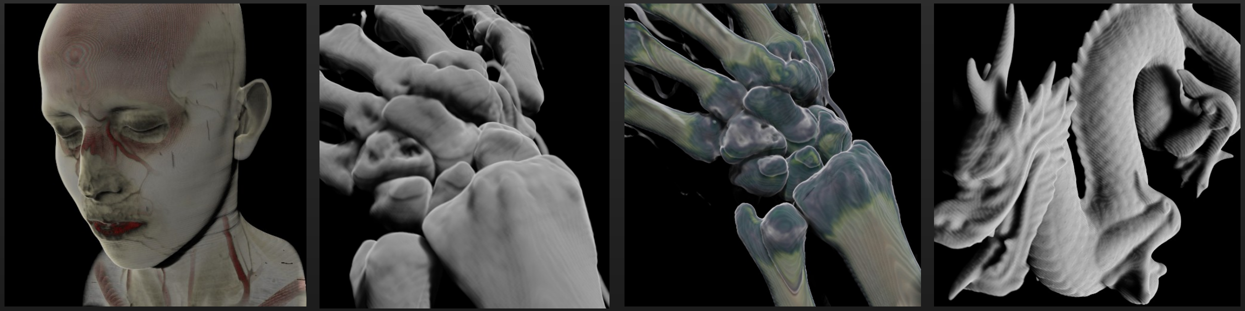
GPU-Accelerated Deep Shadow Maps for Direct Volume Rendering

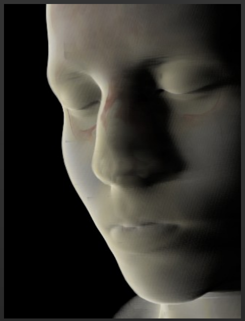


Markus Hadwiger, Andrea Kratz, Christian Sigg*, Katja Bühler

VRVis Research Center, Vienna
****ETH Zurich***

Motivation

High-quality shadows for GPU ray-casting



- Ray-casting
  - Flexibility (adaptive sampling rates)
  - Efficiency (early ray termination, empty space skipping)
  - Image quality

Shadows in volume rendering

- Pre-computed shadow volume [Behrens and Ratering '98]
- Half-angle slicing [Kniss et al. '02]

Image taken from [Kniss et. al 02]

Motivation

Deep shadow maps [Locovic and Veach '00]

- DSMs unify volumes and geometry

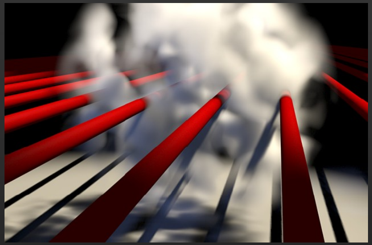
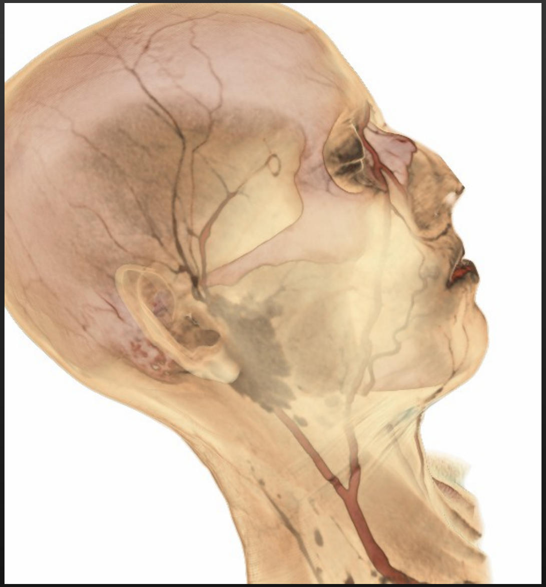


Image taken from [Locovic and Veach '00]

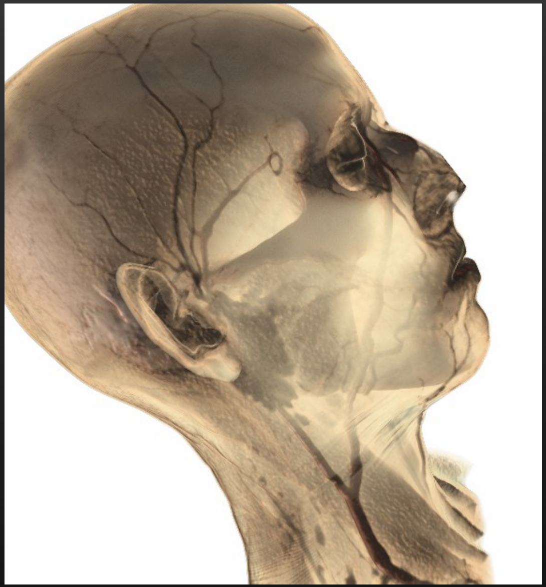
- Opacity shadow maps [Kim and Neumann '01]
  - Regular sampling
- A self-shadowing algorithm for dynamic hair using density clustering [Mertens et al. '04]
  - Irregular sampling with restricted number of bins

Motivation

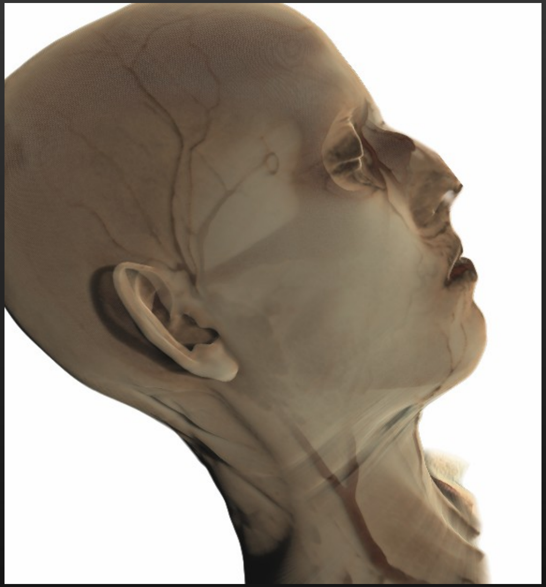
Increased realism



- Unshaded volume rendering



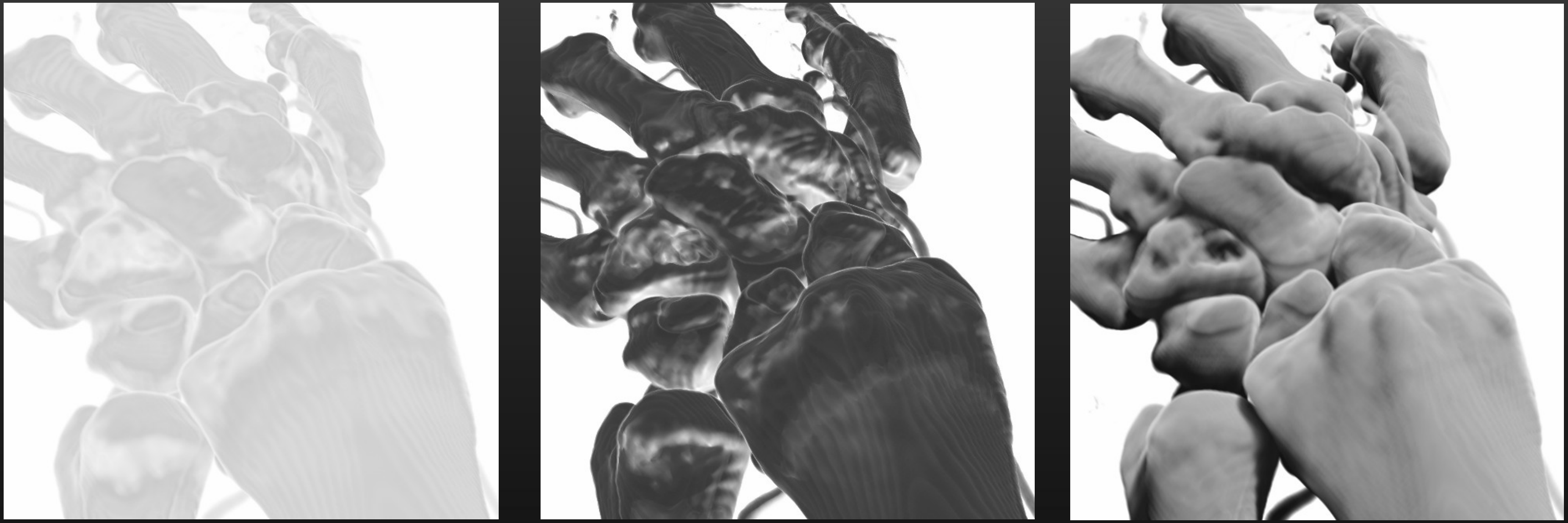
- Gradient-shaded volume rendering



- Volume rendering with shadows

Motivation

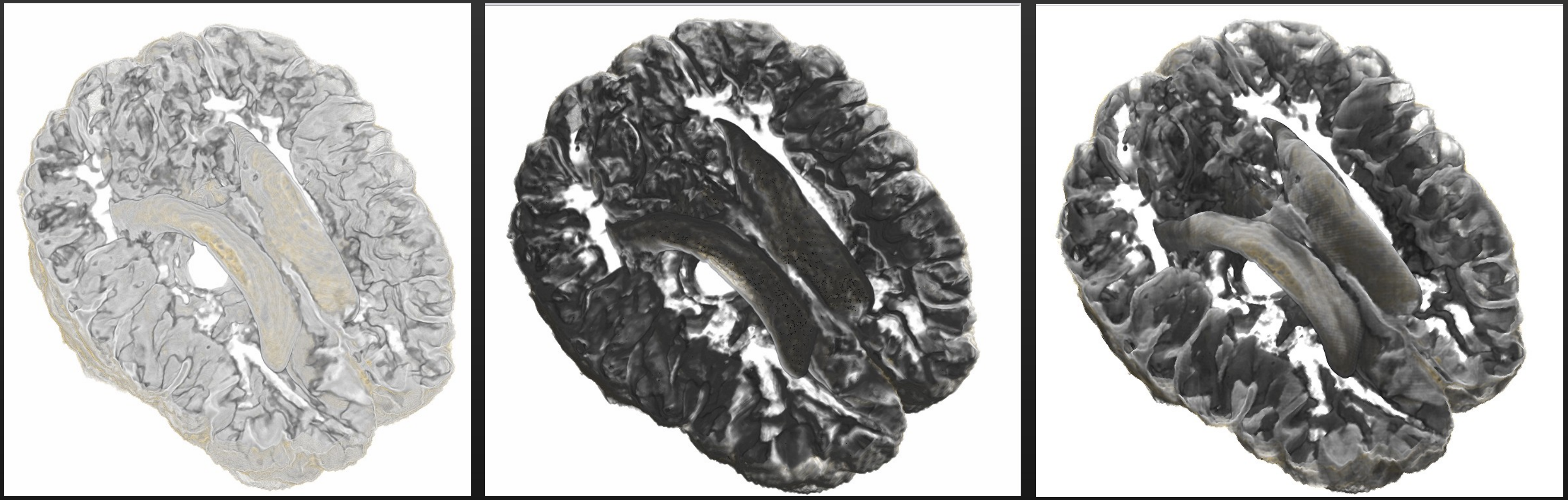
Provide intuitive visual cues for depth and shape



- Improved depth perception
- More contrast
- Meaningful images even with simple transfer functions

Motivation

Enhance overall quality of images

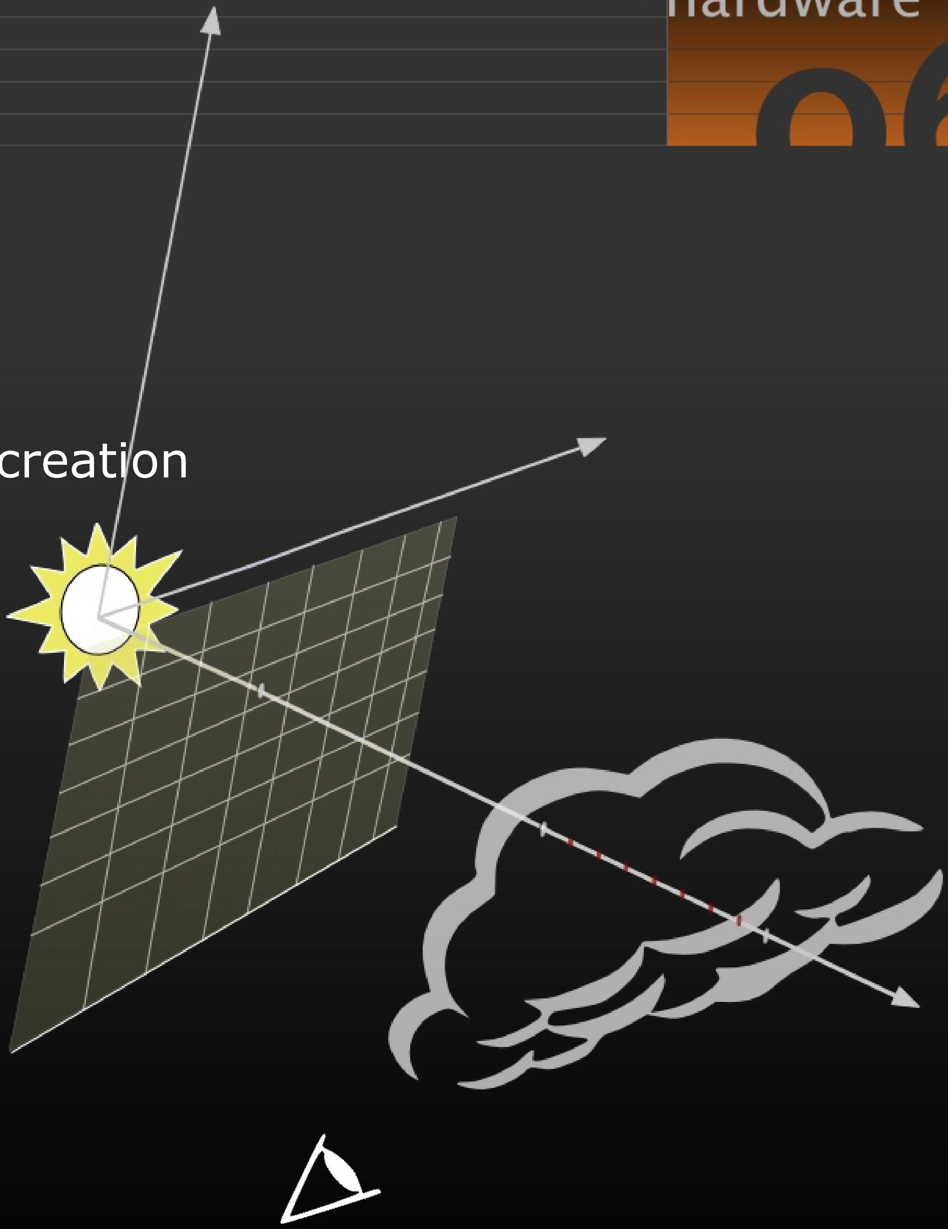


- Gradient-less
- Avoid noise in homogeneous areas
- Reveal fine and detailed structures

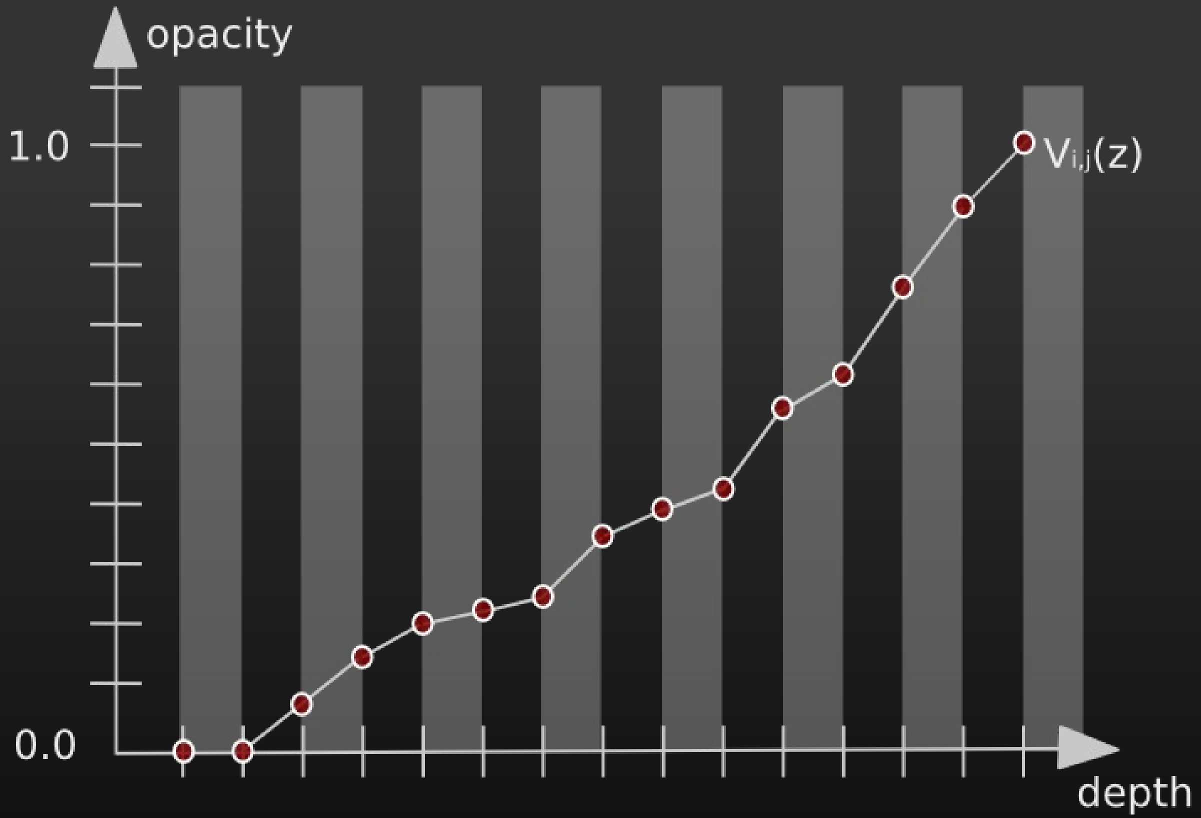
Algorithm

Overview

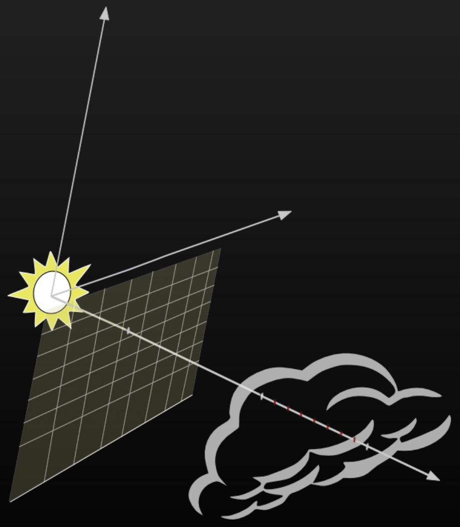
- 2-pass algorithm
  - Deep shadow map (DSM) creation
  - Rendering



Deep Shadow Map Creation



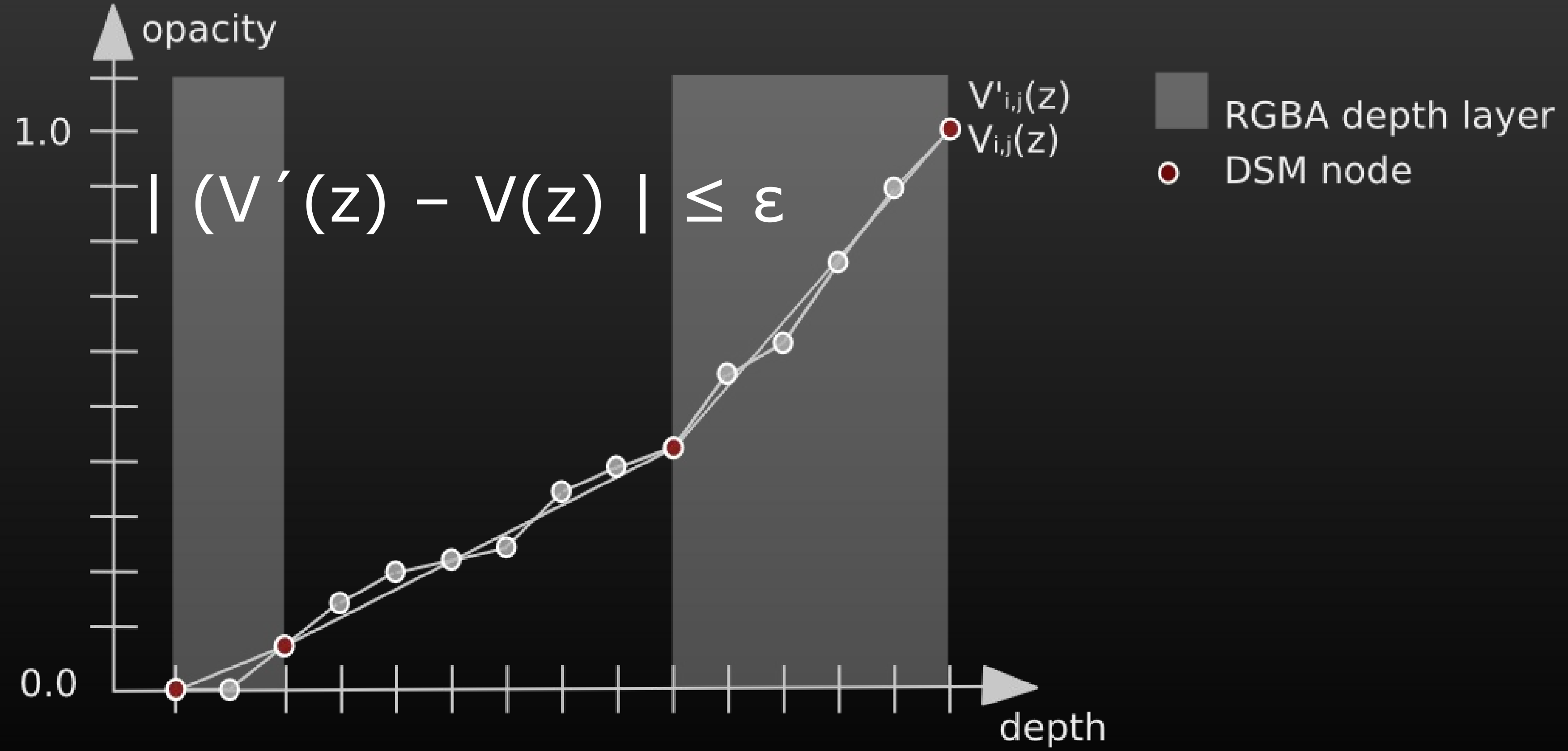
▨	RGBA depth layer
●	DSM node



- Visibility function
- Ray-casting in light space
- Node := (opacity, depth) pair

Deep Shadow Map Creation

Compression

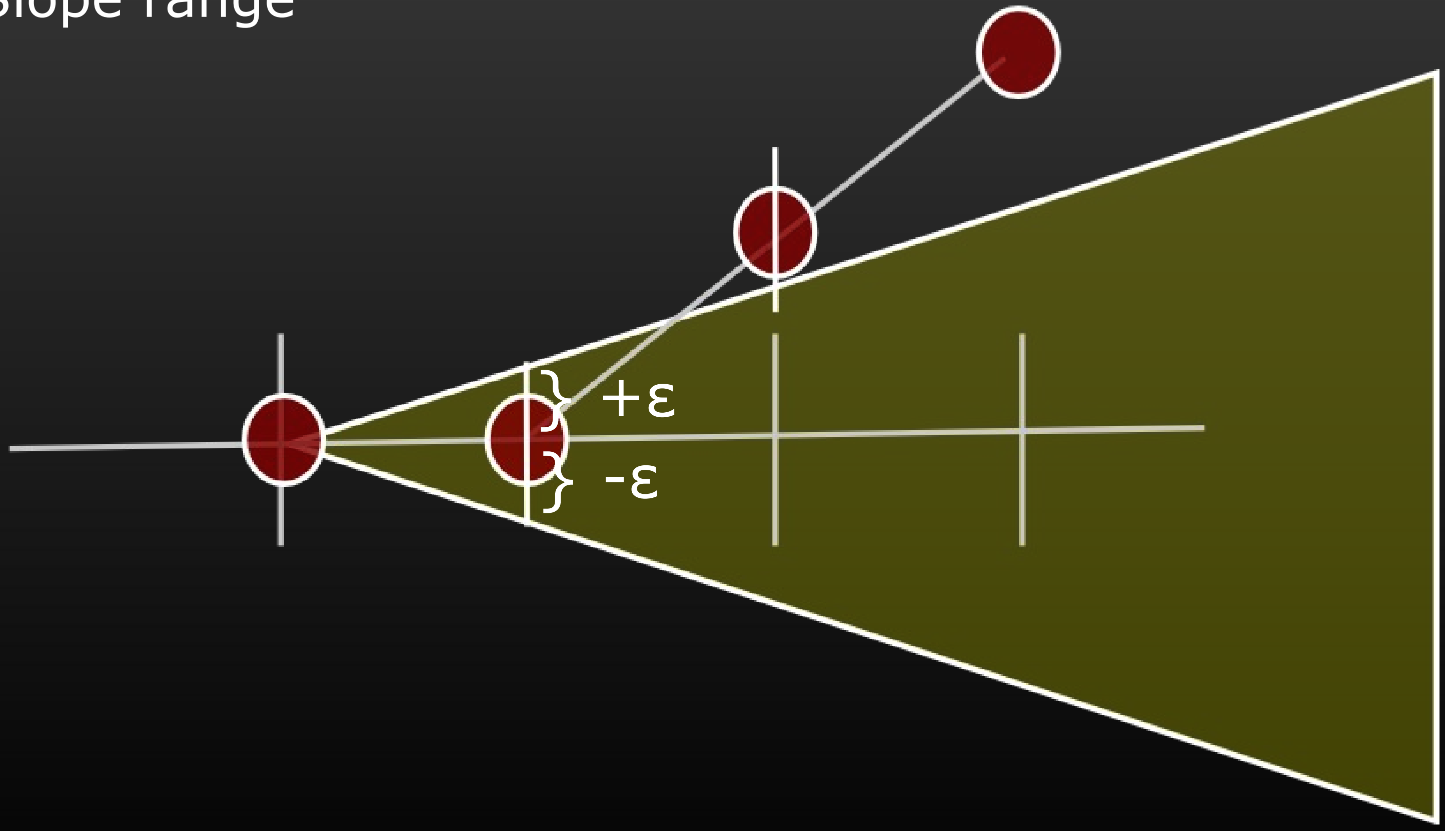


$$| (V'(z) - V(z) | \leq \varepsilon$$

Deep Shadow Map Creation

Determine permissible slopes

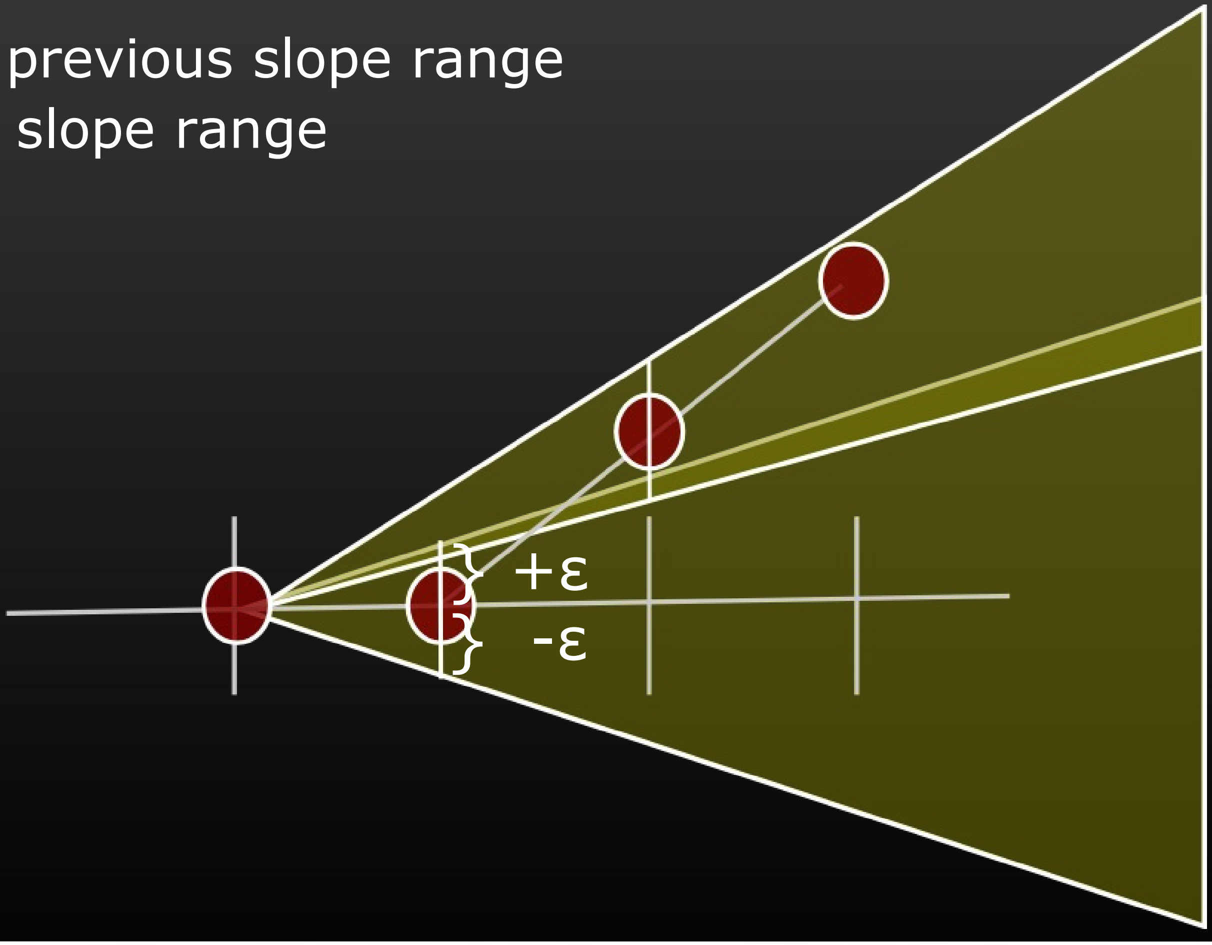
- Slope range



Deep Shadow Map Creation

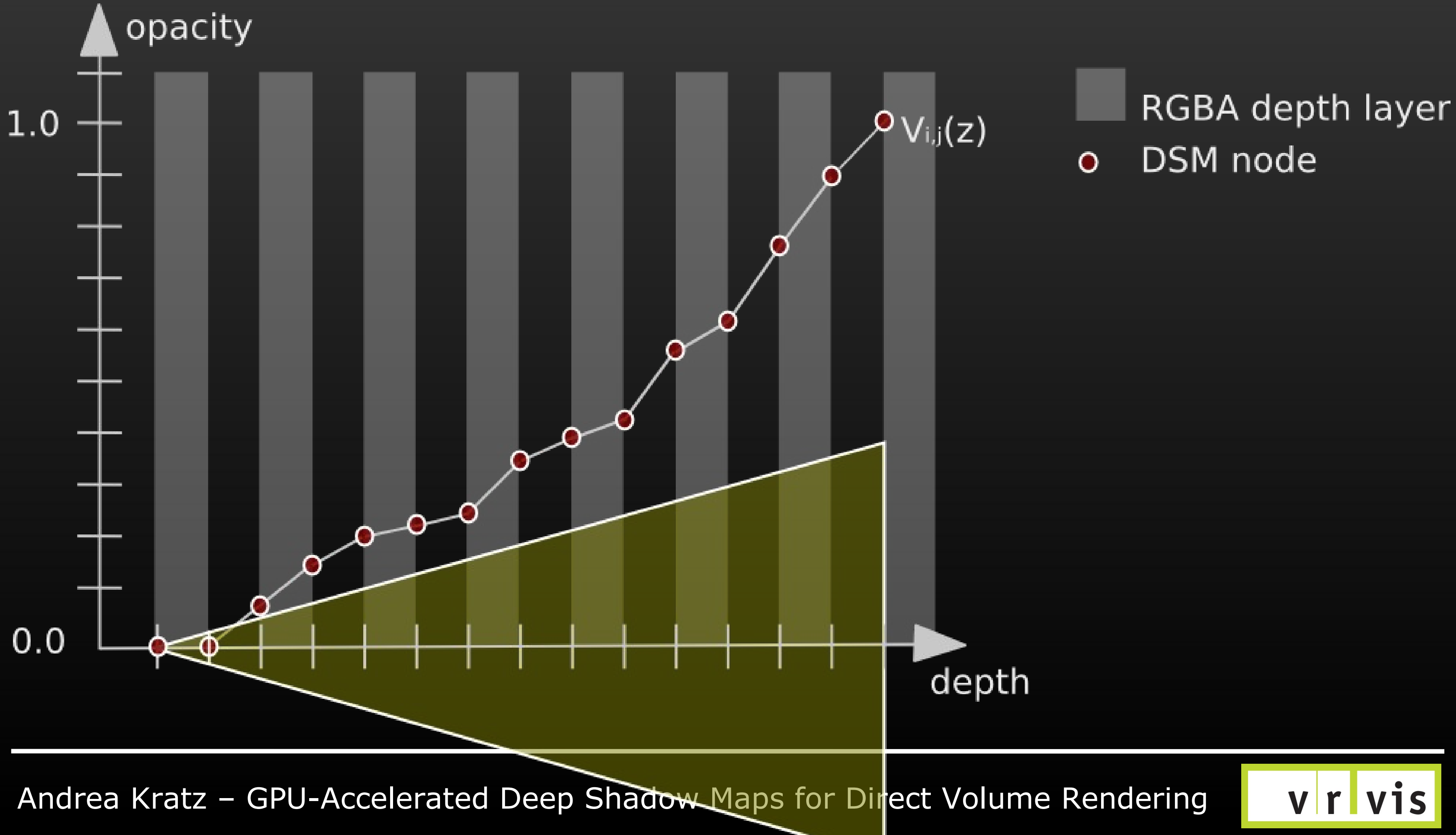
Determine permissible slopes

- Intersect previous slope range with next slope range



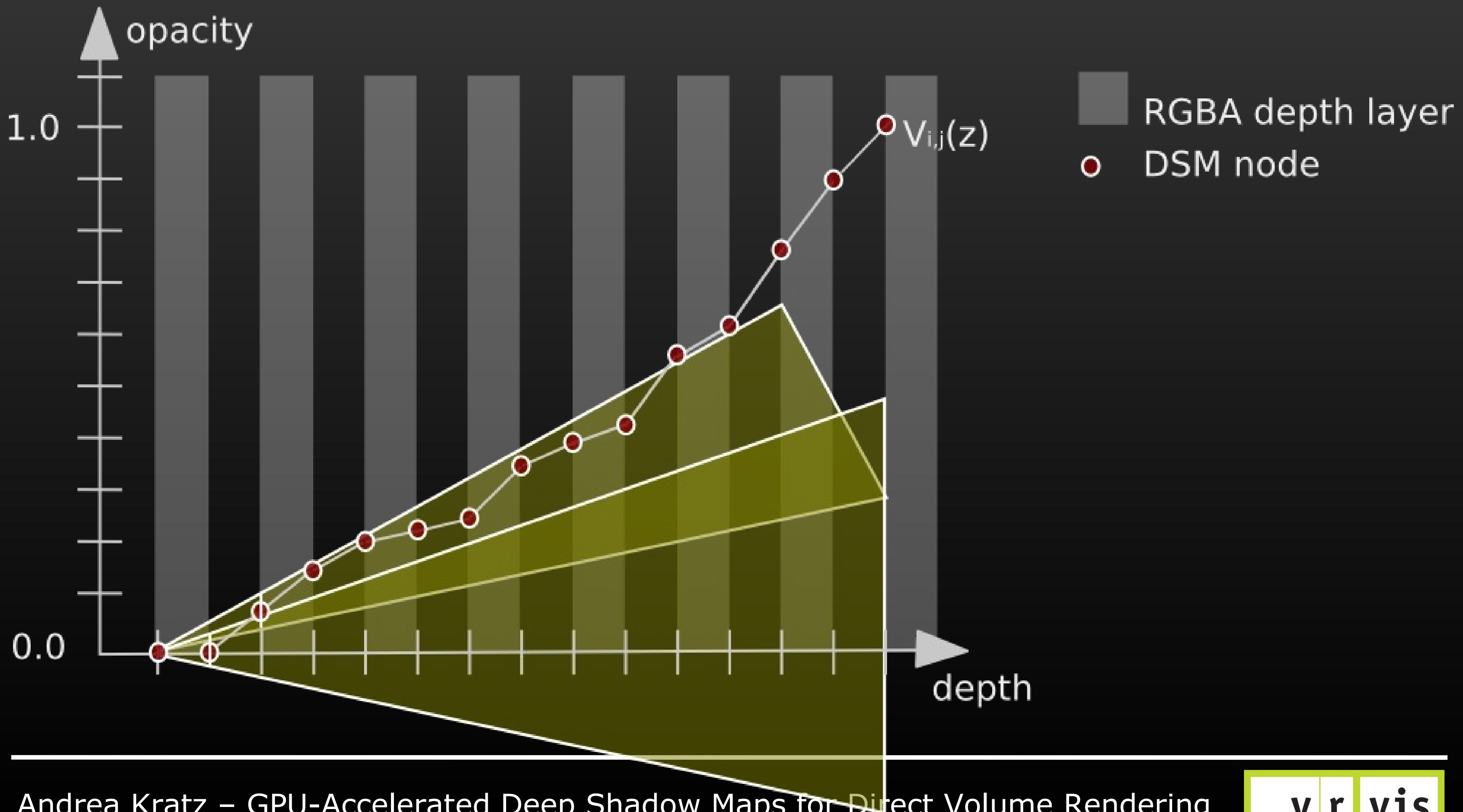
Deep Shadow Map Creation

Determine permissible slopes



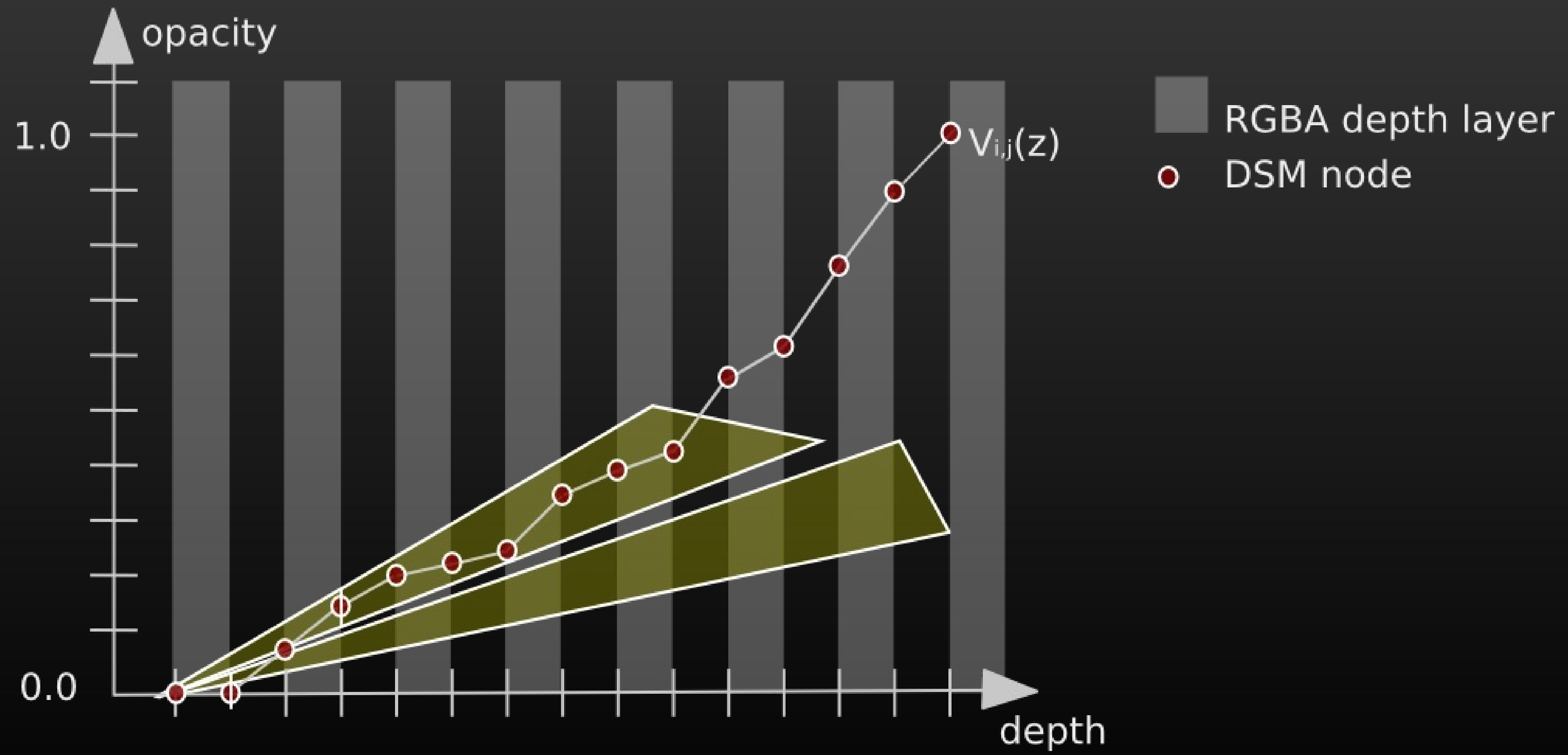
Deep Shadow Map Creation

Intersect previously computed with current slope



Deep Shadow Map Creation

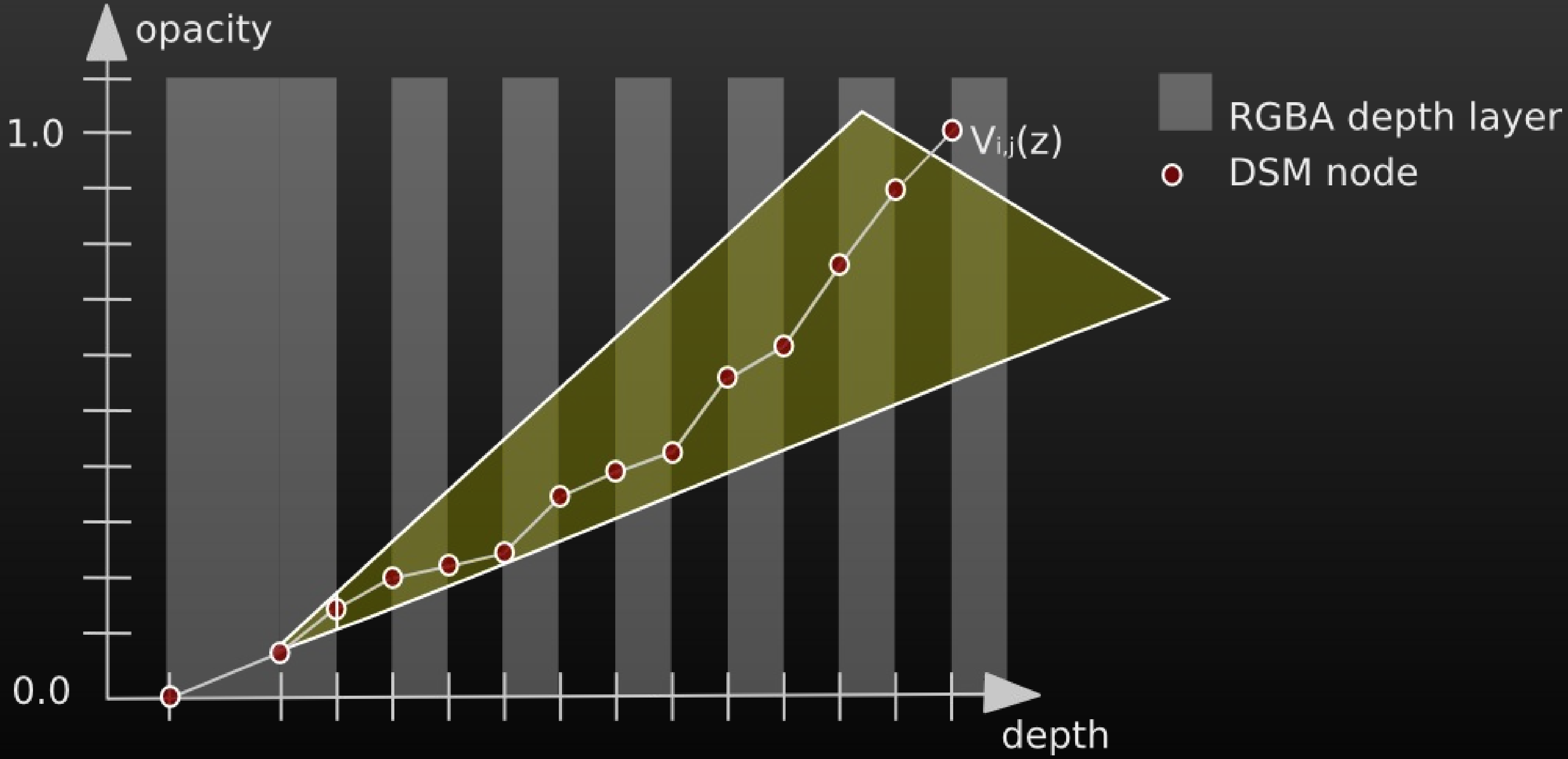
Slope becomes empty:



graphics hardware 06

Deep Shadow Map Creation

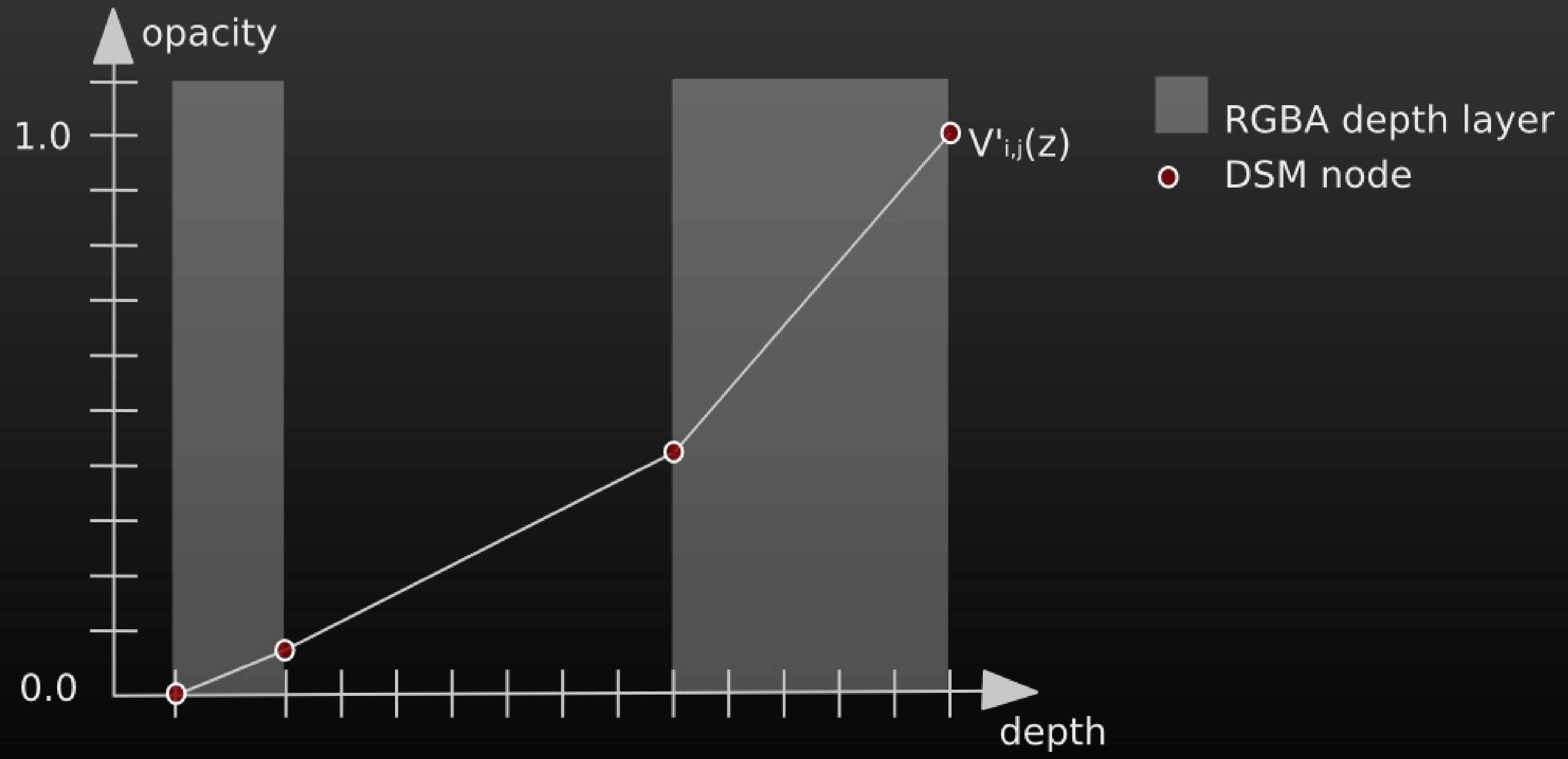
Start at end point of last segment



vrvis

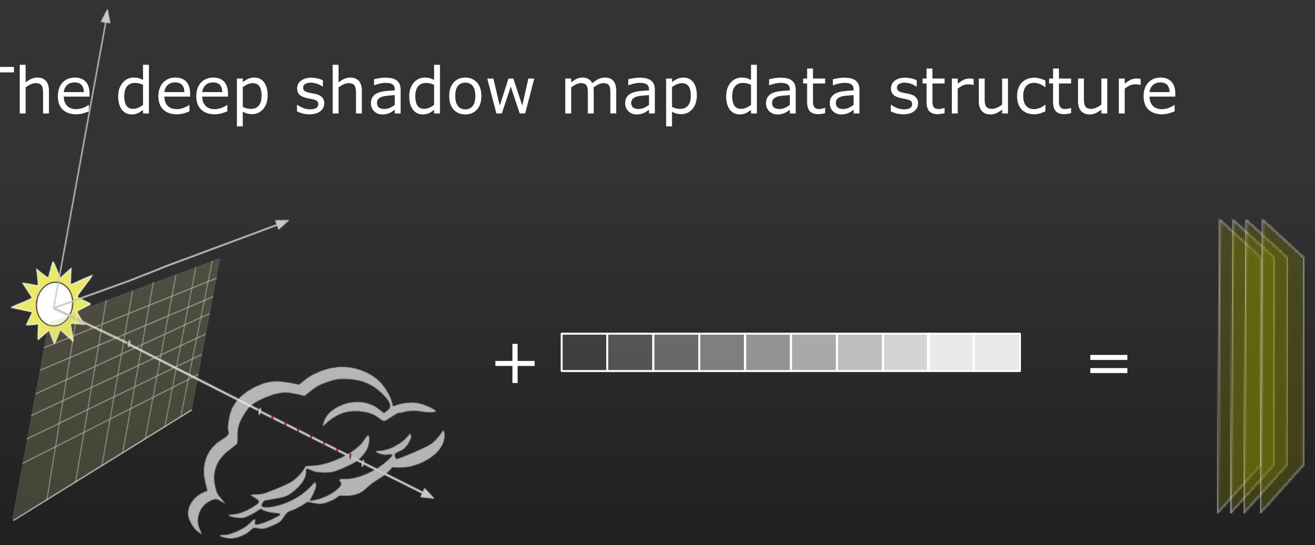
Deep Shadow Map Creation

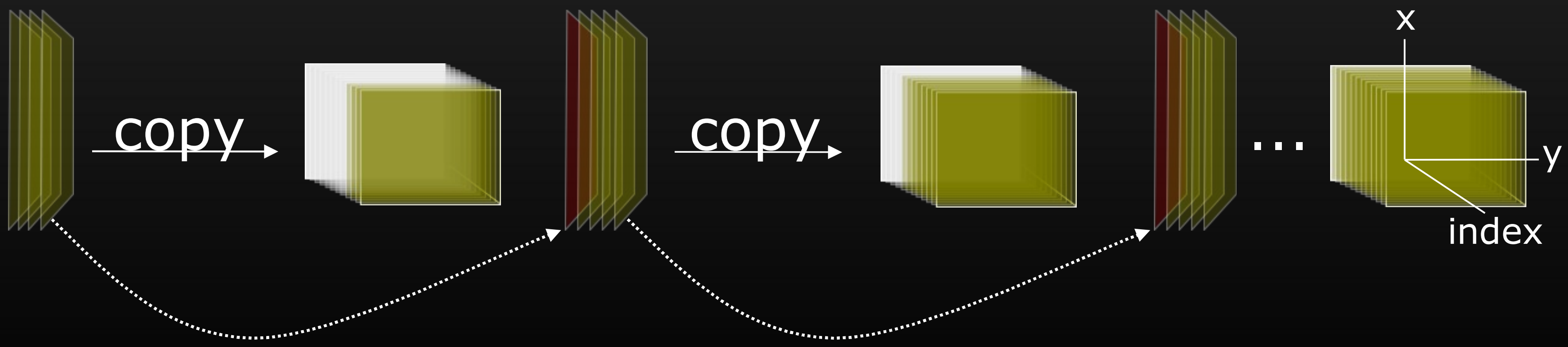
Compressed visibility function $V'_{i,j}(z)$



Deep Shadow Map Creation

The deep shadow map data structure





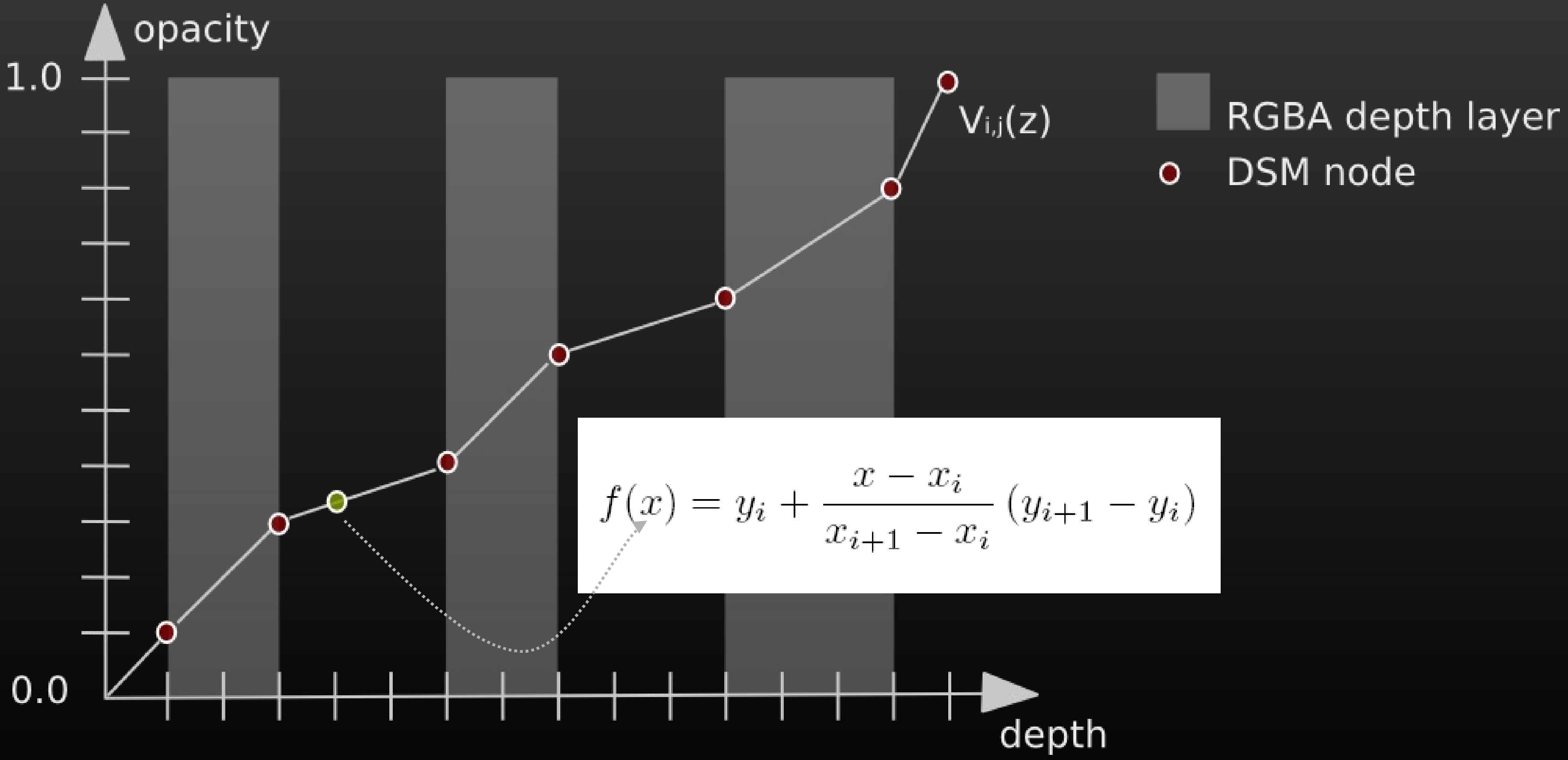
Rendering

Overview

- Ray-casting from viewpoint
- Linear search in DSM data structure for depth
  - Take advantage of spatial coherence
- Modulate pixel color by opacity obtained from DSM

Rendering

Linear search in DSM data structure



$$f(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i)$$

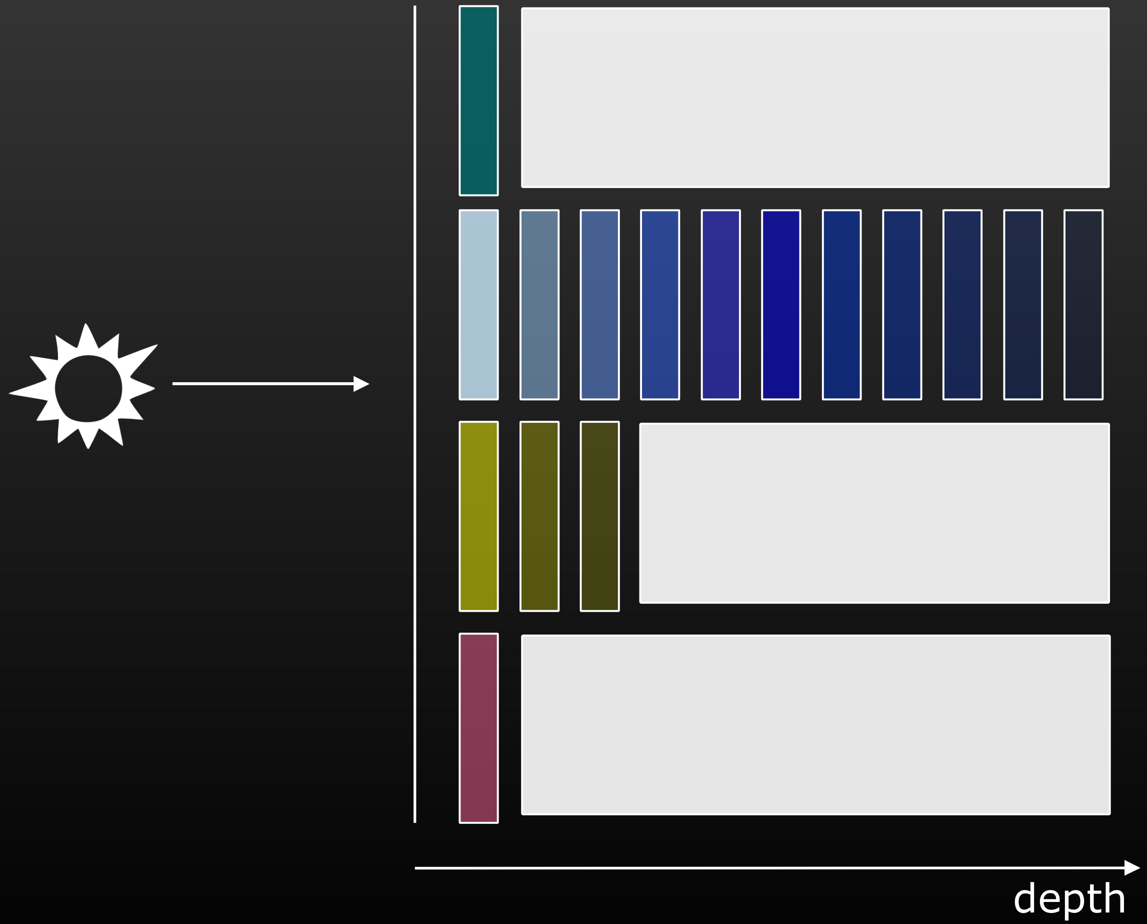
Optimizations

Overview

- Performance
  - Map is created in a screen-tiled manner using occlusion queries
- Memory management
  - Cached blocking scheme to maintain the volume
  - DSM storage in a blocked layout
- Image quality
  - Pre-compression

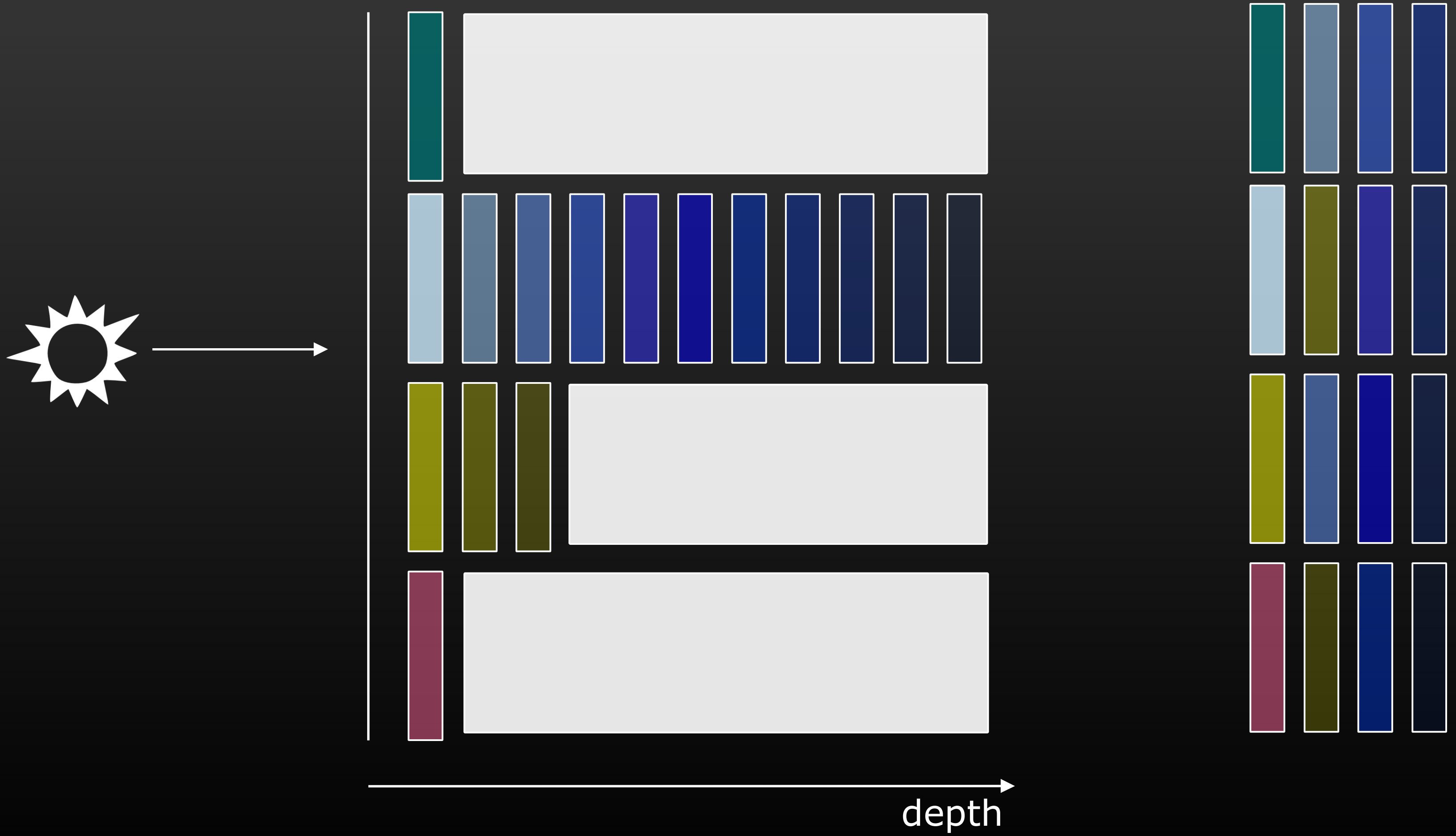
Optimizations

Original DSM vs. Packed DSM



Optimizations

Original DSM vs. Packed DSM



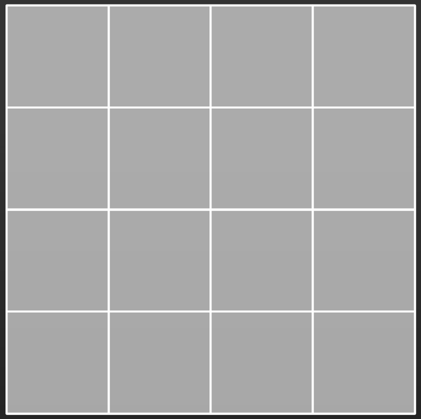
Optimizations

Screen-tiled map creation

Optimizations

Screen-tiled map creation

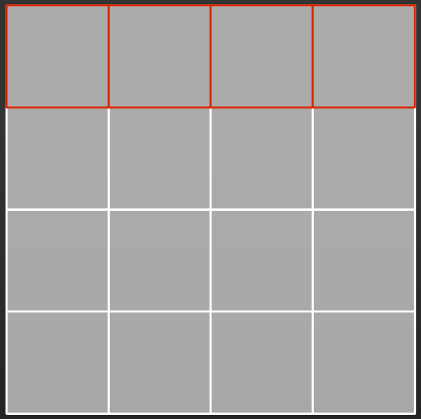
- Separate screen into tiles



Optimizations

Screen-tiled map creation

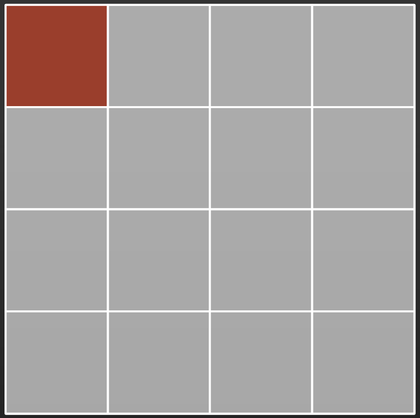
- Separate screen into tiles
- Update active tiles



Optimizations

Screen-tiled map creation

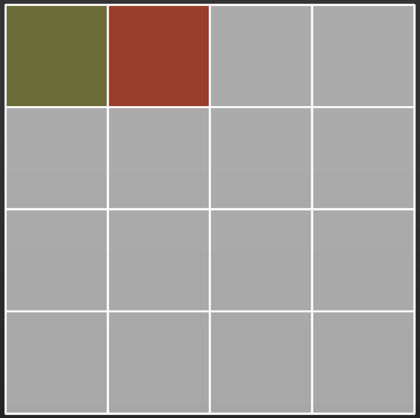
- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile



Optimizations

Screen-tiled map creation

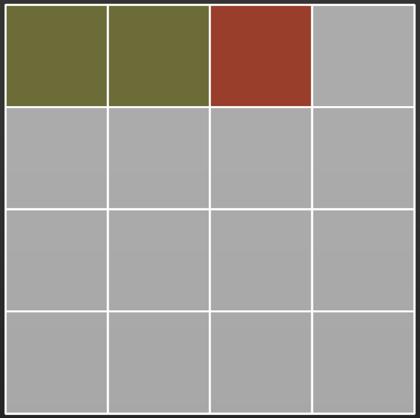
- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile



Optimizations

Screen-tiled map creation

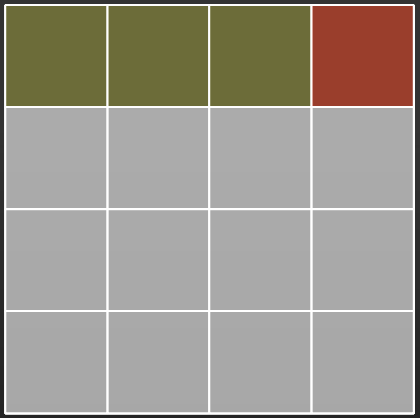
- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile



Optimizations

Screen-tiled map creation

- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile



Optimizations

Screen-tiled map creation

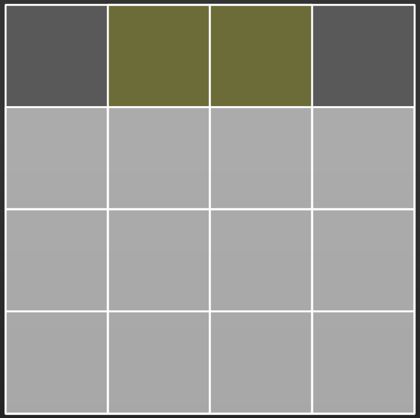
- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile
- To avoid pipeline stalls, check queries only after a specified time_offset

0	50	62	0

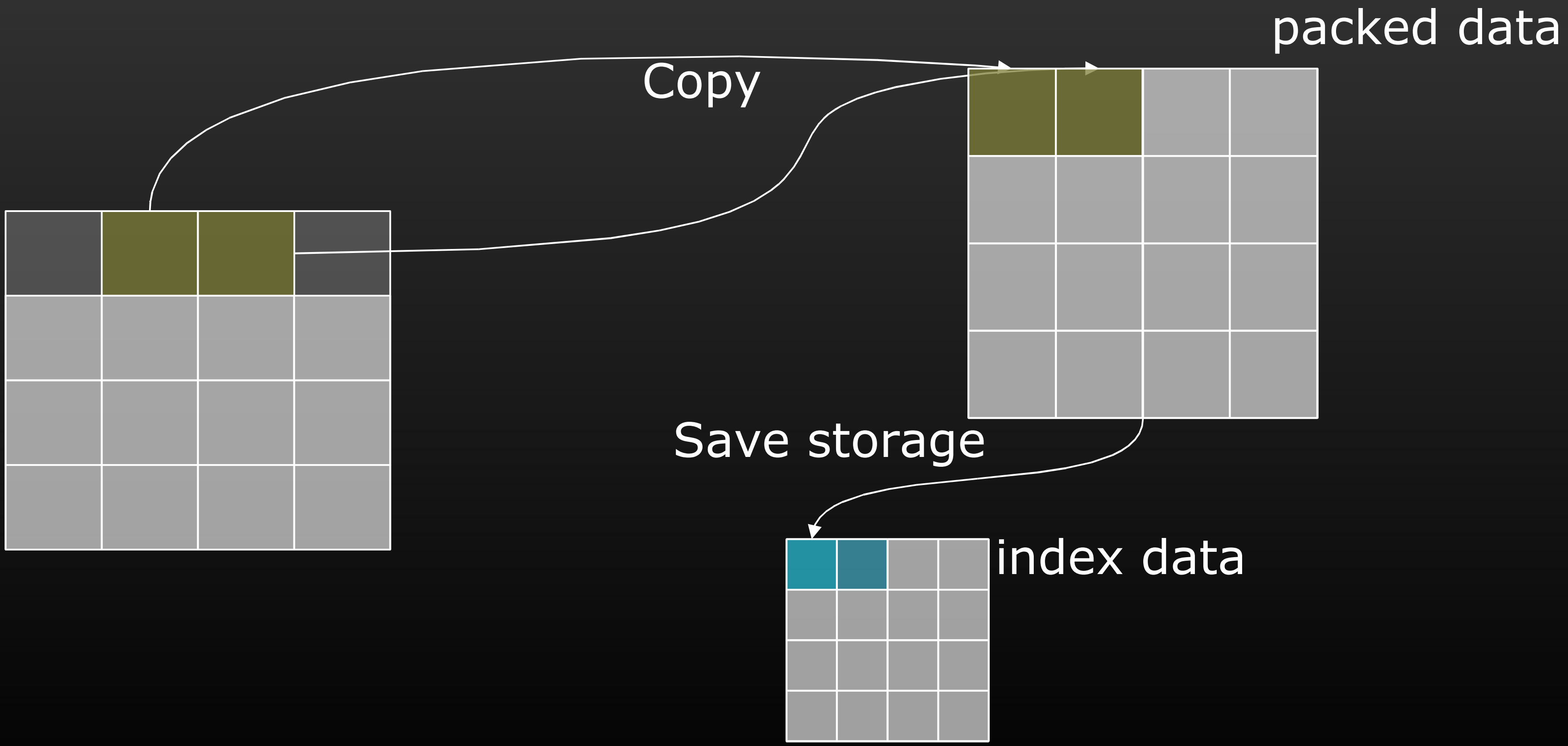
Optimizations

Map creation in a screen-tiled manner

- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile
- To avoid pipeline stalls, check queries
  only after a specified time_offset
- Skip empty tiles and copy non-empty tiles into DSM



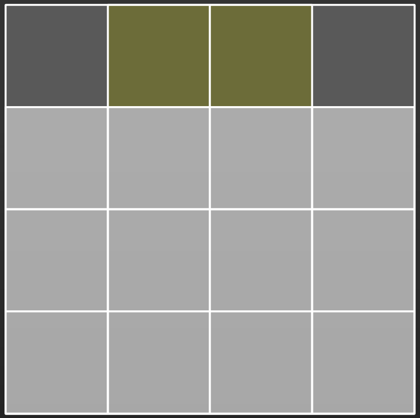
Optimizations



Optimizations

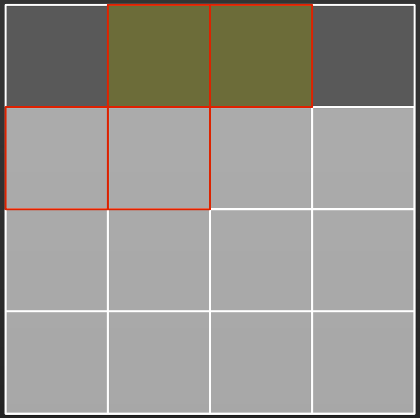
Screen-tiled map creation

- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile
- To avoid pipeline stalls, check queries only after a specified time_offset
- Skip empty tiles and copy non-empty tiles into DSM
- Return to step (2)



Optimizations

Screen-tiled map creation



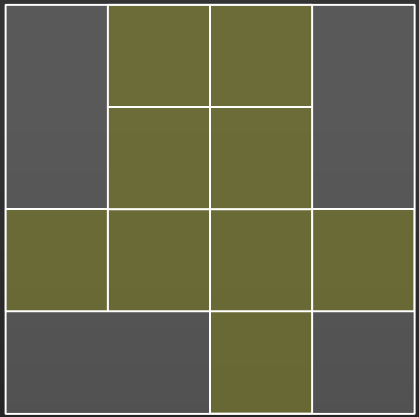
- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile
- To avoid pipeline stalls, check queries

  only after a specified time_offset
- Skip empty tiles and copy non-empty tiles into DSM
- Return to step (2)

Optimizations

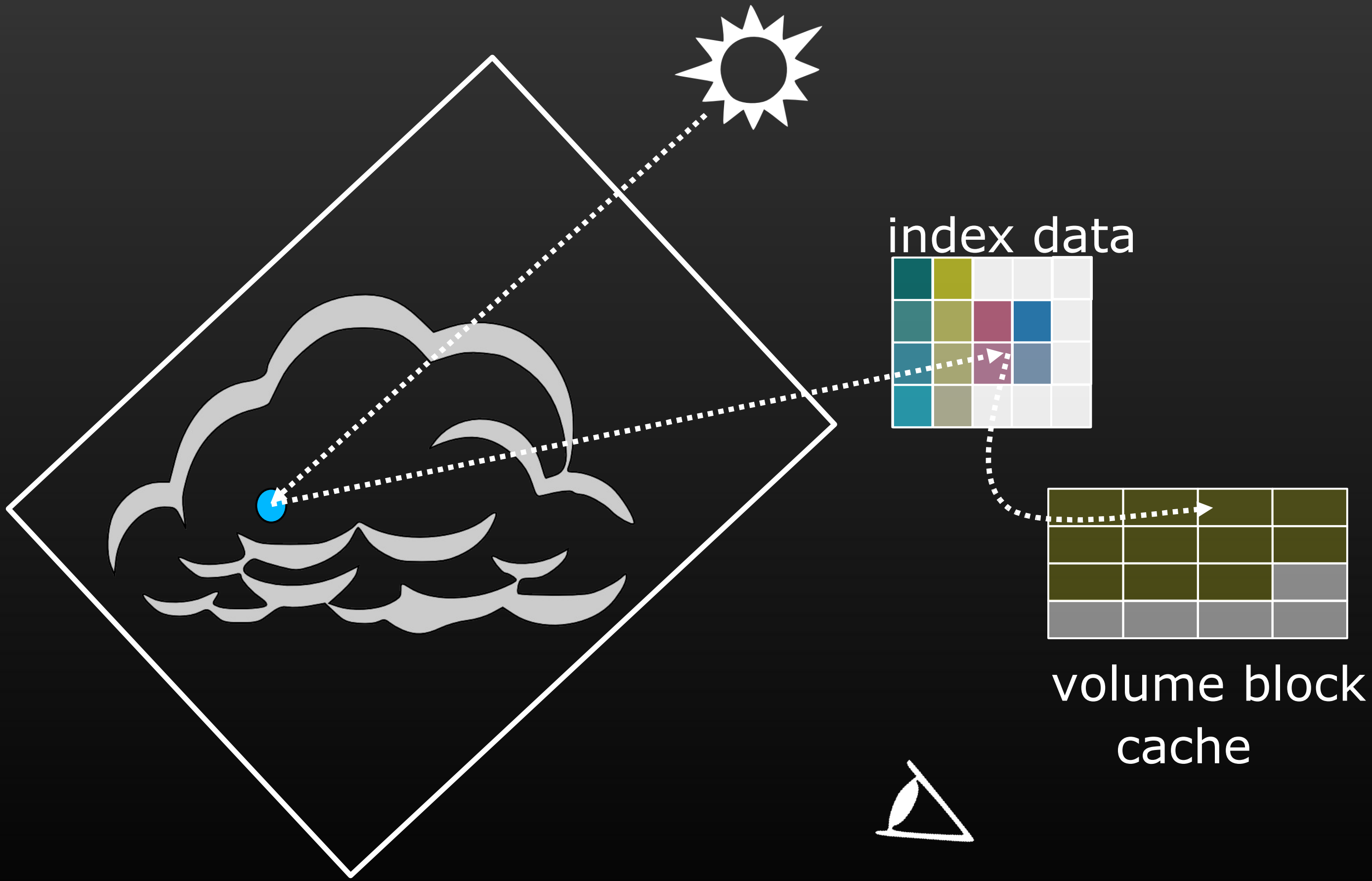
Screen-tiled map creation

- Separate screen into tiles
- Update active tiles
- Maintain occlusion query for every tile
- To avoid pipeline stalls, check queries only after a specified time_offset
- Skip empty tiles and copy non-empty tiles into DSM
- Return to step (2)



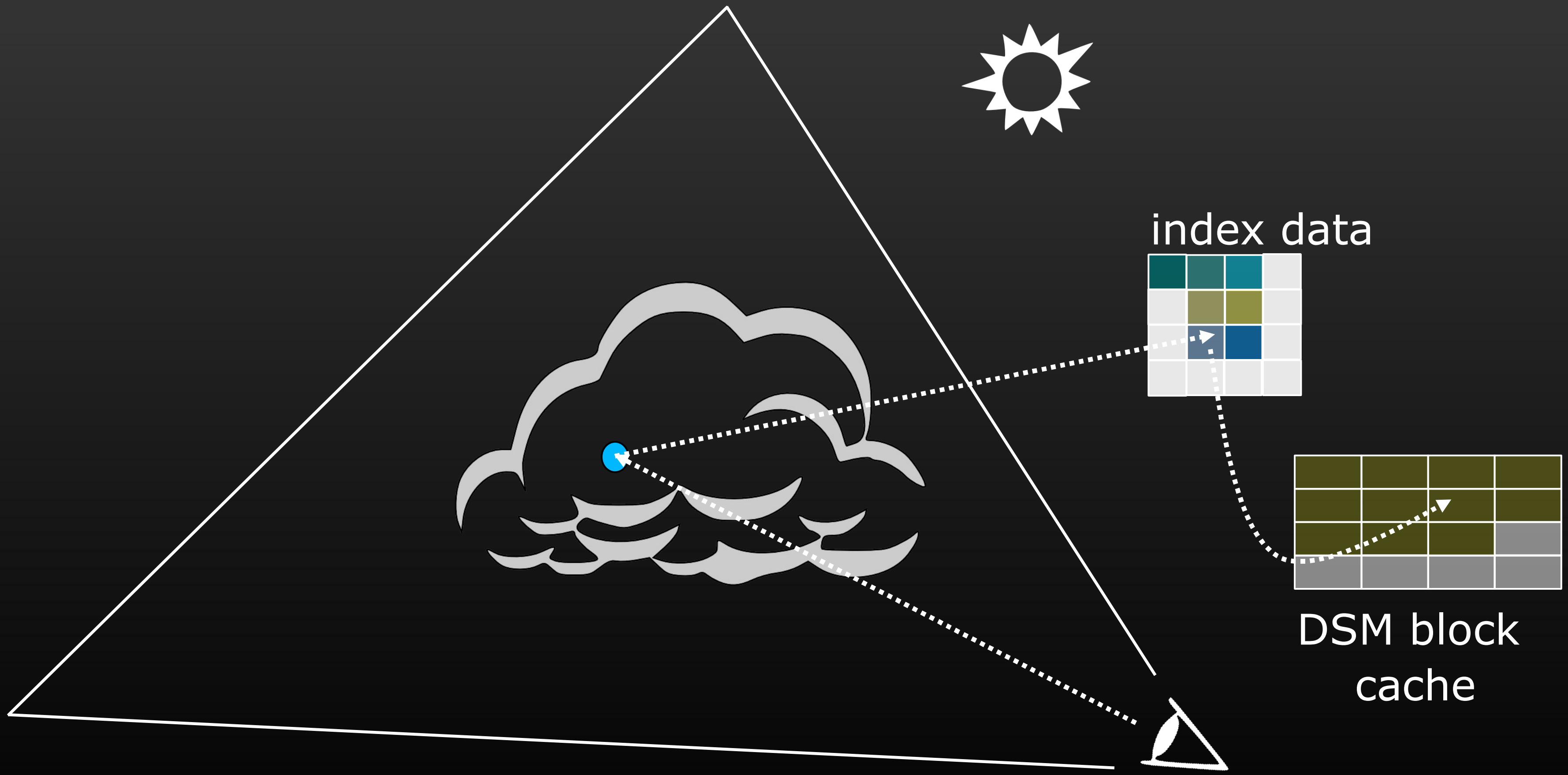
Rendering

Dependent texture fetches (during creation)



Rendering

Dependent texture fetches (during rendering)



Optimizations

Pre-integration and pre-compression

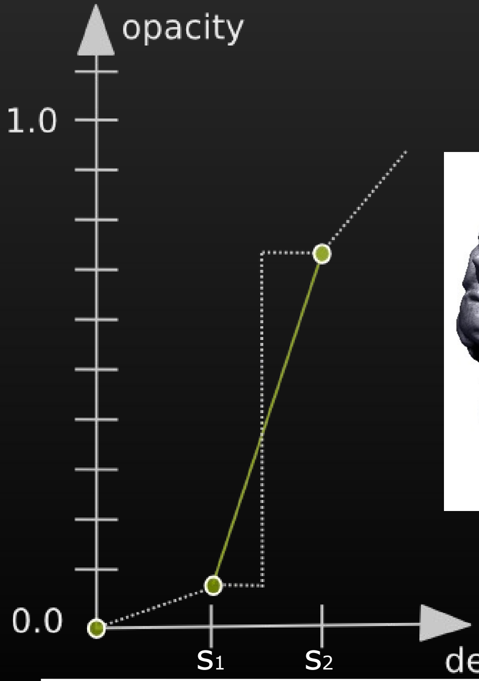
- Problem
  - Capture high frequencies in non-linear transfer functions
- Pre-integration [Engel et al. 2001]
  - Pre-compute the volume rendering integral for all possible combinations of 2 successive samples
- Pre-compression
  - Pre-compute and pre-compress the visibility function for all possible combinations of 2 successive samples

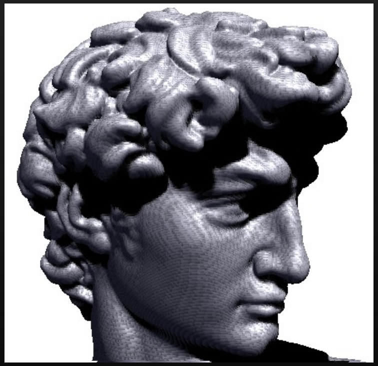
Optimizations

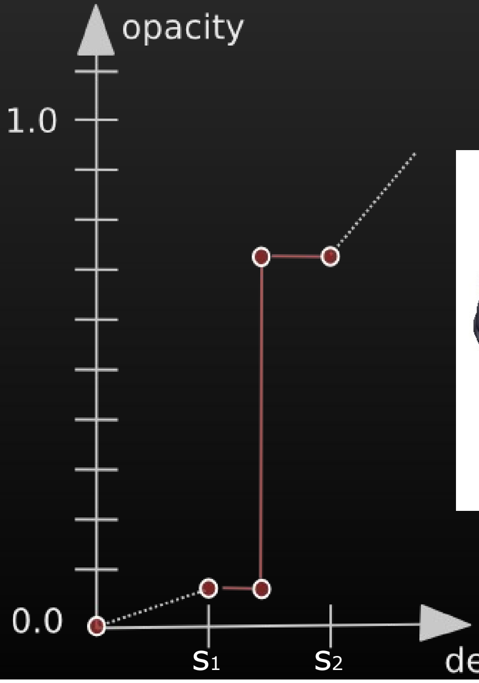
With and without pre-compression

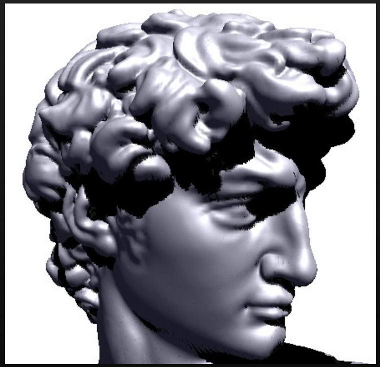
- Miss high frequencies in the opacity transfer function
- Undesired self-shadowing

- Better approximation of the visibility function
- No self-shadowing





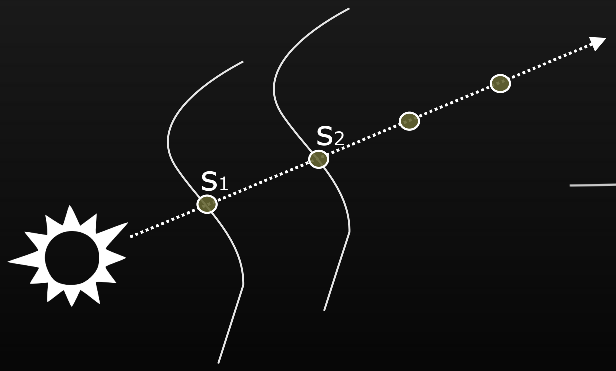


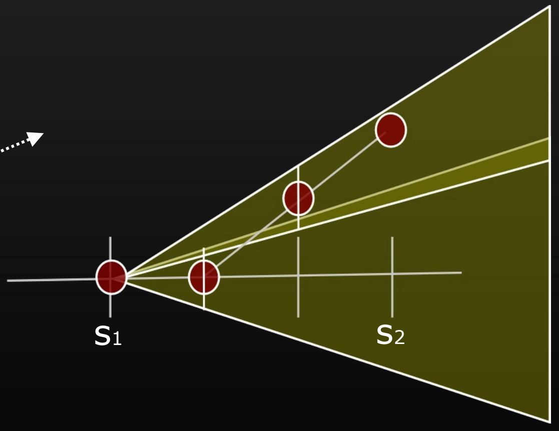


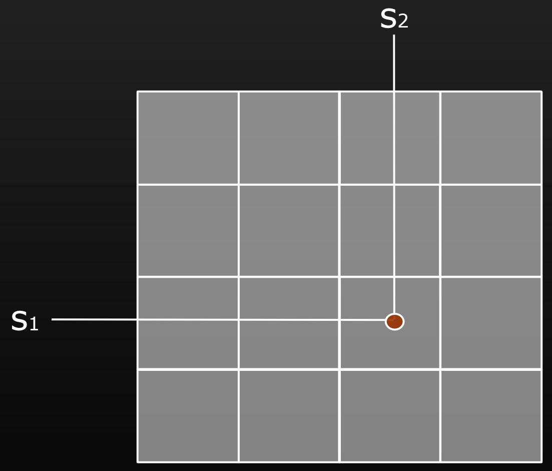
Optimizations

Pre-compression

- Pre-integrate and *compress* the visibility functions
- Sampling rate decoupled from transfer function domain
- Store pre-computed intermediate nodes







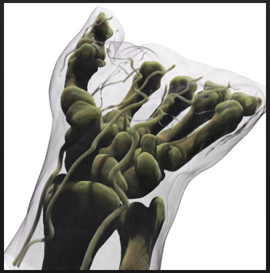
Results

Live Demo

Results

Performance for DSM creation

- ATI Radeon X1900 XTX
- Viewport 512 x 512



[fps]	256 x256	512 x 512
ε = 0.02	~ 2.5	~ 1.5
ε = 0.06	~ 3.5	~ 3

6-8 fps (rendering)

256x128x256

[fps]	256 x256	512 x 512
ε = 0.02	~ 3	~ 1.5
ε = 0.06	~ 4	~ 2.0

6-8 fps (rendering)

512x512x333

Conclusion & Future Work

Conclusion

- High-quality shadows for volume ray-casting
- Reduced amount of memory usage
- Extension of pre-integration to visibility functions

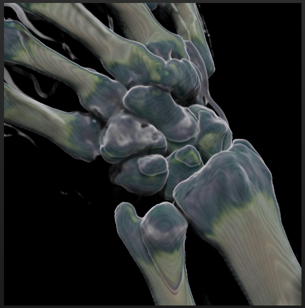
Future work

- Shadow map filtering during rendering
- Colored shadows
- Scattering approximations
- Geometry

Thank you!

For more information, visit

http://medvis.vrvis.at



Funding

- KPlus program of the Austrian government
- Femtech initiative of bmvit, http://www.femtech.at/