texel



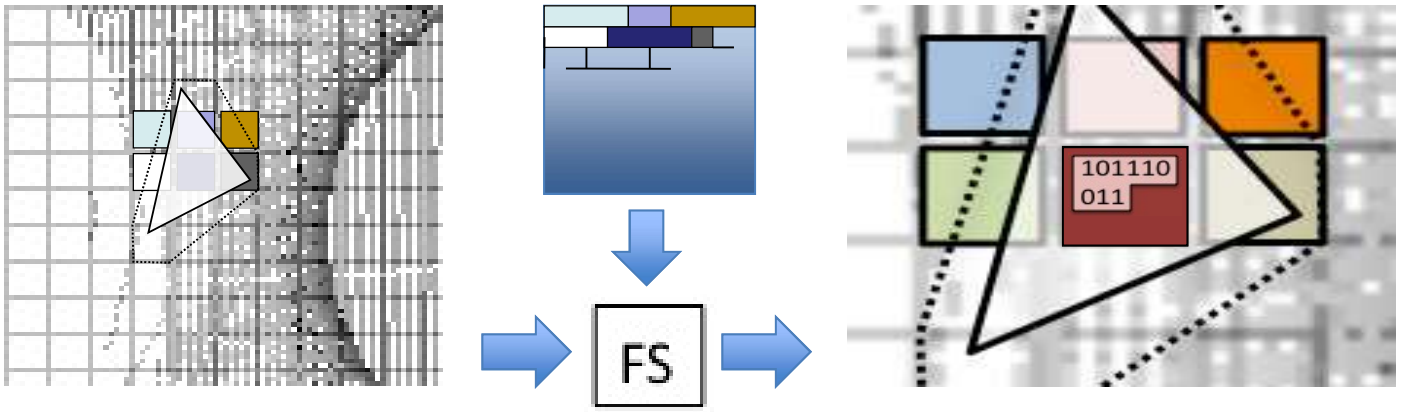




Alias Free Hard Shadows

[Sintorn et al. 2008]

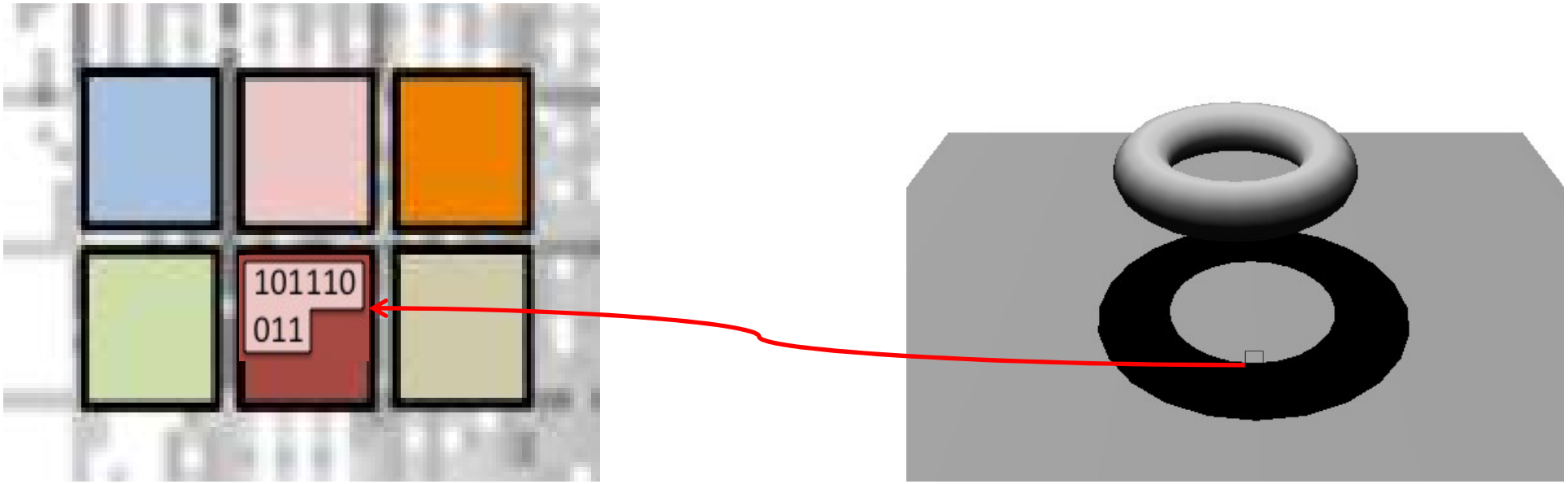
- Render all geometry (conservatively) from lights point of view
- For each generated fragment, test all view-samples in list against triangle
- Set corresponding output bit



Alias Free Hard Shadows

[Sintorn et al. 2008]

- Final screen-space pass
- Use bitmask from previous pass for shadowing

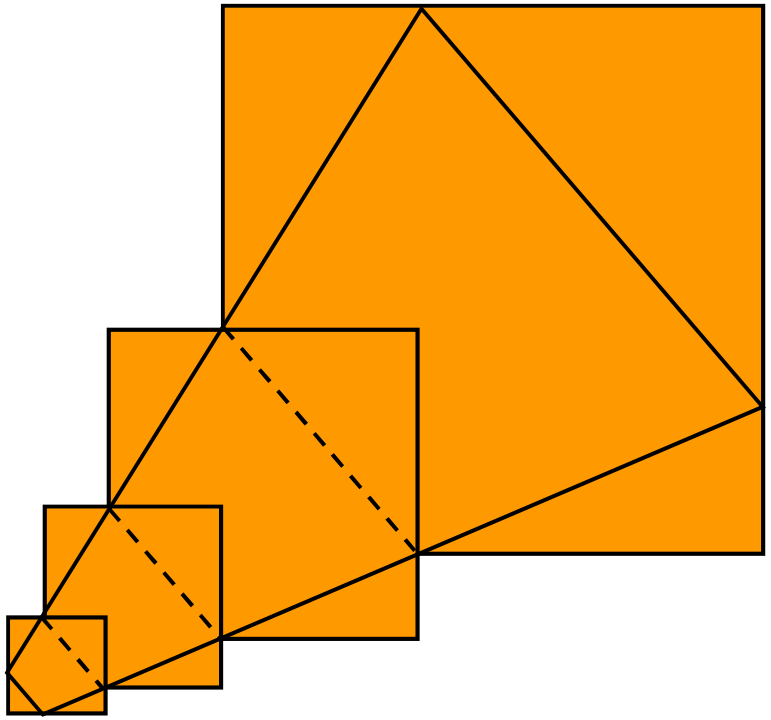


Hard Shadows

Fighting Undersampling - Temporal Reprojection

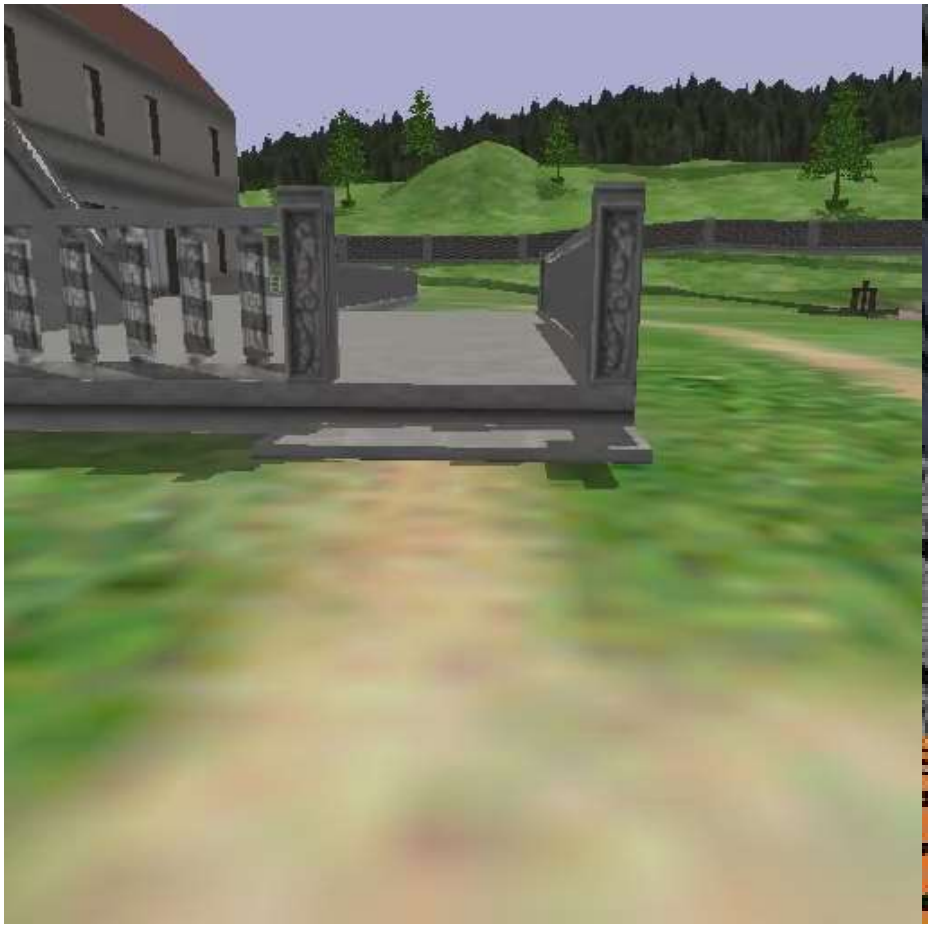
Fighting Temporal Aliasing

- Quick fix: texel snapping
  - Fix coordinate system in world space
  - Shadow maps move at integral multiples of a texel width
  - Gives up warping
  - Popping when zooming



Motivation

- Temporal aliasing
  → temporal smoothing



- Undersampling
  → shadow test confidence

Temporal Smoothing

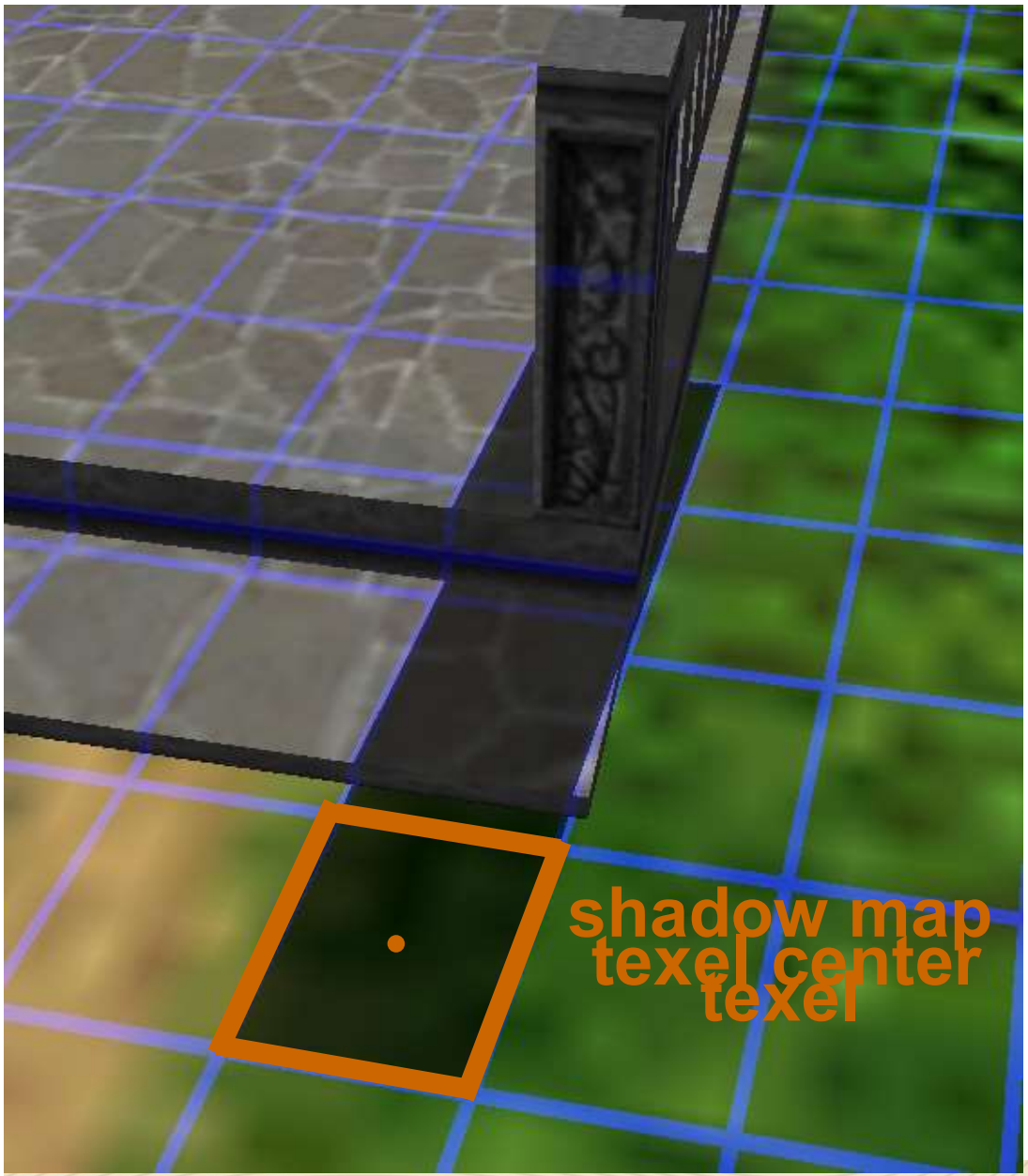
- Shadowing result of previous frame is stored in *cache*

Temporal Smoothing

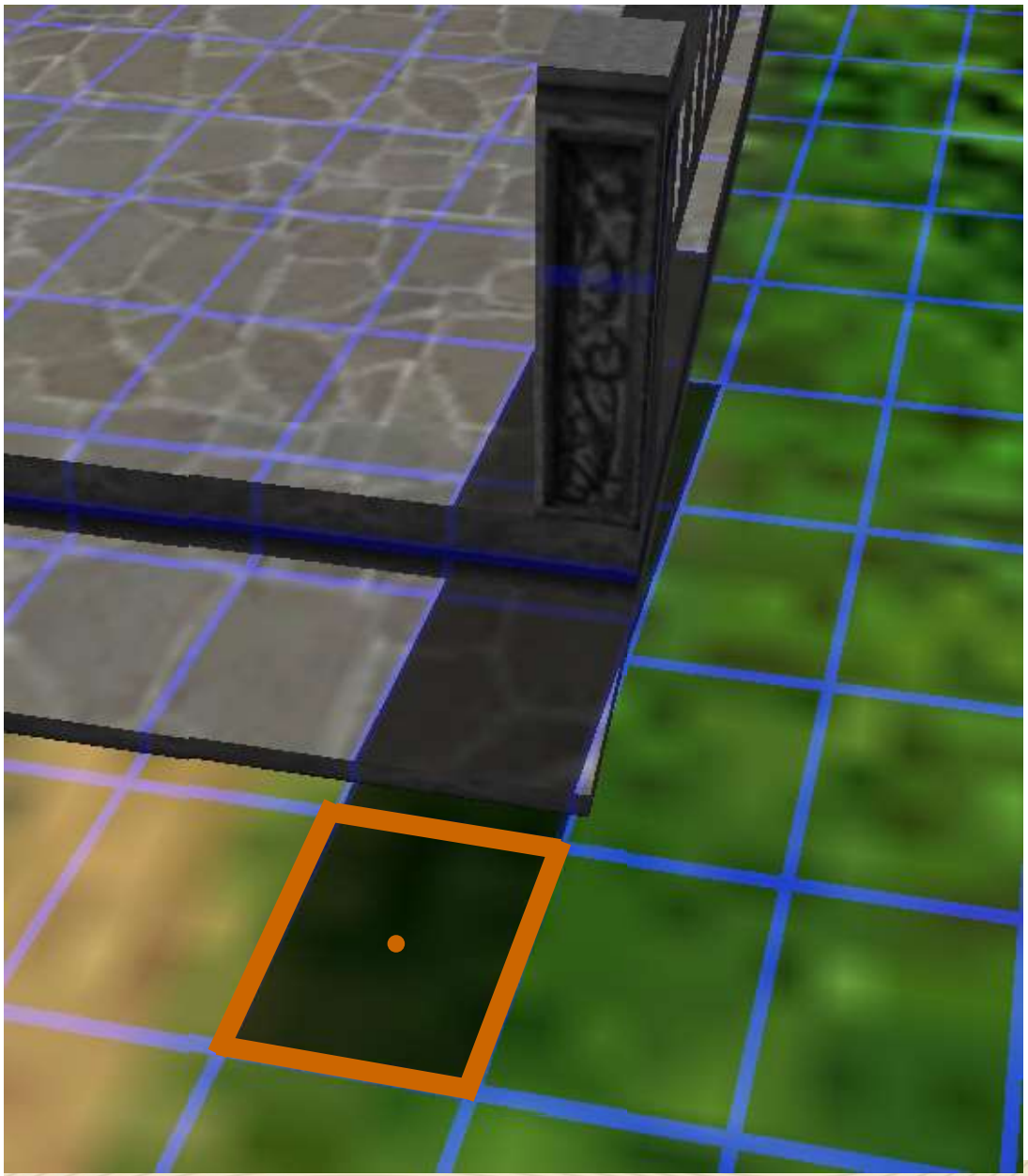
- Shadowing result for each pixel of previous frame (n−1) is stored in *cache*
- Shadowing result of current frame (n)
  cache(n) := w * s(n) + (1−w) * *cache*(n−1)
  - s(n).....*new shadow map test result, binary*
  - *cache*(n-1)..*shadow results of previous frame*
  - *cache*(n)....*shadow results of current frame*
  - w.....*weight, impact of current vs. previous*
    - Large w - fast update, weak smoothing
    - Small w - slow update, strong smoothing

Temporal Smoothing

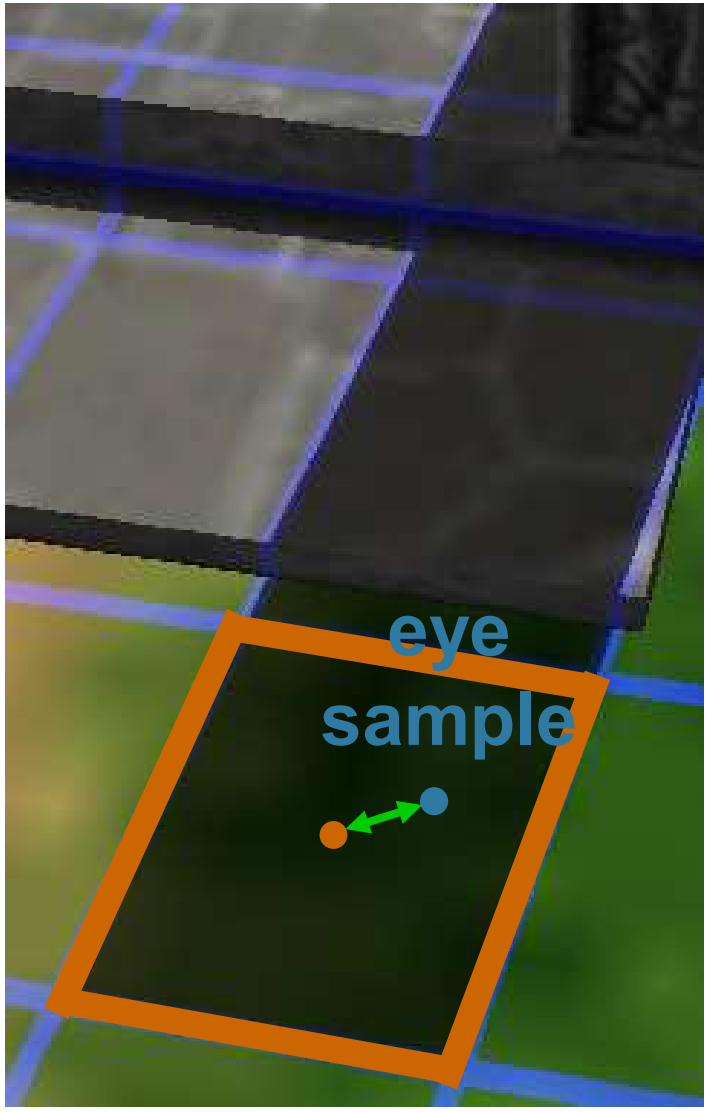
Confidence Estimation



Confidence Estimation



Confidence Estimation

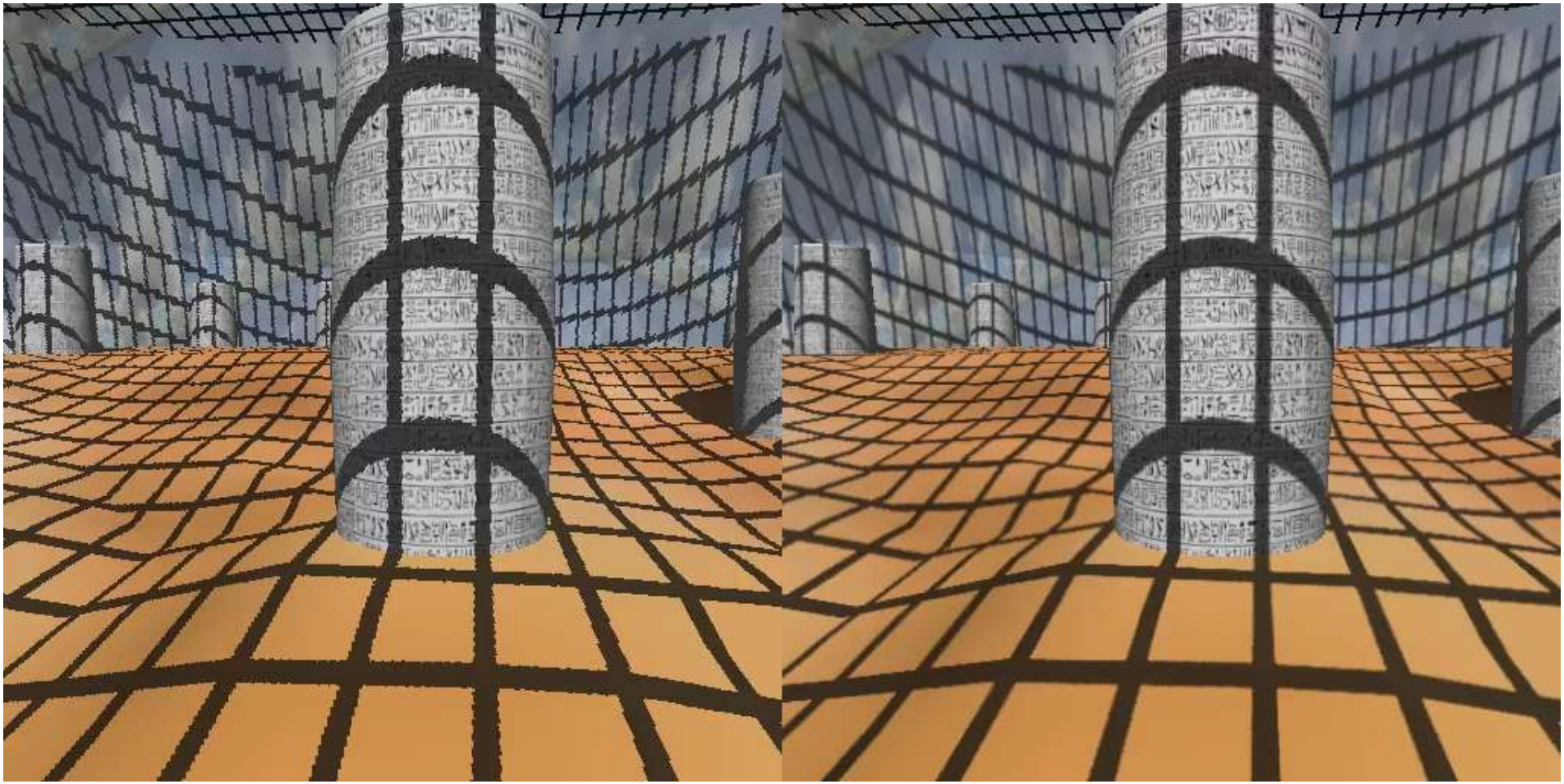


- Shadow map test probably more correct closer to texel center
- Confidence:
  conf := 1- *dist*(**eye**,**texel**)
- Greater confidence
  → greater impact

Confidence and Temporal Smoothing

- Confidence is weight for temporal smoothing

  cache(n):= **conf** * *s*(n) + (1−**conf**) * *cache*(n−1)



Hard Shadows

Fighting Undersampling - Conclusion

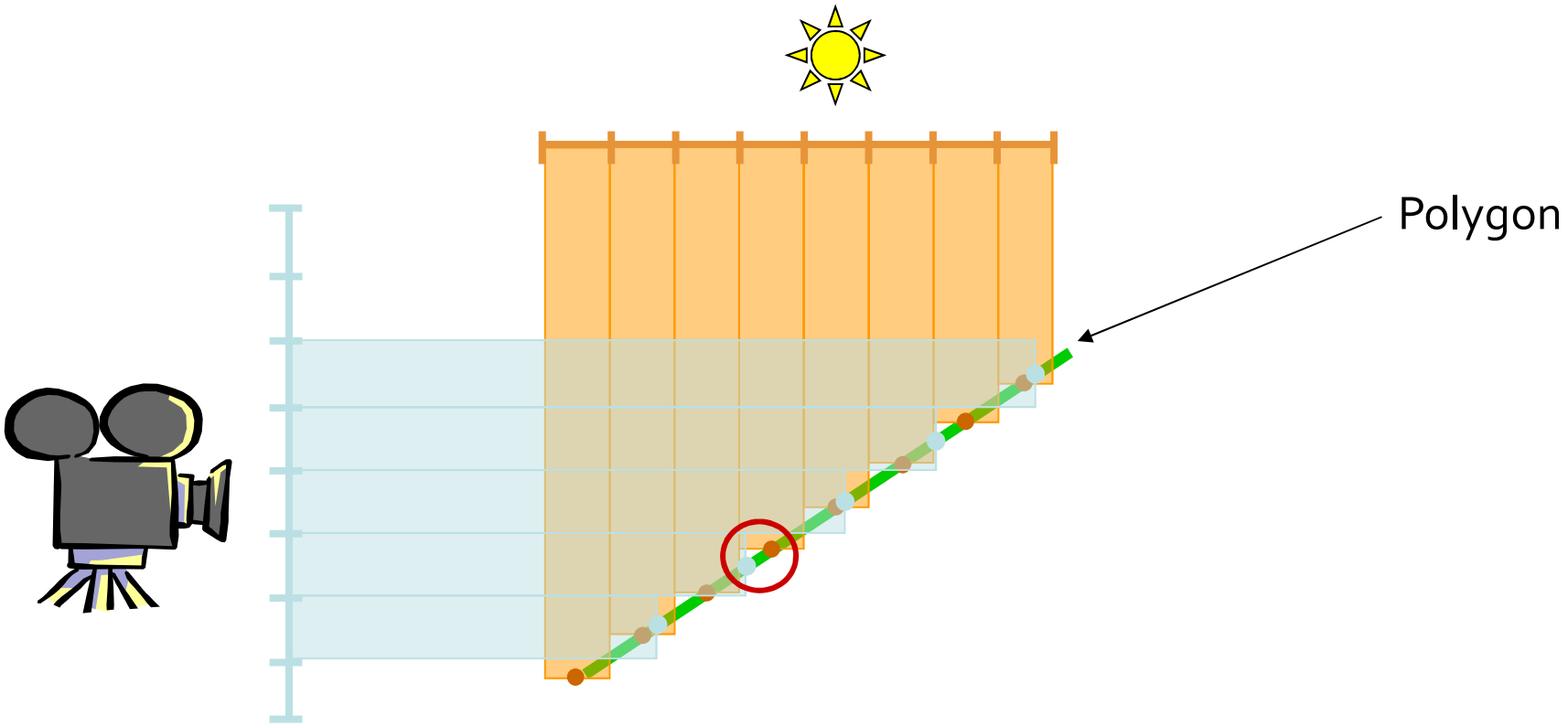
Conclusions

- Fastest speed, single shadow map: **warping**
  - Good for outdoor
- Fast speed, better quality, multiple shadow maps: **z-partitioning**
- High quality, lower speed: **adaptive partitioning**
- Reference quality, even slower: **irregular sampling**

Hard Shadows

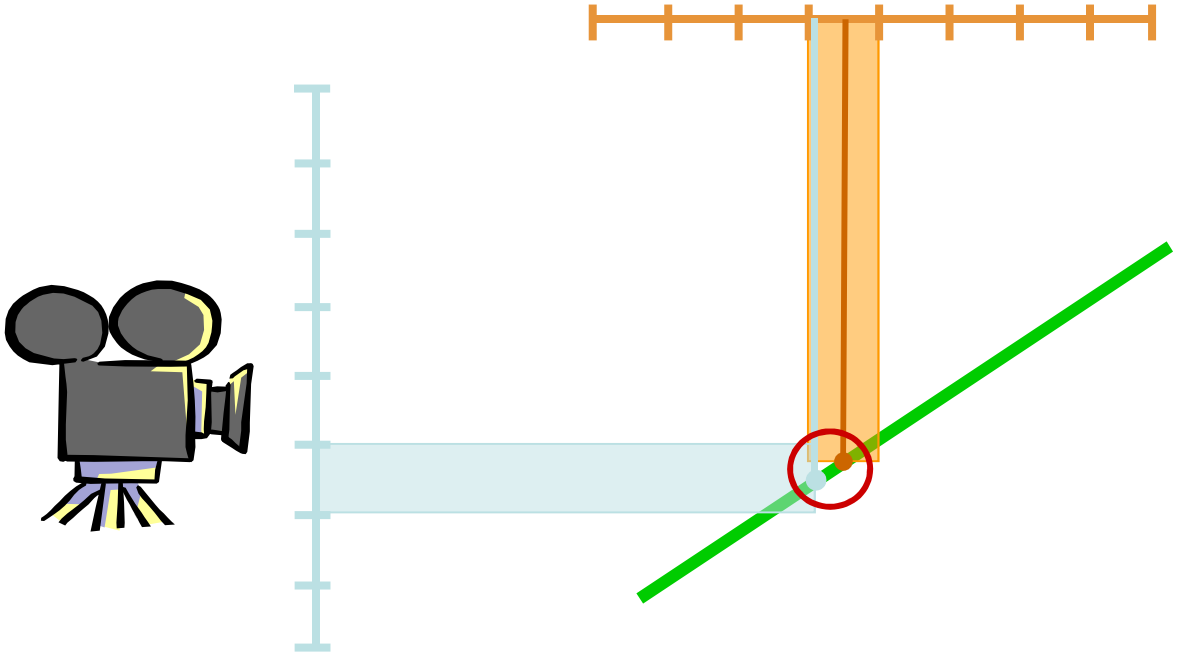
Depth Biasing

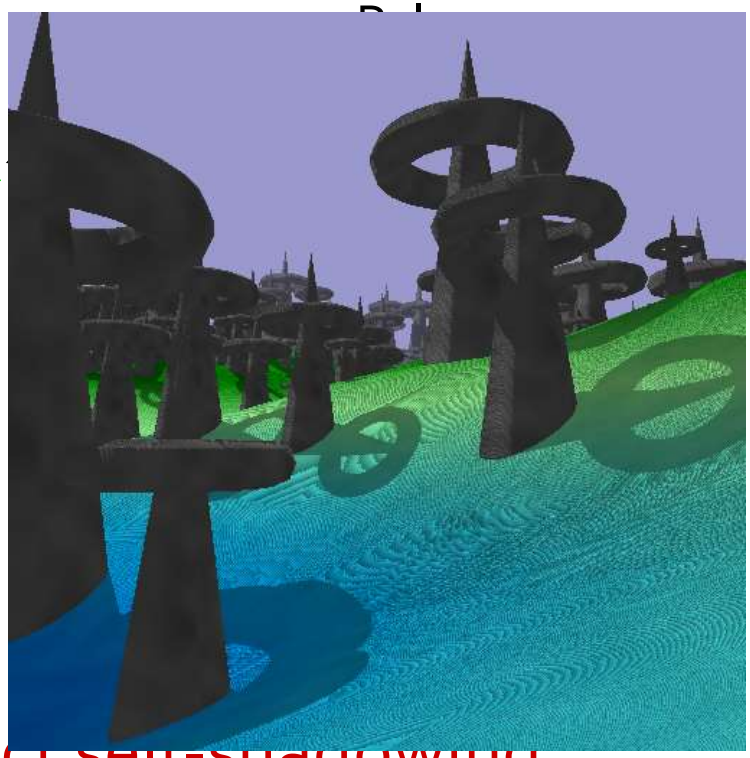
Depth Biasing – Incorrect Self Shadowing



Depth Biasing – Incorrect Self Shadowing

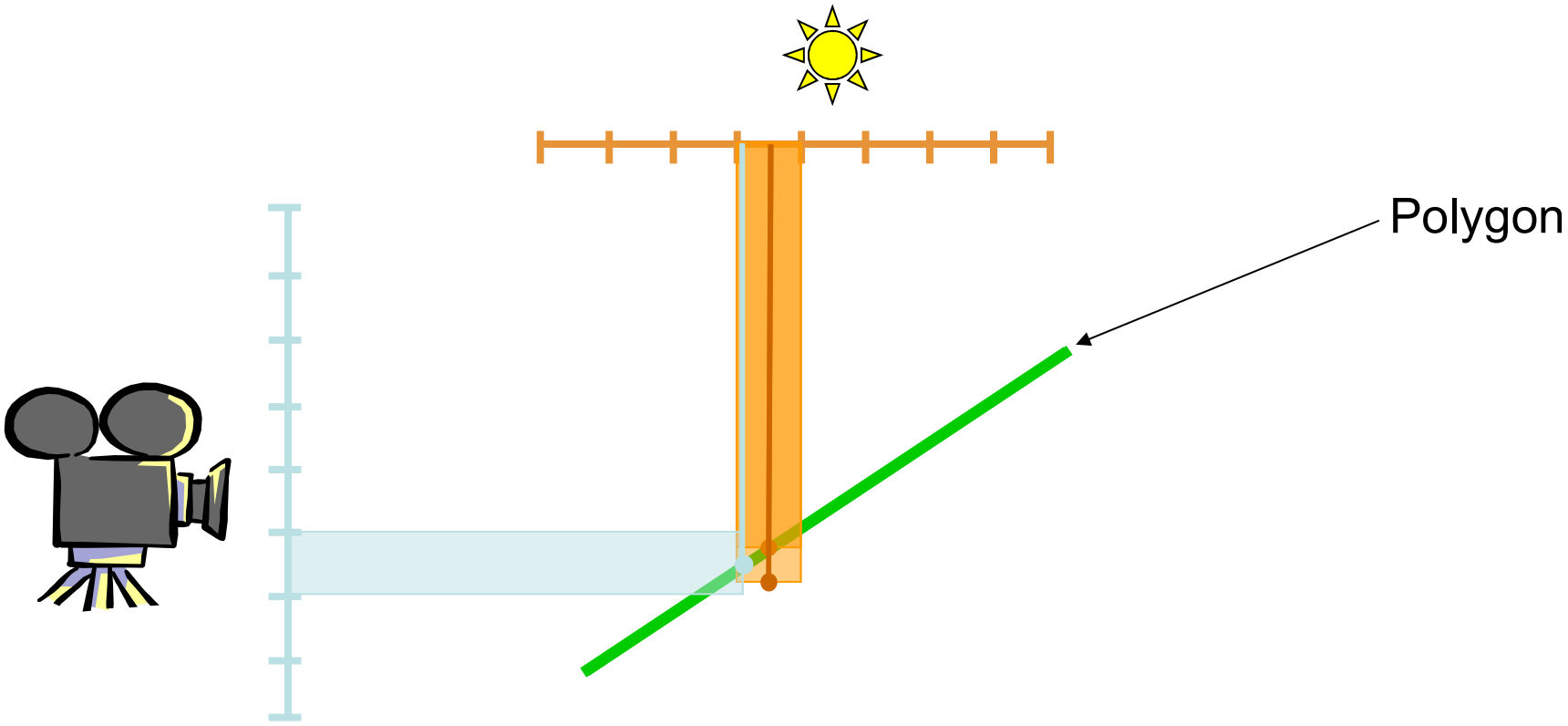
- Numerical instabilities caused by undersampling





$z_{eye} > z_{light}$ ➡ Incorrect self-shadowing

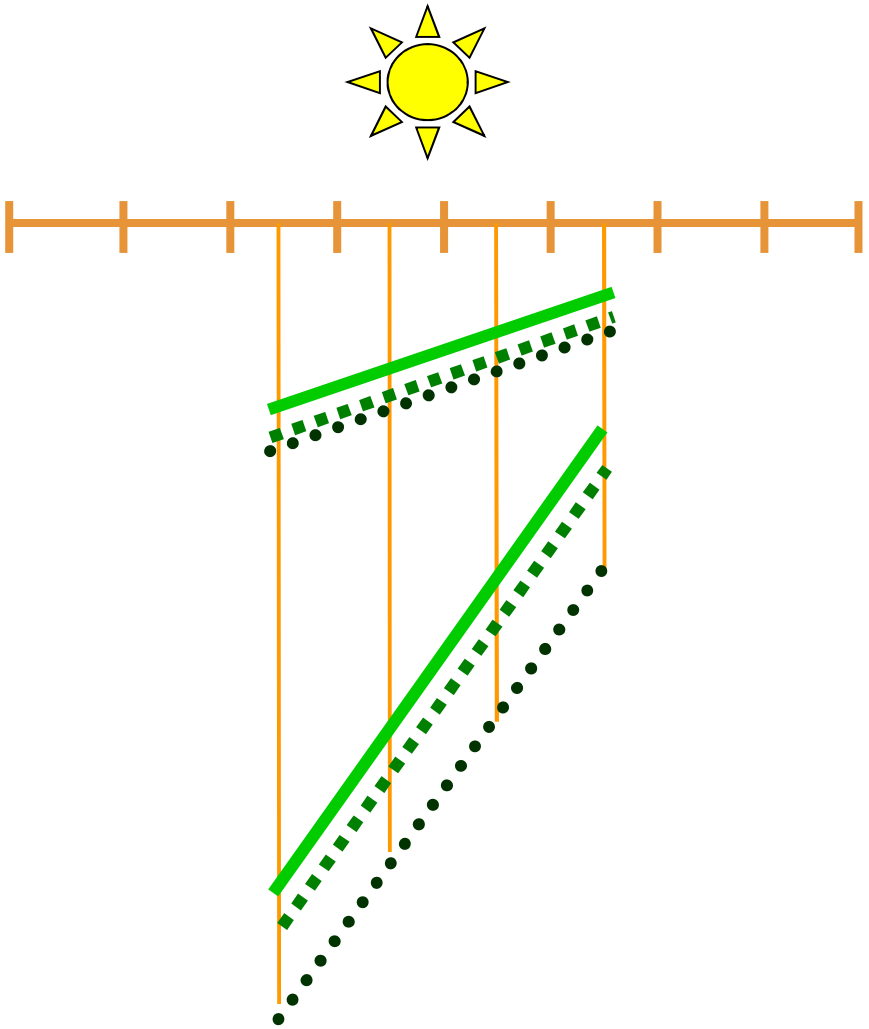
Depth Biasing

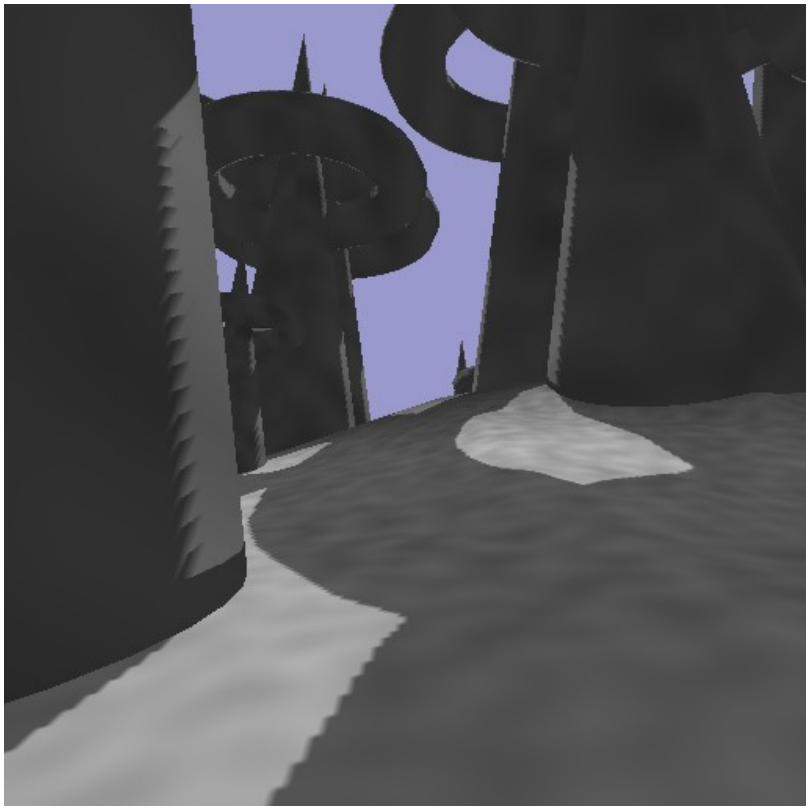


$z_{eye} > z_{light}$ ➡ Incorrect self-shadowing

$z_{eye} < z_{light}$ ➡ No self-shadowing

How to Choose the Bias?





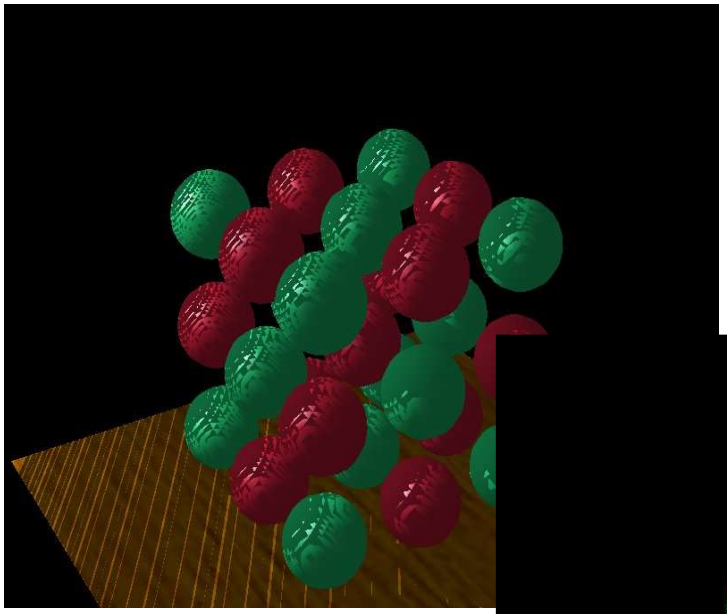
No biasing

Constant biasing

Slope-scale biasing

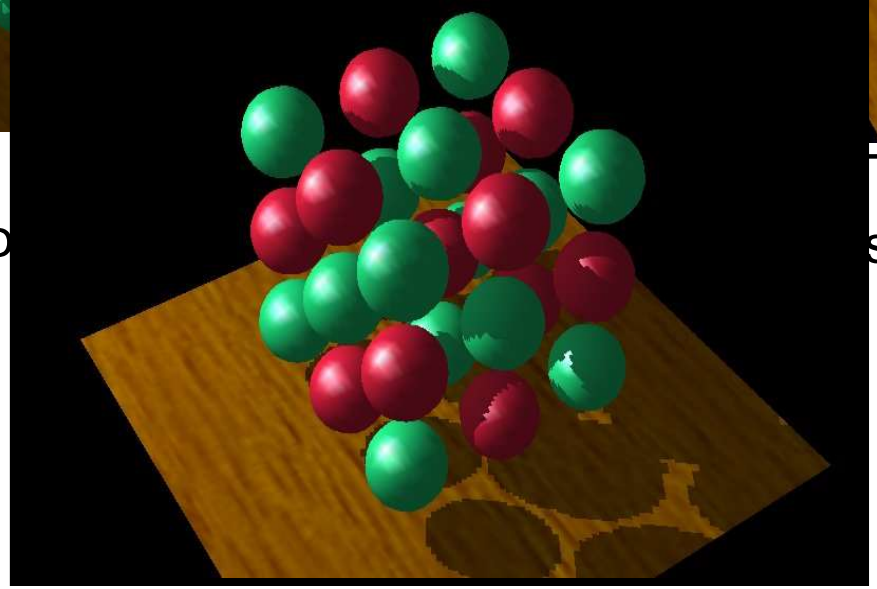
How Much Bias?

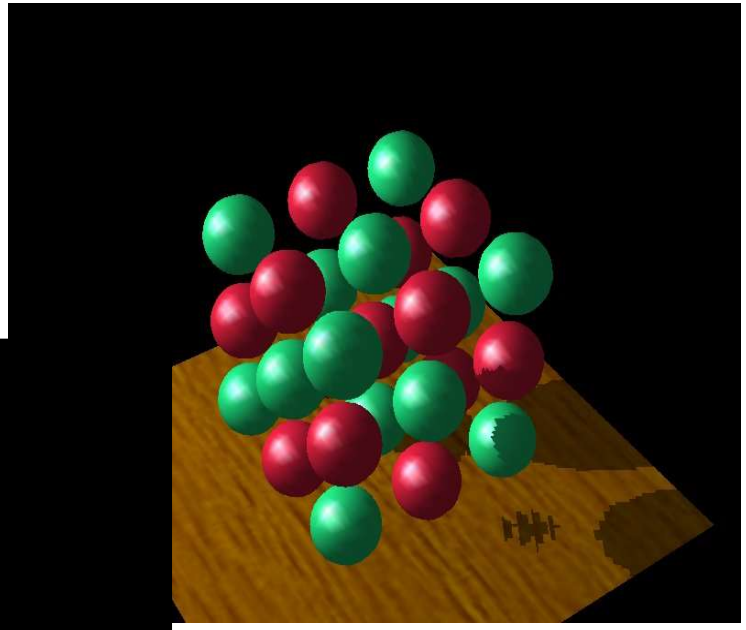
- Depends…



Too little bias, everything begins to shadow

Just right

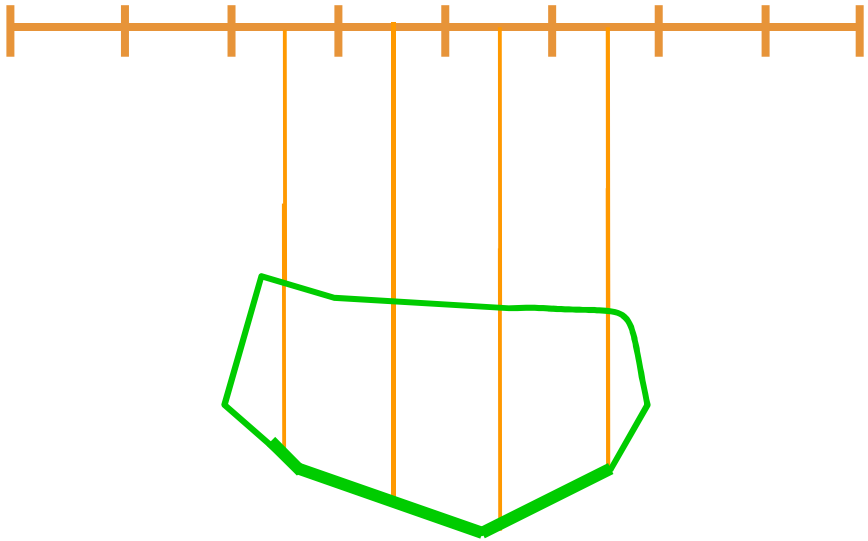


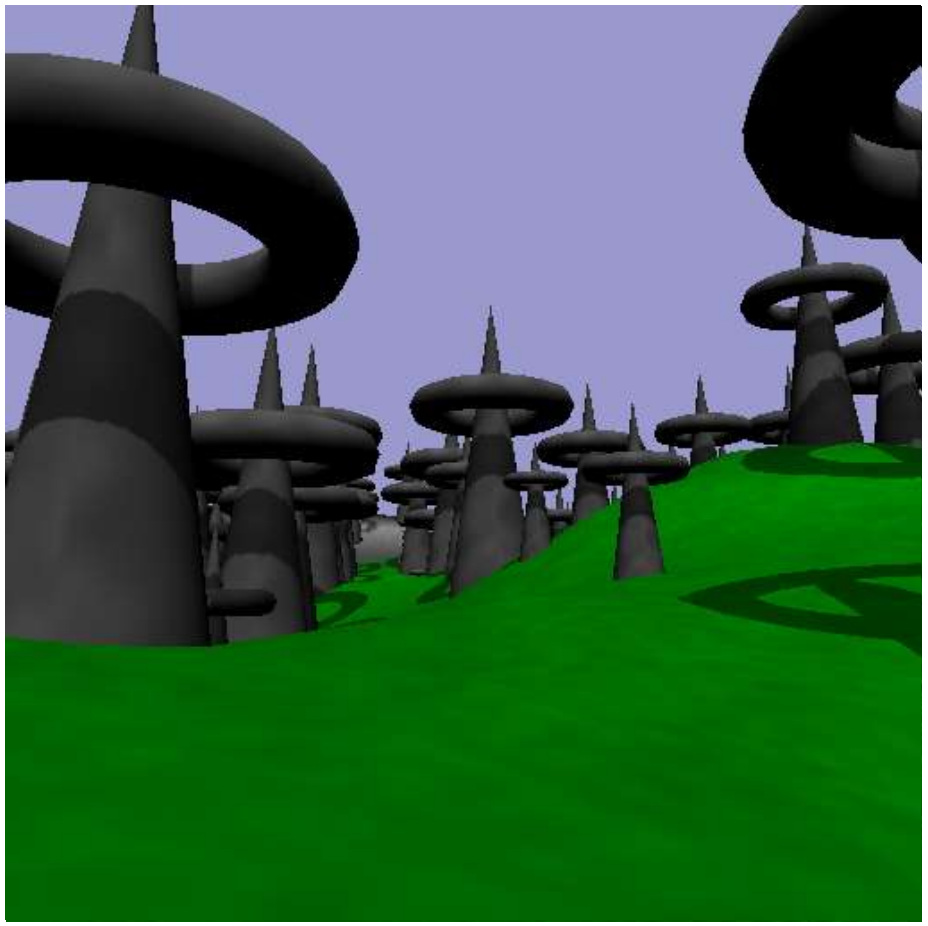


Too much bias, shadow starts too far back

Second-Depth Shadow Mapping

[Wang and Molnar 94]





- Normally: render front-side polygons into SM
- Now: render back-side polygons

Error Overview

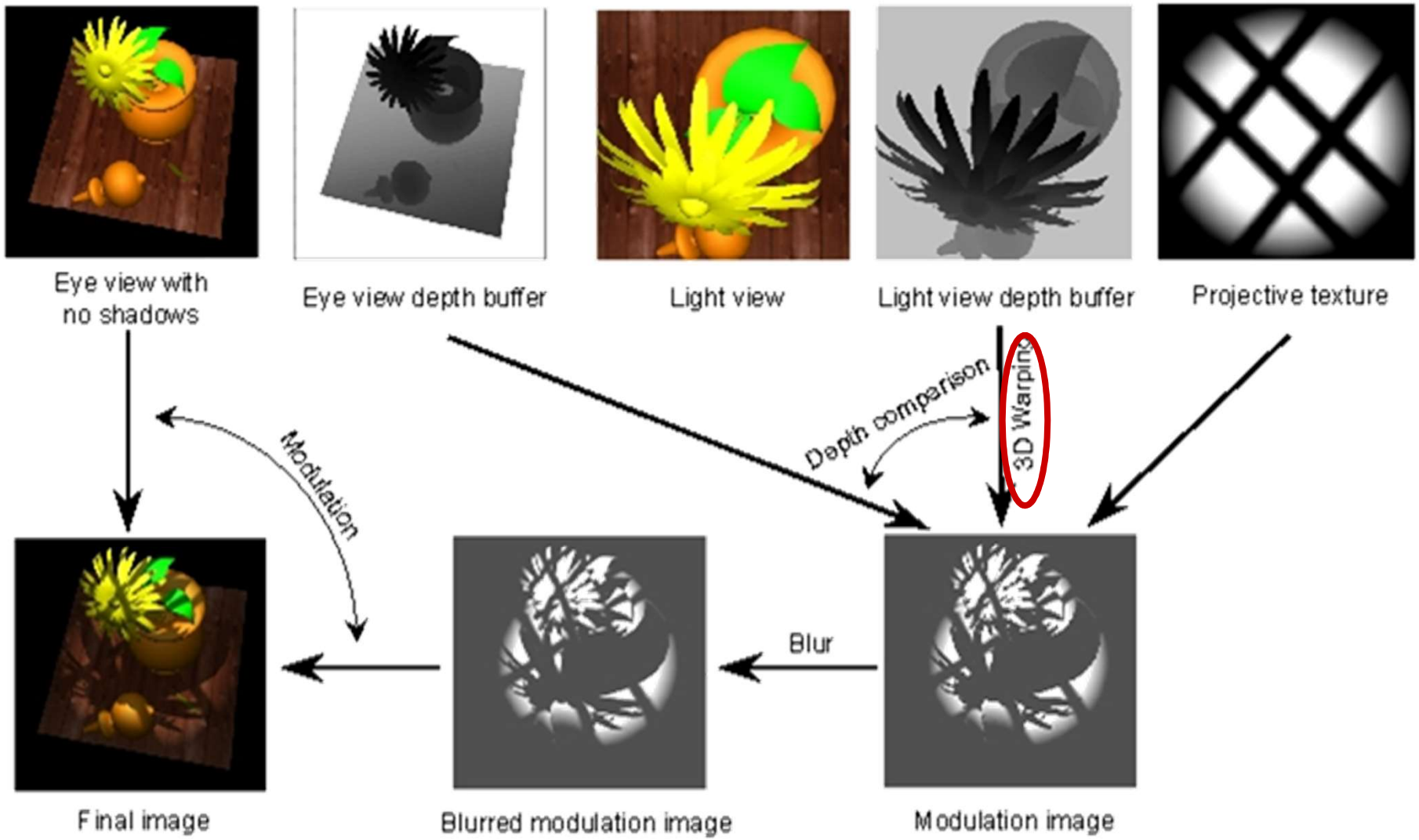
- Undersampling
  - Improve initial sampling
  - Depth Biasing
- Reconstruction error
  - Use different reconstruction algorithm (Silhouette Shadow Maps)
  - Use better reconstruction filters (PCF, PCF with Poisson disk sampling)
- Oversampling
  - Use bandlimiting filters (Convolution Shadow Maps etc., see later)

Hard Shadows

Reconstruction

Forward Shadow Mapping

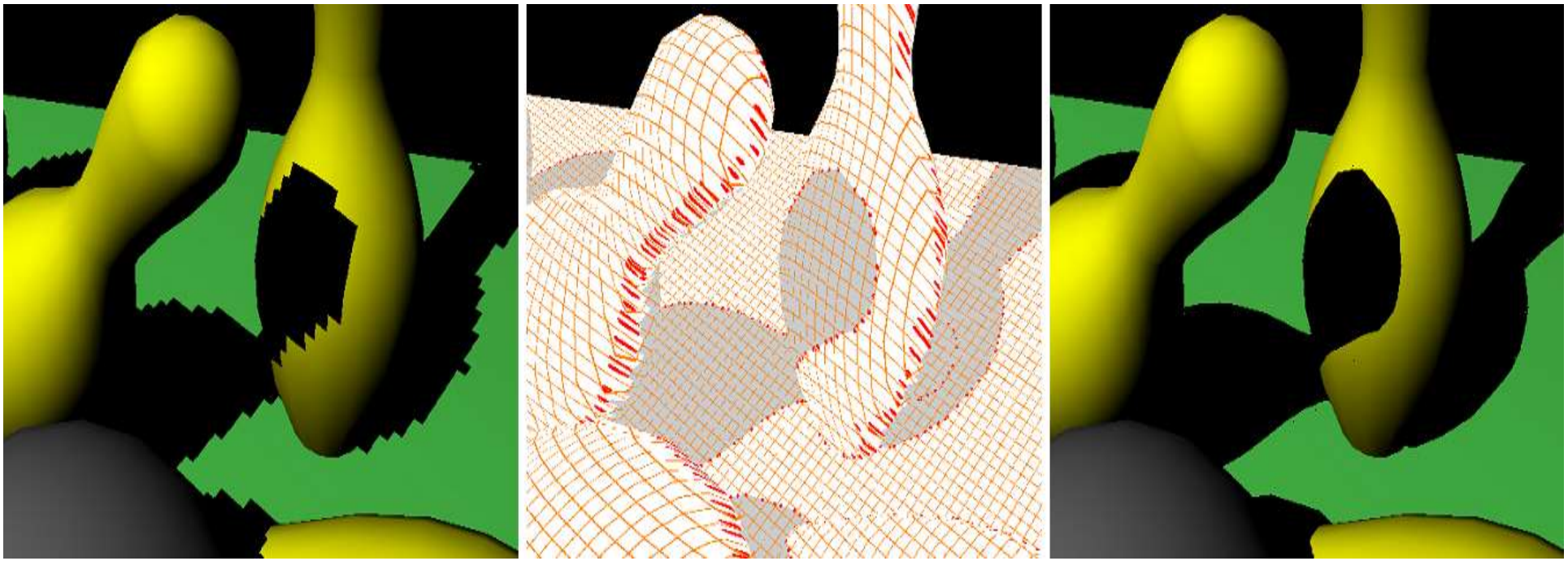
[Zhang 1998]



Shadow Silhouette Maps

[Sen et al. 2003]

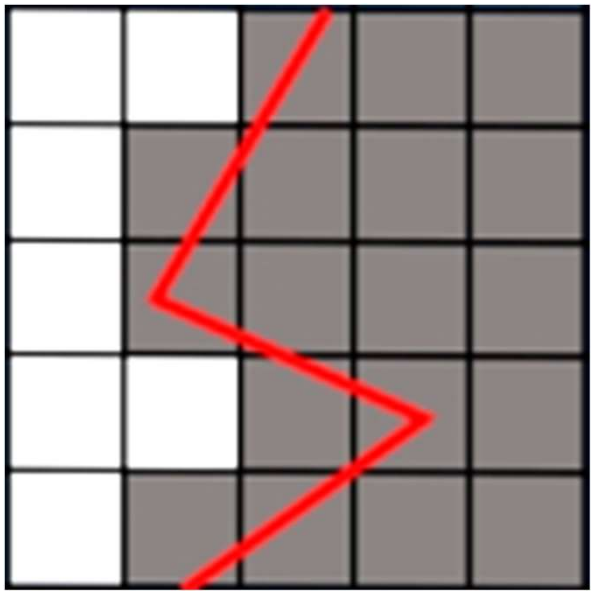
- Use a better silhouette approximation
- Store additional information of shadow edge

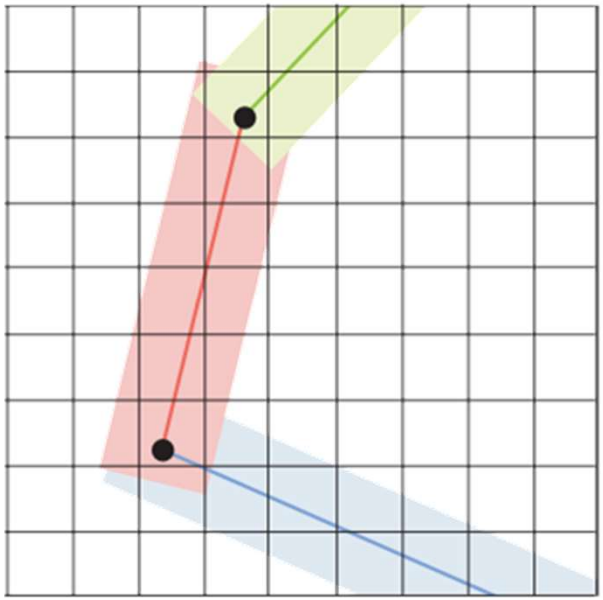


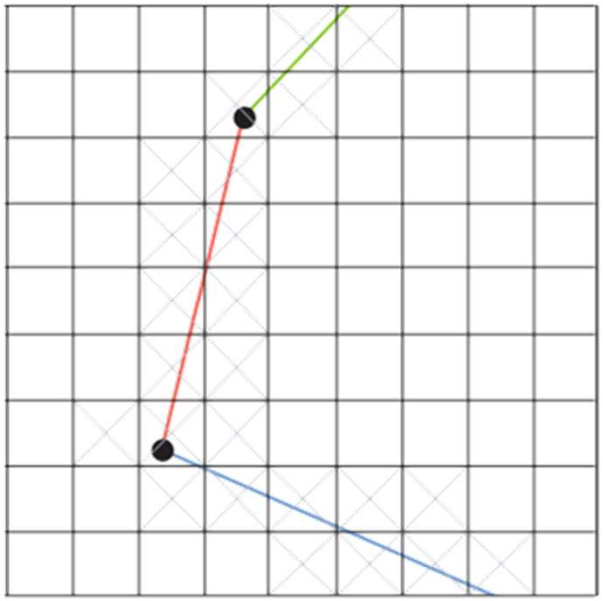
Shadow Silhouette Maps

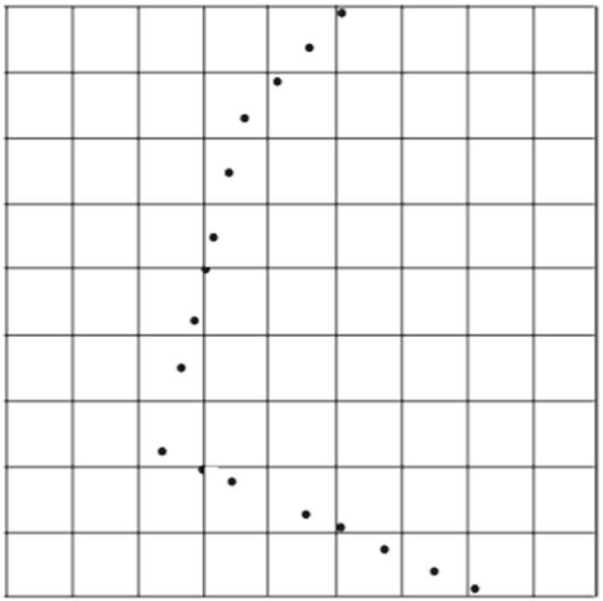
[Sen et al. 2003]

1. Create shadow map
2. Find silhouette edges
3. Rasterize silhouettes
   a. Find points that lie on silhouette edges
   b. Store such points into silhouette map





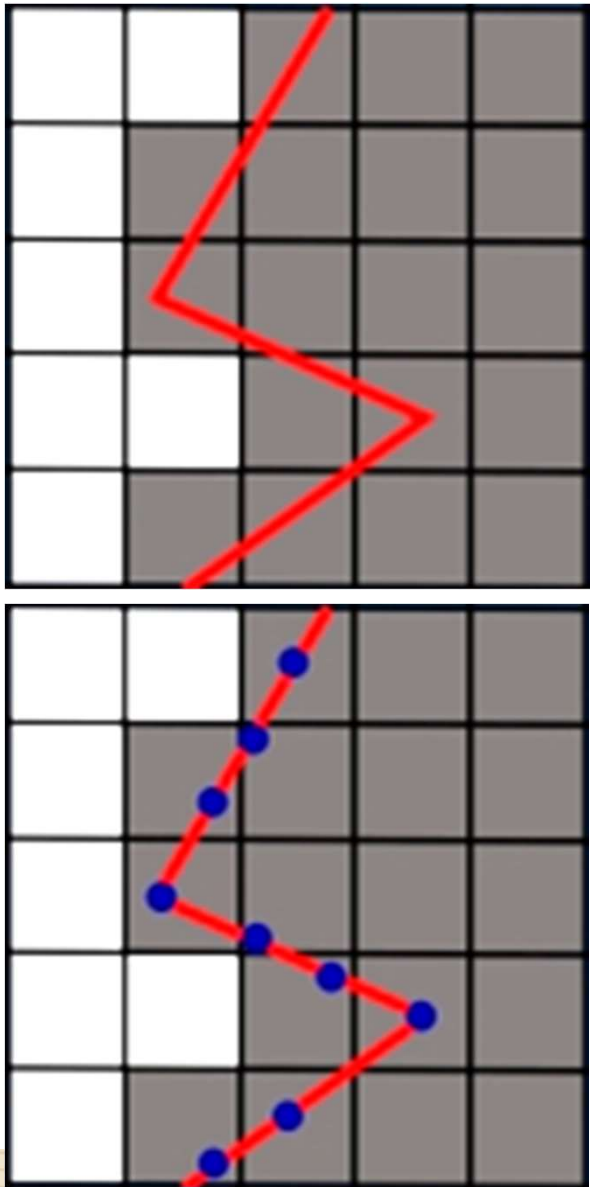




Shadow Silhouette Maps

[Sen et al. 2003]

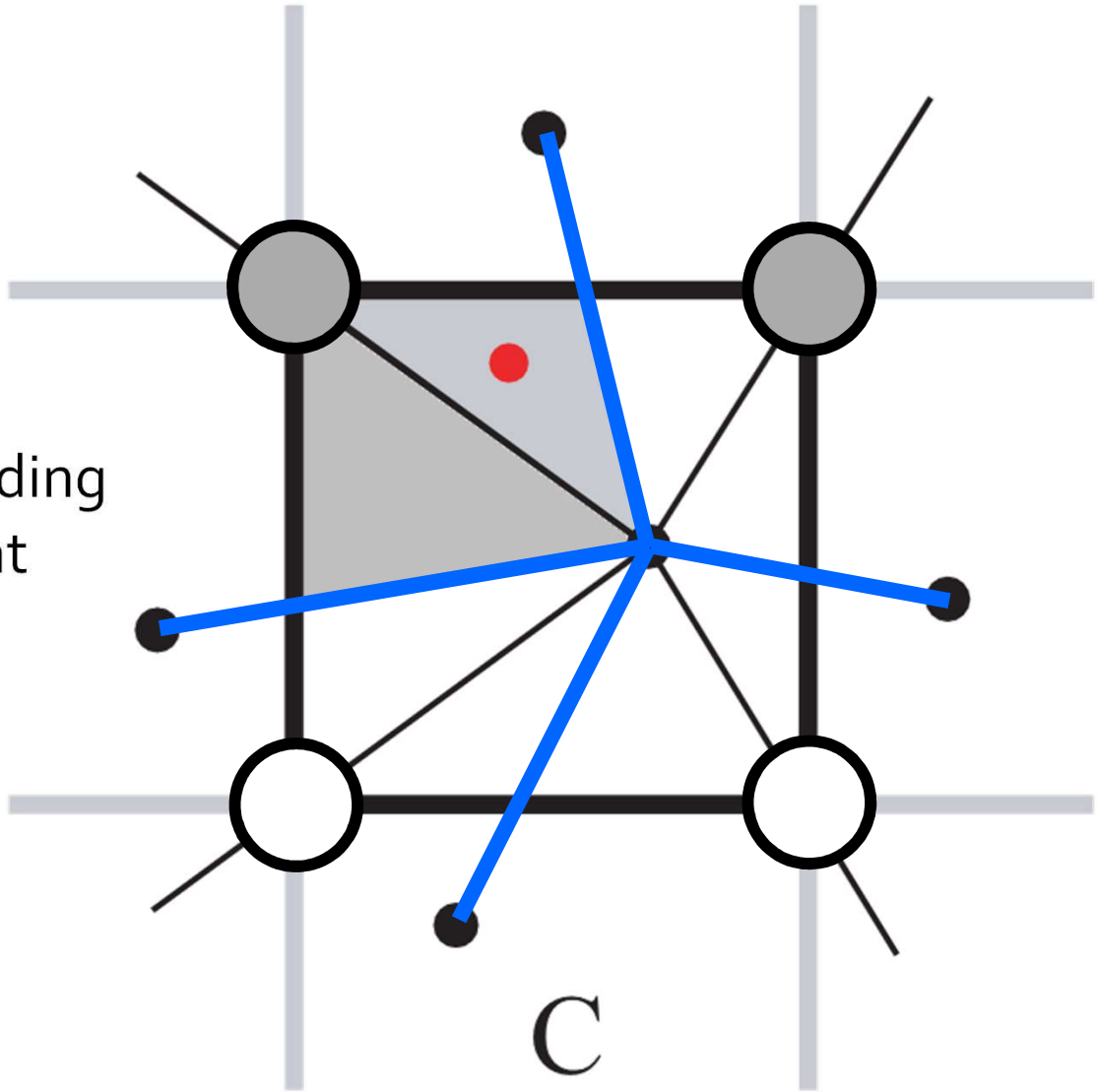
1. Create shadow map
2. Find silhouette edges
3. Rasterize silhouettes
   a. Find points that lie on silhouette edges
   b. Store such points into silhouette map
4. Compute shadows
   a. Non-silhouette pixels use standard shadow map
   b. Silhouette pixels use silhouette map



Shadow Silhouette Maps

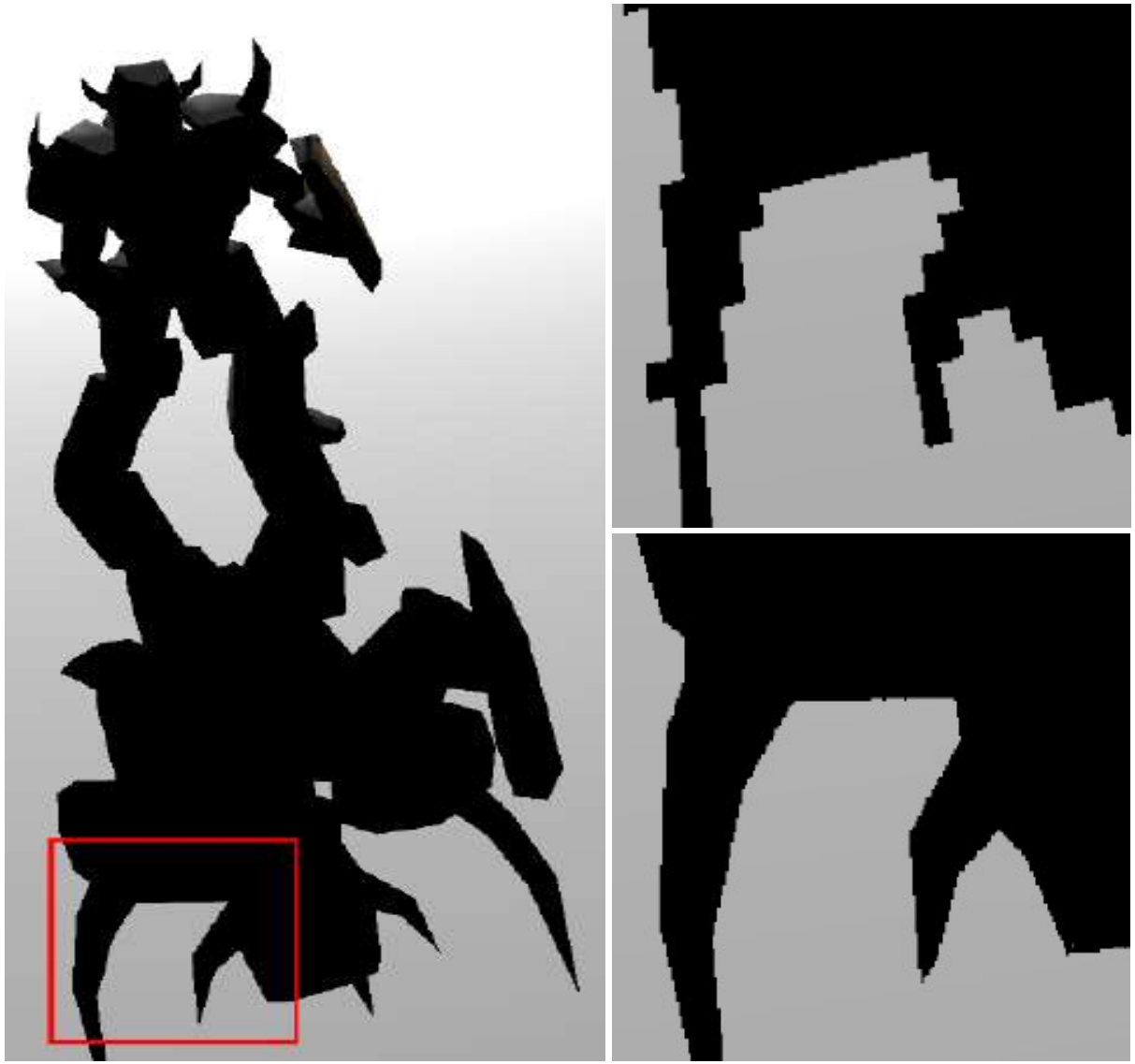
[Sen et al. 2003]

- Fragment
- Silhouette points
  - 1 interior + 4 external
- Create quadrants
- Shade fragment according to shadow test result at corner point



Shadow Silhouette Maps

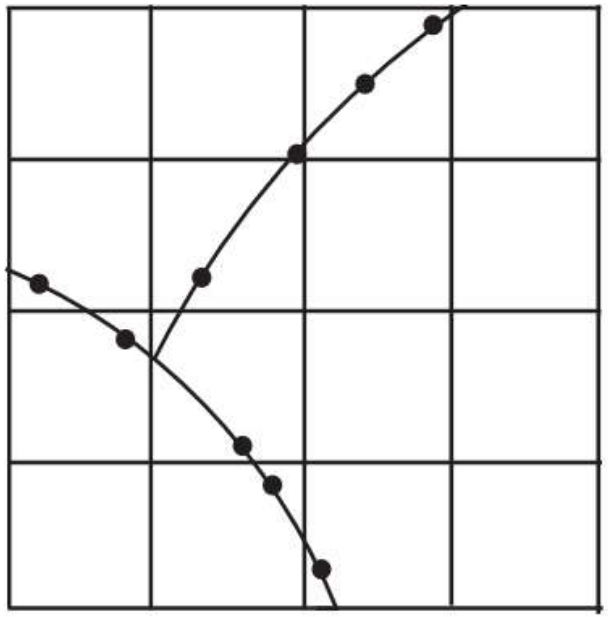
[Sen et al. 2003]

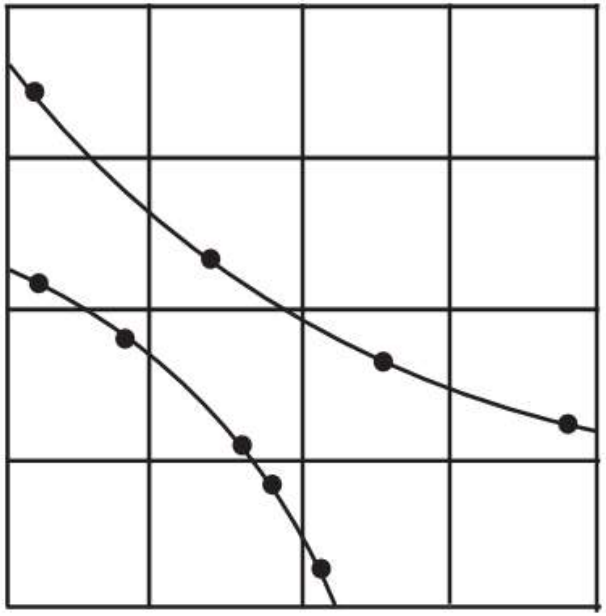


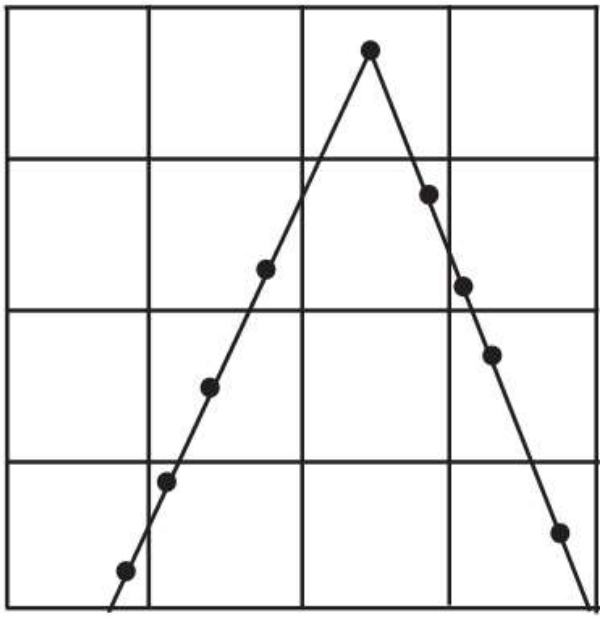
shadow map

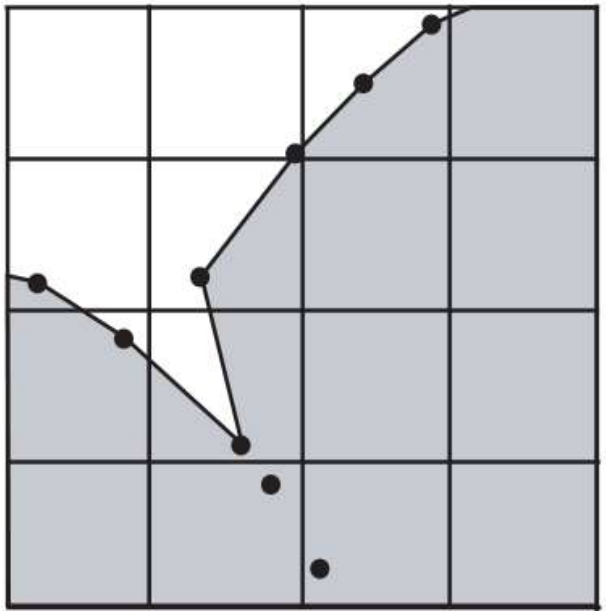
silhouette map

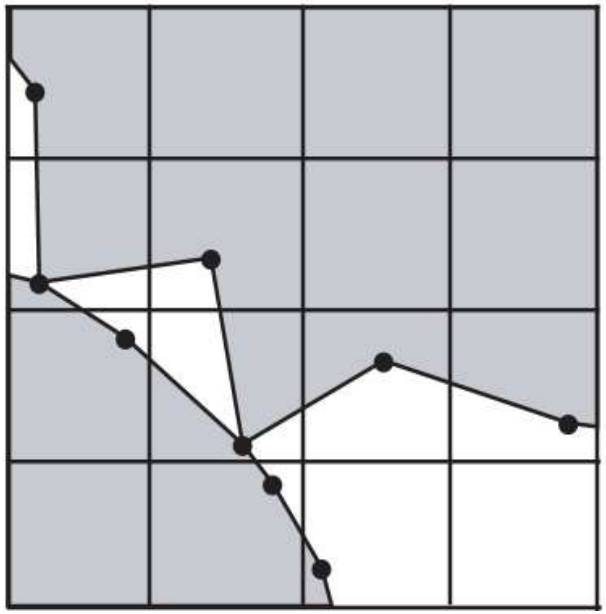
Shadow Silhouette Maps

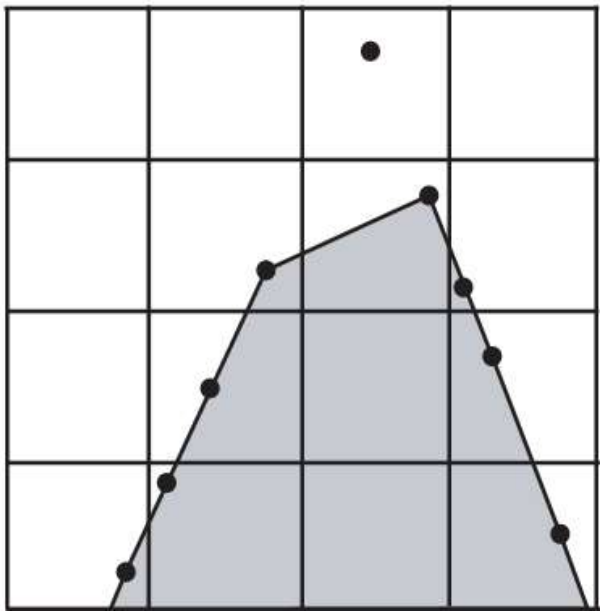


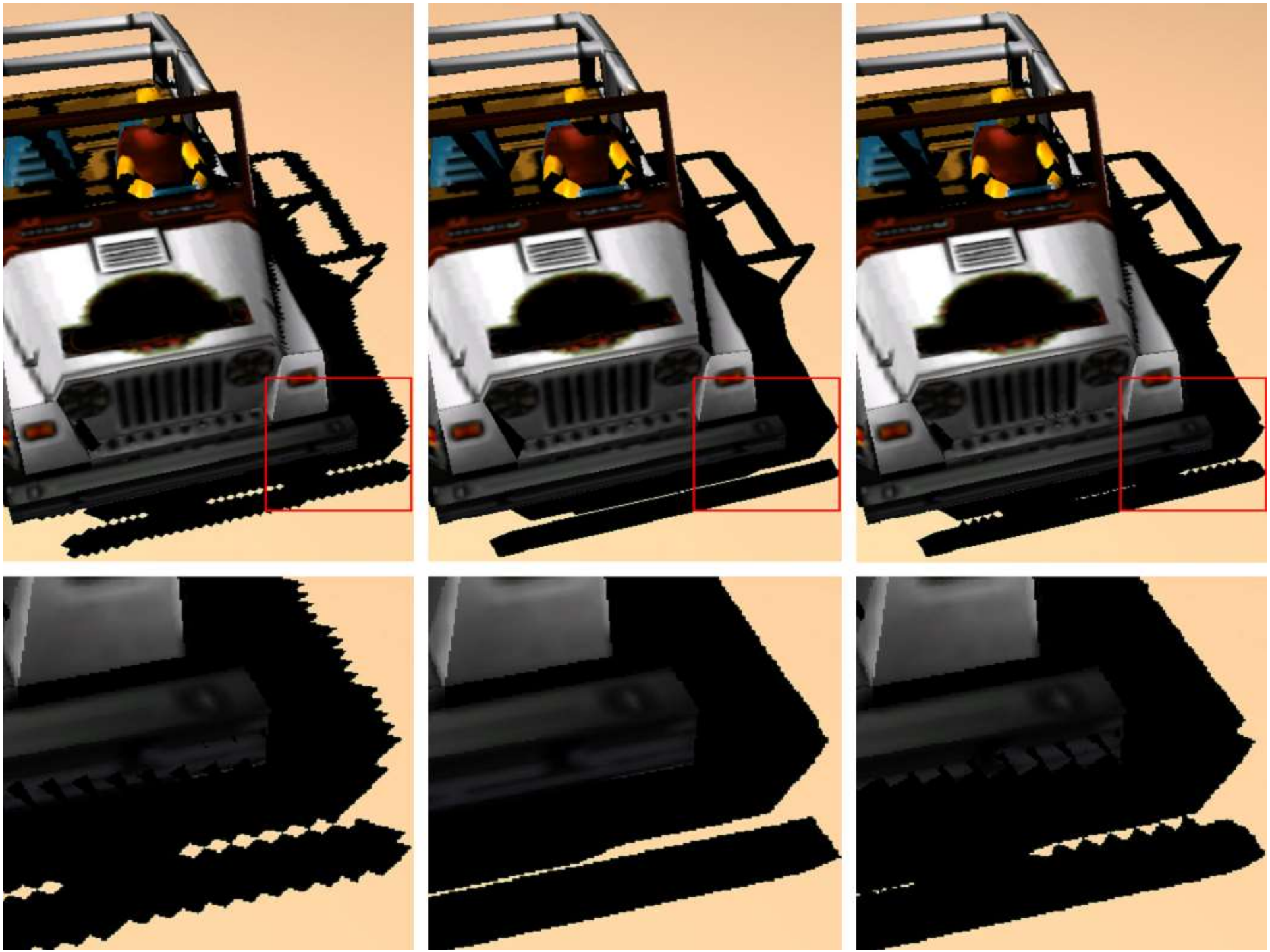








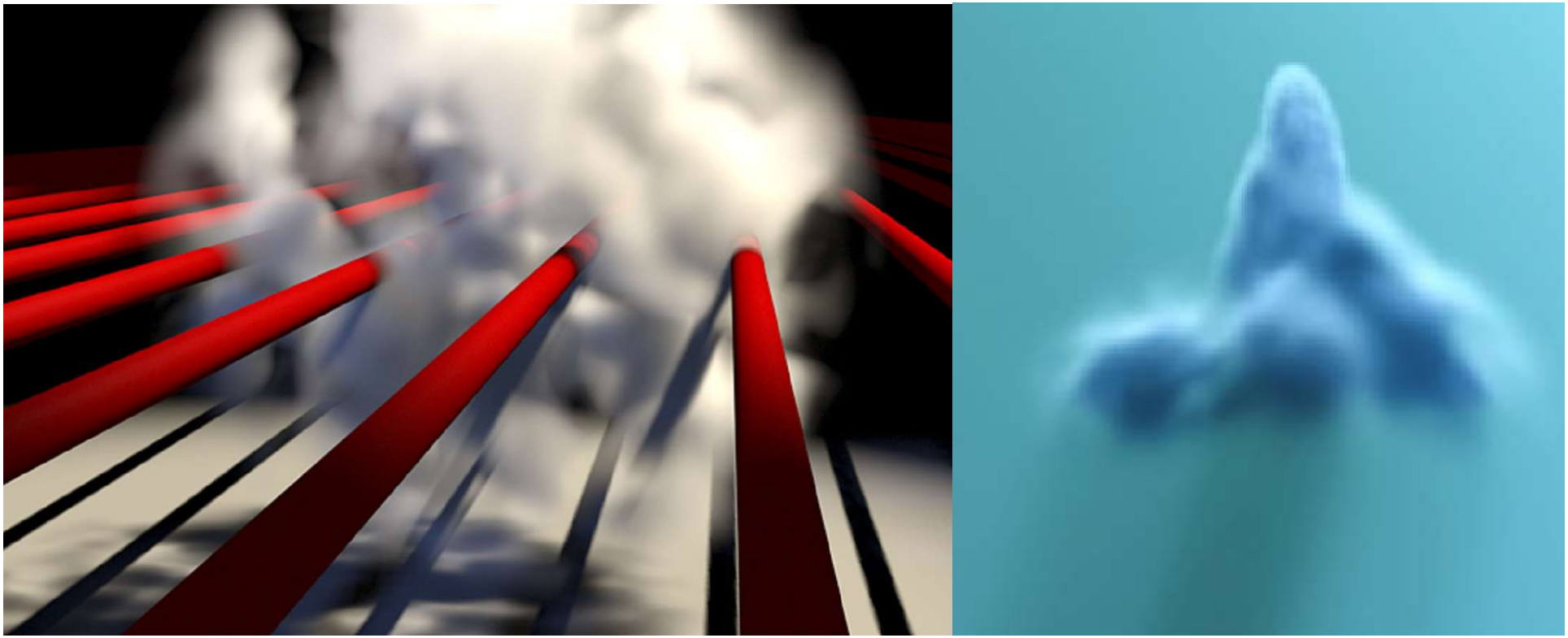




Deep Shadow Maps

[Lokovic and Veach 2000]

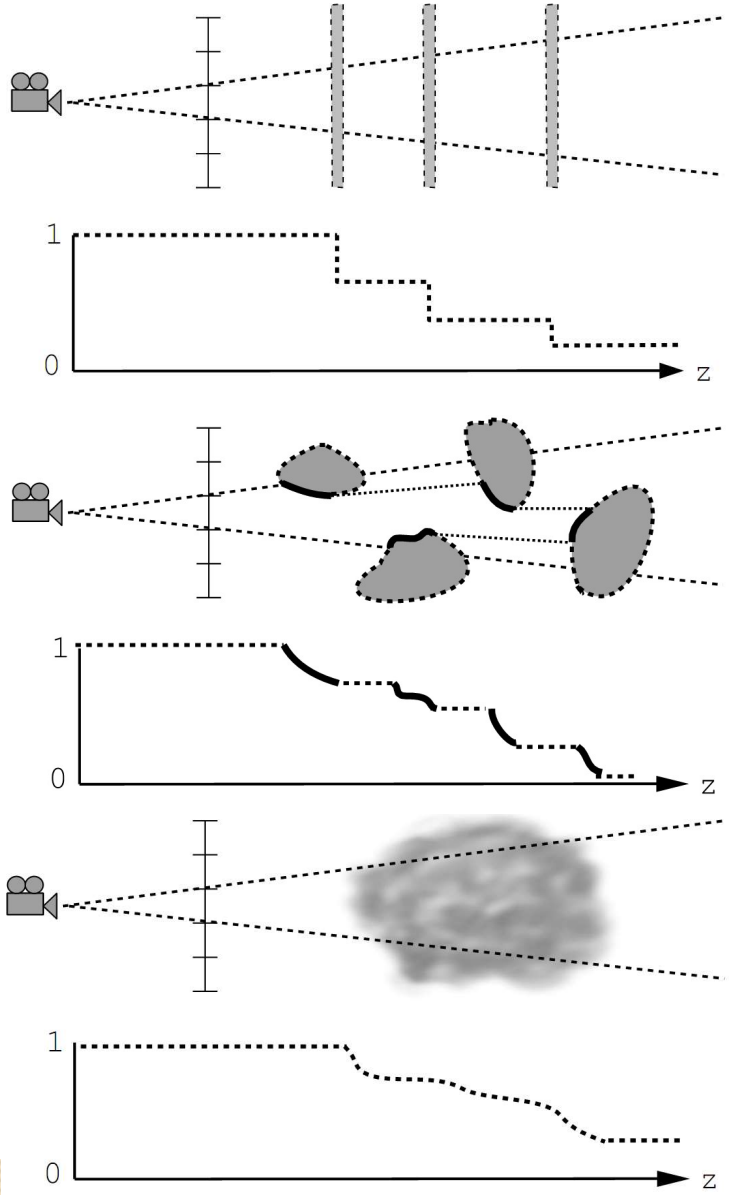
- High quality shadows
- Smoke, hair, fur, …
- Stores function for each texel



Deep Shadow Maps

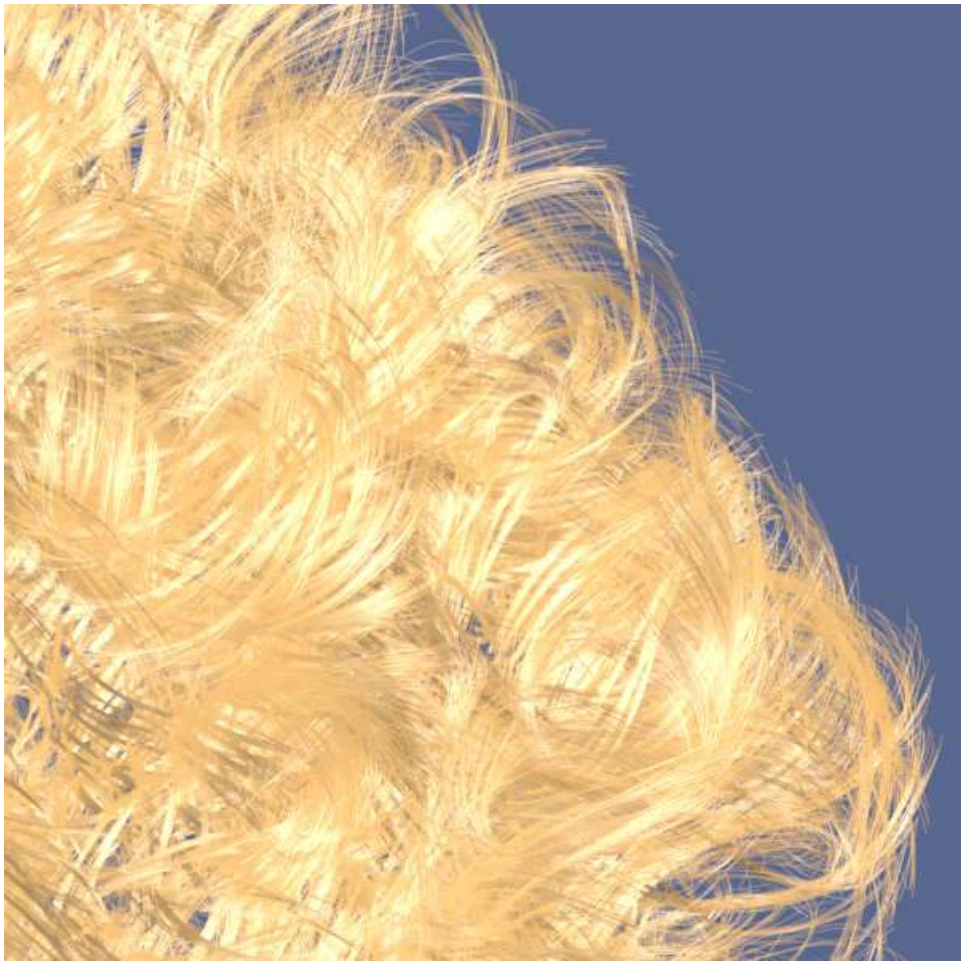
[Lokovic and Veach 2000]

- Transmittance function
  - Calculate reduction of light
- Store compressed version
  - Piecewise linear
  - Fixed number of function samples – store in array



Deep Shadow Maps

[Lokovic and Veach 2000]



without self-shadowing

with self-shadowing