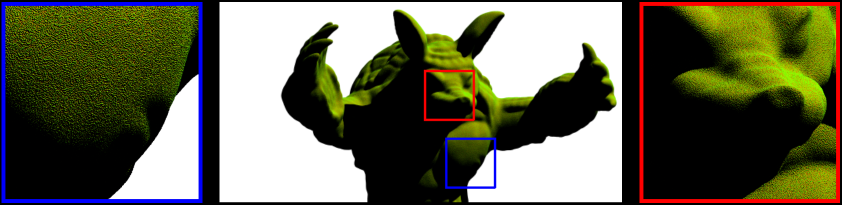
Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees

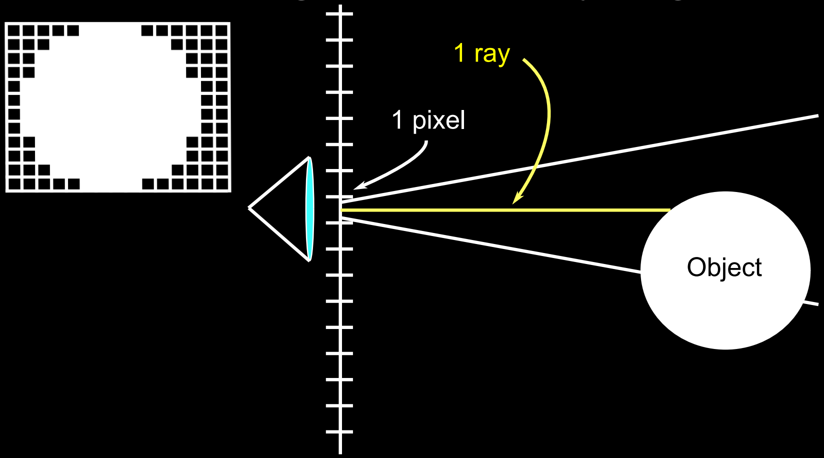
HPG 2012

Eric Heitz   Fabrice Neyret

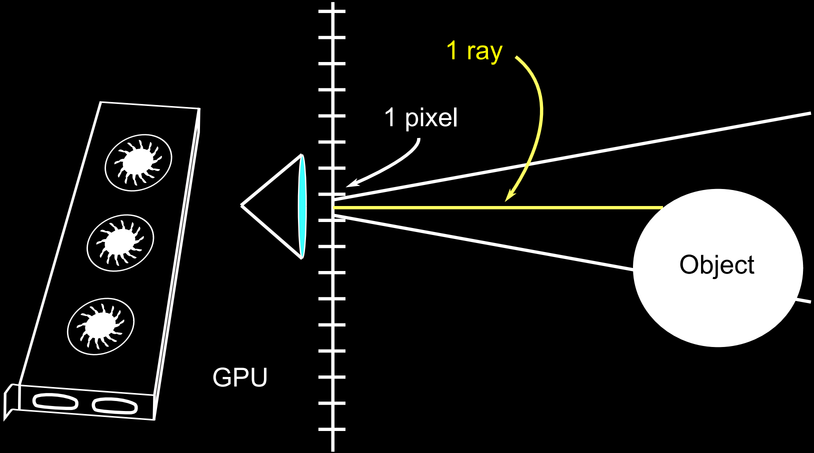
INRIA Rhône-Alpes – Laboratoire Jean Kuntzmann (Université de Grenoble and CNRS)



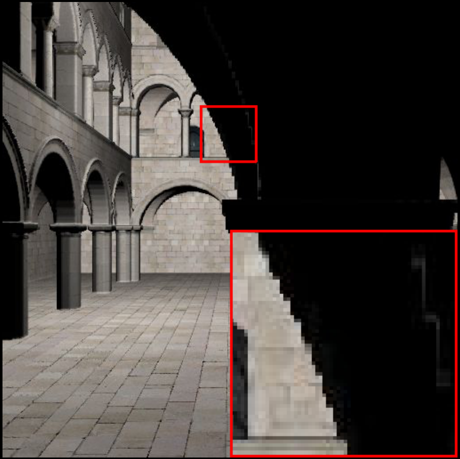
Pixel sampling : rasterization or raytracing..



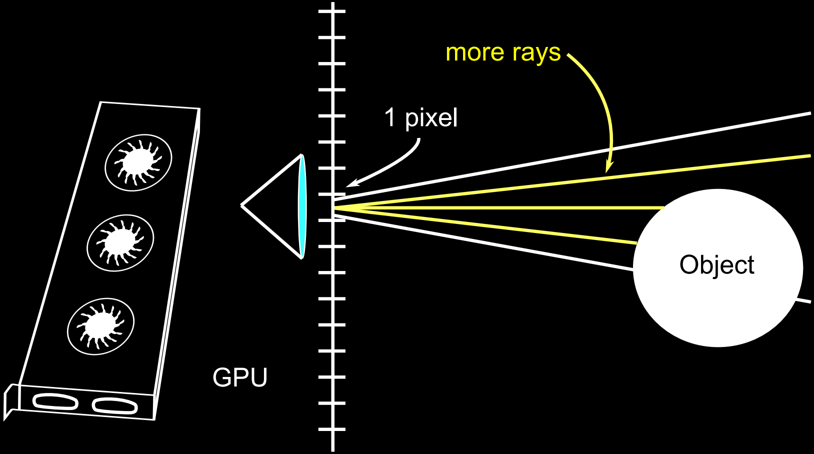
..running on the GPU



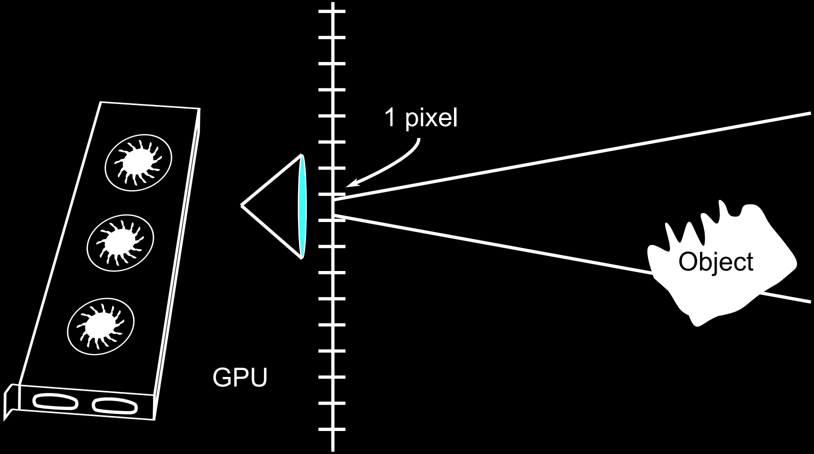
Sampling with only one ray produces aliasing



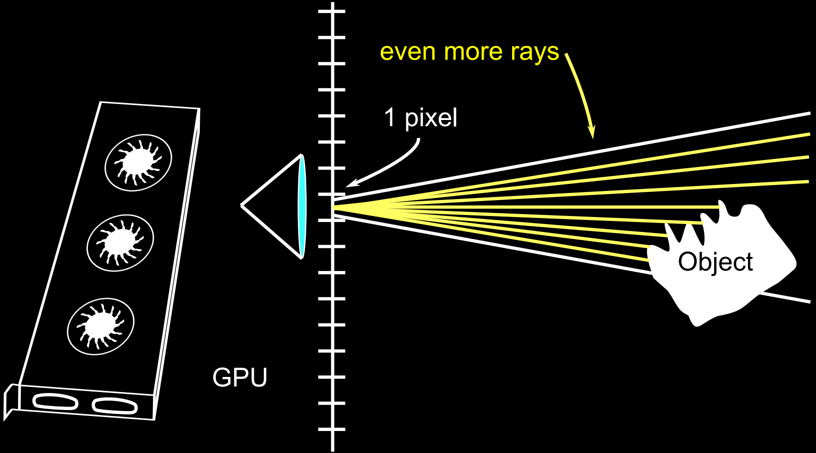
Cast more ray to anti-aliase..



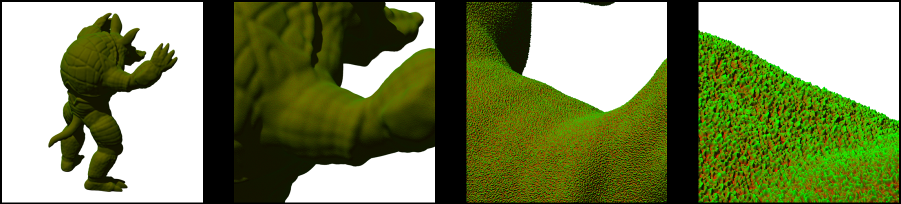
A more complex object..



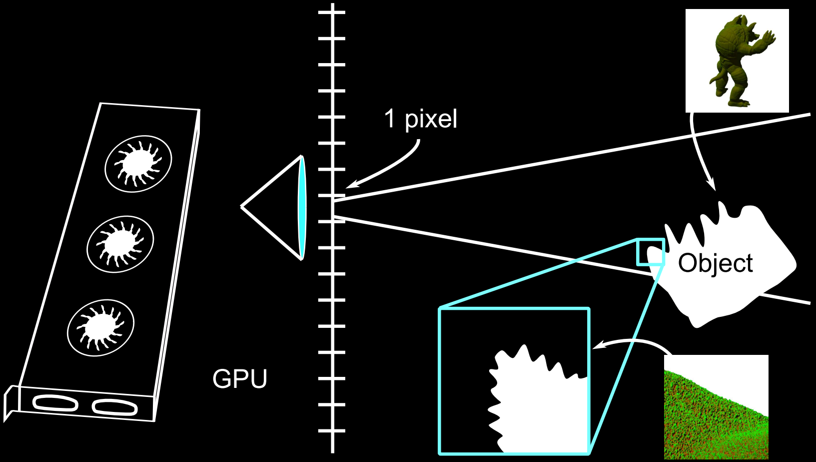
..needs even more rays



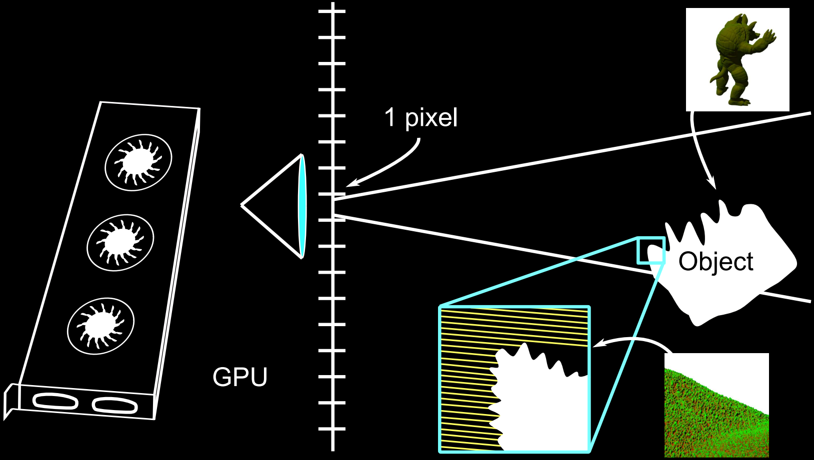
Still more details, more complexity



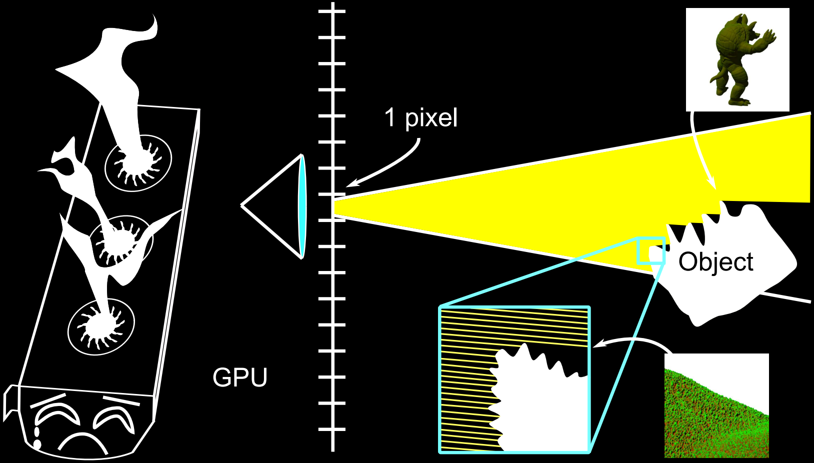
Still more details, more complexity



Every detail needs to be sampled



Unaffordable even for a very big GPU



Scalability in Raytracing

Scalable rendering algorithms

- only 1 ray per pixel instead of oversampling

Scalable rendering representations

- Don't adapt yourself to the scene,
  have the scene adapt to you

Important Features

- **Pre-filtered representations**
- **Interpolation through space and scale**

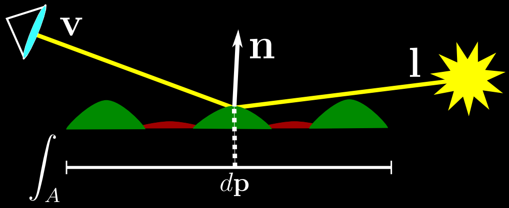
Local Illumination Equation

Integration of the surface A over the pixel footprint

$$I = \frac{\int_A E_{\mathbf{l}}\ a\ \rho_{\mathbf{v},\mathbf{l}}\ V_{\mathbf{l}}\ V_{\mathbf{v}}\ \mathbf{nl}\ \mathbf{nv}\ d\mathbf{p}}{\int_A V_{\mathbf{v}}\ \mathbf{nv}\ d\mathbf{p}}$$

with

$E_{\mathbf{l}}$ **incoming radiance**
a **material albedo**
$\rho_{\mathbf{v},\mathbf{l}}$ **BRDF**
$V_{\mathbf{v}}, V_{\mathbf{l}}$ **visibilities**



Mipmapping Approximation

The terms in the integral are **not correlated**

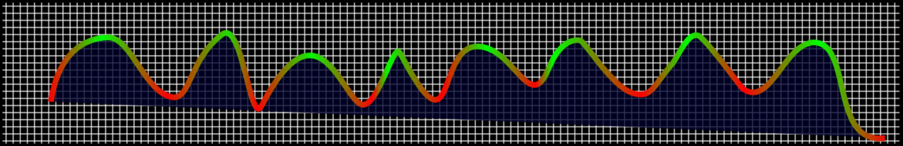
$\rightarrow$ they can be **separately integrated**

$$I = \frac{\int_A E_{\mathbf{l}}\ a\ \rho_{\mathbf{v},\mathbf{l}}\ V_{\mathbf{l}}\ V_{\mathbf{v}}\ \mathbf{nl}\ \mathbf{nv}\ d\mathbf{p}}{\int_A V_{\mathbf{v}}\ \mathbf{nv}\ d\mathbf{p}}$$

becomes

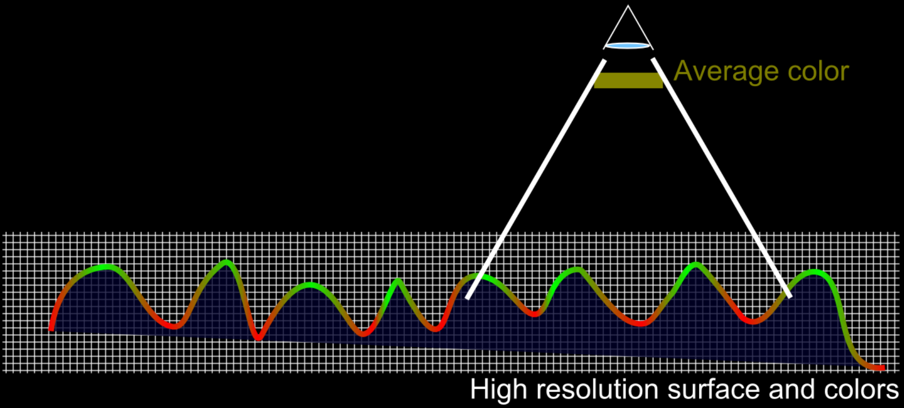
$$I \approx \frac{\int_A E_{\mathbf{l}}\ d\mathbf{p}}{\int_A d\mathbf{p}}\ \frac{\int_A a\ d\mathbf{p}}{\int_A d\mathbf{p}}\ \frac{\int_A \rho_{\mathbf{v},\mathbf{l}}\ \mathbf{nl}\ d\mathbf{p}}{\int_A d\mathbf{p}}\ \frac{\int_A V_{\mathbf{l}}\ d\mathbf{p}}{\int_A d\mathbf{p}}$$

The effect of correlation

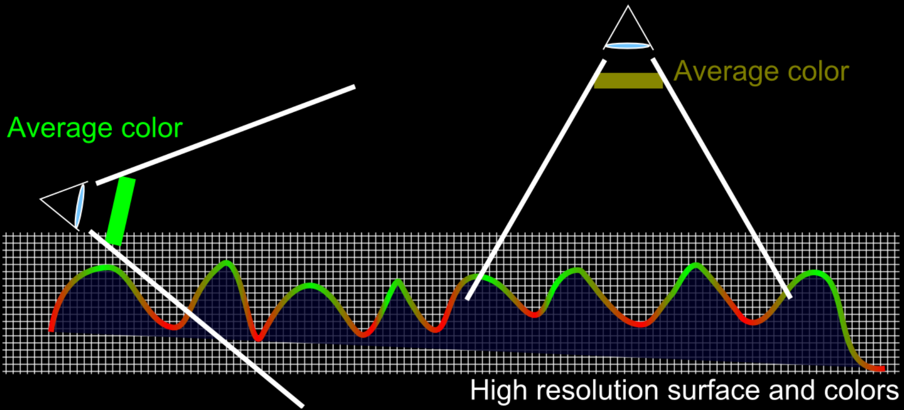


High resolution surface and colors

The effect of correlation

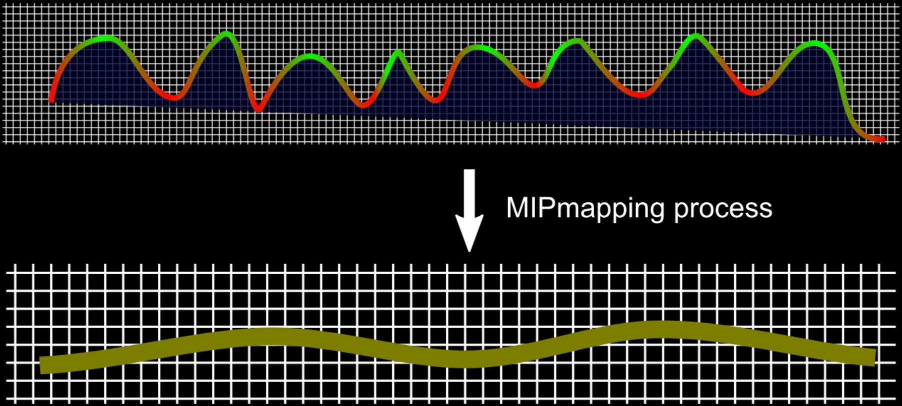


The effect of correlation

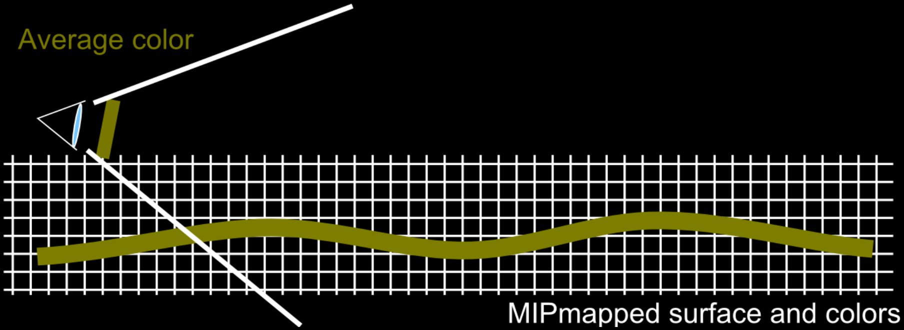


$$\int a\, V_{\mathbf{v}} \neq \int a \int V_{\mathbf{v}}$$

The effect of correlation



The effect of correlation

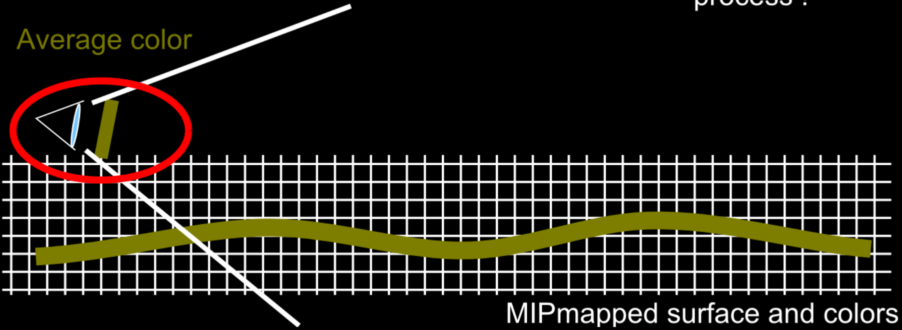


The effect of correlation

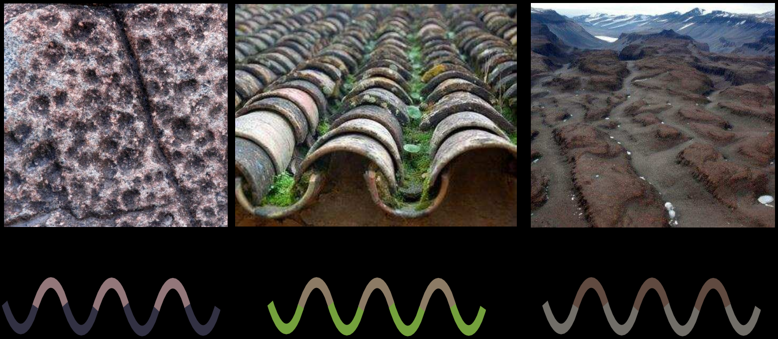
Masking effects are lost with separate MIPmapping

→

Visibility has to be taken into account in the pre-filtering process !



Many real-world surfaces show such correlations



Important Features

- Pre-filtered representations
  - Mipmapping does not work with correlations
- Interpolation through space and scale

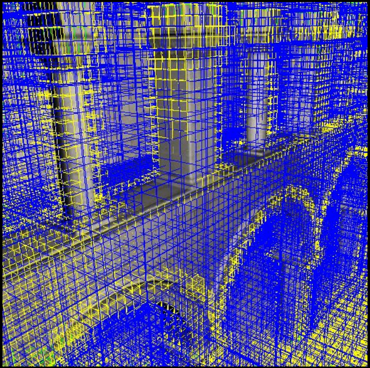
Important Features

- Pre-filtered representations
  - Mipmapping does not work with correlations
- Interpolation through space and scale
  - Meshes do not filter properly
  - Meshes have parametrization and topological problems

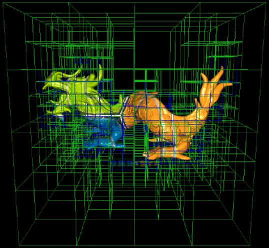
Sparse Voxel Octrees (SVO)

Properties

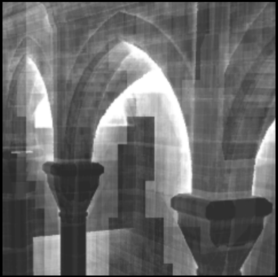
- Spatial organization
- No parametrization problem
- LoD representation
- GPU implementation



Previous Work

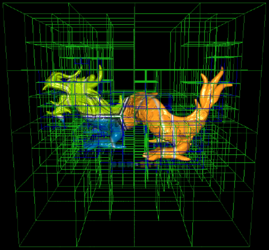


GigaVoxels [CNLE09]



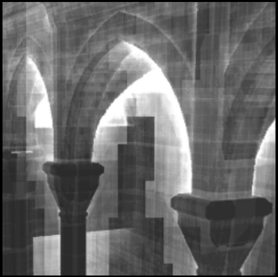
Efficient Sparse Voxel Octrees [LK10]

Previous Work



GigaVoxels [CNLE09]

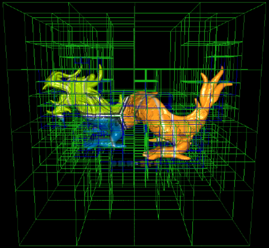
- Interpolation



Efficient Sparse Voxel Octrees [LK10]

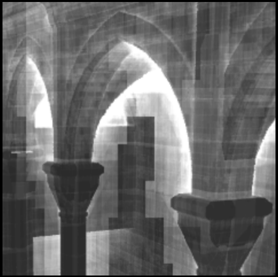
- No interpolation

Previous Work



GigaVoxels [CNLE09]

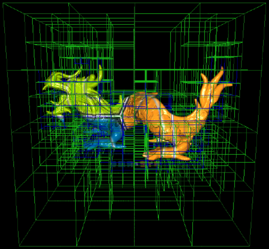
- Interpolation
- Volume rendering

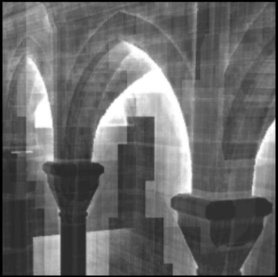


Efficient Sparse Voxel Octrees [LK10]

- No interpolation
- Surface rendering

Previous Work





GigaVoxels [CNLE09]

- Interpolation
- Volume rendering
- Pre-filtering
  - Material pre-filtering
  - No normal pre-filtering
  - No visibility pre-filtering

Efficient Sparse Voxel Octrees [LK10]

- No interpolation
- Surface rendering
- No pre-filtering
  - No material pre-filtering
  - No normal pre-filtering
  - No visibility pre-filtering

Our Sparse Voxel Octree Model

- **Interpolation** : GigaVoxels [CNLE09] data-structure

Our Sparse Voxel Octree Model

- **Interpolation : GigaVoxels [CNLE09] data-structure**

Contributions

- **Surface-based representation : 1. Macro-surface representation**

Our Sparse Voxel Octree Model

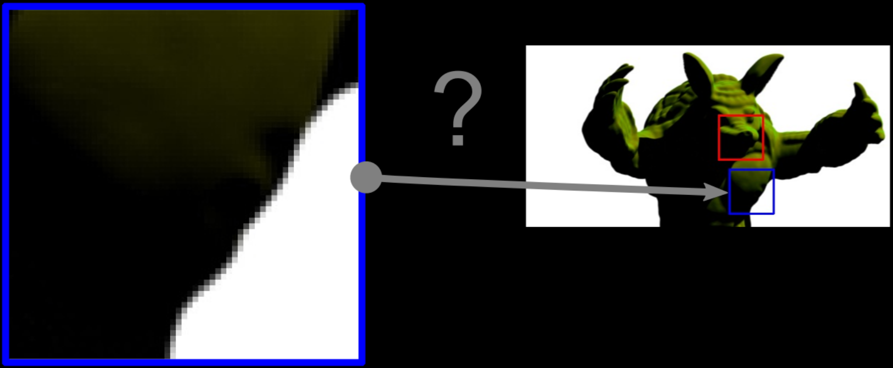
- Interpolation : GigaVoxels [CNLE09] data-structure

Contributions

- Surface-based representation : 1. Macro-surface representation
- Pre-filtering (bottom-up build) : 2. Micro-surface representation
  - Material pre-filtering
  - Normal pre-filtering
  - Visibility pre-filtering

1. **Macro-surface model**
2. **Micro-surface model**

How to render anti-aliased surfaces ?
(with 1 ray per pixel)



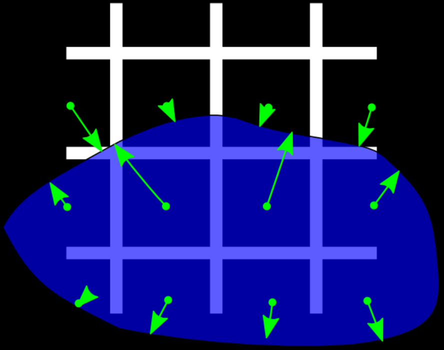
1. Macro-surface model

Rendering with Subpixel Occlusion Distributions

- The A-buffer, an antialiased hidden surface method
  [Carpenter 84]
- Tracing ray differentials
  [Igehy 99]

Surface Representation

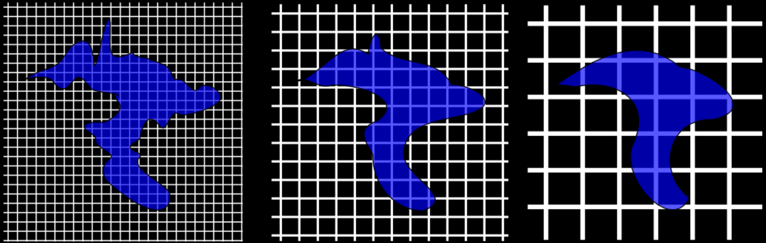
A signed distance field keeps the subvoxel surface position



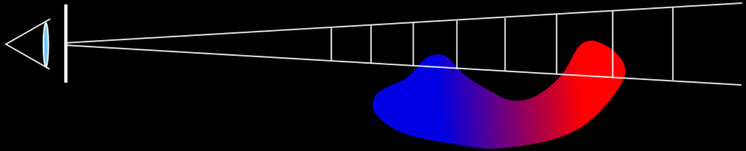
1. Macro-surface model

Surface Representation

We build a LoD surface representation
At runtime, the pixel footprint matches the size of a voxel

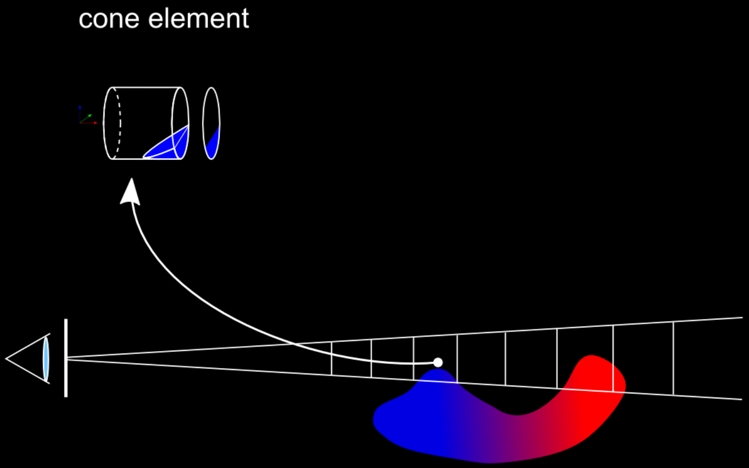


Differential Cone Tracing Scheme



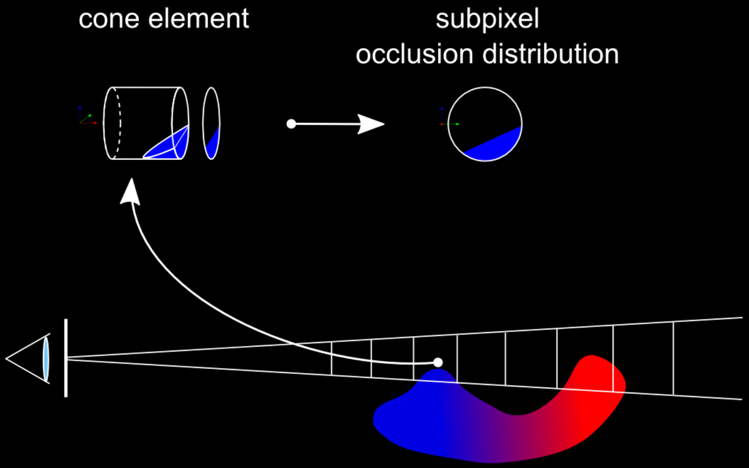
1. Macro-surface model

Differential Cone Tracing Scheme



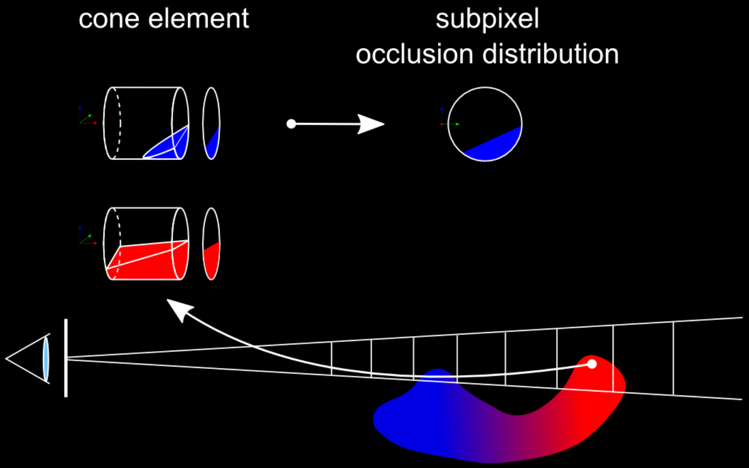
1. Macro-surface model

Differential Cone Tracing Scheme



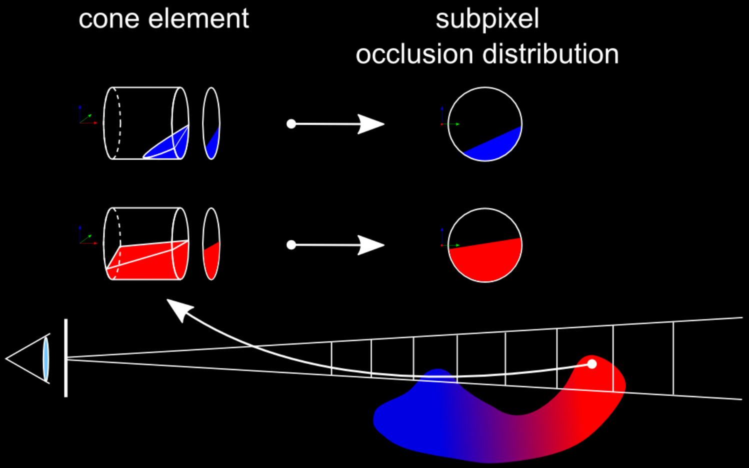
1. Macro-surface model

Differential Cone Tracing Scheme



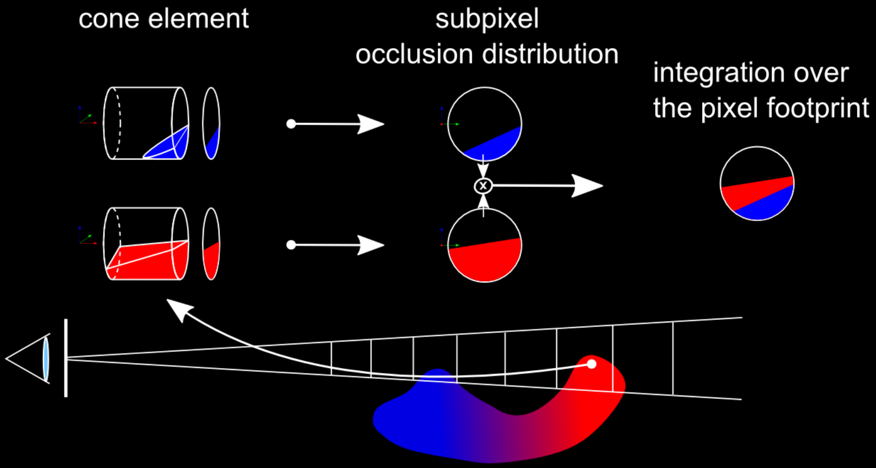
1. Macro-surface model

Differential Cone Tracing Scheme



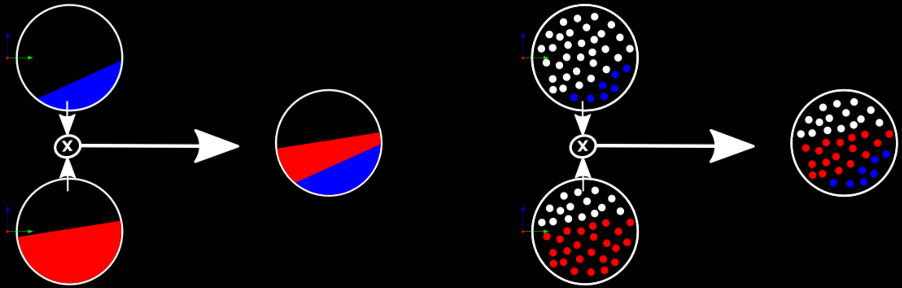
1. Macro-surface model

Differential Cone Tracing Scheme



1. Macro-surface model

We represent the subpixel occlusion distribution with pre-computed binary mask and combine them in a A-buffer way



1. Macro-surface model

Results

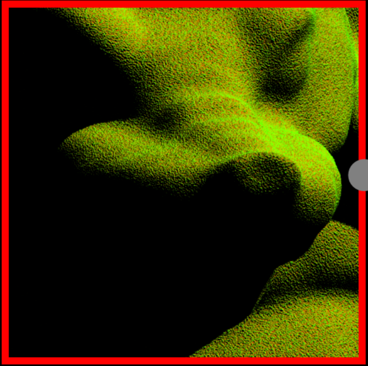
1. Macro-surface model

Results

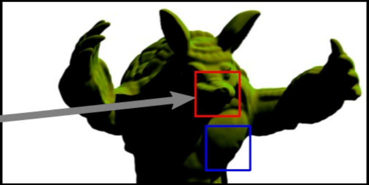
1. Macro-surface model

1. Macro-surface model
2. Micro-surface model

How to represent complex details from far-away ?



?



$$I = \frac{\int_A E_{\mathbf{l}}\, a\, \rho_{\mathbf{v},\mathbf{l}}\, V_{\mathbf{l}}\, V_{\mathbf{v}}\, \mathbf{nl}\, \mathbf{nv}\, d\mathbf{p}}{\int_A V_{\mathbf{v}}\, \mathbf{nv}\, d\mathbf{p}}$$

A Hierarchy of Scales

- Anisotropic reflection models

  [Kajiya 85]

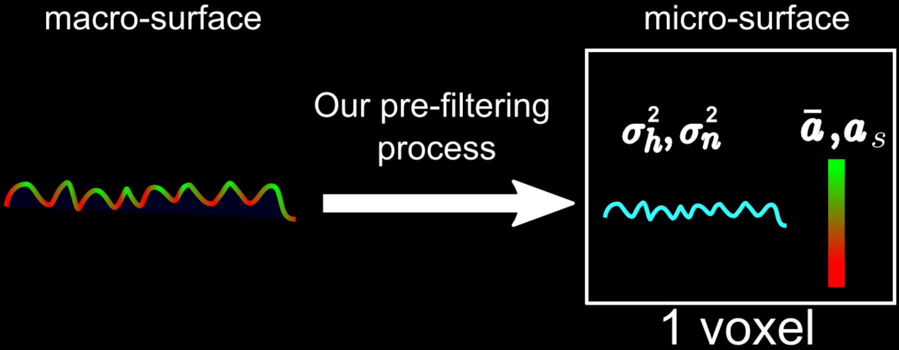
- Normal distribution functions and multiple surfaces

  [Fournier 92]

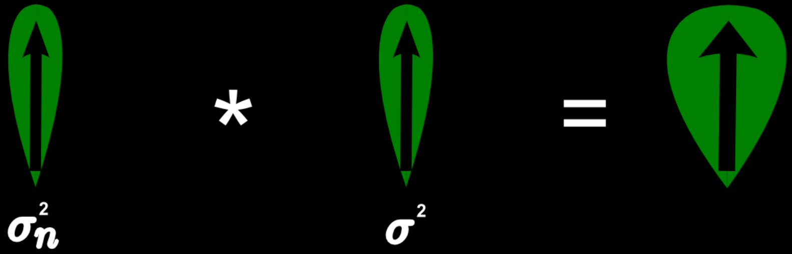
- Smooth transitions between bump rendering algorithms

  [Becker and Max 93]

Pre-filtering Process



Normal Pre-filtering Strategy



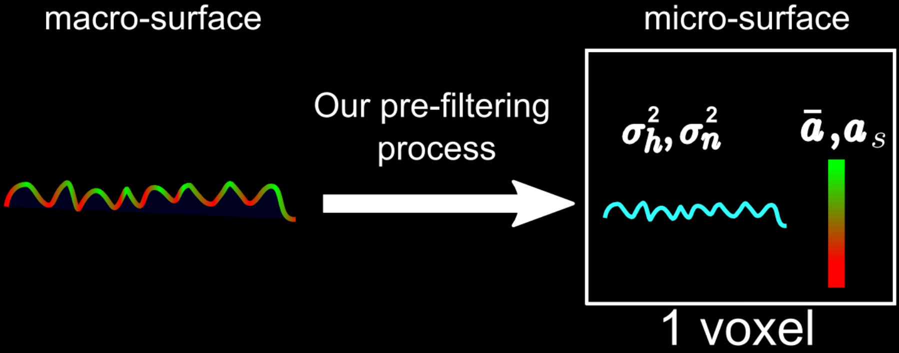
Gaussian slope statistics of the micro-surface

ground material roughness (BRDF with Gaussian distribution)

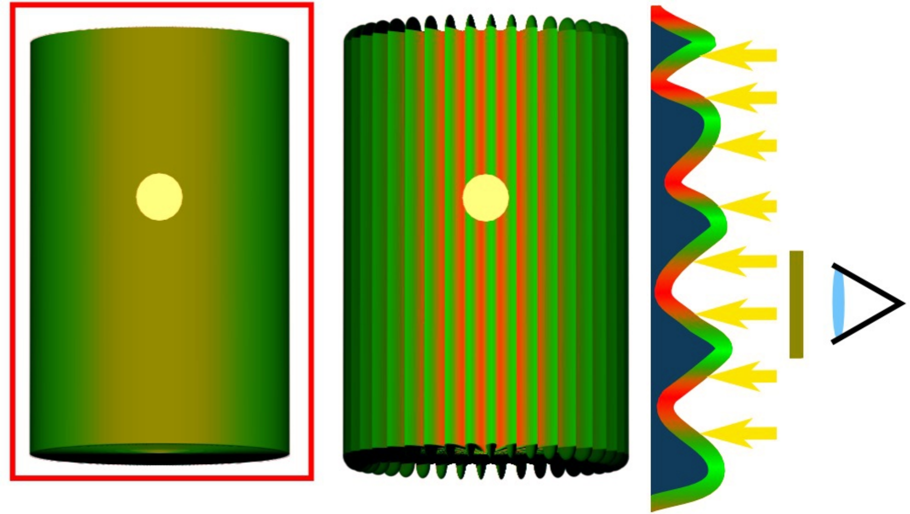
macro-BRDF used for shading

Normal Pre-filtering Result

Pre-filtering Process

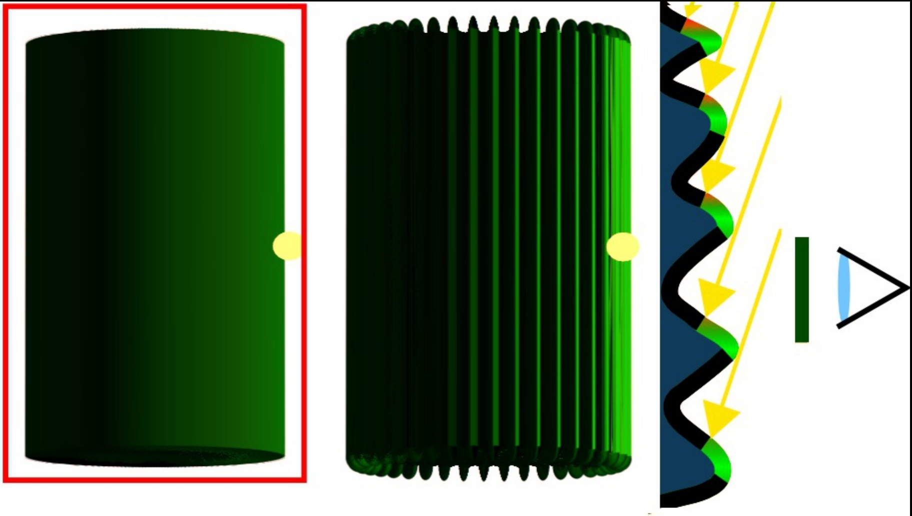


Pre-filtering Masking Effects



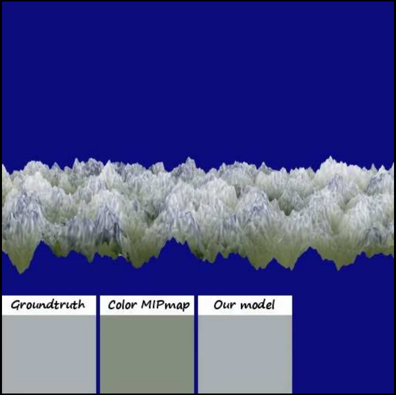
| Our micro-surface model || Macroscopic behaviour |

Pre-filtering Shadowing Effects



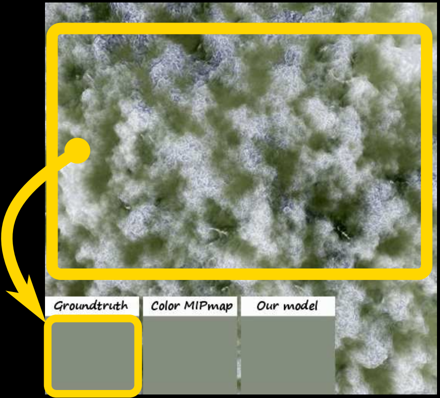
| Our micro-surface model || Macroscopic behaviour |

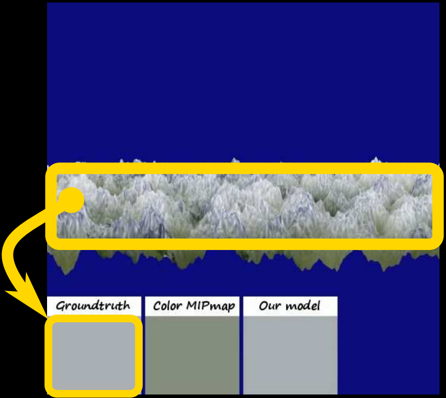
Validation on Real Textures



A real-world texture
showing strong correlation between colors and heights

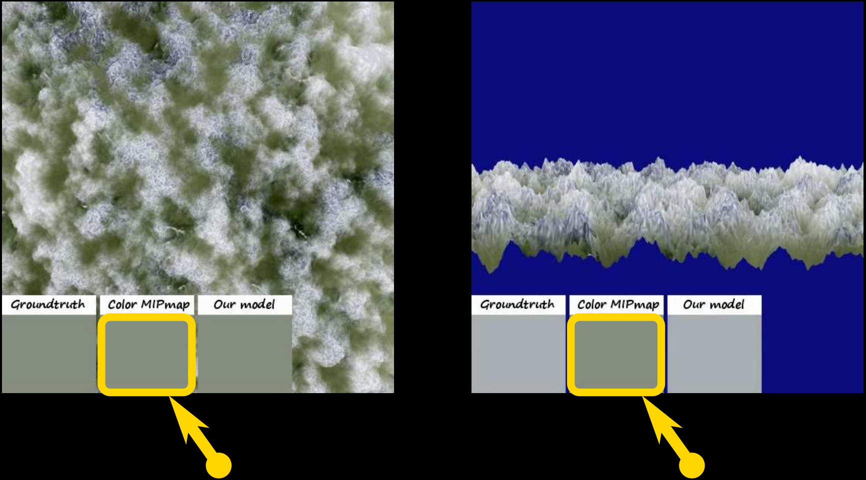
Validation on Real Textures





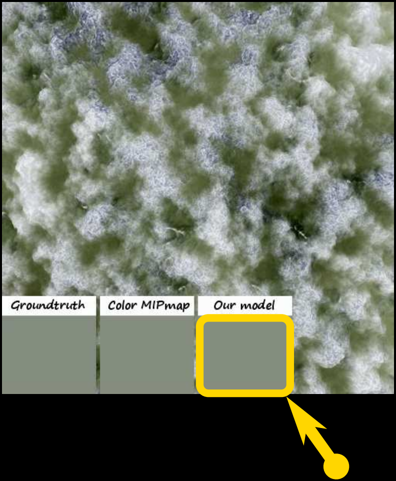
The groundtruth is the average visible color
It is view-dependent

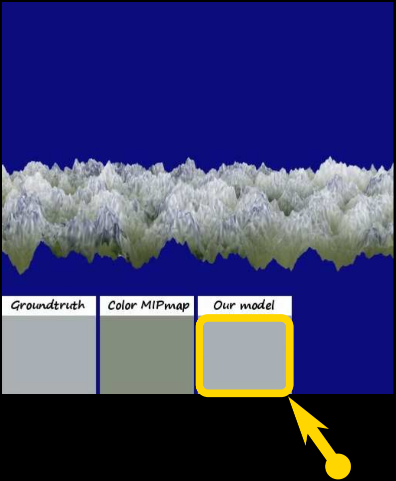
Validation on Real Textures



The mipmapped color is the average of the texture
It is NOT view-dependent

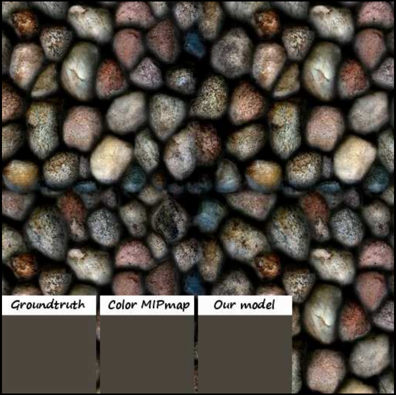
Validation on Real Textures





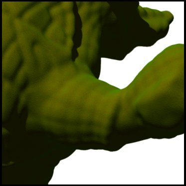
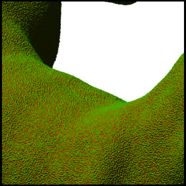
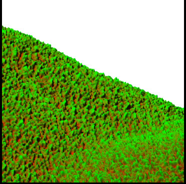
Our model is view-dependent
It is a better approximation than mipmap

Validation on Real Textures



Our model is view-dependent
It is a better approximation than mipmap

Performances

fps	57	37	25	19
μs/pix	0.26	0.13	0.22	0.32

The timing per covered pixel is consistent through scales

Conclusion

Properties of our model

- Scalable
- Accurate appearance pre-filtering
- Seamless LoD transition

- Differential cone tracing scheme
  - works also with implicit distance fields
- Micro-surface representation
  - works also with textures and height-maps

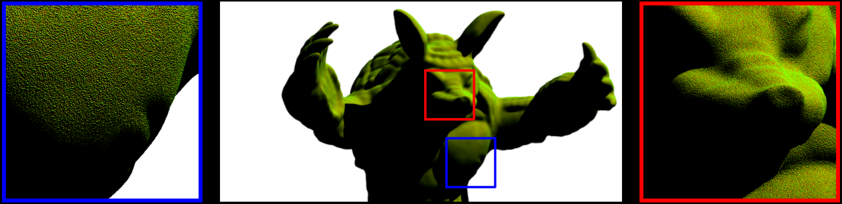
Future Work

Limitation of our approach

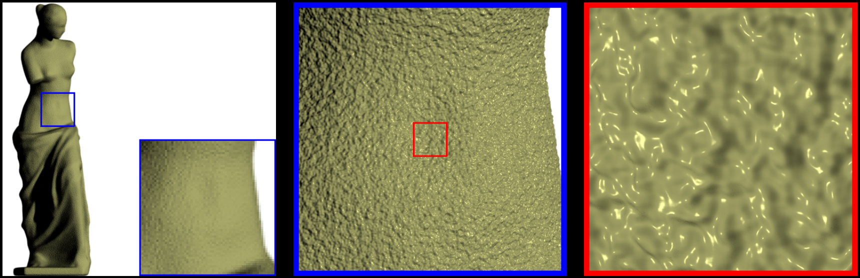
- B-rep objects

Other desirable features

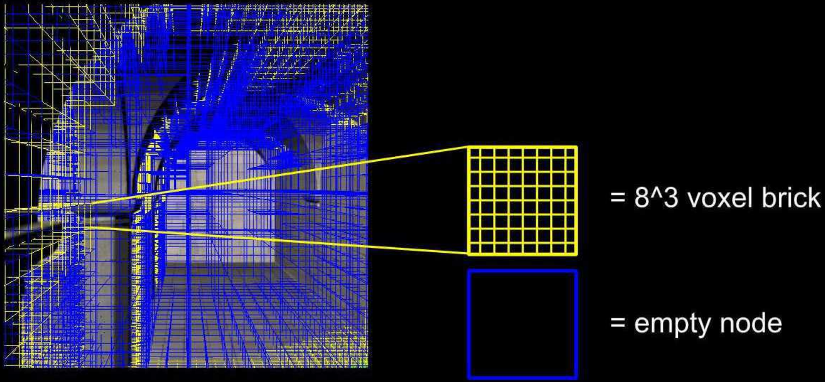
- Semi-transparent material, volume rendering
- Pre-filtering other kinds of correlation



Thank you for your attention!



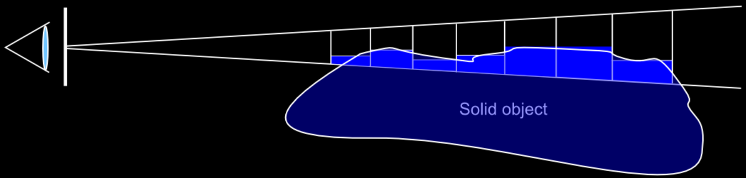
The GigaVoxels [CNLE09] octree structure stores voxel bricks at each node instead of only one voxel
→ hardware interpolation possible

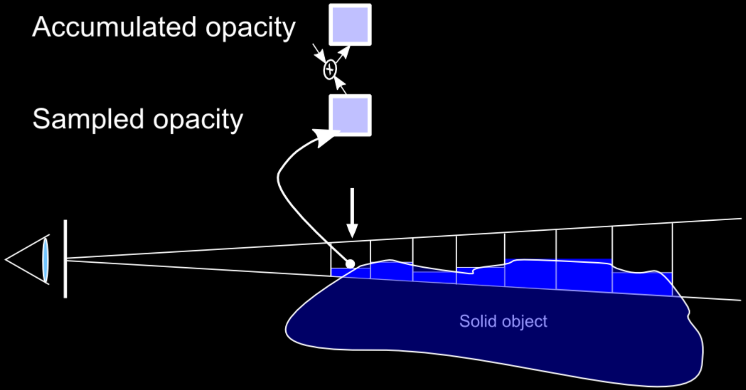


GigaVoxels octree structure

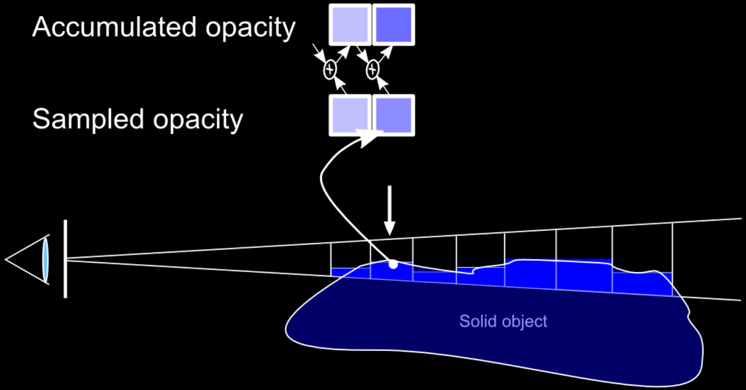
Accumulated opacity

Sampled opacity

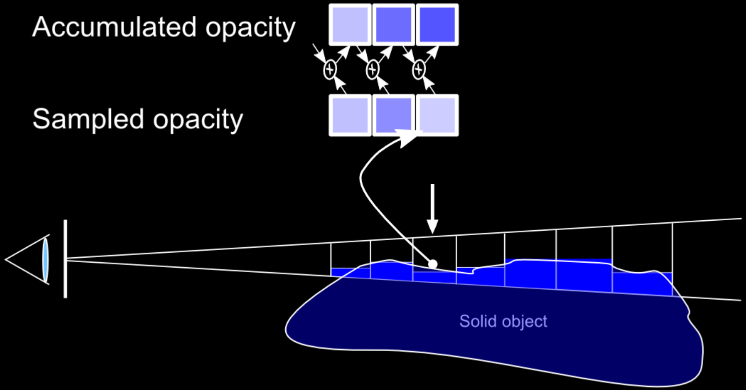




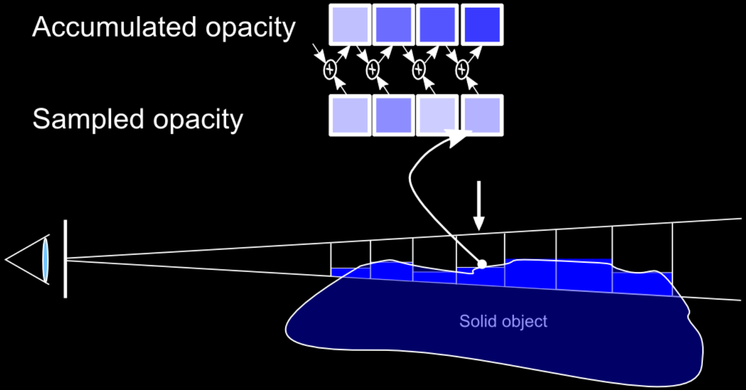
Density-based rendering



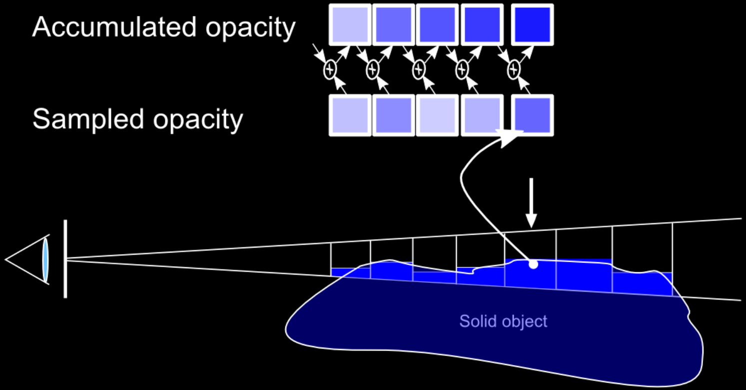
Density-based rendering



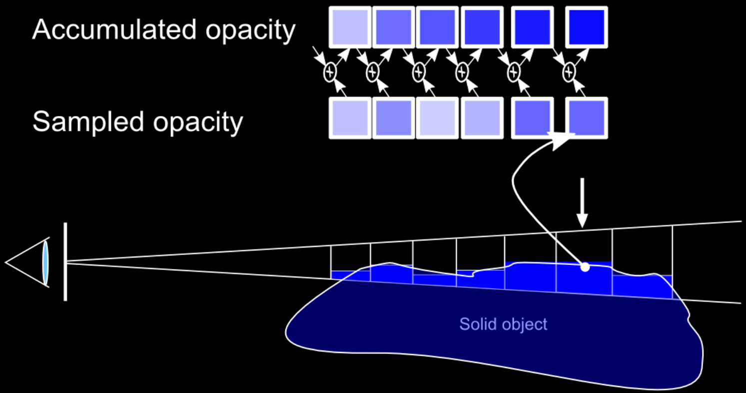
Density-based rendering



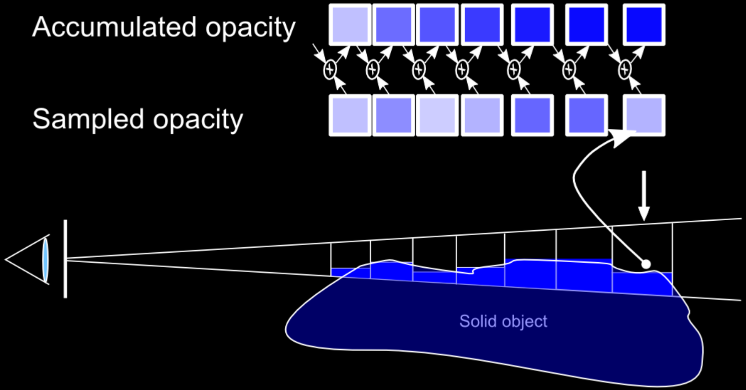
Density-based rendering



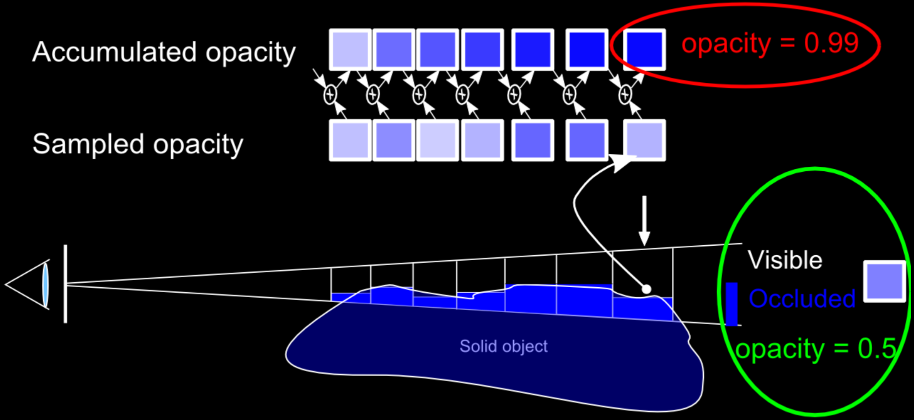
Density-based rendering



Density-based rendering



Density-based rendering



Density-based rendering