

A Meshless Hierarchical Representation for Light Transport

Jaakko Lehtinen^{1,2}

Matthias Zwicker³

Emmanuel Turquin^{4,5}

Janne Kontkanen⁶

Frédo Durand¹

François Sillion^{5,4}

Timo Aila⁷

¹MIT CSAIL

²TKK

³UCSD

⁴Grenoble University

⁵INRIA

⁶PDI/DreamWorks

⁷NVIDIA Research



I believe most of us will agree that interactive Global illumination with moving lights and cameras in complex environments such as this one is a challenging problem.



I believe most of us will agree that interactive Global illumination with moving lights and cameras in complex environments such as this one is a challenging problem.



I believe most of us will agree that interactive Global illumination with moving lights and cameras in complex environments such as this one is a challenging problem.



I believe most of us will agree that interactive Global illumination with moving lights and cameras in complex environments such as this one is a challenging problem.

How?

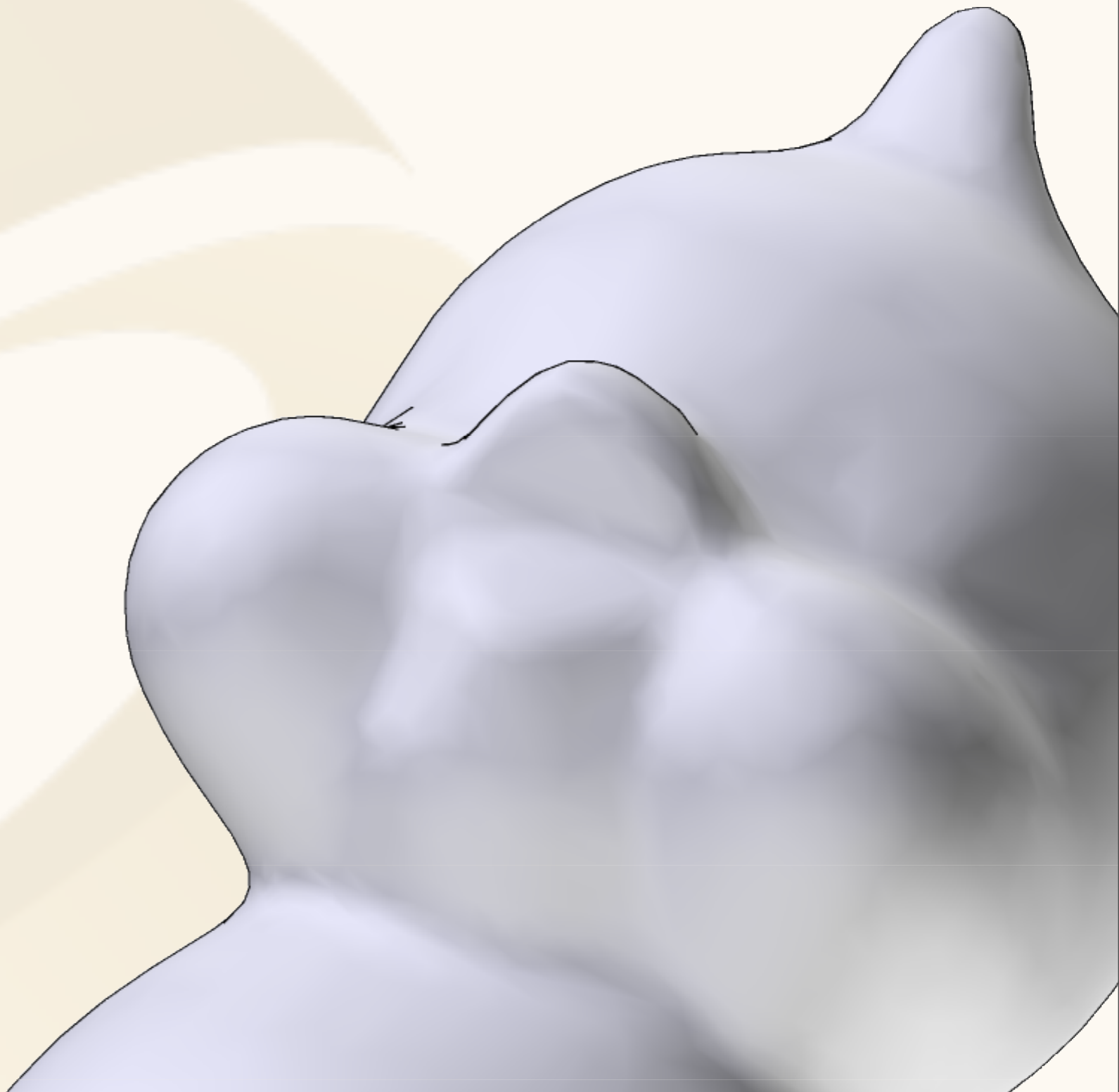
- **Store lighting on surfaces**
 - Enables quick reconstruction from any viewpoint
- **Examples**
 - PRT techniques store spatially varying transfer matrices [Sloan 02, Ng 03, ..., ..., ...]
 - FEM techniques store irradiance/radiosity/radiance

Many techniques tackle this problem by precomputing and storing some sort of lighting functions on the surfaces of the scene, often using basis functions. The solutions can then be easily visualized from any viewpoint.

For example, Precomputed Radiance Transfer techniques store spatially varying transfer matrices that encode the appearance of surface points in terms of input lighting, while traditional finite element methods store radiance or radiosity in fixed lighting conditions.

Some Usual Approaches

- **Piecewise linear vertex basis**
 - Flexible, easy
 - Not adaptive

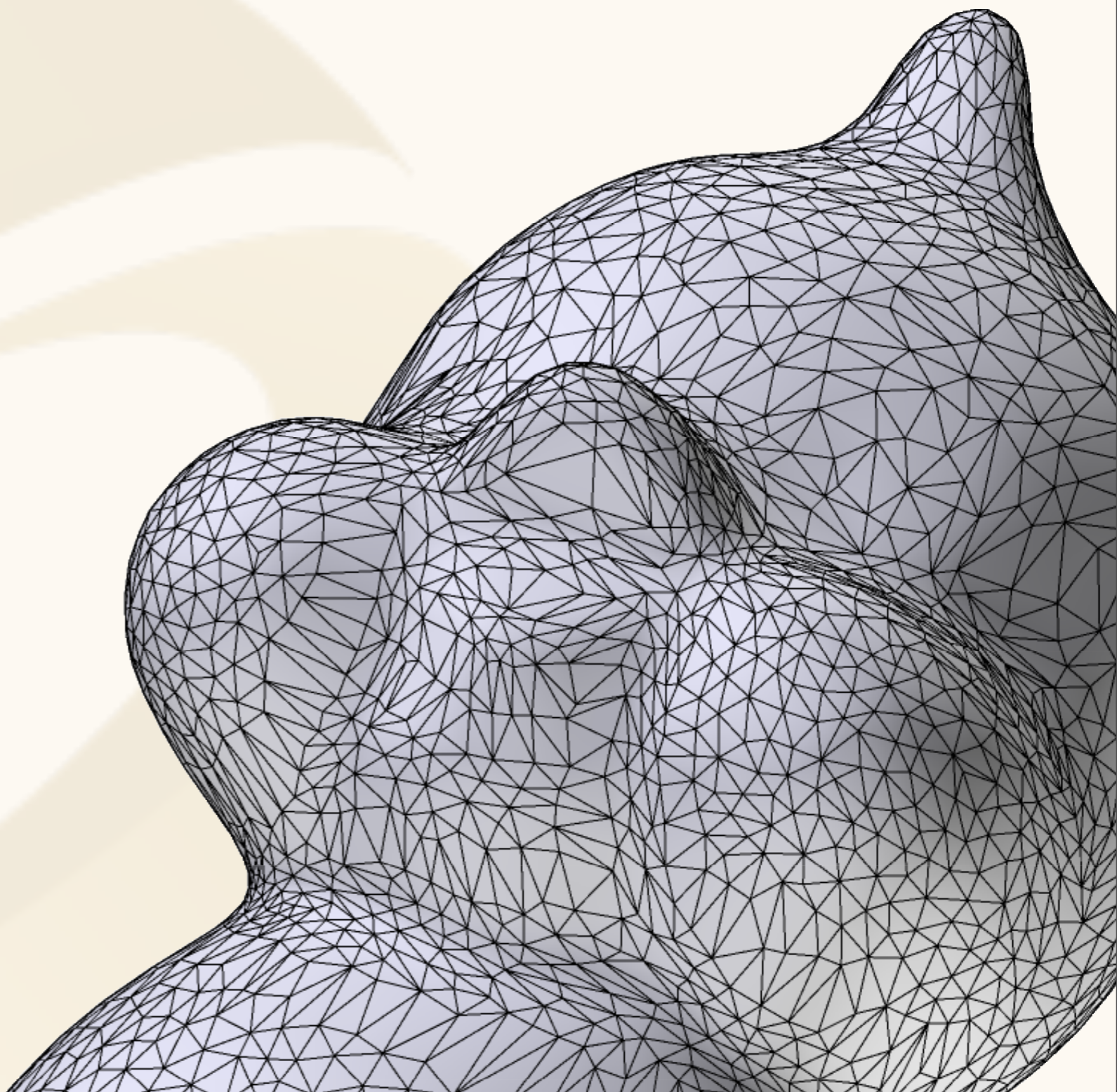


Let's take a look at the most common approach for capturing spatial variation, linear interpolation over triangles. The lighting function is sampled at the vertices, and the results are linearly blended across the triangle. While this is easy and general, you have to sample at all of them to get a complete reconstruction, which is a lot of work in a complex scene.

In other words, the sampling cannot be adapted to the frequency content of the signal being approximated. Similar arguments apply to other nonhierarchical bases.

Some Usual Approaches

- **Piecewise linear vertex basis**
 - Flexible, easy
 - Not adaptive

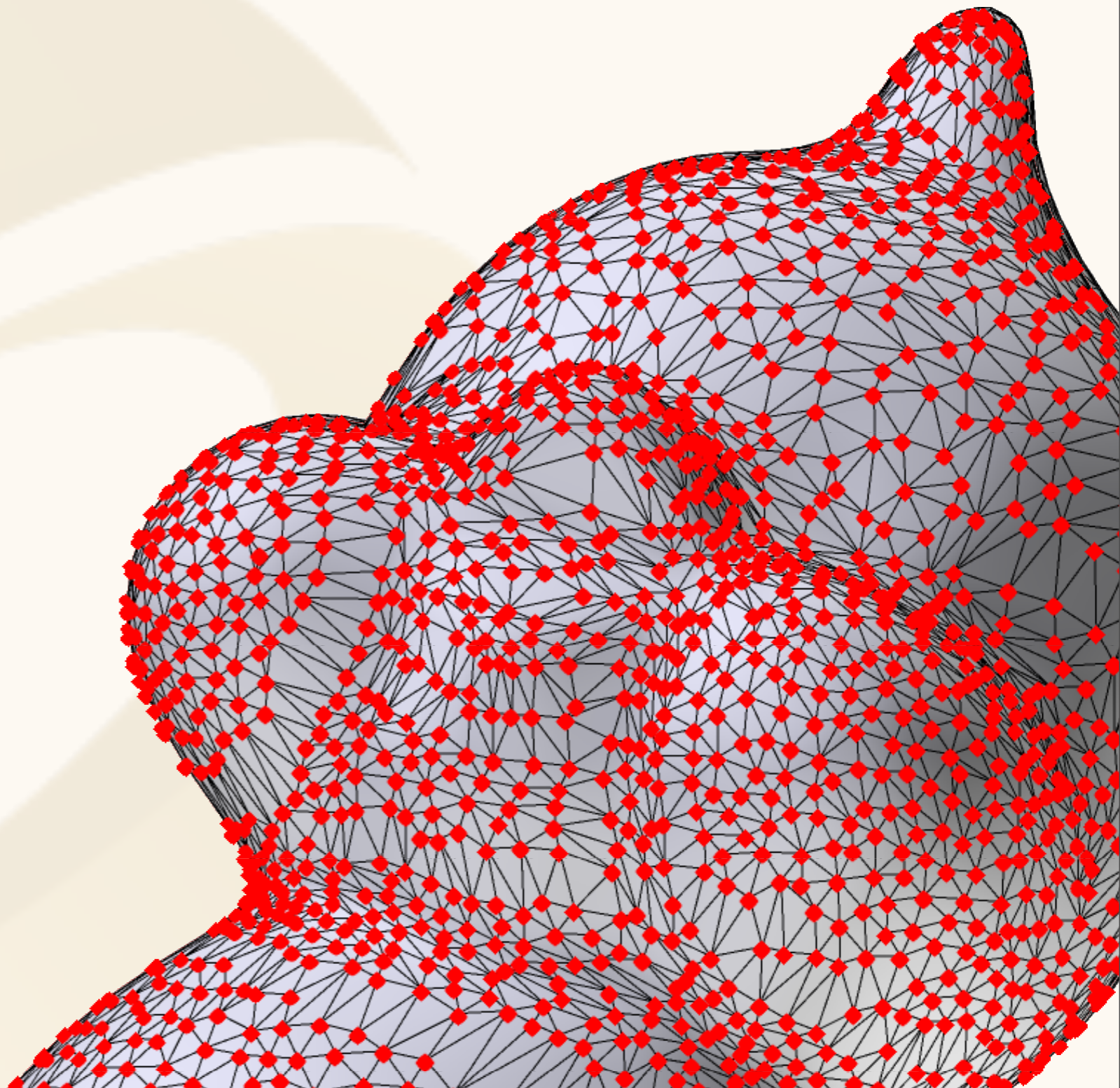


Let's take a look at the most common approach for capturing spatial variation, linear interpolation over triangles. The lighting function is sampled at the vertices, and the results are linearly blended across the triangle. While this is easy and general, you have to sample at all of them to get a complete reconstruction, which is a lot of work in a complex scene.

In other words, the sampling cannot be adapted to the frequency content of the signal being approximated. Similar arguments apply to other nonhierarchical bases.

Some Usual Approaches

- **Piecewise linear vertex basis**
 - Flexible, easy
 - Not adaptive



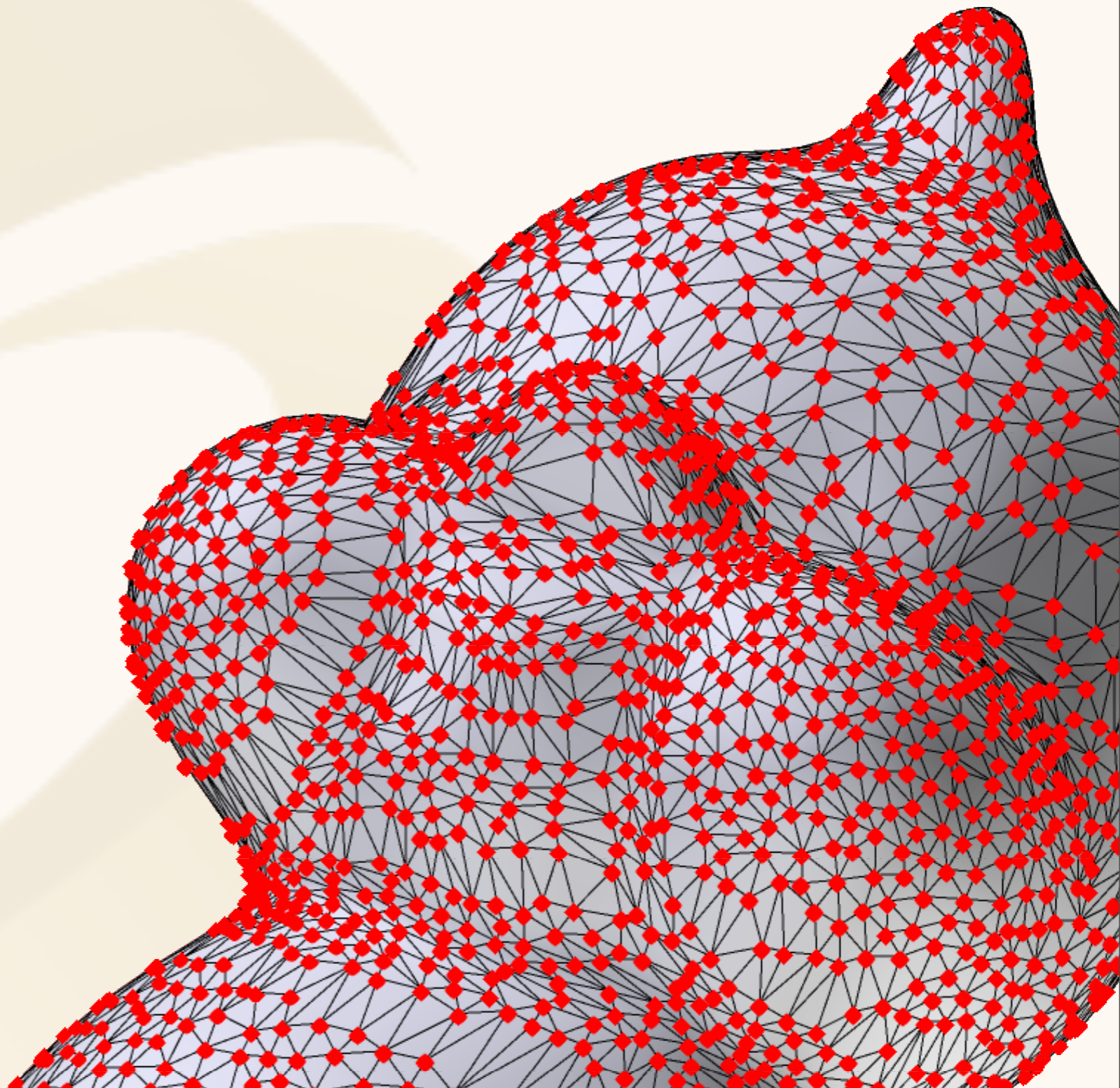
Let's take a look at the most common approach for capturing spatial variation, linear interpolation over triangles. The lighting function is sampled at the vertices, and the results are linearly blended across the triangle. While this is easy and general, you have to sample at all of them to get a complete reconstruction, which is a lot of work in a complex scene.

In other words, the sampling cannot be adapted to the frequency content of the signal being approximated. Similar arguments apply to other nonhierarchical bases.

Some Usual Approaches

- **Piecewise linear vertex basis**

- Flexible, easy



Let's take a look at the most common approach for capturing spatial variation, linear interpolation over triangles. The lighting function is sampled at the vertices, and the results are linearly blended across the triangle. While this is easy and general, you have to sample at all of them to get a complete reconstruction, which is a lot of work in a complex scene.

In other words, the sampling cannot be adapted to the frequency content of the signal being approximated. Similar arguments apply to other nonhierarchical bases.

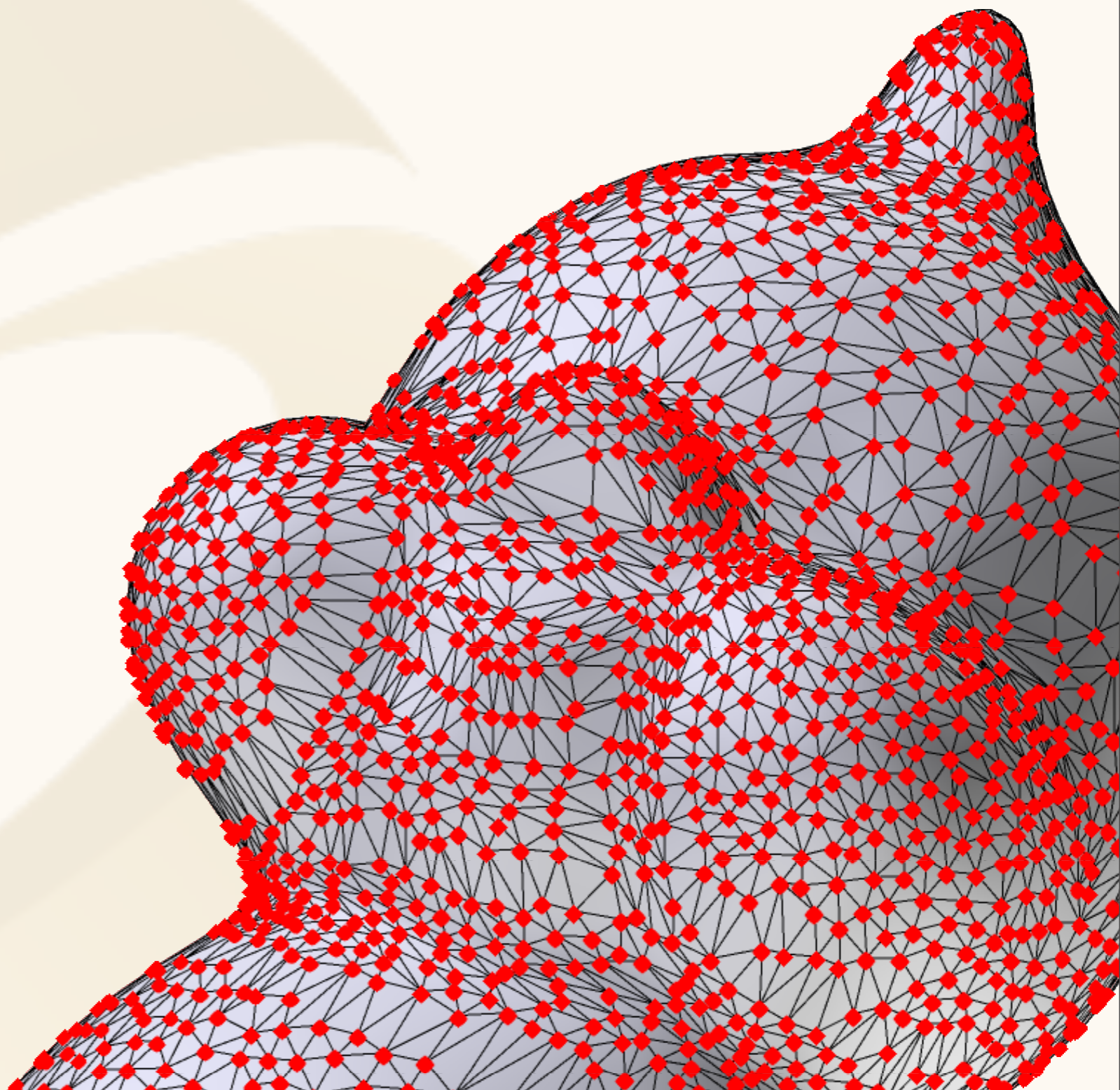
Some Usual Approaches

- **Piecewise linear vertex basis**

- Flexible, easy



- Not adaptive



Let's take a look at the most common approach for capturing spatial variation, linear interpolation over triangles. The lighting function is sampled at the vertices, and the results are linearly blended across the triangle. While this is easy and general, you have to sample at all of them to get a complete reconstruction, which is a lot of work in a complex scene.

In other words, the sampling cannot be adapted to the frequency content of the signal being approximated. Similar arguments apply to other nonhierarchical bases.

Some Usual Approaches

- **Piecewise linear vertex basis**

- Flexible, easy



- Not adaptive



- **Hierarchical**

- Fast

- Difficult

To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches

- **Piecewise linear vertex basis**

- Flexible, easy



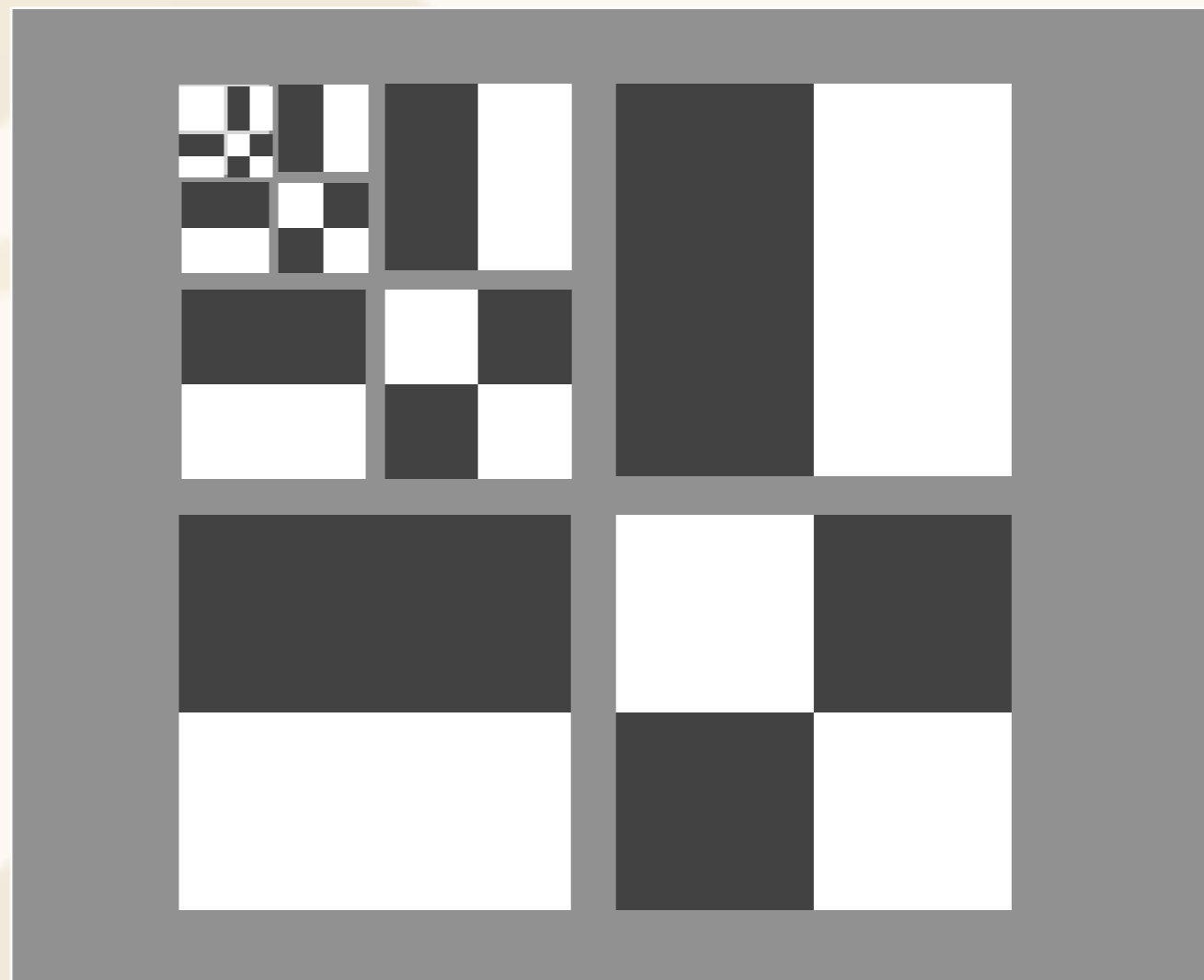
- Not adaptive



- **Hierarchical**

- Fast

- Difficult



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches

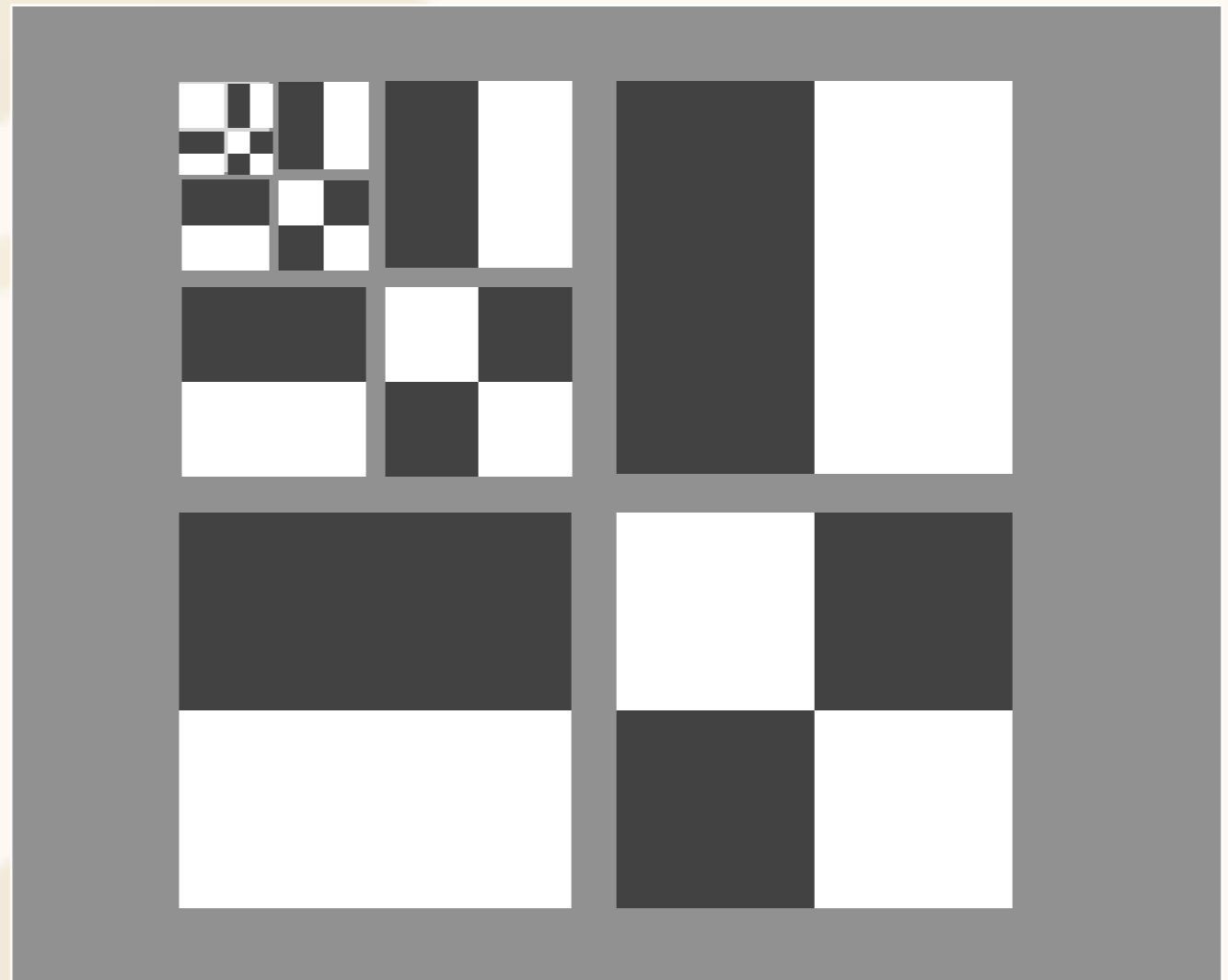
- **Piecewise linear vertex basis**

- Flexible, easy ✓

- Not adaptive ✗

- **Hierarchical**

- Fast ✓



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches

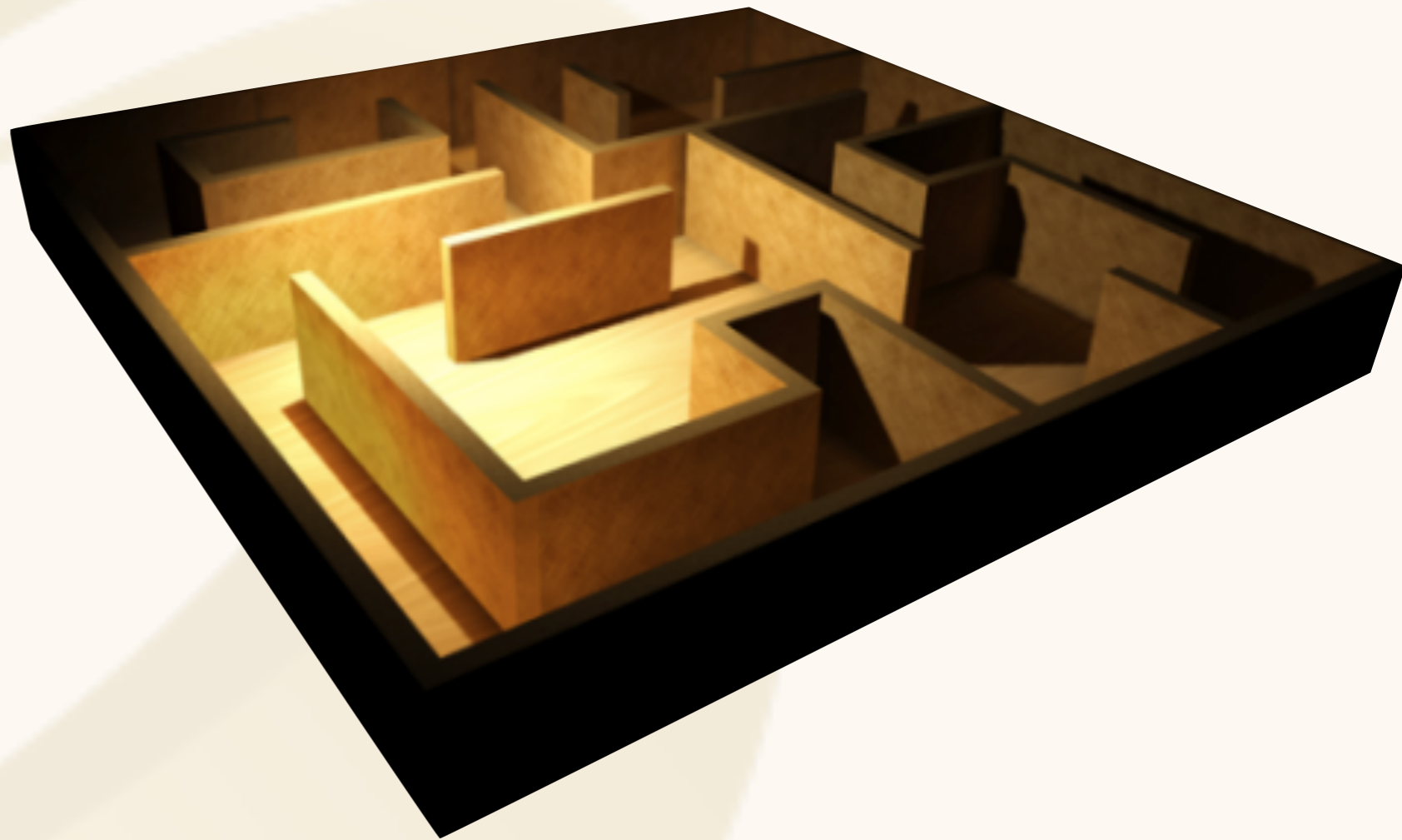
- **Piecewise linear vertex basis**

- Flexible, easy ✓

- Not adaptive ✗

- **Hierarchical**

- Fast ✓



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches



- **Piecewise linear vertex basis**

- Flexible, easy



- Not adaptive



- **Hierarchical**

- Fast



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches



- **Piecewise linear vertex basis**

- Flexible, easy

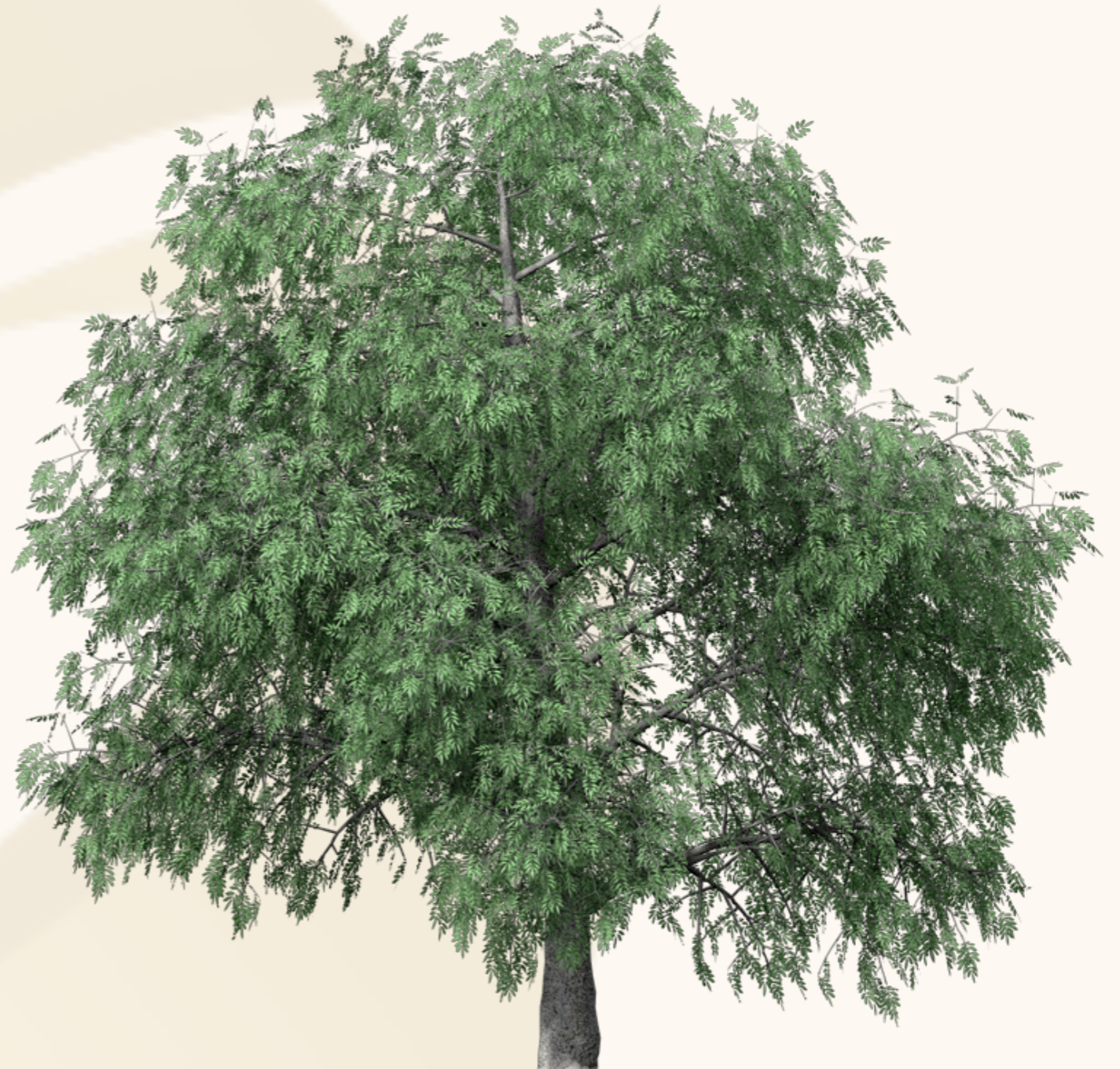


- Not adaptive



- **Hierarchical**

- Fast



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Some Usual Approaches



- **Piecewise linear vertex basis**

- Flexible, easy



- Not adaptive

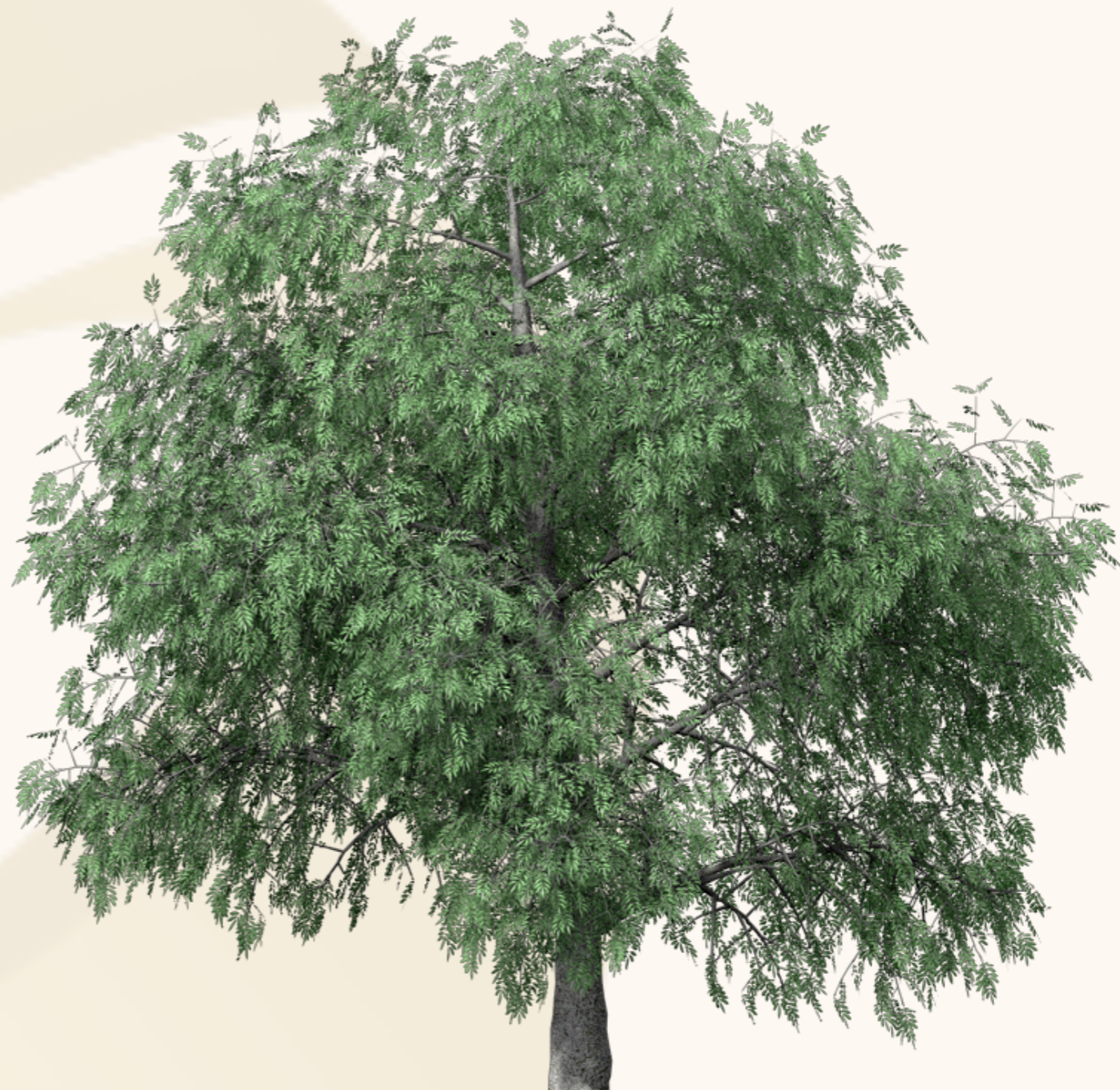


- **Hierarchical**

- Fast



- Difficult



To get adaptive resolution, you can, for instance, paste your favorite wavelet basis on the surfaces. The multiresolution representation allows computations to take place at the appropriate level of detail; this makes many algorithms really much faster. This is all great when the geometry is simple enough such that it allows a nice 2D parameterization.

But in cases when you have complex geometry, perhaps with topologically disjoint components such as cobblestones, or even tree foliage, or similar, you're pretty much out of luck.

Goals for Representation

Flexible and Simple

Smooth Reconstruction

Hierarchical

Independent of Geometric Representation

In all, we would like a basis that

shares the simplicity and ease of use of the piecewise linear per-vertex interpolation,

offers pleasing smooth reconstruction on smooth surfaces, with no artificial seams

that is hierarchical to enable adaptive computations,

and furthermore, which would be as decoupled from the actual surface representation as possible.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

Rendering on GPU

Discussion and Conclusions

Previous Work

Hierarchical Surface Bases

- **HR**

[Hanrahan 91, ...]

- **Wavelets**

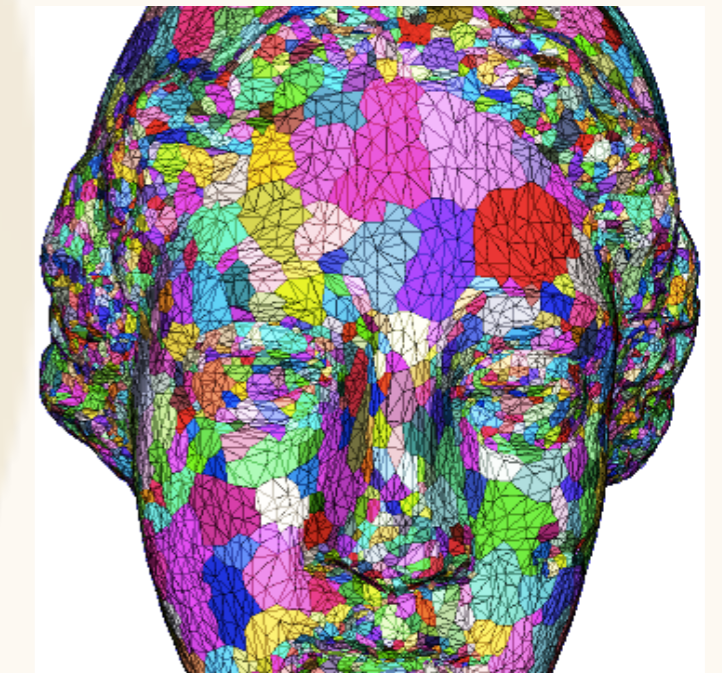
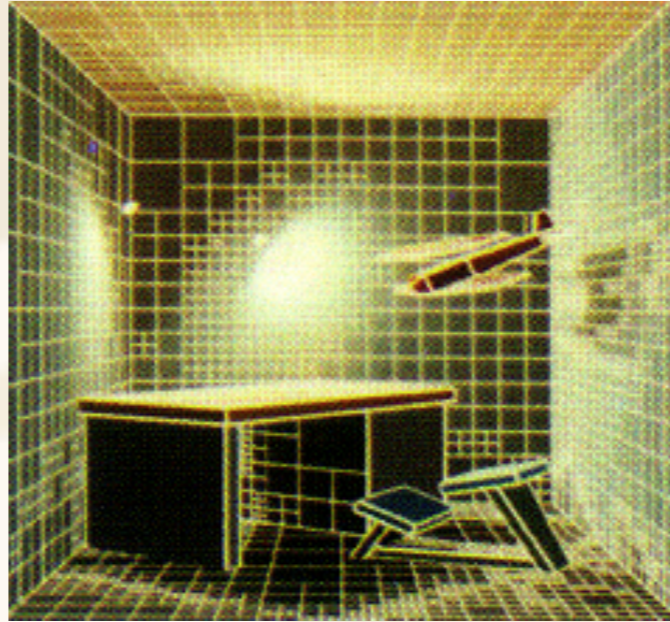
[Gortler 93, Lecot 05, Kontkanen 06, ...]

- **Volume clusters**

[Sillion 95, ...]

- **Face clusters**

[Willmott 99, ...]



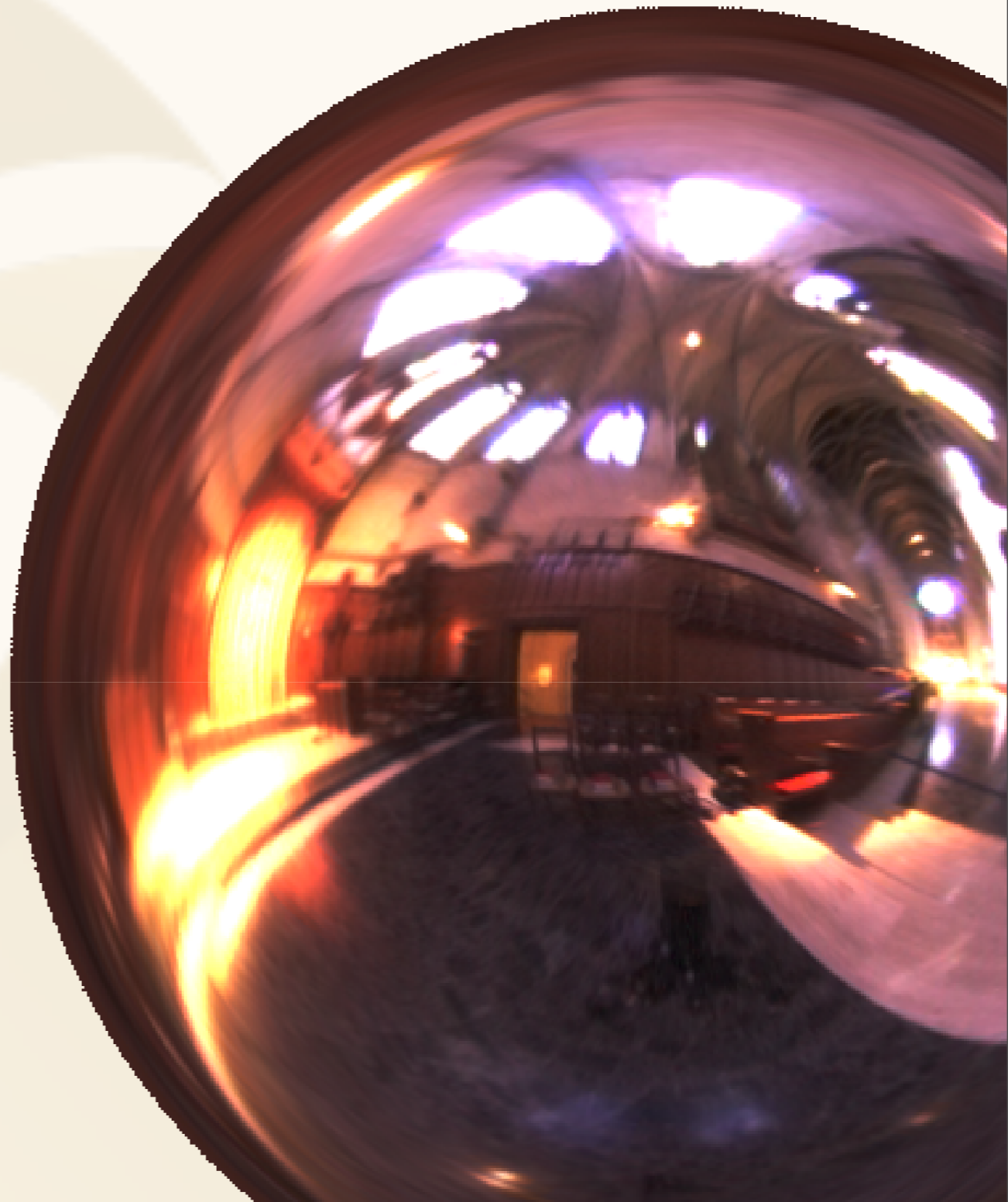
Lots of work has been done on hierarchical illumination algorithms. This includes hierarchical radiosity, wavelet radiosity and its glossy derivatives, volume clustering techniques and face clustering techniques.

We would like to employ similar multiresolution algorithms, but without the need to parameterize or mesh the surfaces, as is required by all these previous approaches.

Previous Work

PRT

- **Focus on angular domain for illumination**
 - hierarchical, wavelets
[Ng 03, Liu 04, Wang 07, ...]
- **Spatial: usually per-vertex, no hierarchy**
 - **Exception:**
[Kontkanen 06] has hierarchy, but only simple geometry



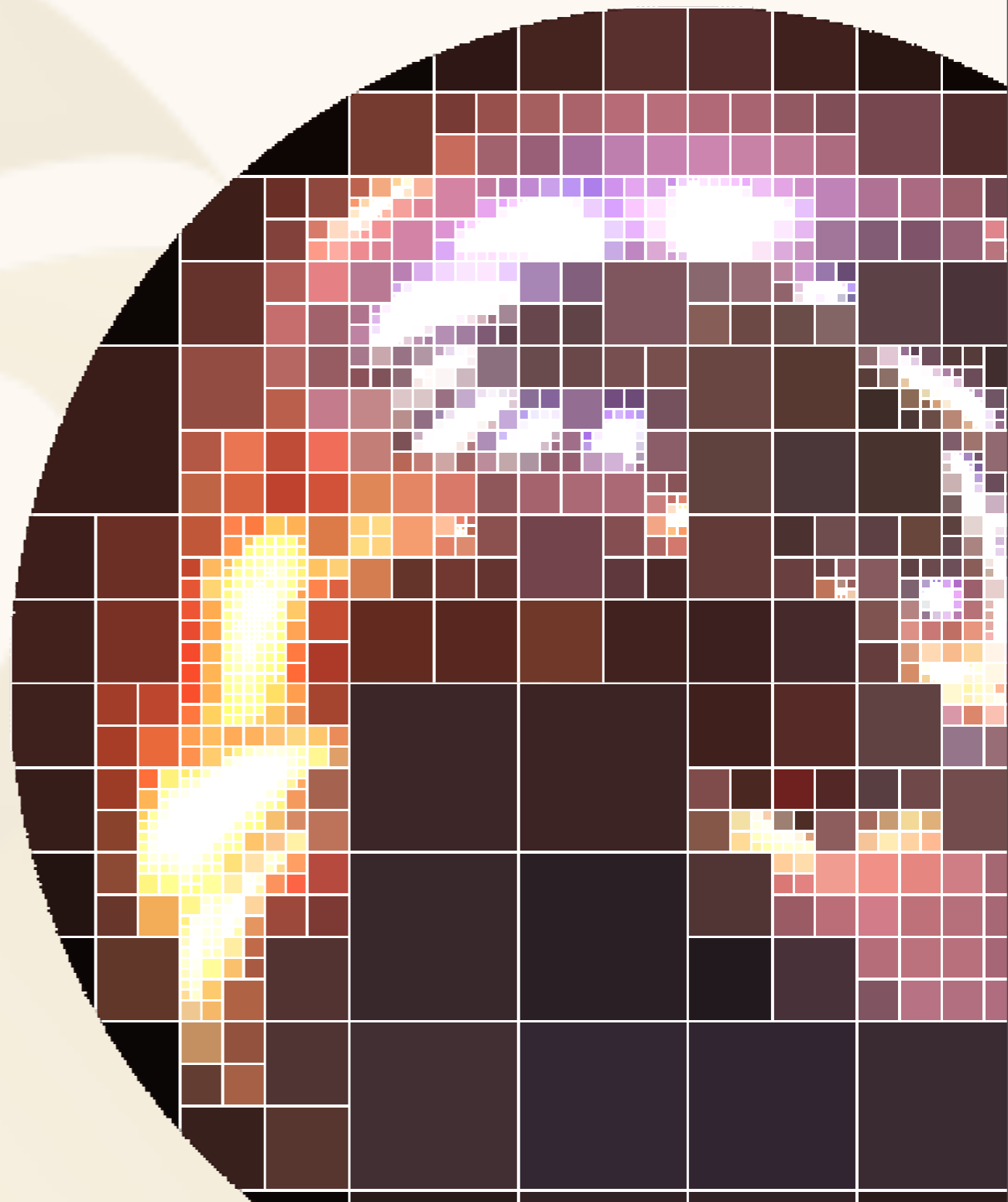
Most Precomputed Radiance Transfer or PRT work concentrates more on the efficient representation of distant incident illumination. Hierarchical wavelet bases are often employed to enable all-frequency relighting.

However, in the spatial domain, these methods usually resort to standard non-hierarchical piecewise linear basis functions, which means slow precomputation and necessitates compression to make the dataset small enough. The Kontkanen PRT technique for local light sources is a notable exception; they use Haar wavelets on the surfaces, but unfortunately this rules out complex scenes.

Previous Work

PRT

- **Focus on angular domain for illumination**
 - hierarchical, wavelets
[Ng 03, Liu 04, Wang 07, ...]
- **Spatial: usually per-vertex, no hierarchy**
 - **Exception:**
[Kontkanen 06] has hierarchy, but only simple geometry



Most Precomputed Radiance Transfer or PRT work concentrates more on the efficient representation of distant incident illumination. Hierarchical wavelet bases are often employed to enable all-frequency relighting.

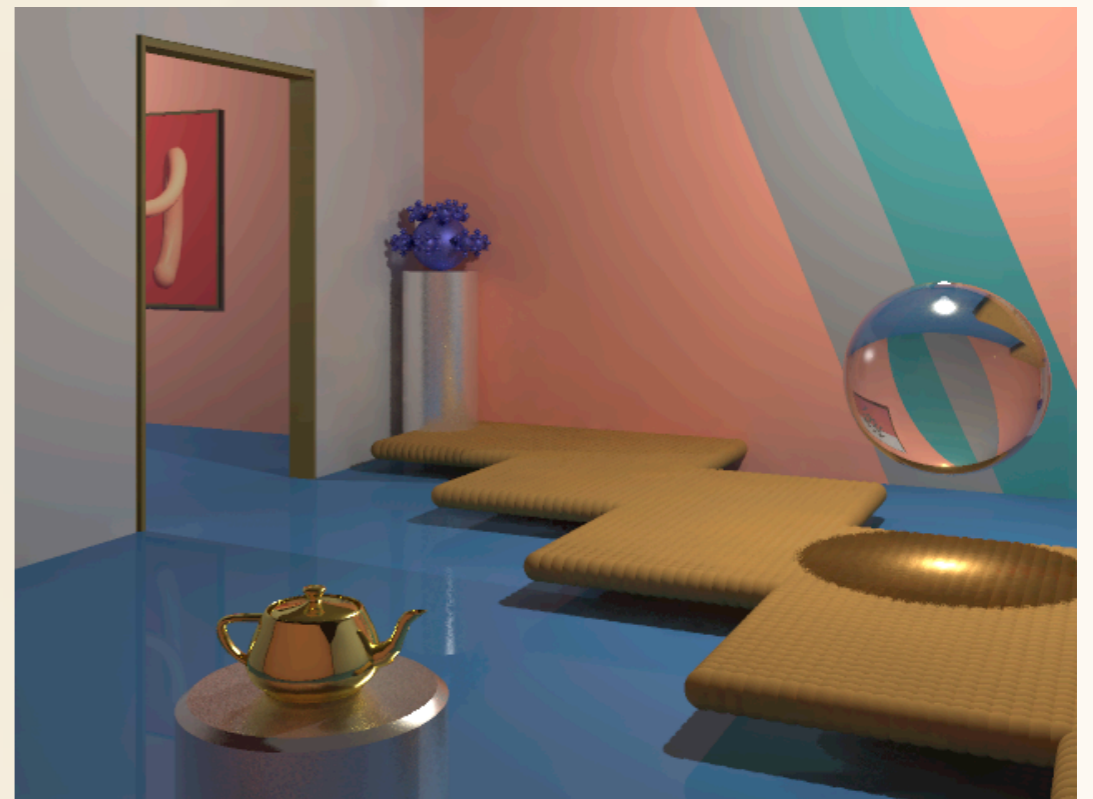
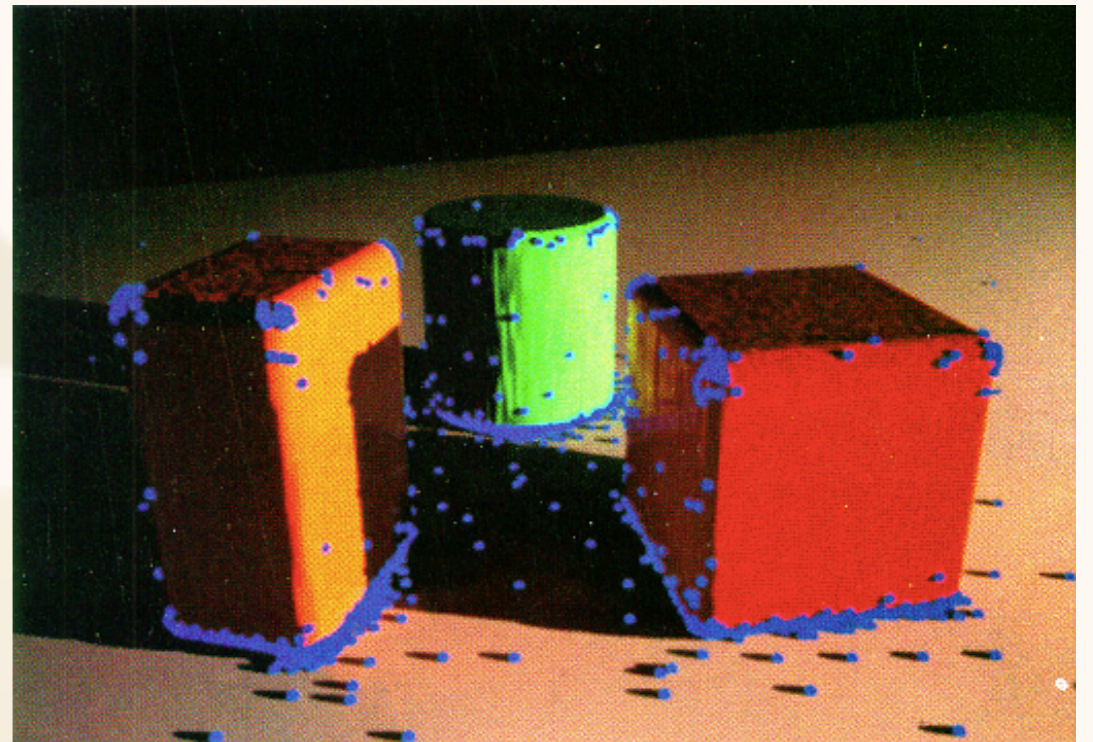
However, in the spatial domain, these methods usually resort to standard non-hierarchical piecewise linear basis functions, which means slow precomputation and necessitates compression to make the dataset small enough. The Kontkanen PRT technique for local light sources is a notable exception; they use Haar wavelets on the surfaces, but unfortunately this rules out complex scenes.

Previous Work

Off-line Rendering

- **Point-based illumination representation and caching**
 - Irradiance caching [Ward 88]
 - Photon map [Jensen 96]

**Point samples
abstract geometry**



Point samples have been found really useful in offline rendering. The irradiance cache interpolates sparse illumination samples, while Photon mapping represents irradiance through the density of photons. In both approaches, the point samples decouple the geometric representation from the illumination algorithm, which is a very attractive property.

However in contrast to these techniques, for PRT and related techniques we need basis functions and a projection operator, and furthermore the solution needs to be visualized directly without final gathering. In addition, a multiresolution basis is essential for fast algorithms.

Previous Work

Meshless Representations

- **Meshless simulation and animation**

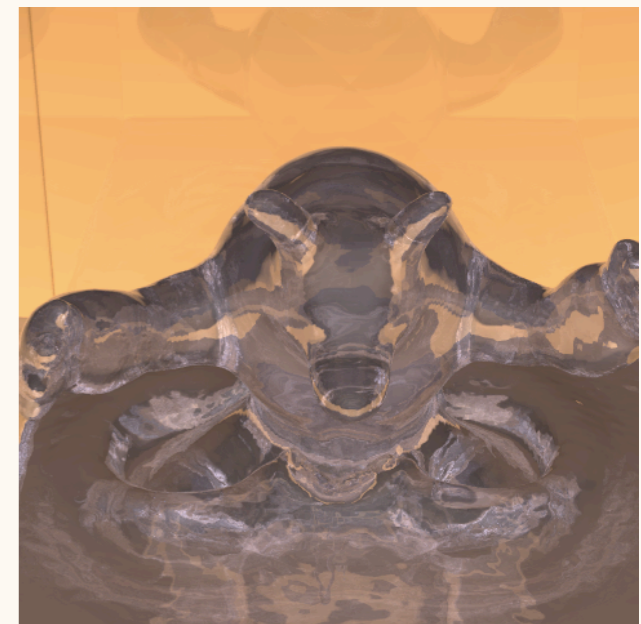
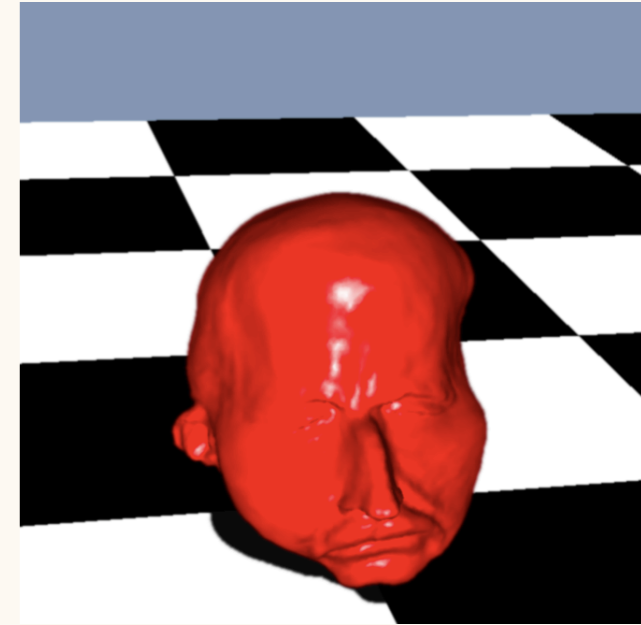
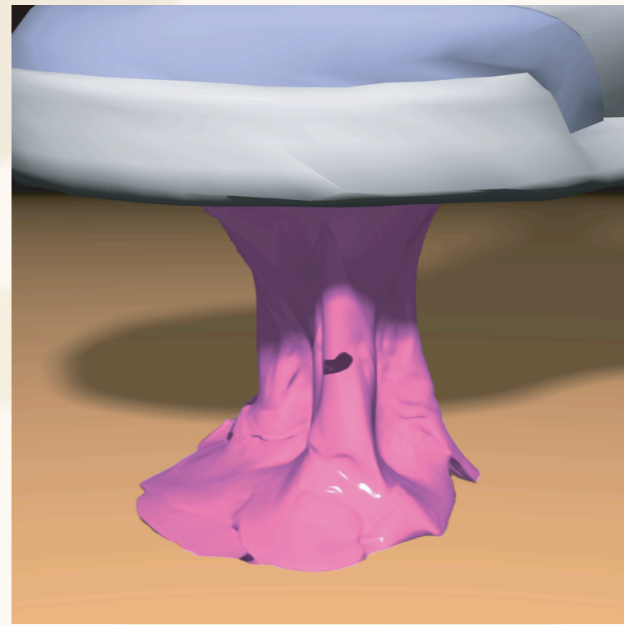
[Desbrun 96, Müller 03, Müller 04, Pauly 05, ...]

- **Point-based modeling**

[Pfister 00, Amenta 04, ...]

- **(Hierarchical) scattered data interpolation**

[Shepard 68, ...]



So-called Meshless Finite Element Techniques use basis functions defined without the use of a mesh. In graphics, meshless methods have been applied in simulation and animation of deformable bodies and fluids.

Furthermore, using points as a rendering and modeling primitive has been explored for example in the Surfel work and pointset surfaces defined through Moving Least Squares.

All of these techniques build on scattered data interpolation.

Contributions

- **Meshless hierarchical function basis**
 - geometry-independent multi-res algorithms
- **Algorithm for generating the basis**
- **Method for rendering directly on GPU**
- **Novel direct-to-indirect PRT algorithm**
 - complex geometry, efficient precomputation

We draw inspiration from all the work just outlined and present a novel meshless hierarchical function basis for light transport computations. As the name implies, the basis is not tied to a mesh and thus does not require meshing, clustering or parameterization, and allows multiscale representation of illumination on arbitrary geometry.

We also describe a simple algorithm for generating the basis, and a technique for rendering directly from the basis on the GPU.

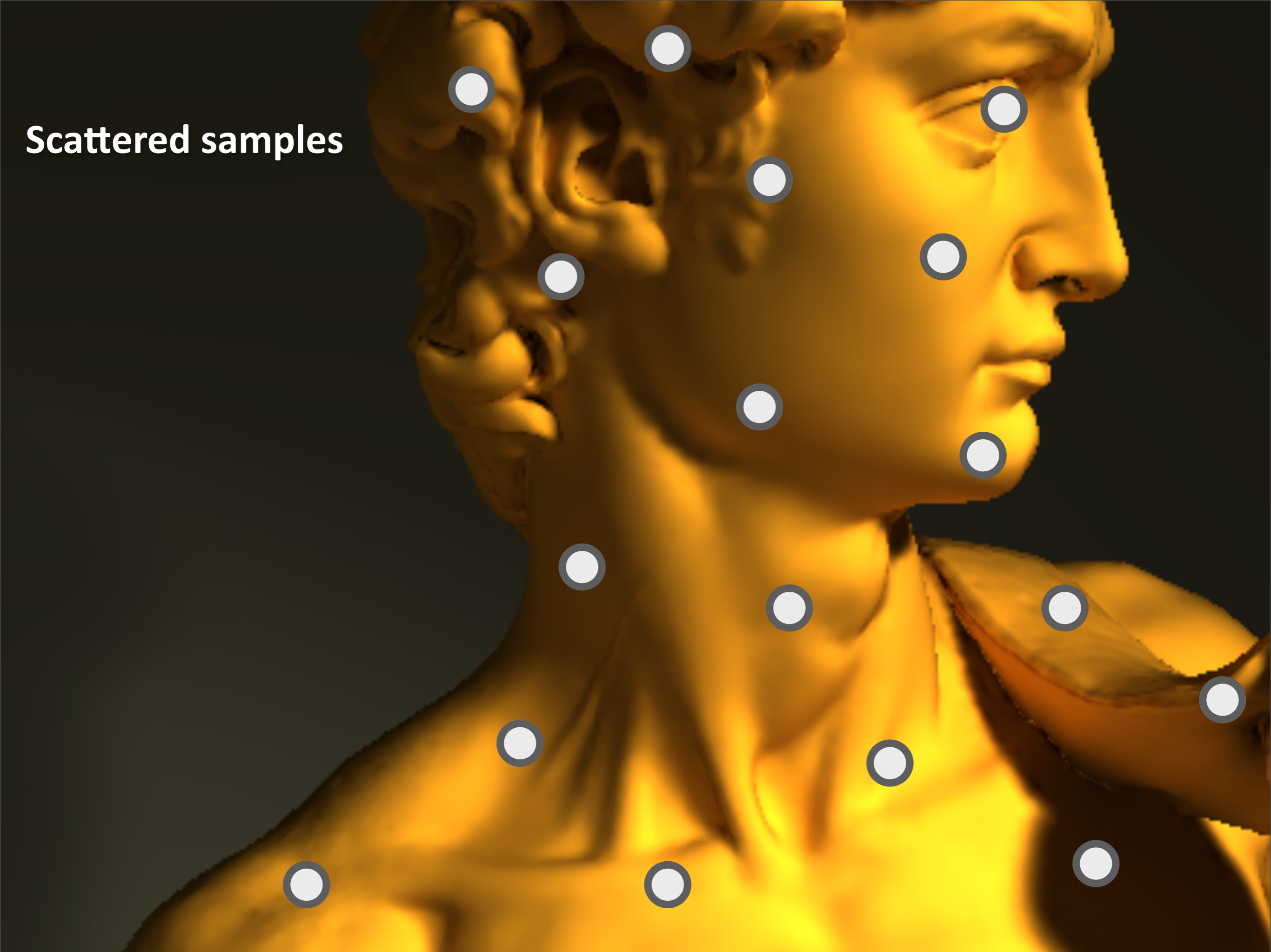
Finally, we apply the basis to PRT and describe an algorithm that allows interactive global illumination with moving local light sources on complex multi-million triangle scenes, and that has a fast precomputation step thanks to the hierarchy.

Overview of our approach



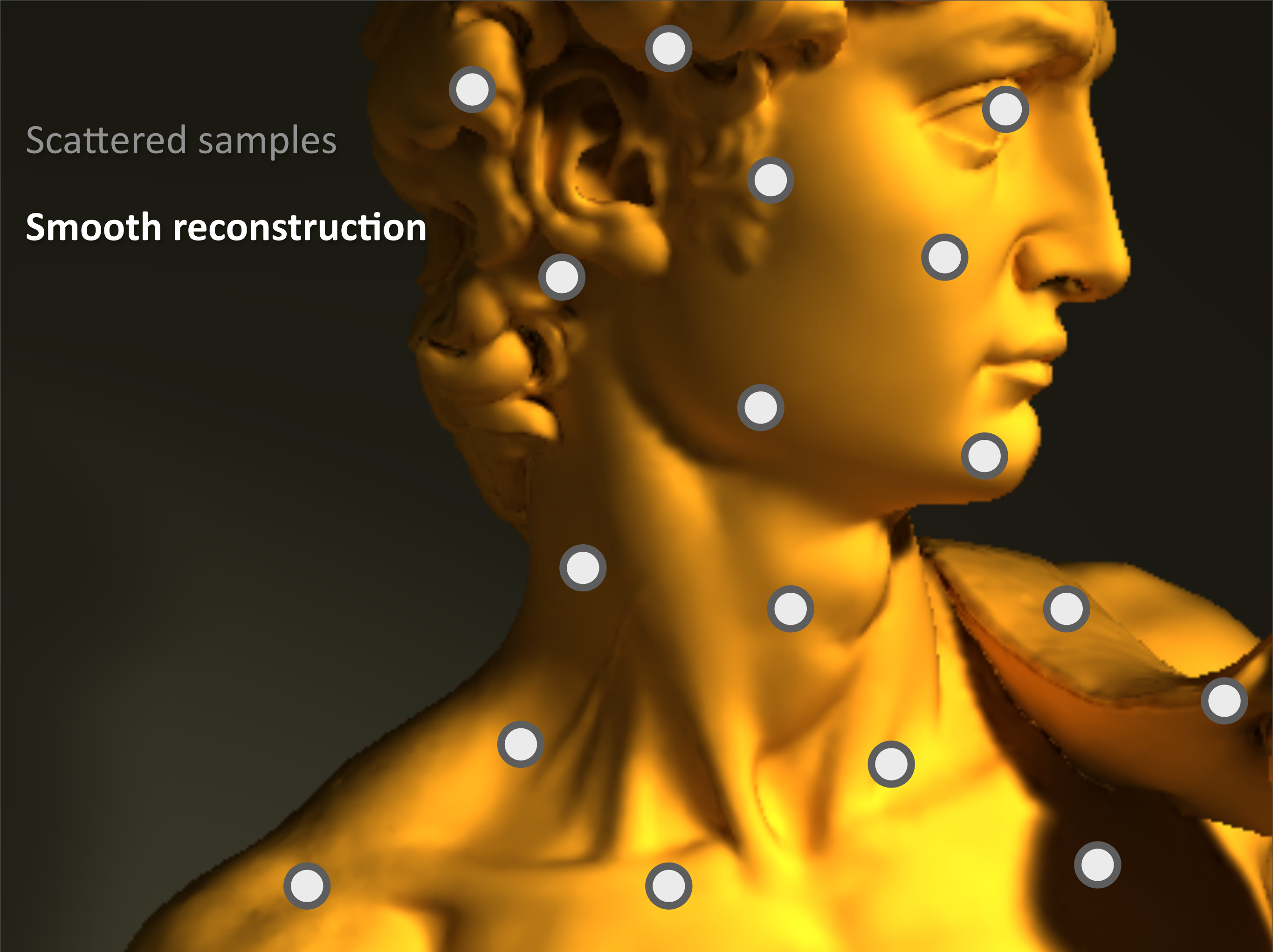
Let's see an overview of how the basis works. Here we illustrate the approximation of surface irradiance.

Scattered samples



Our basis builds on point samples scattered in 3D on the surfaces of our scene.

The function that we want to approximate is sampled at the points. The samples only represent lighting, not the geometry.

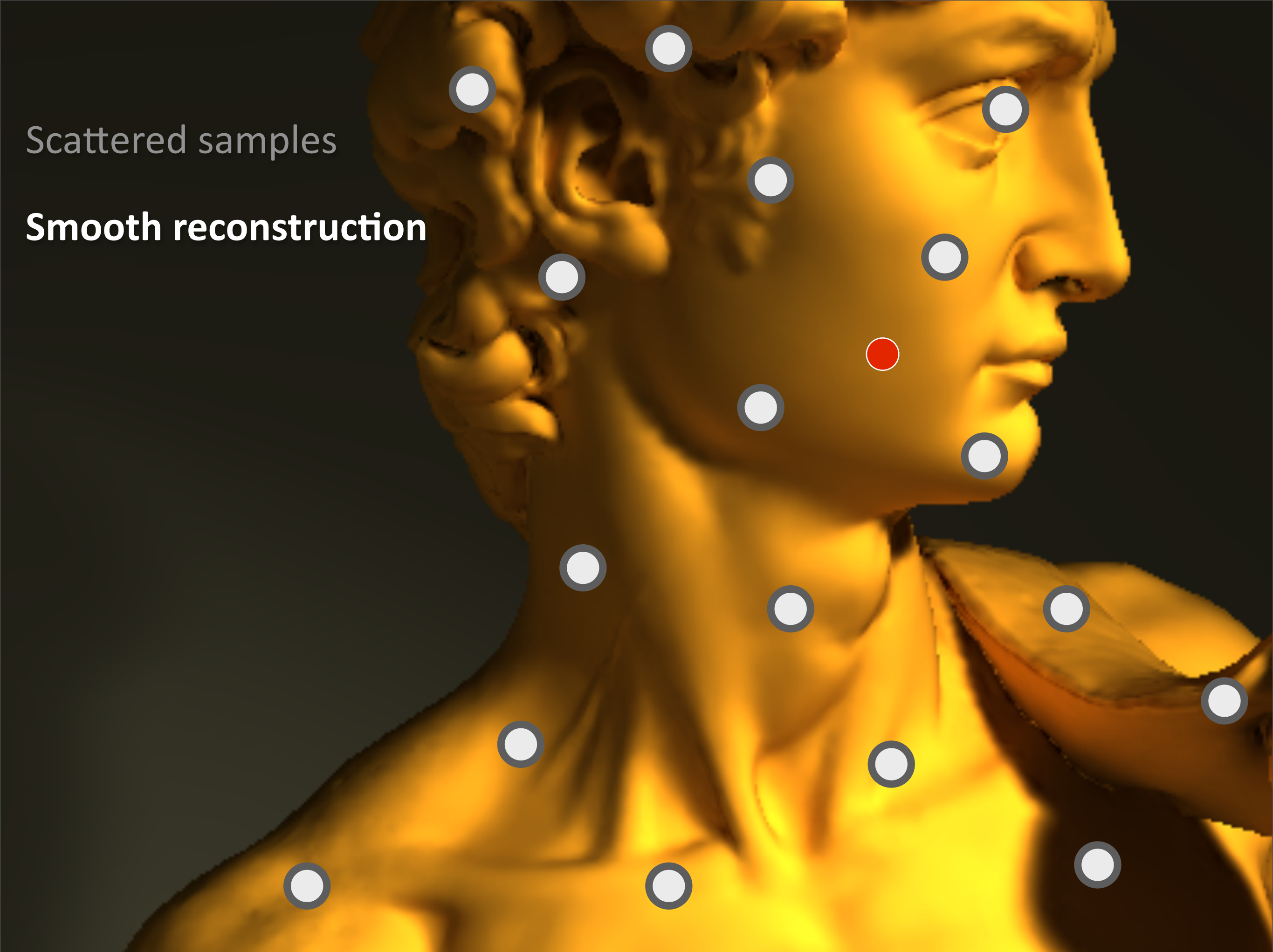
A golden bust of a classical figure, possibly a woman, shown in profile. The bust is set against a dark background. Scattered across the surface of the bust are several white circular markers with grey outlines, representing sample points. The lighting is dramatic, highlighting the contours of the face and the texture of the hair.

Scattered samples

Smooth reconstruction

For ANY point in the scene, we can use a scattered data approximation procedure to smoothly approximate the function based on the nearby point samples.

For any query point, here denoted in red, we take some weighted average of the point values from the nearby samples.

A golden bust of a classical figure, possibly a woman, is shown in profile. The bust is rendered in a warm, golden-yellow color. Scattered across the surface of the bust are several small, white circular points with dark gray outlines. One of these points, located on the cheek area, is highlighted in red. The background is dark, making the golden bust stand out.

Scattered samples

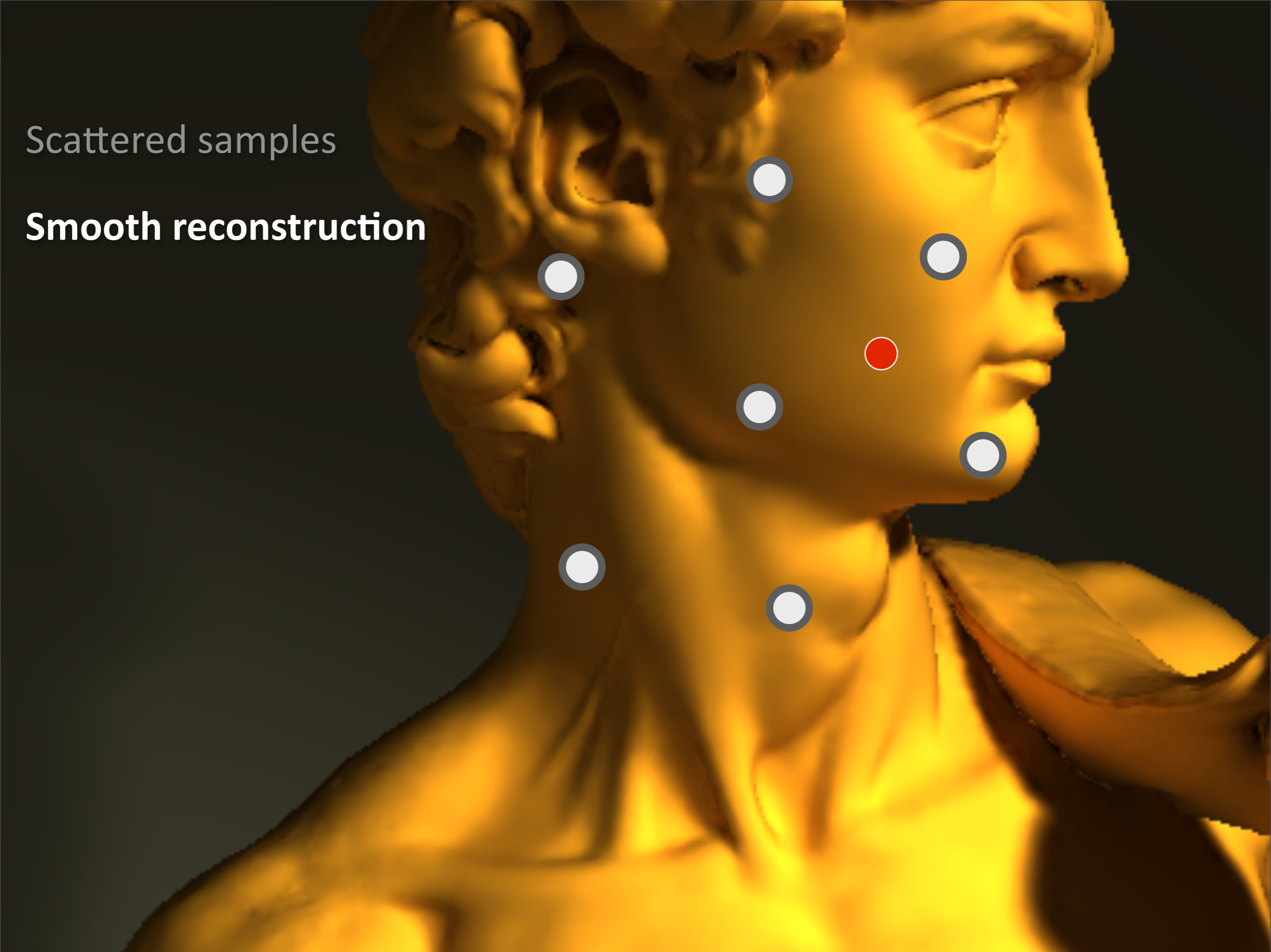
Smooth reconstruction

For ANY point in the scene, we can use a scattered data approximation procedure to smoothly approximate the function based on the nearby point samples.

For any query point, here denoted in red, we take some weighted average of the point values from the nearby samples.

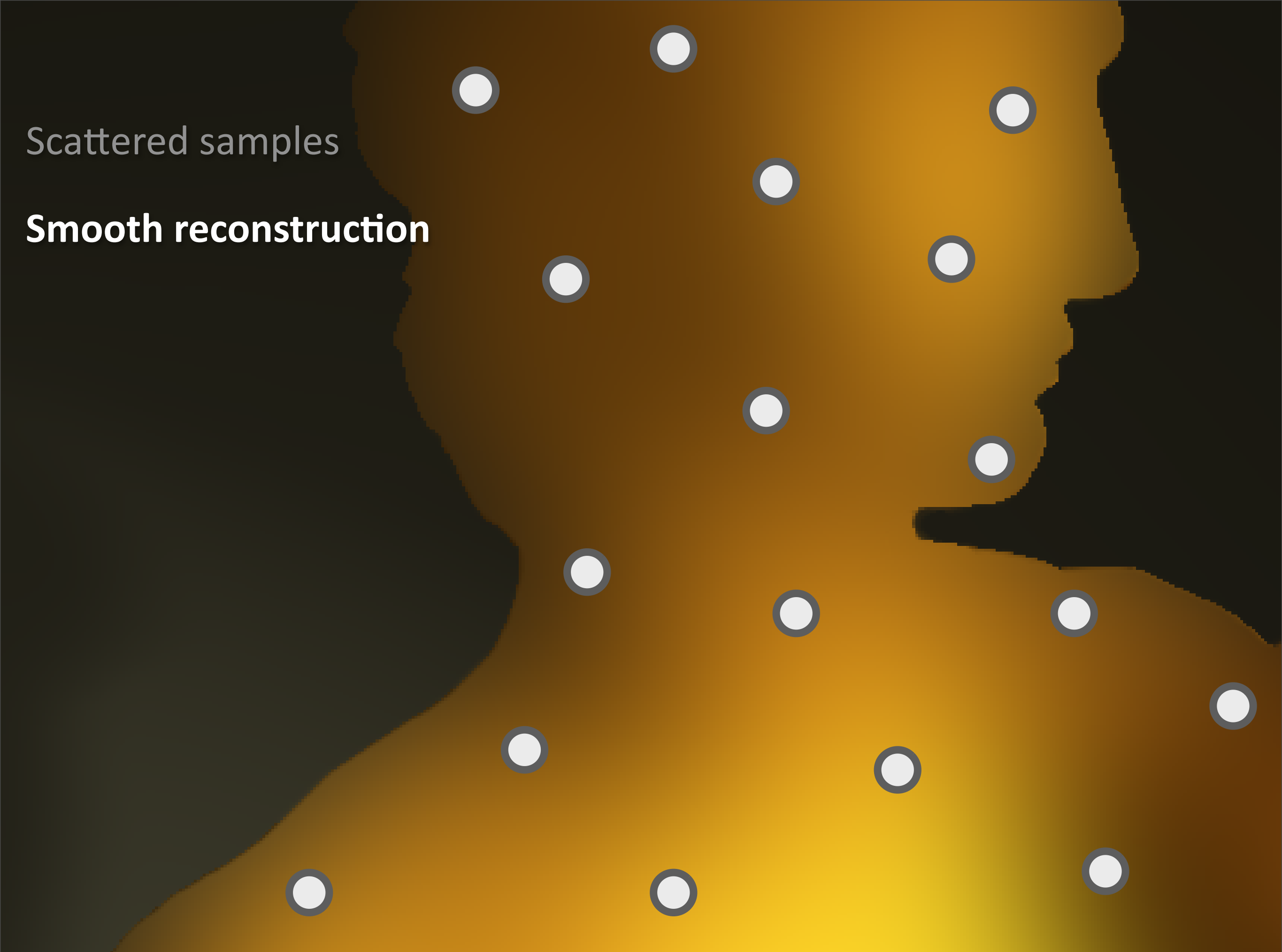
Scattered samples

Smooth reconstruction



For ANY point in the scene, we can use a scattered data approximation procedure to smoothly approximate the function based on the nearby point samples.

For any query point, here denoted in red, we take some weighted average of the point values from the nearby samples.



Scattered samples

Smooth reconstruction

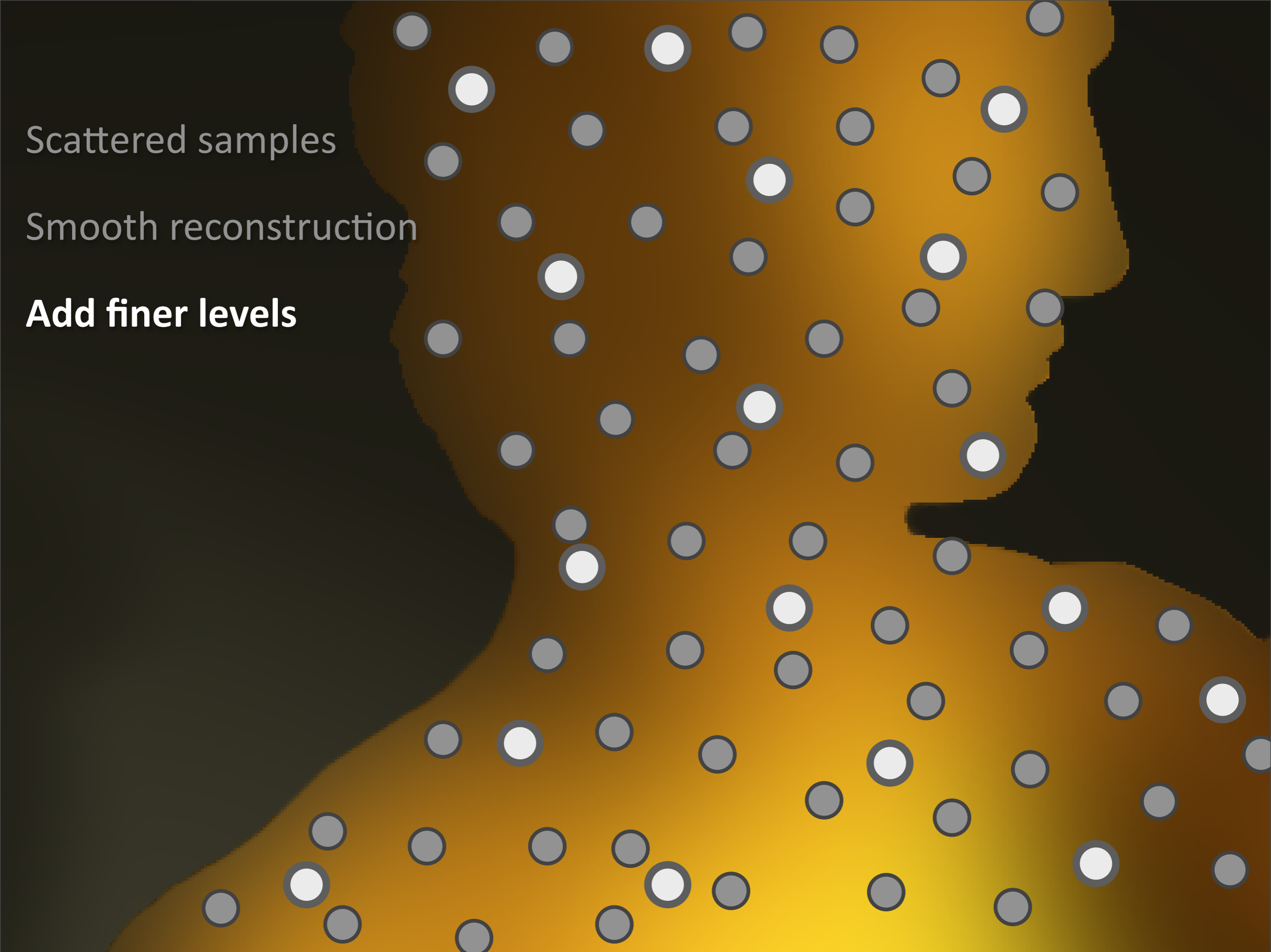
When this is done for all points, we get a coarse, blurry approximation of the original function.

If the function varies faster than the spacing of our points, we obviously cannot capture its behaviour correctly.

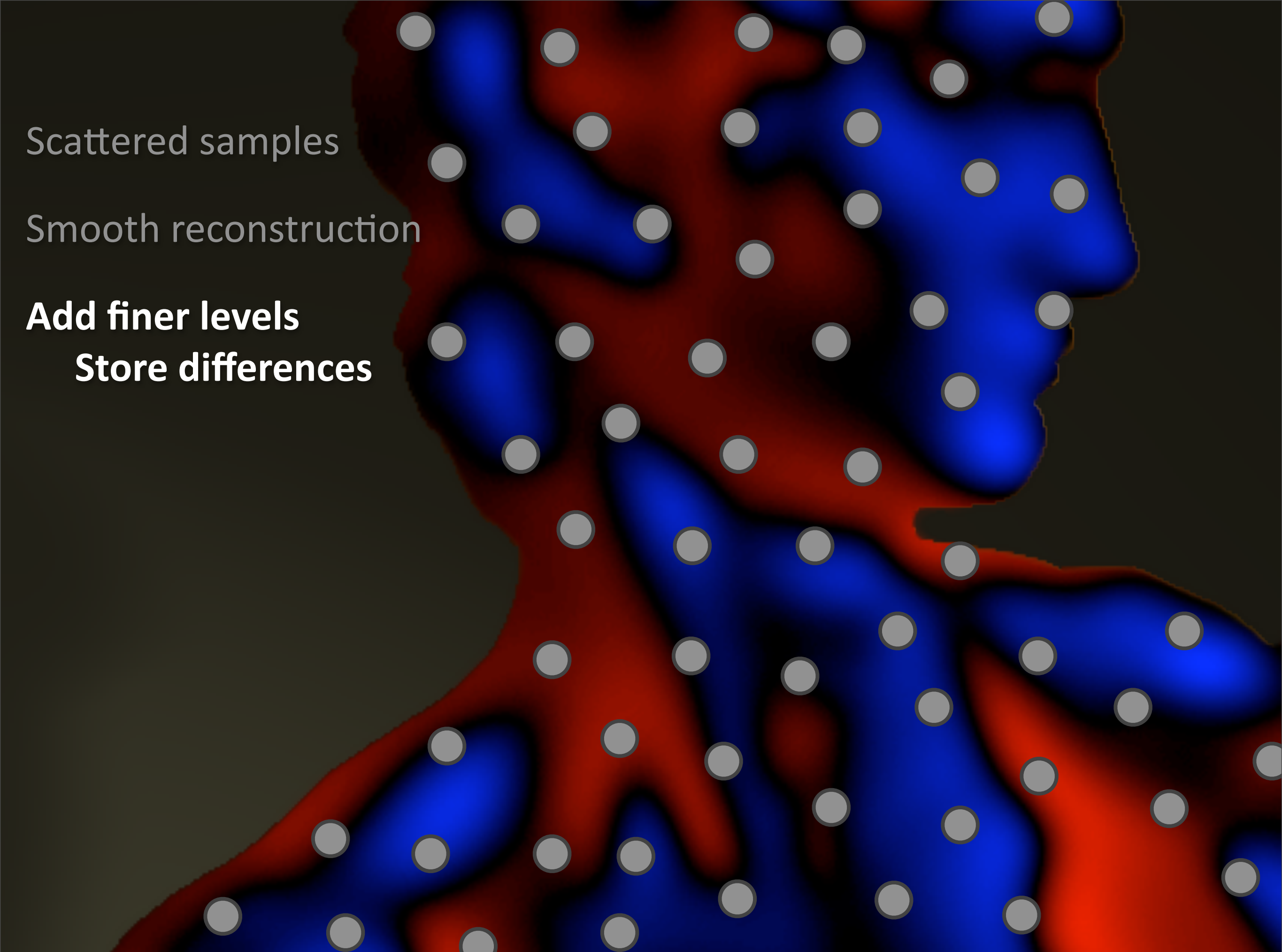
Scattered samples

Smooth reconstruction

Add finer levels



To add detail missing from the coarse approximation, we introduce a finer pointset.



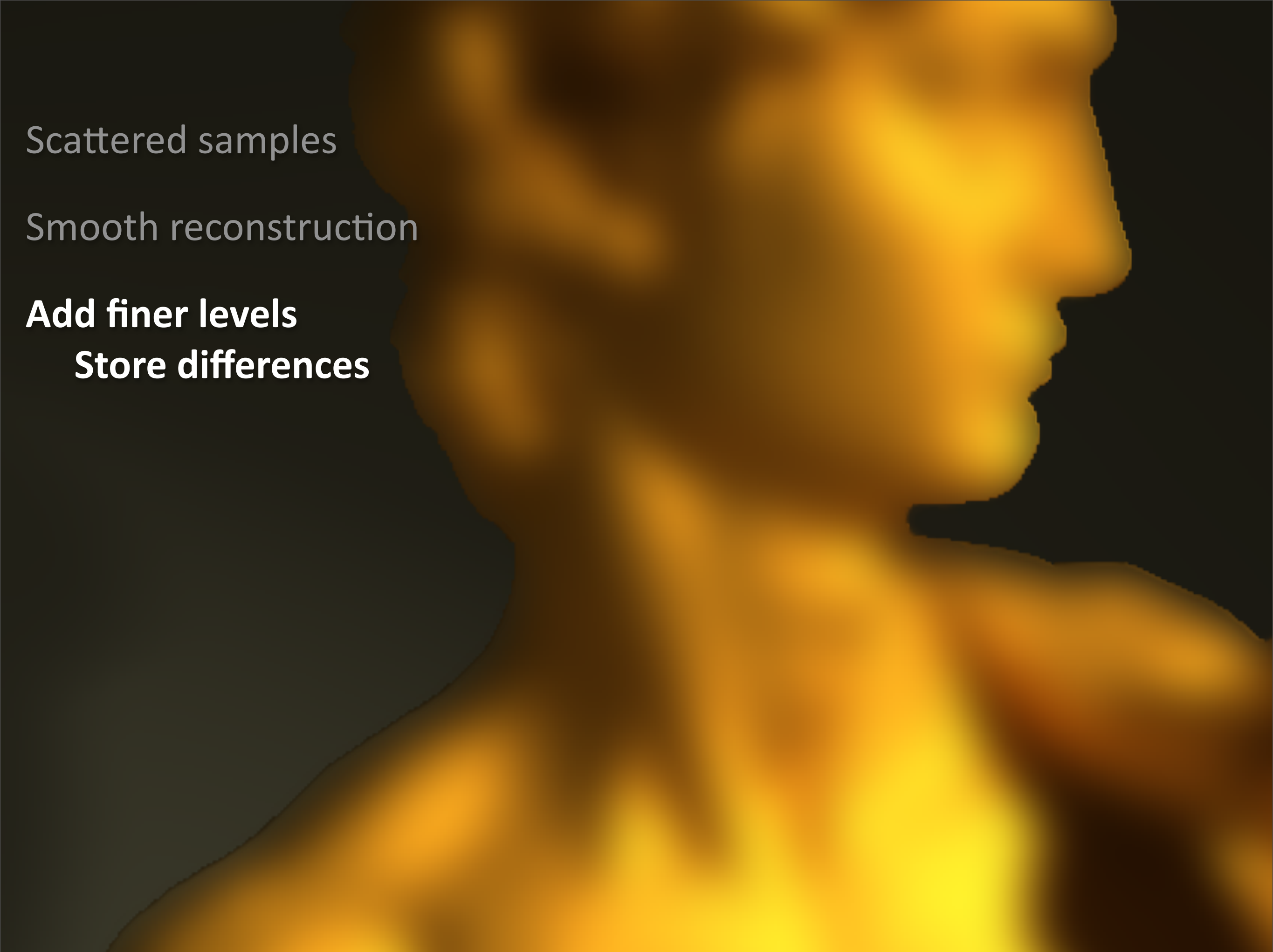
Scattered samples

Smooth reconstruction

Add finer levels

Store differences

At these finer points, we compute the DIFFERENCE between the function values and the previous coarse approximation. These differences are approximated using the finer points.



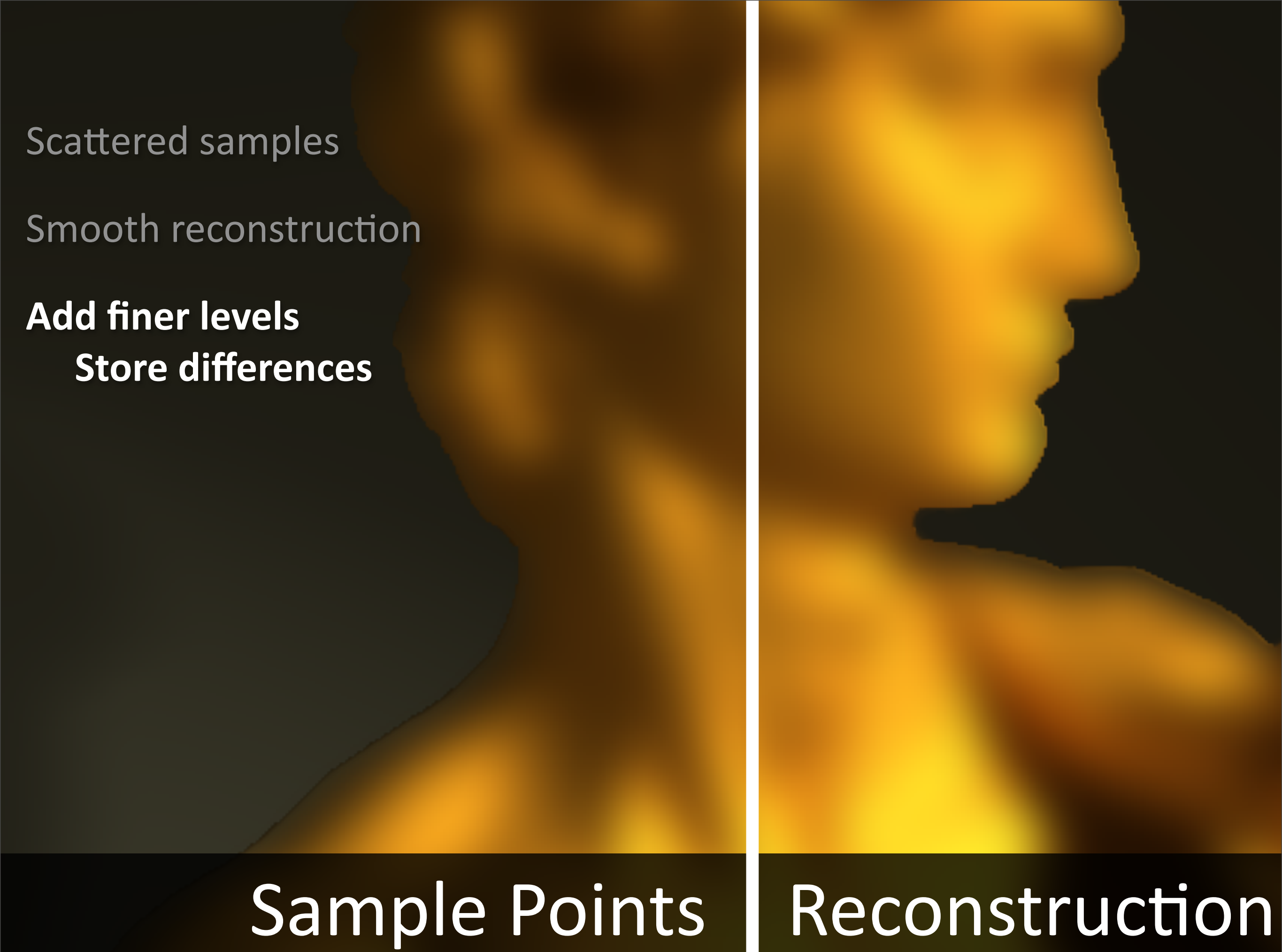
Scattered samples

Smooth reconstruction

Add finer levels

Store differences

When the differences are added to the coarse approximation, we get something that is closer to the true function, but not quite there yet.



Scattered samples

Smooth reconstruction

Add finer levels

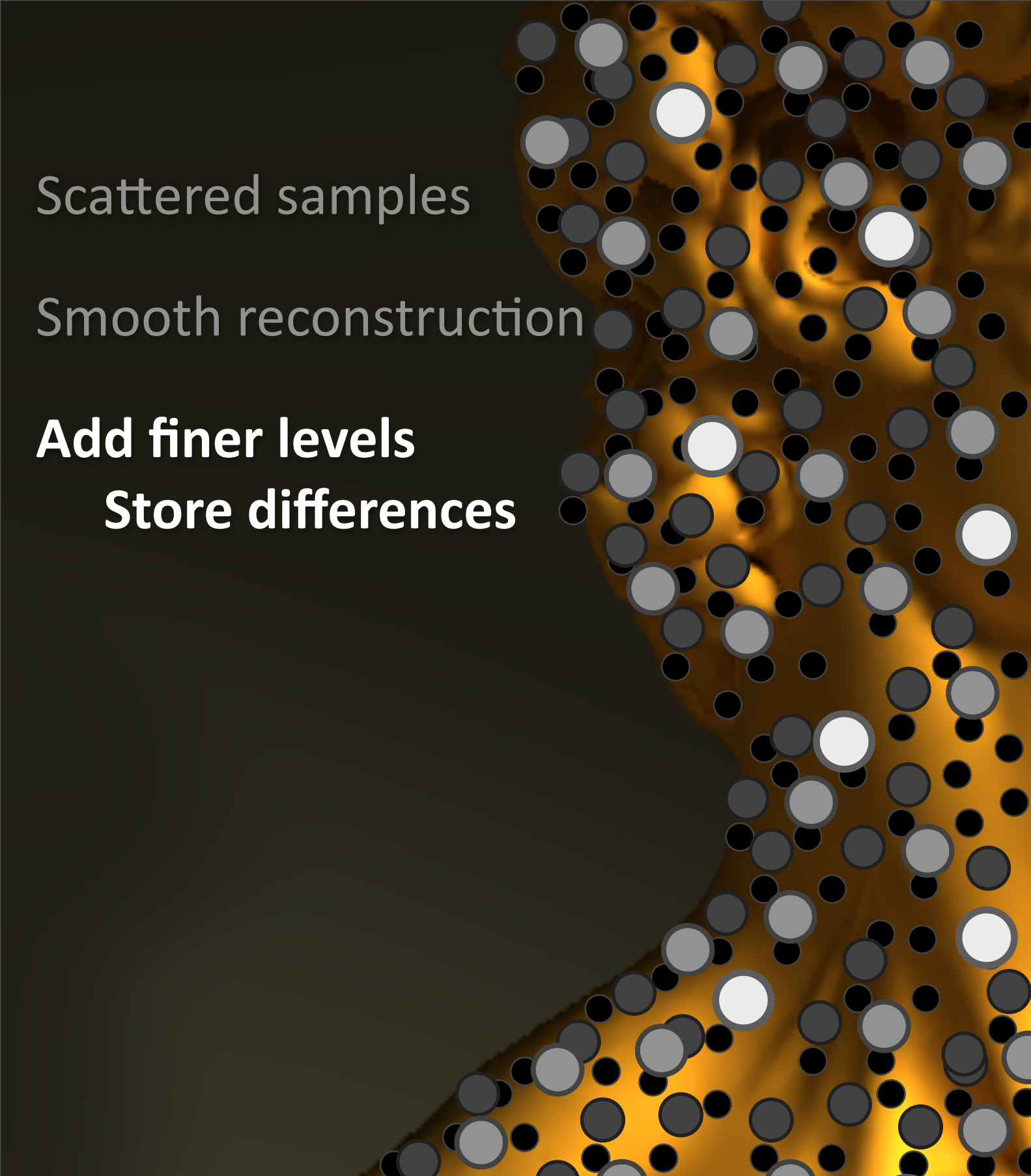
Store differences

Sample Points

Reconstruction

To get even closer to the original function, we keep adding levels of finer and finer points, always computing and storing the differences to the previous approximation.

Now, wherever the original function is smooth, its behaviour will be captured well already on the coarser levels of the hierarchy, which means that the differences on finer levels will be small. This key property allows adaptive computations analogous to wavelets, but without the need for parametrization or meshing.



Scattered samples

Smooth reconstruction

Add finer levels

Store differences

Sample Points



Reconstruction

To get even closer to the original function, we keep adding levels of finer and finer points, always computing and storing the differences to the previous approximation.

Now, wherever the original function is smooth, its behaviour will be captured well already on the coarser levels of the hierarchy, which means that the differences on finer levels will be small. This key property allows adaptive computations analogous to wavelets, but without the need for parametrization or meshing.

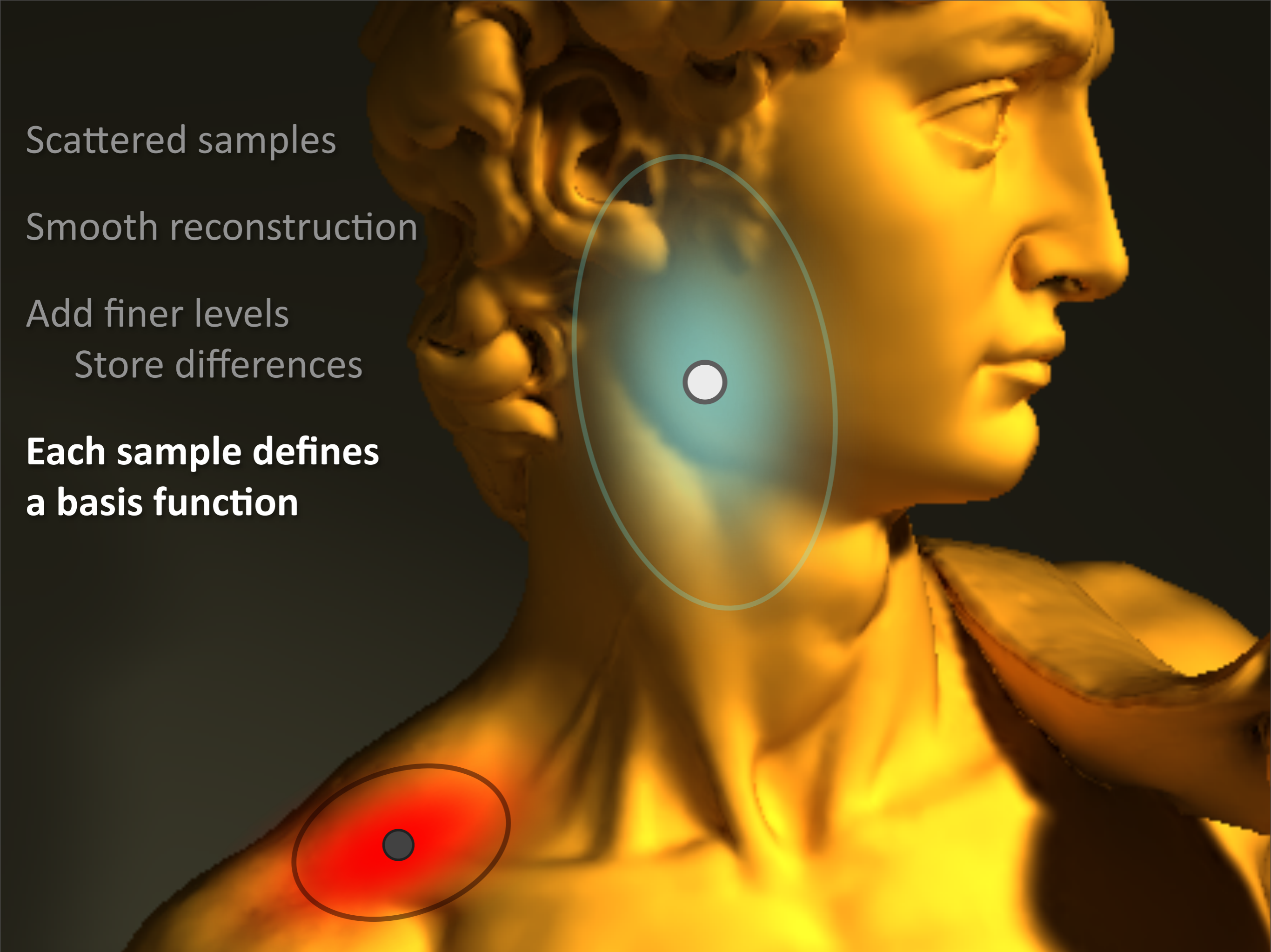
Scattered samples

Smooth reconstruction

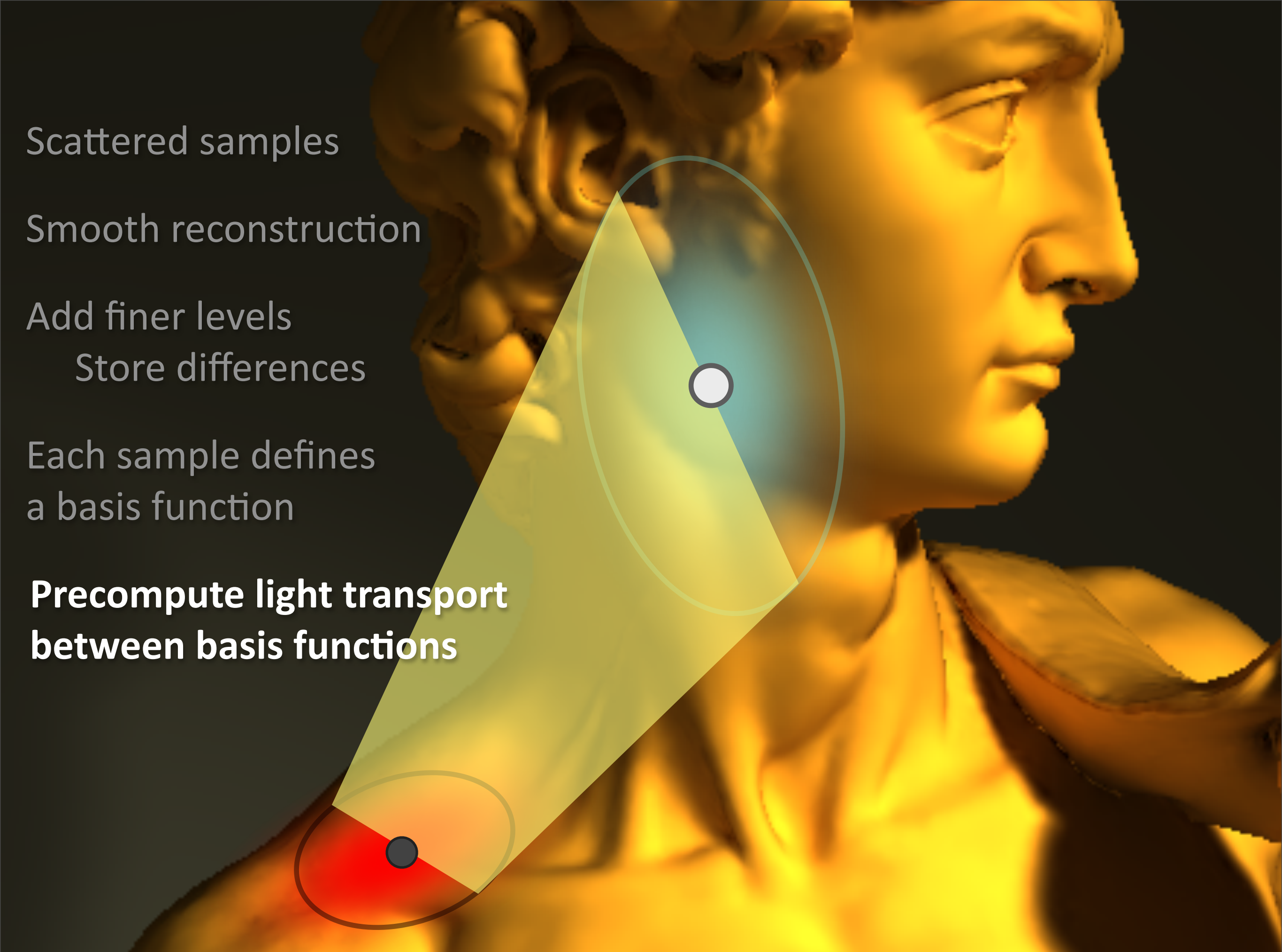
Add finer levels

Store differences

**Each sample defines
a basis function**



Also, it turns out that this intuitive process can be seen as a linear basis projection, such that each of the points corresponds to a single basis function.



Scattered samples

Smooth reconstruction

Add finer levels

Store differences

Each sample defines
a basis function

**Precompute light transport
between basis functions**

In the end, we will apply the function basis to PRT by hierarchically precomputing light transport between basis functions.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

Rendering on GPU

Discussion and Conclusions

Now let's see how we build our meshless basis functions.

Smooth Reconstruction

- Take local weighted averages of nearby samples [Shepard 68]
- Each sample is associated with a weight function



Let's take a look at how the approximation works.

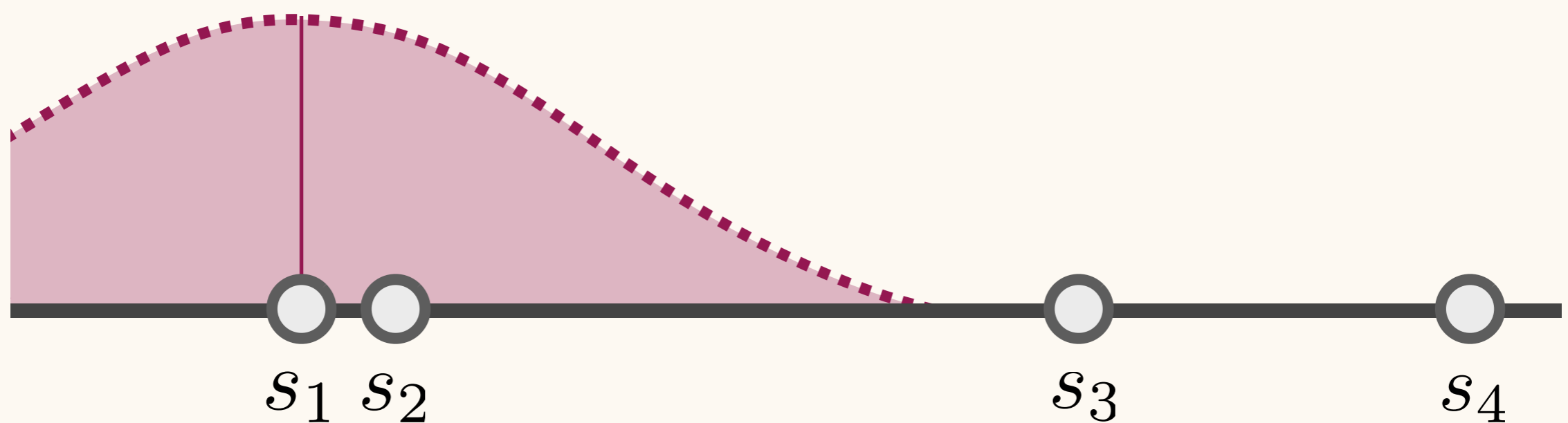
Specifically, we're going to take some weighted average of the nearby point samples to reconstruct the function. These weights in the average are given by a smooth weight function that is associated with each sample.

The weight functions determine the influence a given sample point has to its surroundings. This is called Shepard approximation.

We cannot use the weight functions directly as basis functions. That would be radial basis function or RBF approximation, which would necessitate large linear systems. Instead we want to use the sampled values directly as basis coefficients. The Shepard scheme enables this by a normalization step.

Smooth Reconstruction

- Take local weighted averages of nearby samples [Shepard 68]
- Each sample is associated with a weight function



Let's take a look at how the approximation works.

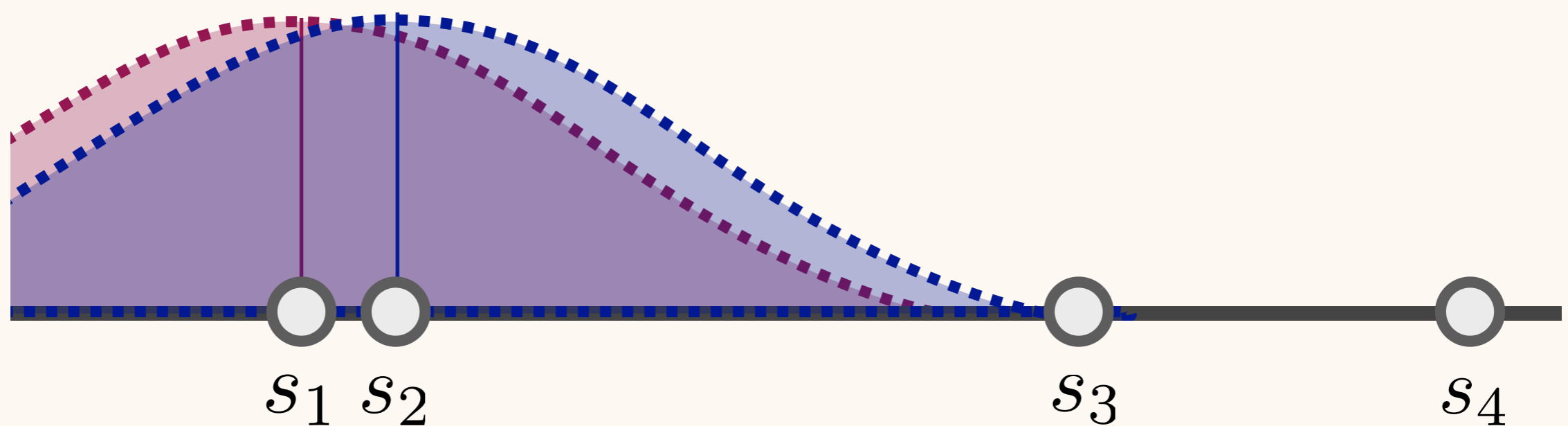
Specifically, we're going to take some weighted average of the nearby point samples to reconstruct the function. These weights in the average are given by a smooth weight function that is associated with each sample.

The weight functions determine the influence a given sample point has to its surroundings. This is called Shepard approximation.

We cannot use the weight functions directly as basis functions. That would be radial basis function or RBF approximation, which would necessitate large linear systems. Instead we want to use the sampled values directly as basis coefficients. The Shepard scheme enables this by a normalization step.

Smooth Reconstruction

- Take local weighted averages of nearby samples [Shepard 68]
- Each sample is associated with a weight function



Let's take a look at how the approximation works.

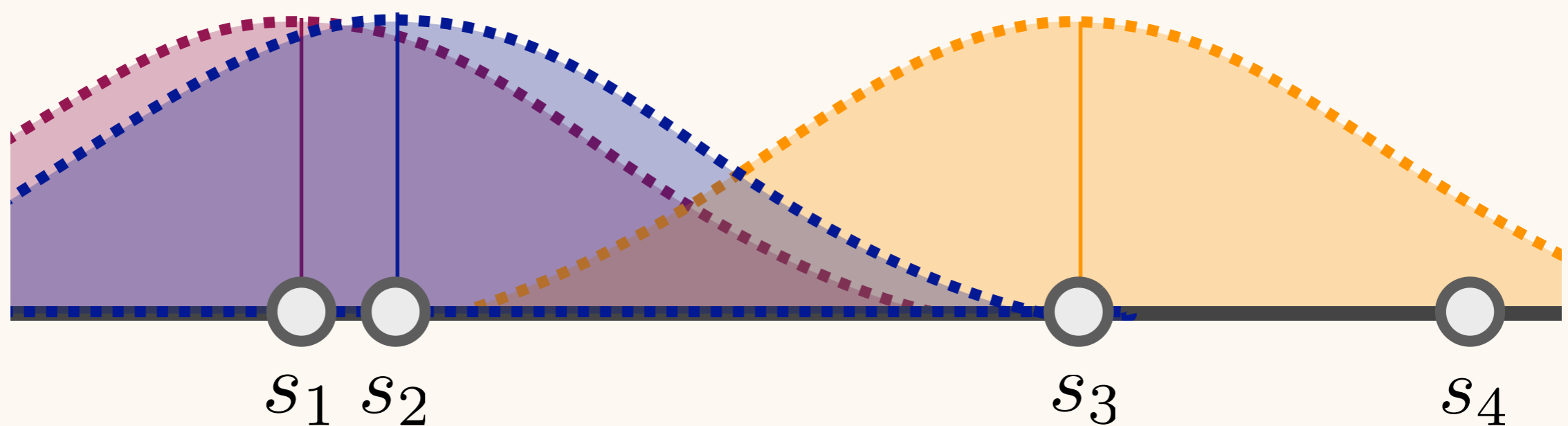
Specifically, we're going to take some weighted average of the nearby point samples to reconstruct the function. These weights in the average are given by a smooth weight function that is associated with each sample.

The weight functions determine the influence a given sample point has to its surroundings. This is called Shepard approximation.

We cannot use the weight functions directly as basis functions. That would be radial basis function or RBF approximation, which would necessitate large linear systems. Instead we want to use the sampled values directly as basis coefficients. The Shepard scheme enables this by a normalization step.

Smooth Reconstruction

- Take local weighted averages of nearby samples [Shepard 68]
- Each sample is associated with a weight function



Let's take a look at how the approximation works.

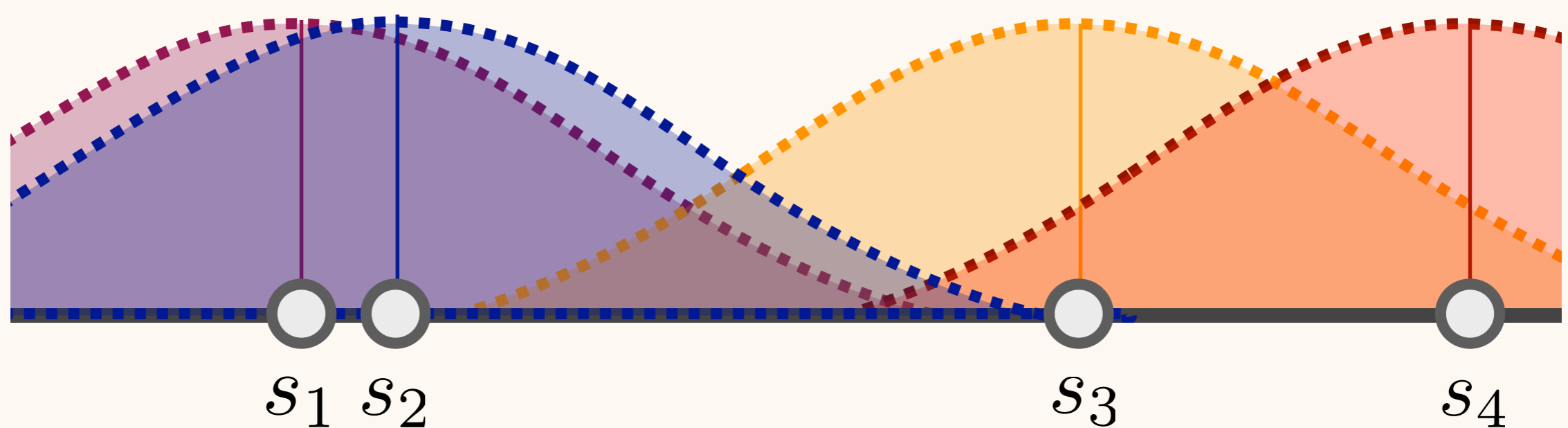
Specifically, we're going to take some weighted average of the nearby point samples to reconstruct the function. These weights in the average are given by a smooth weight function that is associated with each sample.

The weight functions determine the influence a given sample point has to its surroundings. This is called Shepard approximation.

We cannot use the weight functions directly as basis functions. That would be radial basis function or RBF approximation, which would necessitate large linear systems. Instead we want to use the sampled values directly as basis coefficients. The Shepard scheme enables this by a normalization step.

Smooth Reconstruction

- Take local weighted averages of nearby samples [Shepard 68]
- Each sample is associated with a weight function



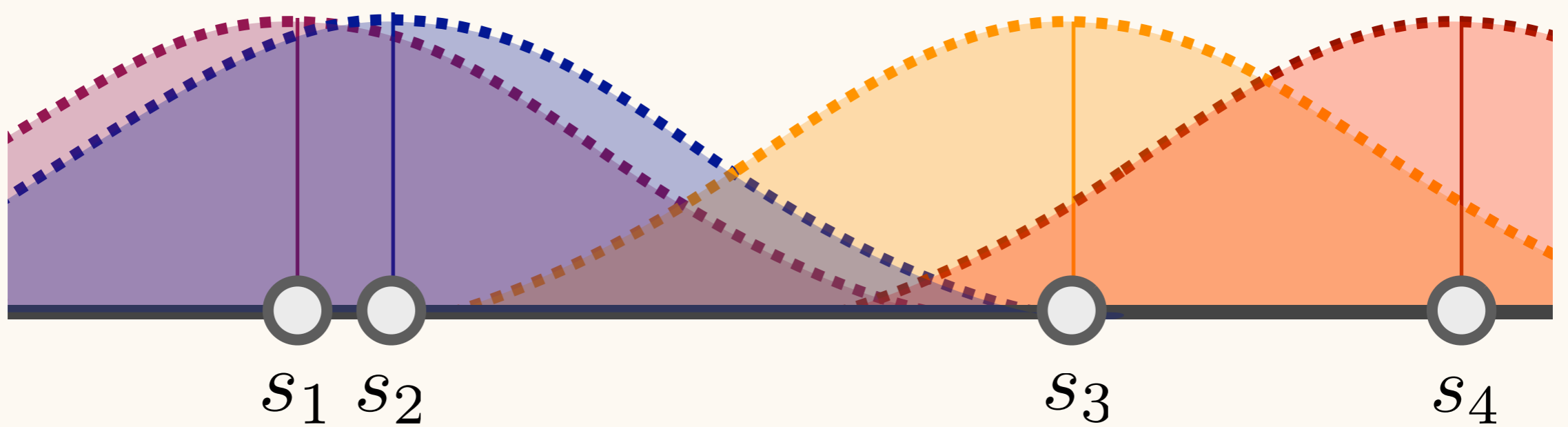
Let's take a look at how the approximation works.

Specifically, we're going to take some weighted average of the nearby point samples to reconstruct the function. These weights in the average are given by a smooth weight function that is associated with each sample.

The weight functions determine the influence a given sample point has to its surroundings. This is called Shepard approximation.

We cannot use the weight functions directly as basis functions. That would be radial basis function or RBF approximation, which would necessitate large linear systems. Instead we want to use the sampled values directly as basis coefficients. The Shepard scheme enables this by a normalization step.

From Weights to Basis Functions



To get from these weight functions to basis functions, we are going to normalize each weight function according to the Shepard scheme with the sum of all weight functions at all points in the domain.

The sum used for normalization is shown here in blue.

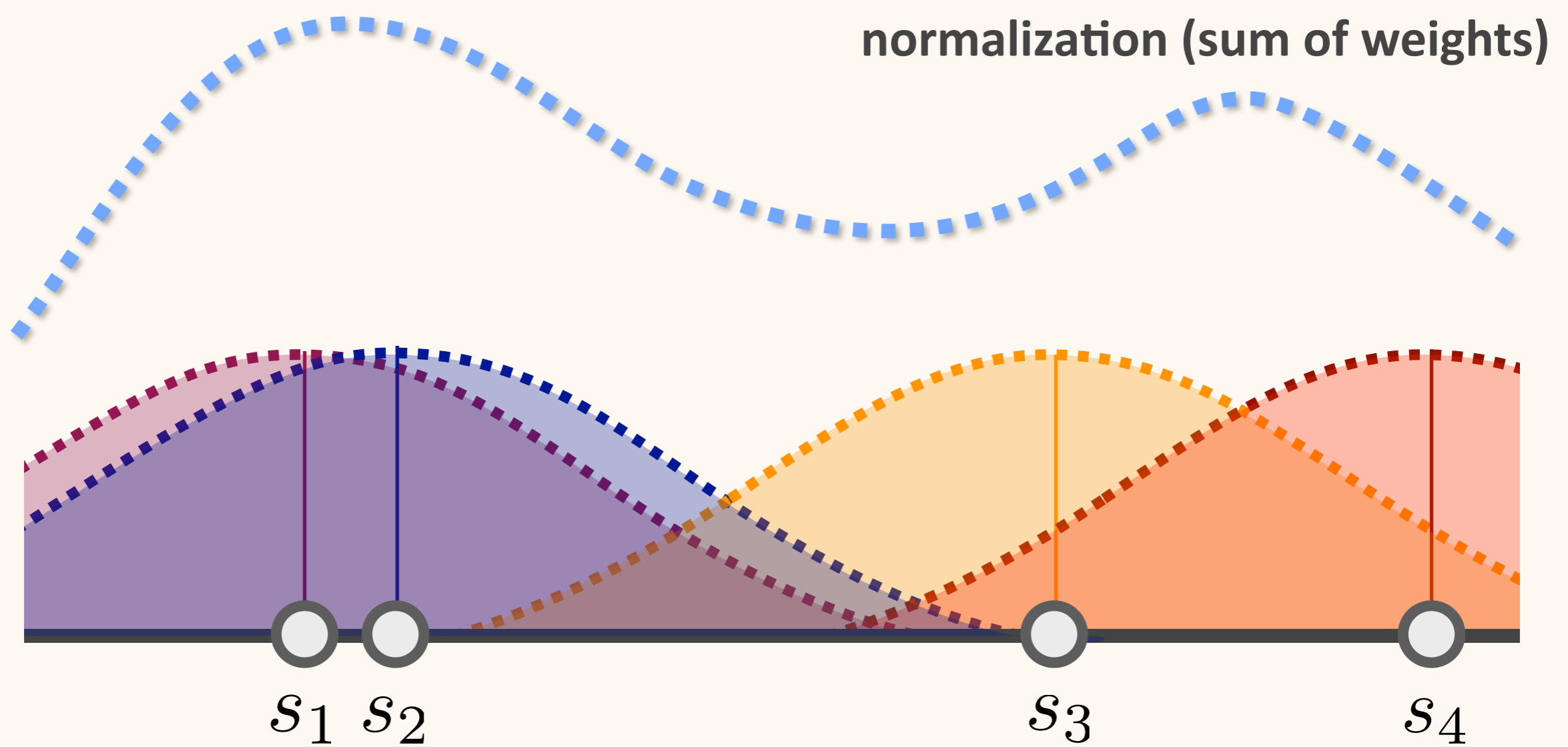
Now, let me concentrate on one of the weight functions.

We're now going to divide the weight by the normalizing sum..

.. and this leads us to the normalized basis function.

The procedure is exactly the same for all weight functions. Note how the distribution of the samples affects the shape of the resulting basis functions.

From Weights to Basis Functions



To get from these weight functions to basis functions, we are going to normalize each weight function according to the Shepard scheme with the sum of all weight functions at all points in the domain.

The sum used for normalization is shown here in blue.

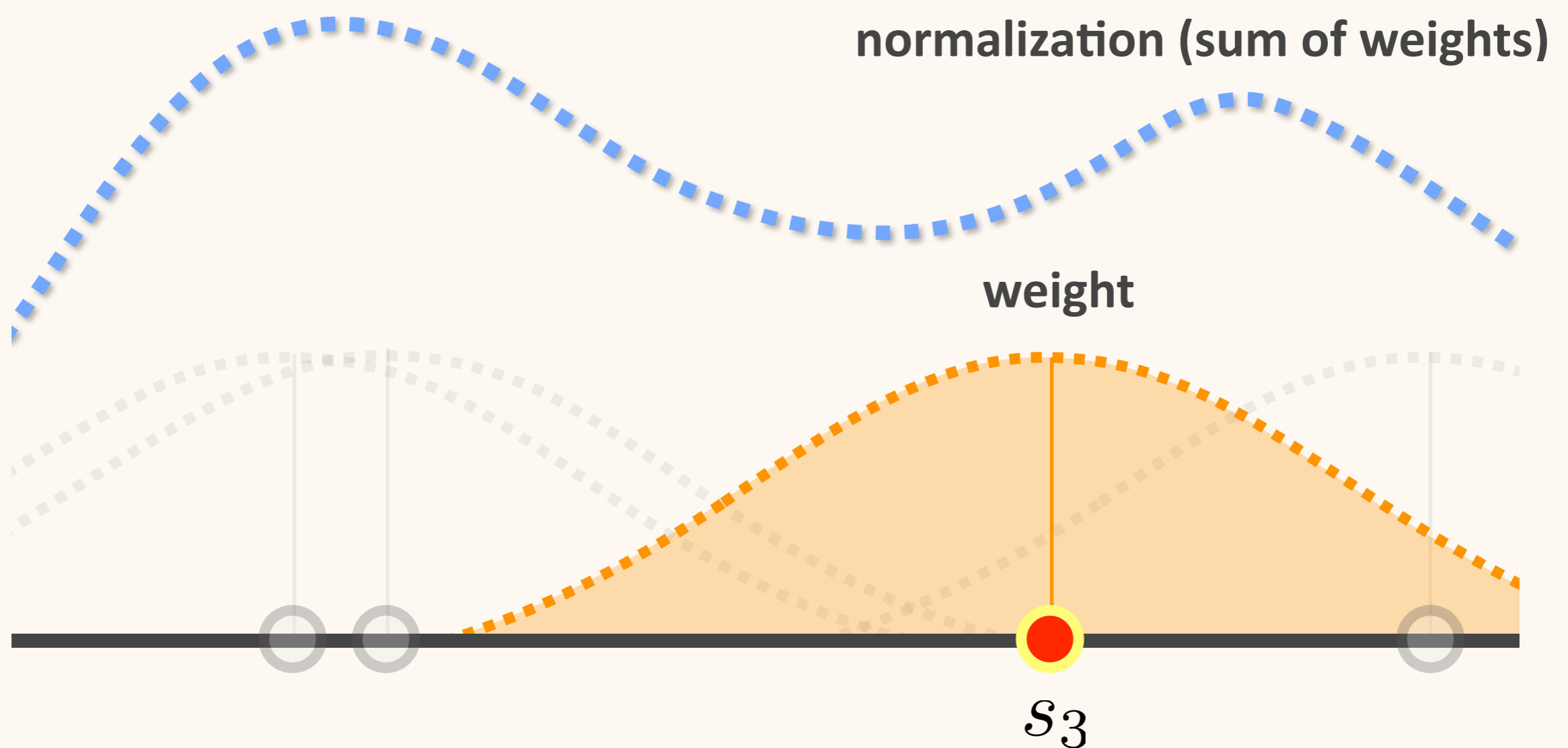
Now, let me concentrate on one of the weight functions.

We're now going to divide the weight by the normalizing sum..

.. and this leads us to the normalized basis function.

The procedure is exactly the same for all weight functions. Note how the distribution of the samples affects the shape of the resulting basis functions.

From Weights to Basis Functions



To get from these weight functions to basis functions, we are going to normalize each weight function according to the Shepard scheme with the sum of all weight functions at all points in the domain.

The sum used for normalization is shown here in blue.

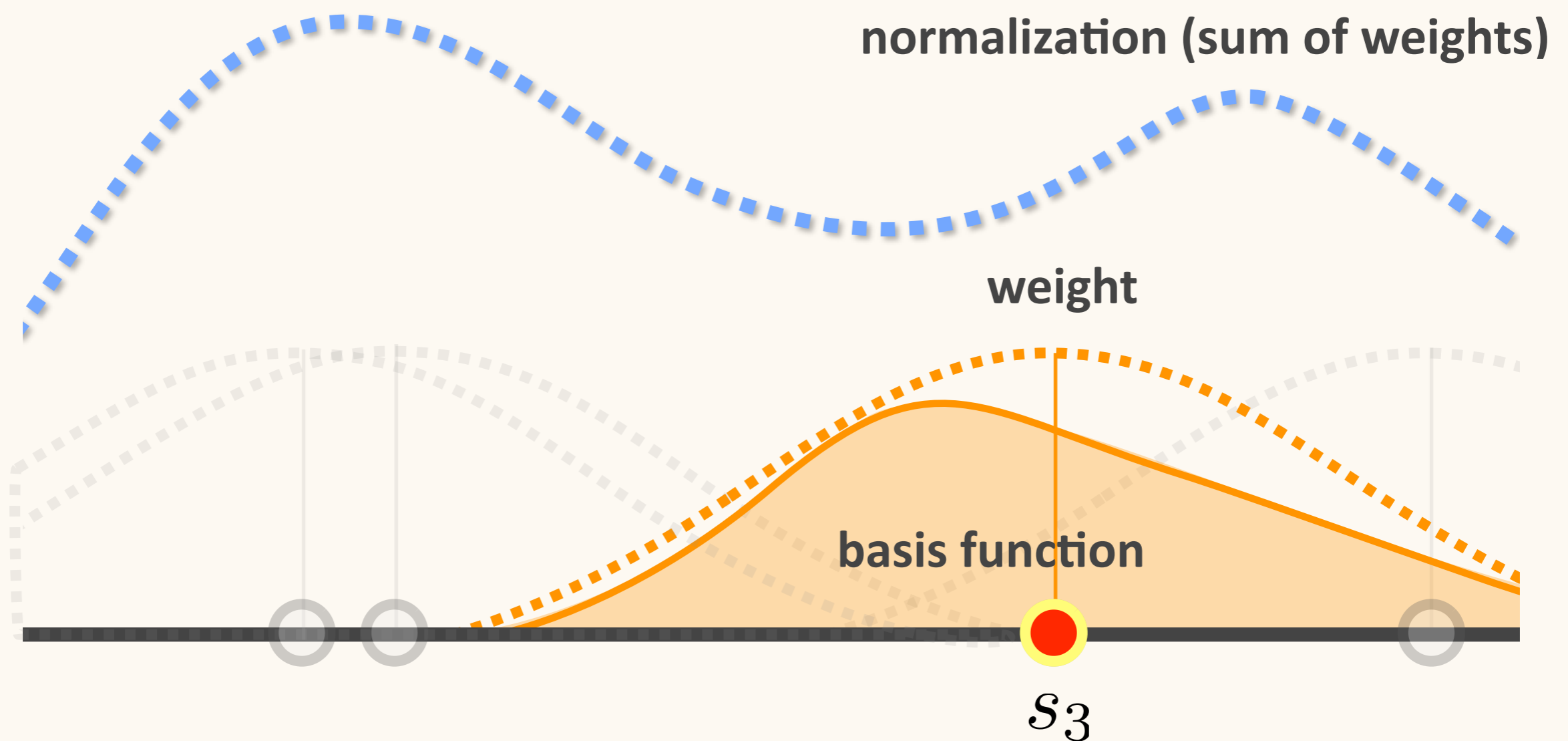
Now, let me concentrate on one of the weight functions.

We're now going to divide the weight by the normalizing sum..

.. and this leads us to the normalized basis function.

The procedure is exactly the same for all weight functions. Note how the distribution of the samples affects the shape of the resulting basis functions.

Basis Function = Normalized Weight



To get from these weight functions to basis functions, we are going to normalize each weight function according to the Shepard scheme with the sum of all weight functions at all points in the domain.

The sum used for normalization is shown here in blue.

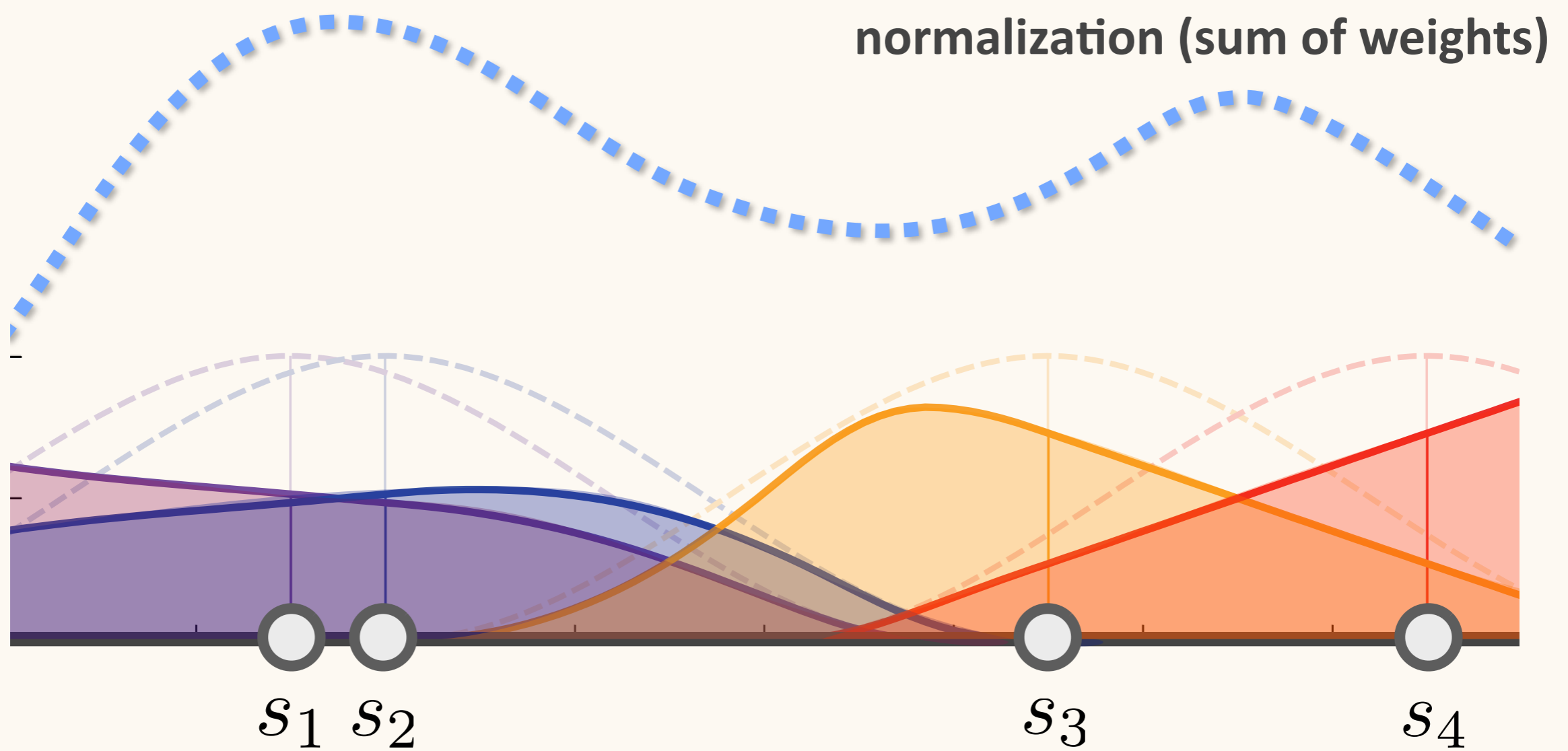
Now, let me concentrate on one of the weight functions.

We're now going to divide the weight by the normalizing sum..

.. and this leads us to the normalized basis function.

The procedure is exactly the same for all weight functions. Note how the distribution of the samples affects the shape of the resulting basis functions.

Basis Function = Normalized Weight



To get from these weight functions to basis functions, we are going to normalize each weight function according to the Shepard scheme with the sum of all weight functions at all points in the domain.

The sum used for normalization is shown here in blue.

Now, let me concentrate on one of the weight functions.

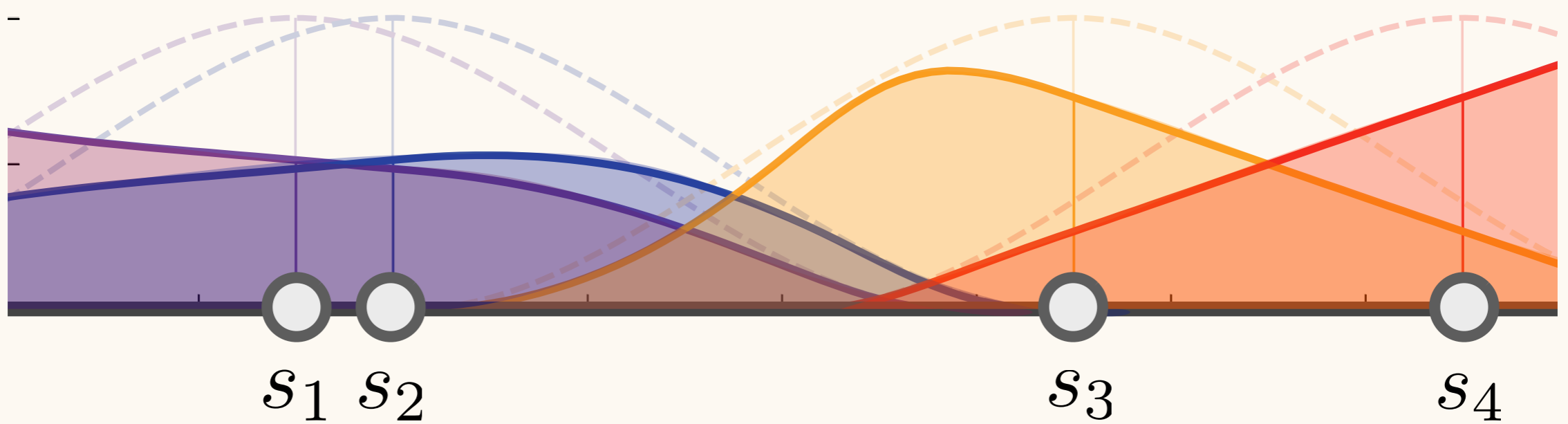
We're now going to divide the weight by the normalizing sum..

.. and this leads us to the normalized basis function.

The procedure is exactly the same for all weight functions. Note how the distribution of the samples affects the shape of the resulting basis functions.

Basis Function = Normalized Weight

- **Weight functions are 5D**
 - position and normal distance
 - reconstruction respects normal discontinuities (corners)



Here I've illustrated the weight and basis functions in 1D. In practice, the distance function we use is a five-dimensional combination of 3D distance and distance between normals. This means that on smooth surfaces the approximation will be smooth regardless of any patch or primitive boundaries, but normal discontinuities such as corners will yield a discontinuous approximation, just as we want.

Properties

- **Linear function basis**
 - analogous to piecewise linear, wavelets, ...
 - projection operator [Gortler 93, ...]
- **Wavelet-like sparsity**
 - allows multi-res and coarse-to-fine computations
- **Independent of geometric representation**
- **Related ideas: Laplacian pyramid, lifting scheme**

This construction leads to a linear, finite function basis which is analogous to any other linear basis such as piecewise constants, wavelets, etc.

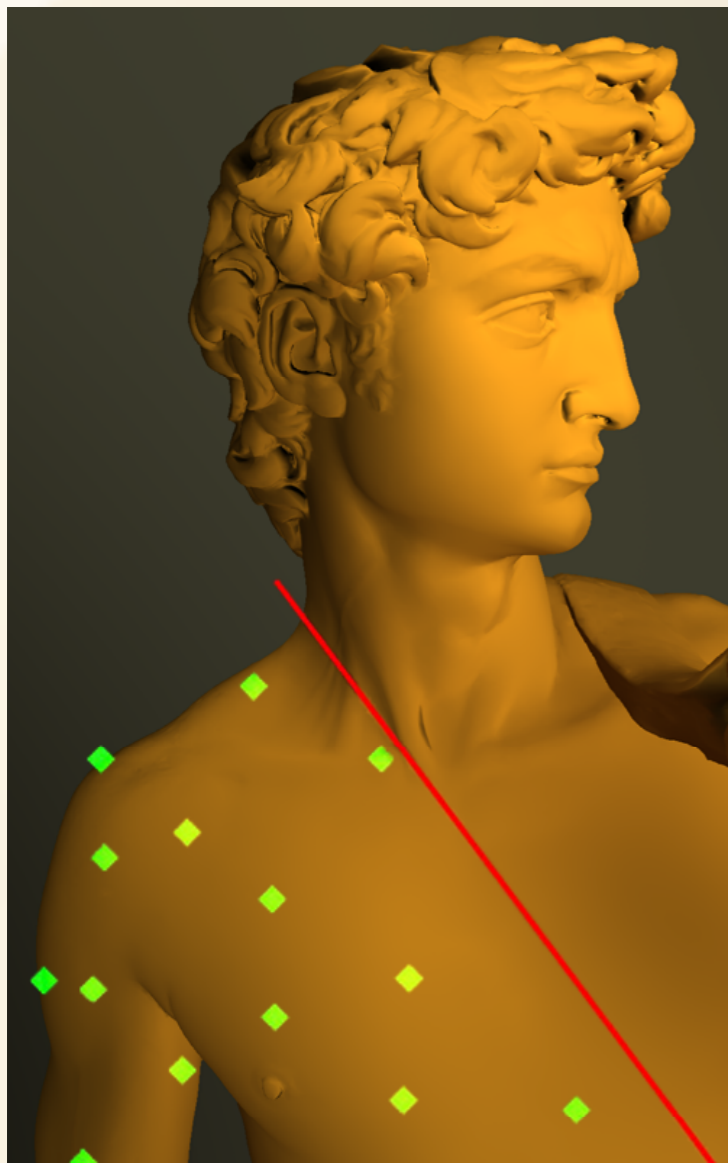
As I already mentioned, the differencing between the hierarchy levels leads to wavelet-like sparsity, which is the key to hierarchical, coarse-to-fine algorithms.

Furthermore, the basis is independent of the underlying geometric representation, because all we need for evaluating the basis functions are pointwise positions and normals.

The construction has some similarities to certain other multiresolution ideas. Please refer to the paper for details.

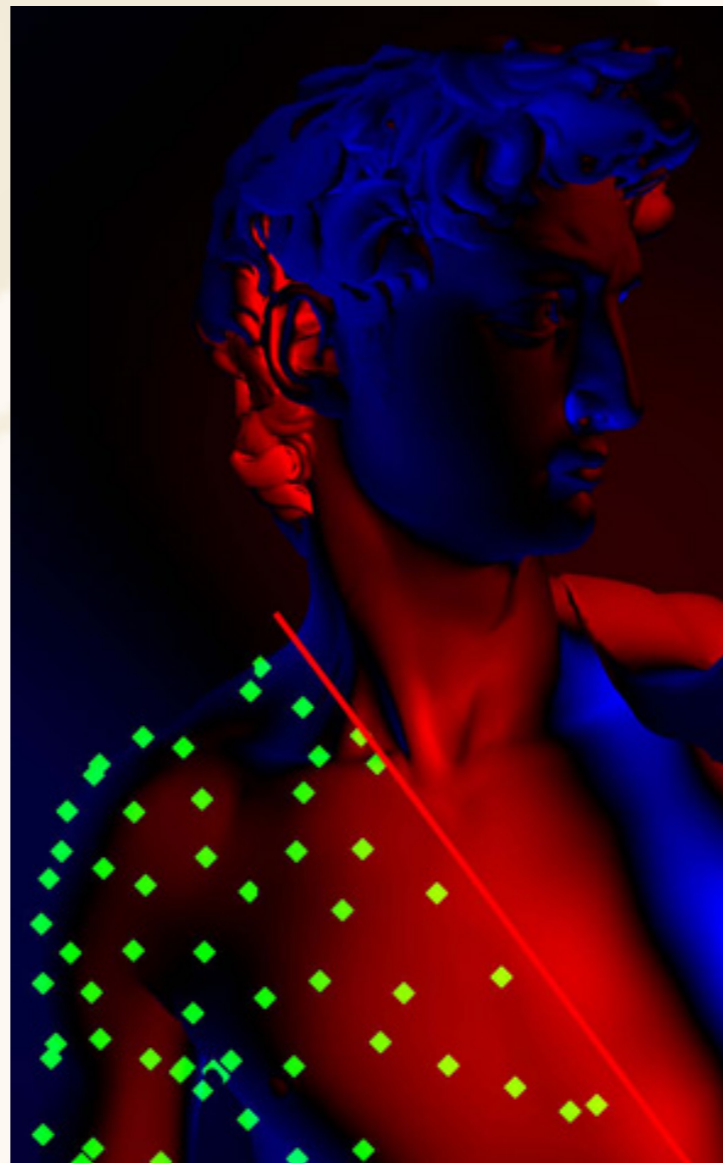
Approximation Example

Coarse



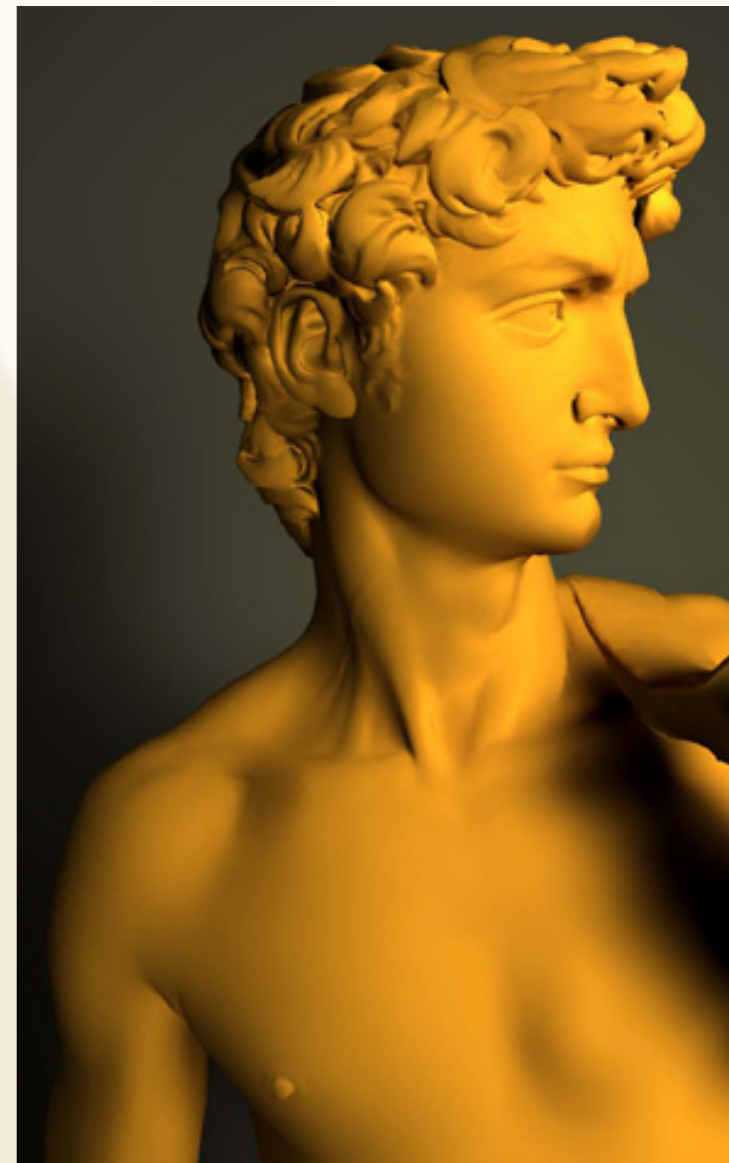
+

Difference



=

Finer



Red = positive
Blue = negative

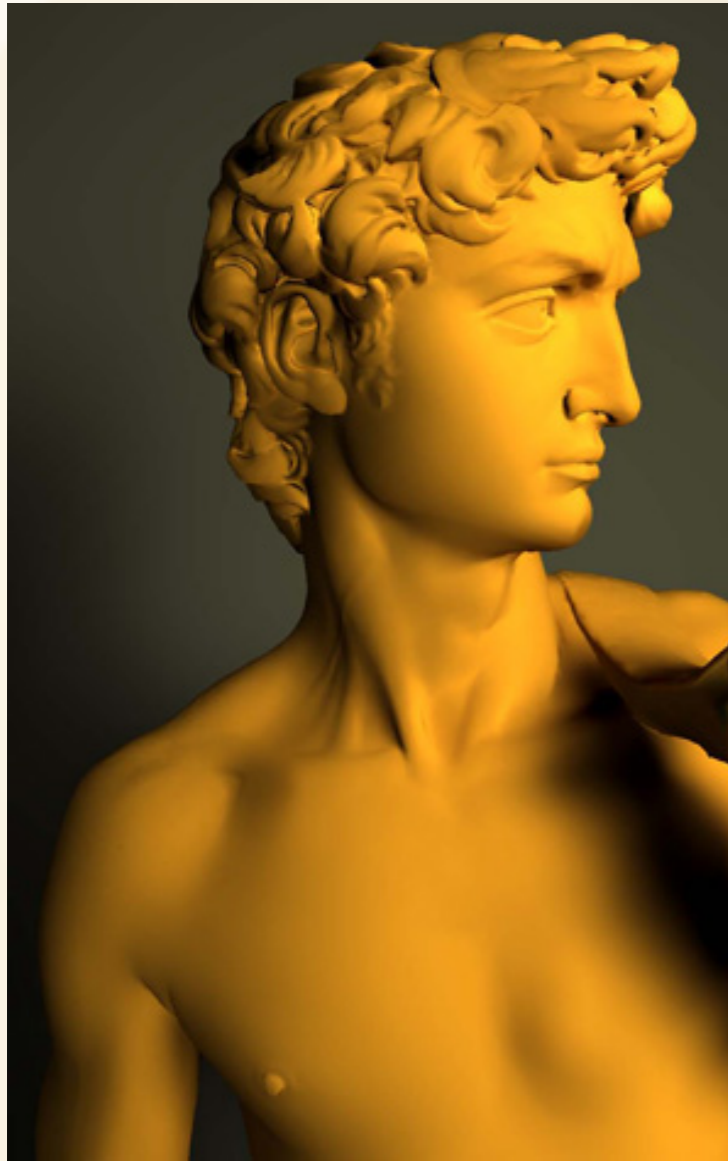
Let's look at a concrete example where we approximate irradiance over the David.

The coarse approximation on the left is computed from few points and is thus quite blurry. Once we add in the differences computed at the finer set of points in the middle, we get more detail, as shown on the right, but it's still kind of blurry.

Approximation Example

Difference

Finer



Red = positive
Blue = negative

Let's look at a concrete example where we approximate irradiance over the David.

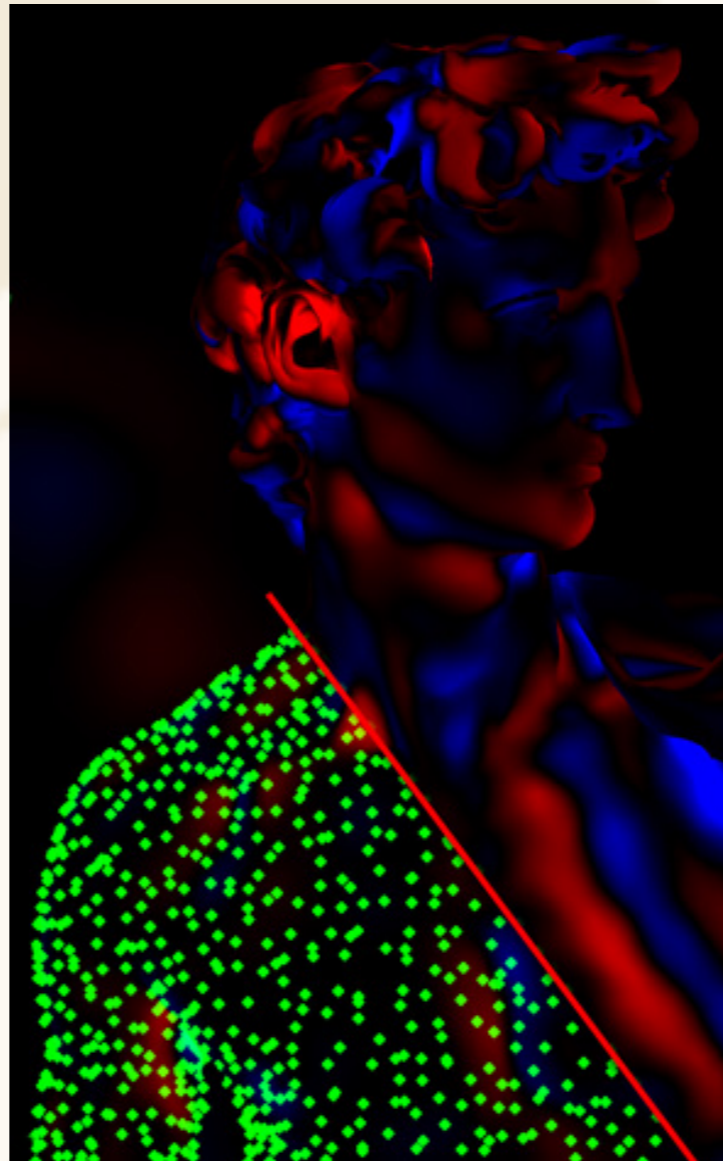
The coarse approximation on the left is computed from few points and is thus quite blurry. Once we add in the differences computed at the finer set of points in the middle, we get more detail, as shown on the right, but it's still kind of blurry.

Approximation Example

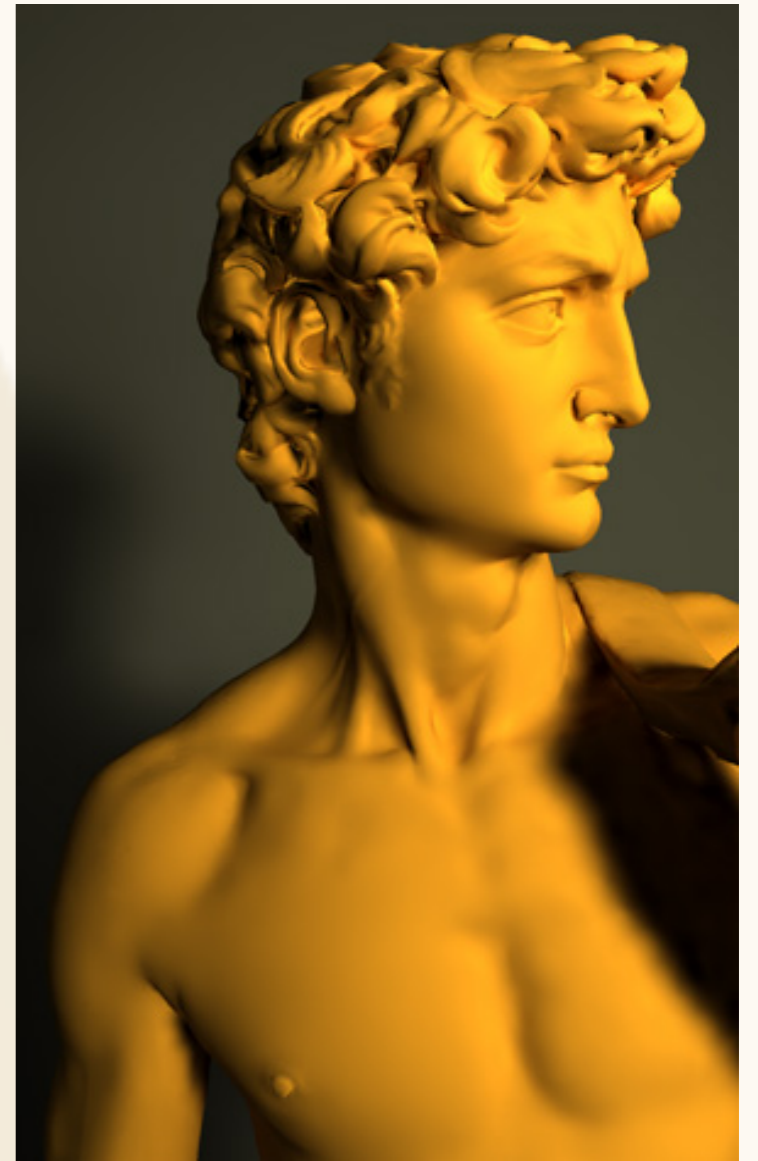
Previous



Difference



Finer



+

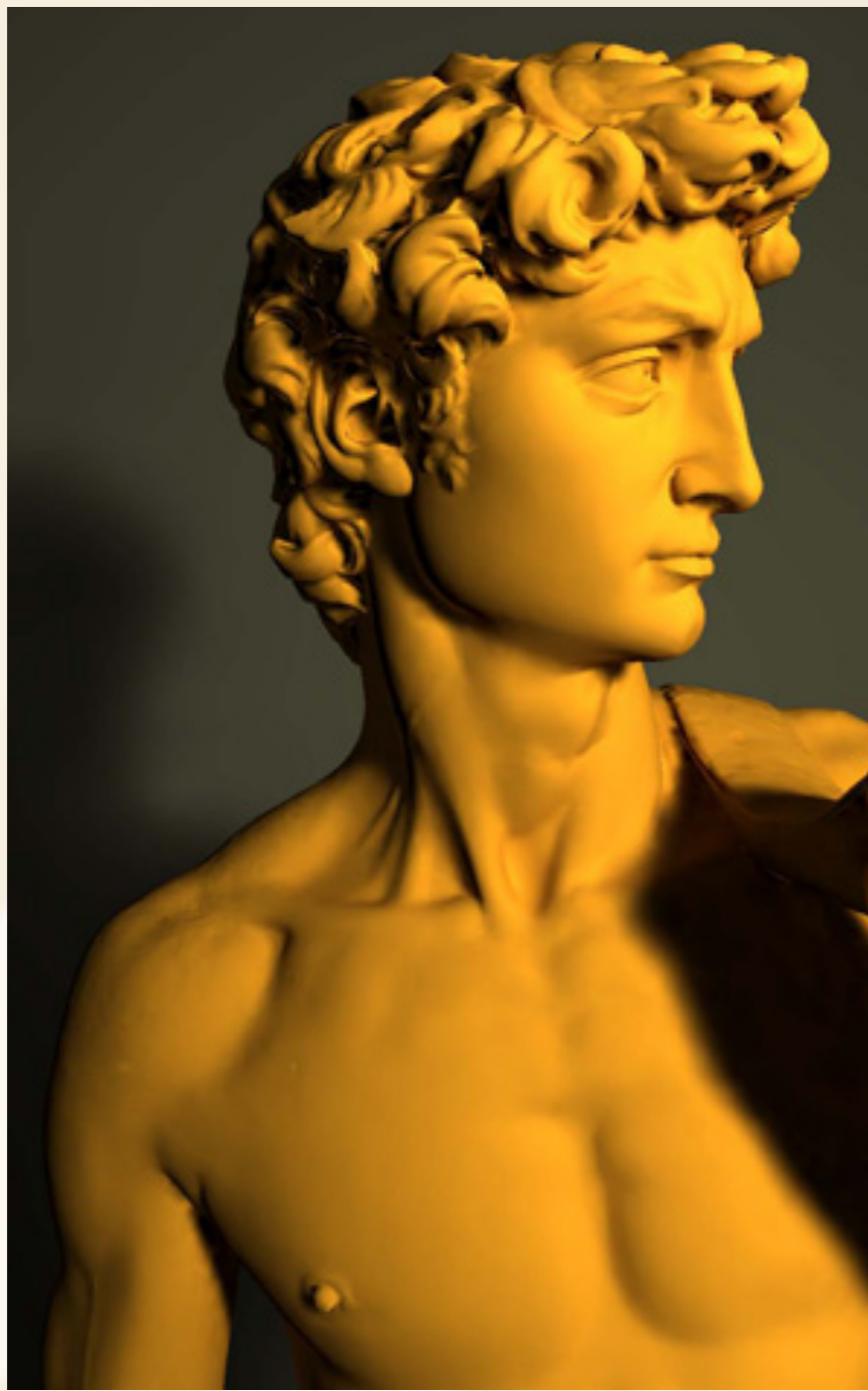
=

Red = positive
Blue = negative

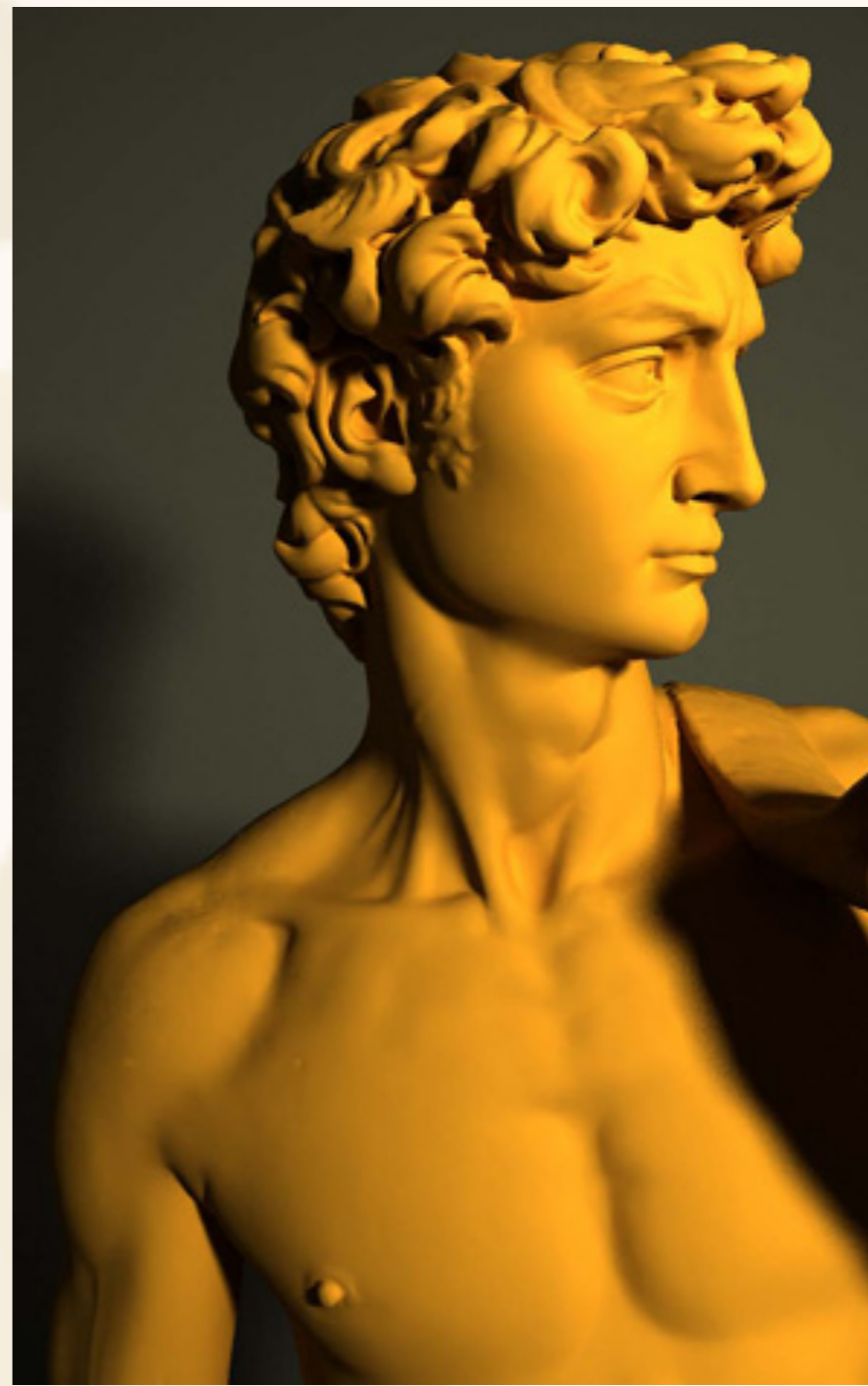
Now we just keep adding more levels of points, and thus we get closer and closer to the ground truth.

Notice how in the smooth areas the differences become small, like on the cheek.

Approximation Example



Meshless reconstruction



Per-pixel Reference

This is a comparison to a per-pixel ground truth. I hope you'll agree that the approximation is quite convincing.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

Rendering on GPU

Discussion and Conclusions

Basis Construction

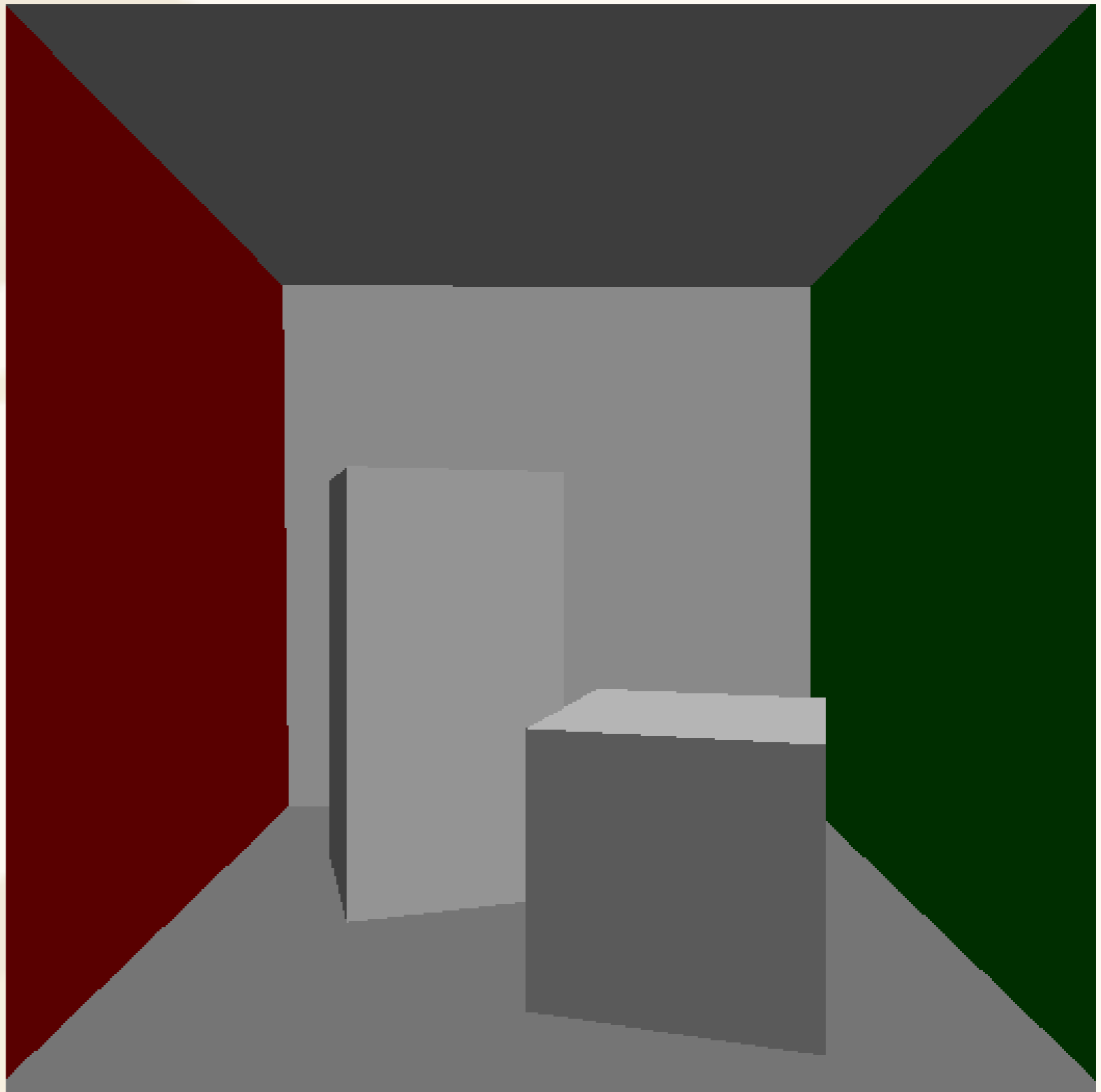
- **Requirements**

- uniform coverage (Poisson disk)
- points only on visible surfaces
- independence of geometric representation

To get a good representation of the illumination over the surfaces, we want the distribution of the points to be uniform with respect to the approximation weights. This is accomplished by a Poisson disk distribution. Furthermore, we want the algorithm to only place points on surfaces visible from any reasonable viewpoint, and the algorithm should be as independent of the geometric representation as possible.

Basis Construction

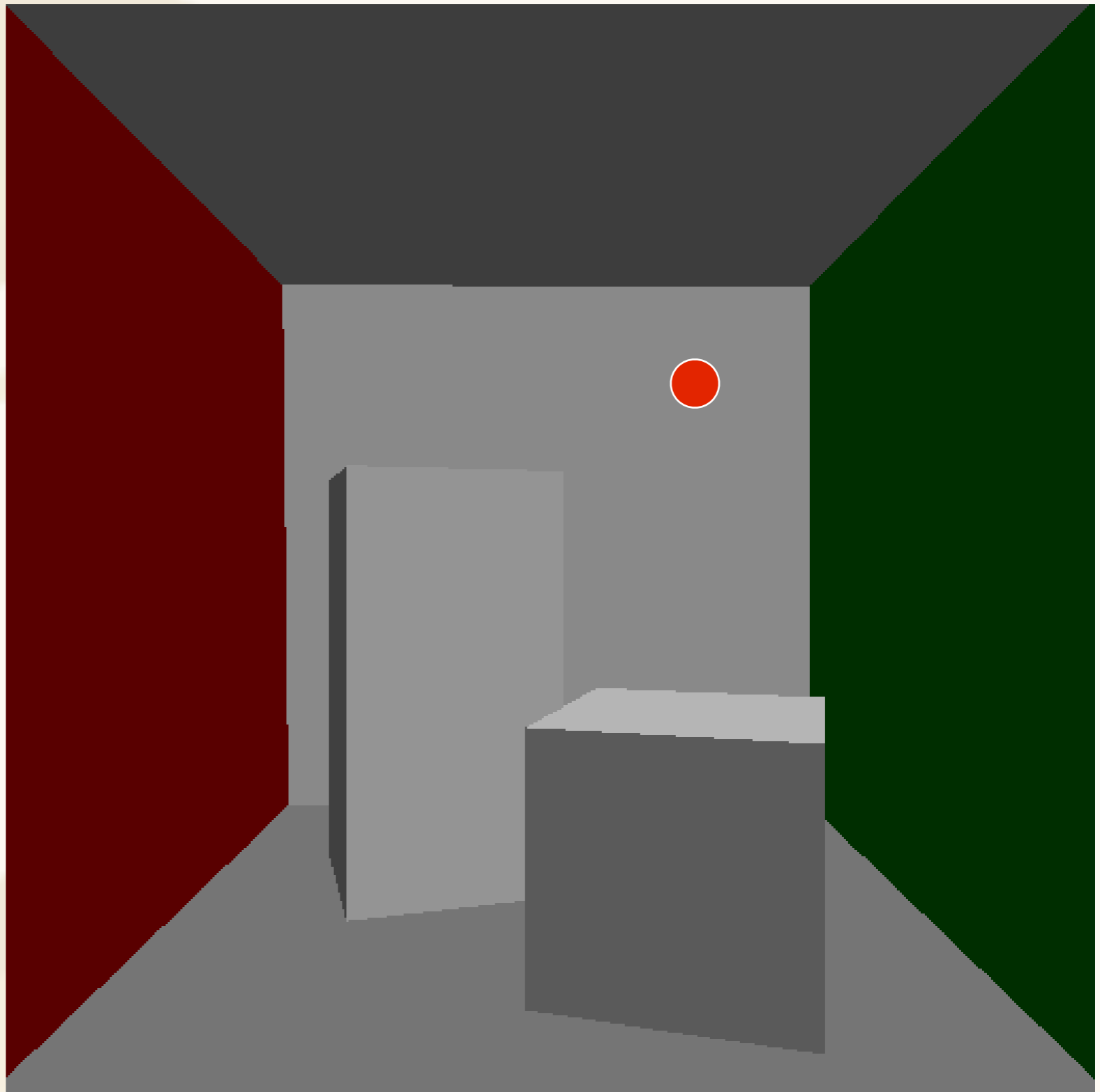
Specify 1 seed point



We start off by having the user specify 1 seed point in the scene. This point needs to be somewhere in the free space within the scene where it sees such surfaces that we want our basis points on.

Basis Construction

Specify 1 seed point

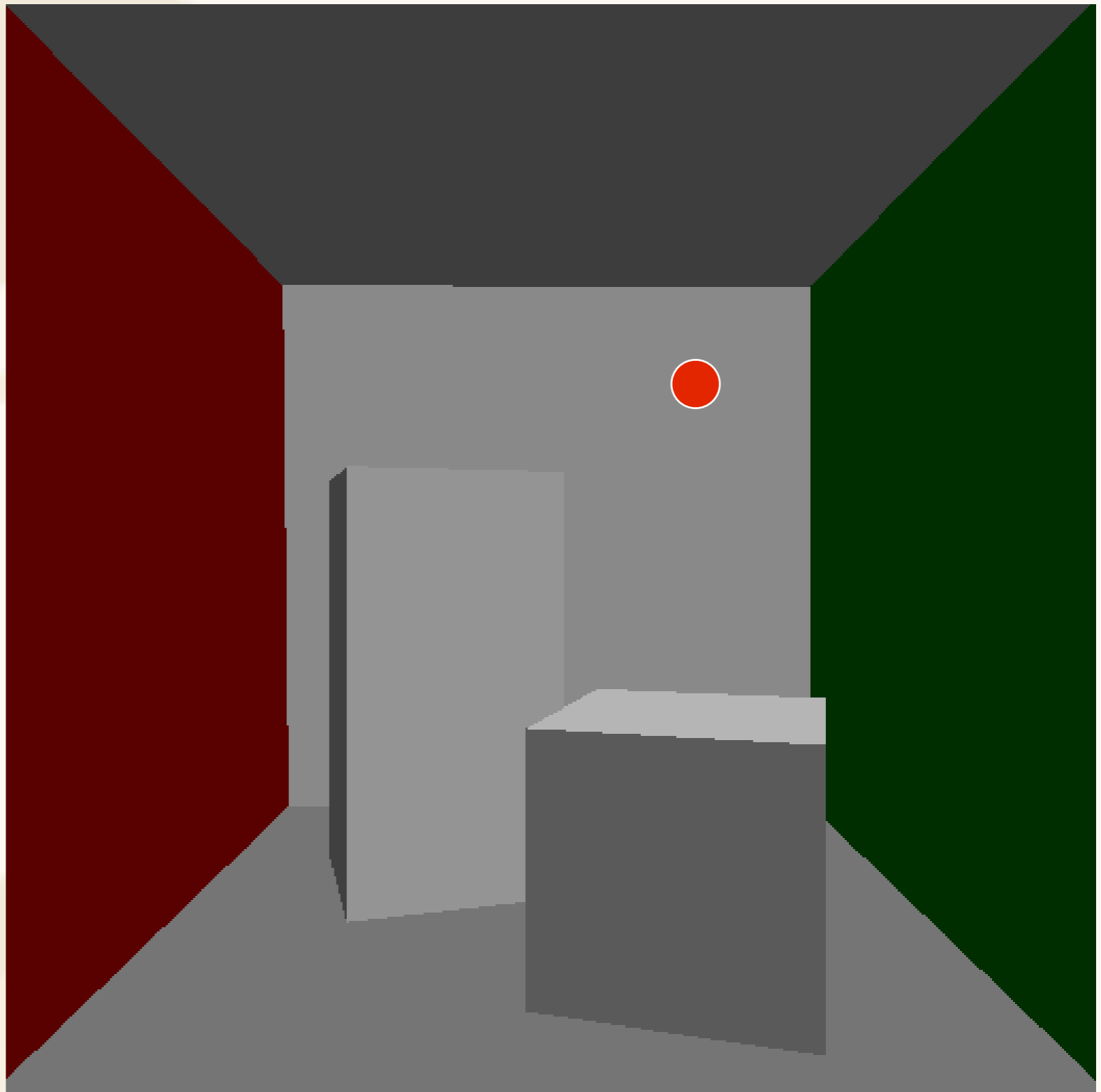


We start off by having the user specify 1 seed point in the scene. This point needs to be somewhere in the free space within the scene where it sees such surfaces that we want our basis points on.

Basis Construction

Specify 1 seed point

**Generate
candidate particles**
- ray tracing

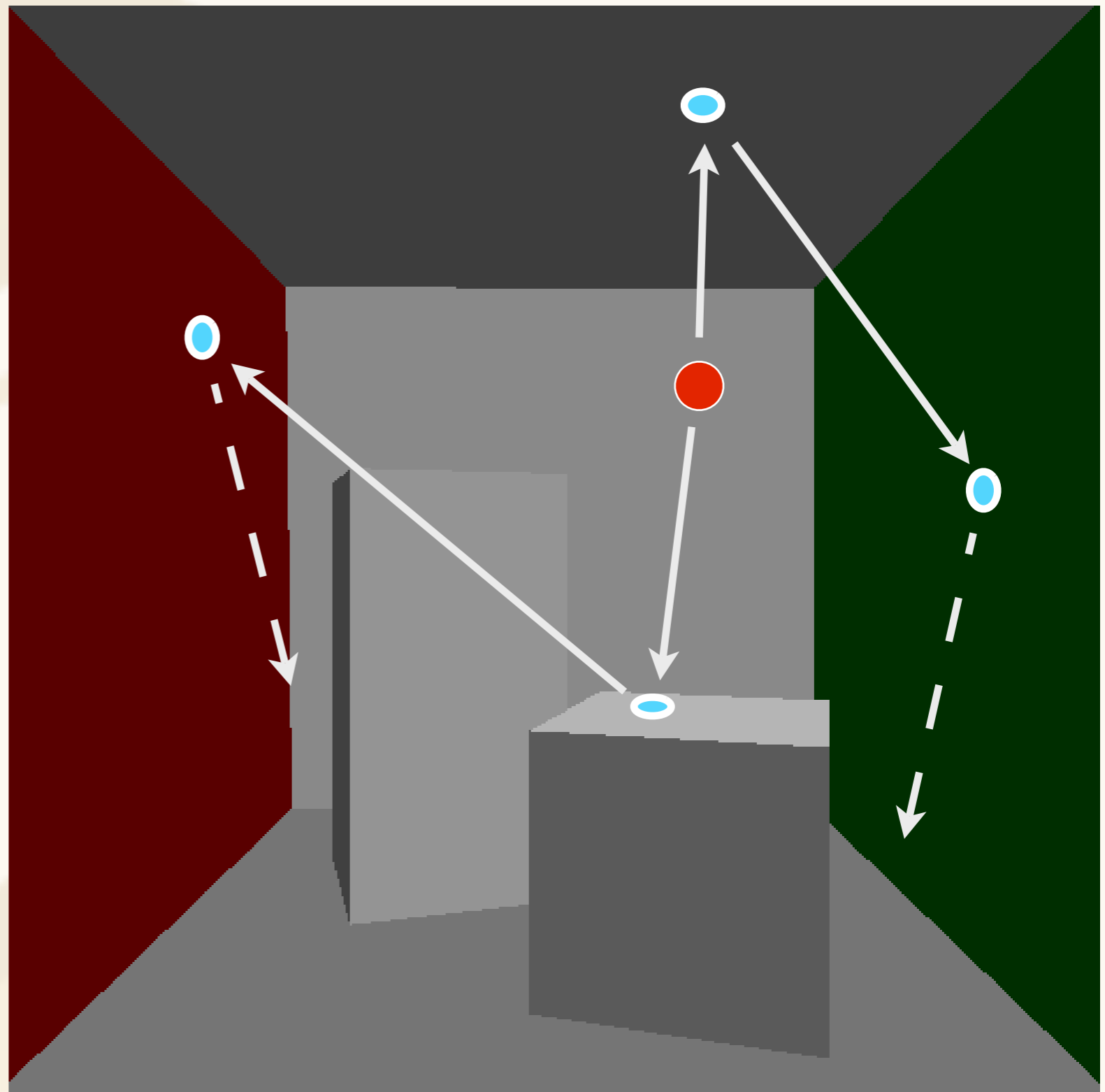


Then, we generate a set of candidate samples by tracing rays backwards from the seed point. When the ray hits the scene, we store a candidate, then let the ray reflect randomly, deposit a new candidate at the new intersection, etc. We let the rays bounce 30 times.

Basis Construction

Specify 1 seed point

**Generate
candidate particles**
- ray tracing

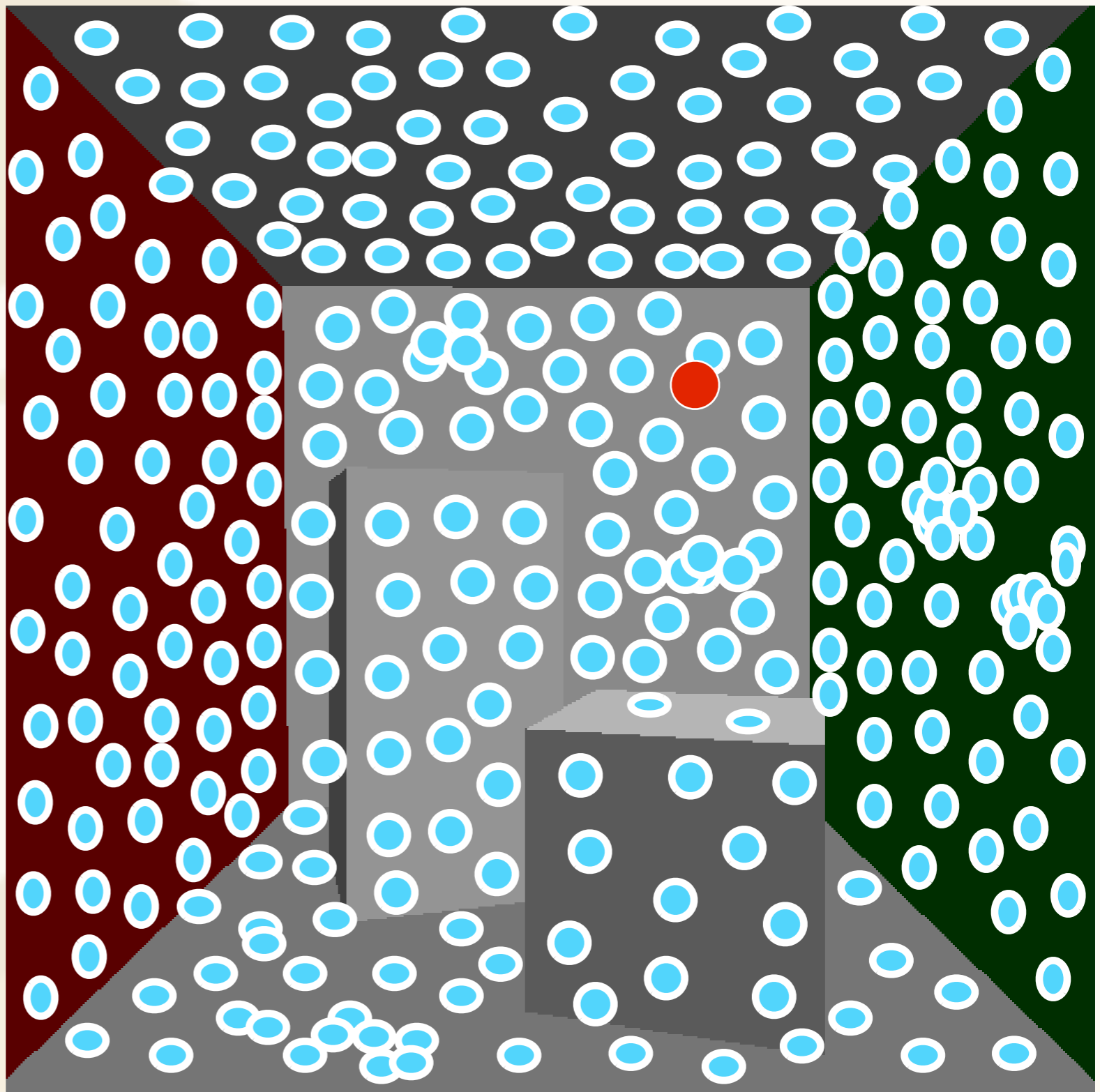


Then, we generate a set of candidate samples by tracing rays backwards from the seed point. When the ray hits the scene, we store a candidate, then let the ray reflect randomly, deposit a new candidate at the new intersection, etc. We let the rays bounce 30 times.

Basis Construction

Specify 1 seed point

**Generate
candidate particles**
- ray tracing



This results in a large set of candidates. Note that these points are not yet Poisson disk distributed.

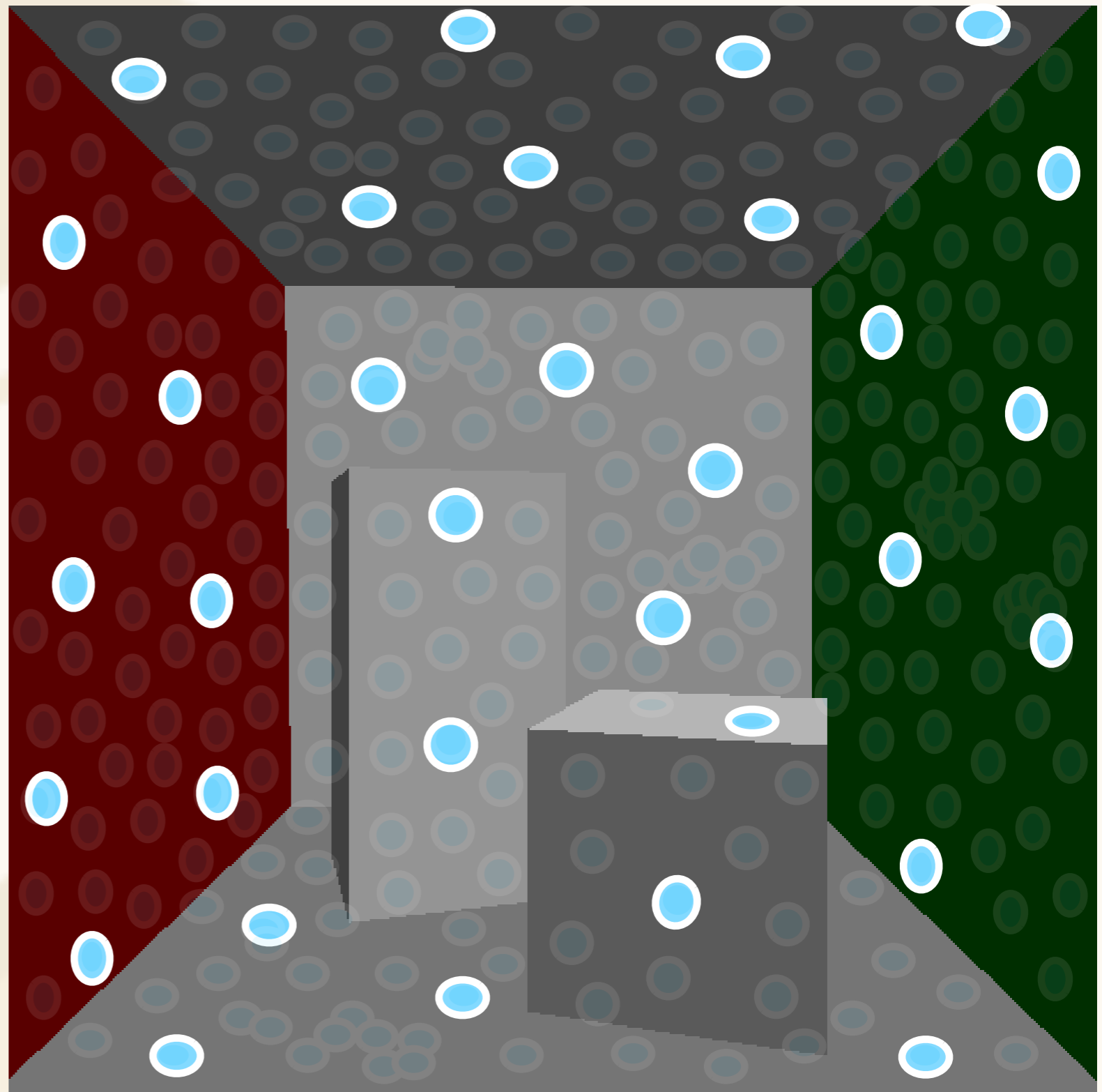
Basis Construction

Specify 1 seed point

Generate
candidate particles
- ray tracing

**Reject samples that fail
Poisson criterion
(Dart throwing)**

- Radius related to support size
- Obeys 5D metric induced by weight functions



Once we have the candidates, we use a simple dart throwing algorithm to pick a subset of the points that respects the Poisson criterion according to a 5D metric induced by our weight functions. The acceptance radius is related to the support sizes of the weight functions. Finer levels of the hierarchy have smaller radiuses.

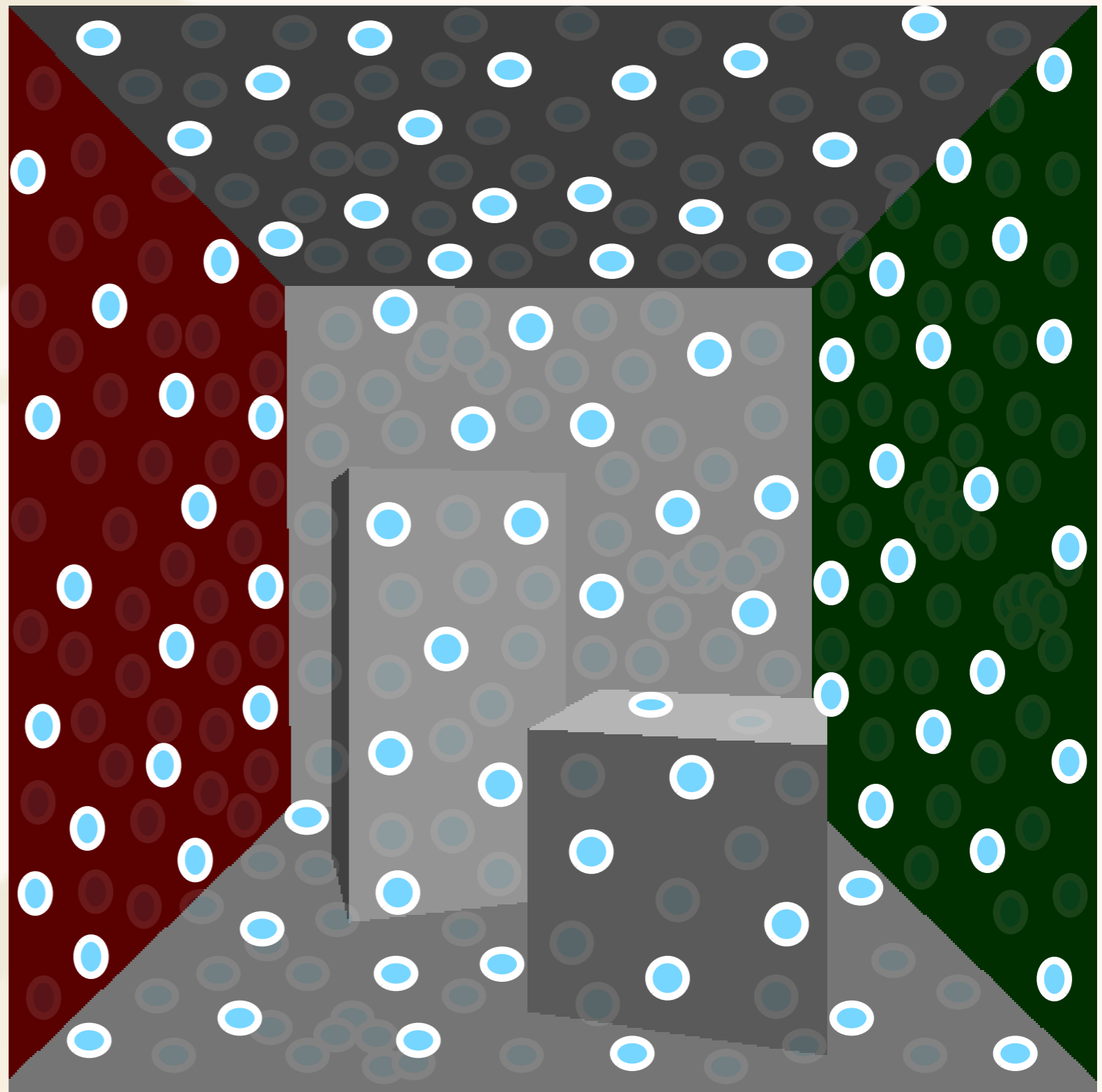
Basis Construction

Specify 1 seed point

Generate
candidate particles
- ray tracing

**Reject samples that fail
Poisson criterion
(Dart throwing)**

- Radius related to support size
- Obeys 5D metric induced by weight functions



The same geometry-independent process applies to all levels of hierarchy.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

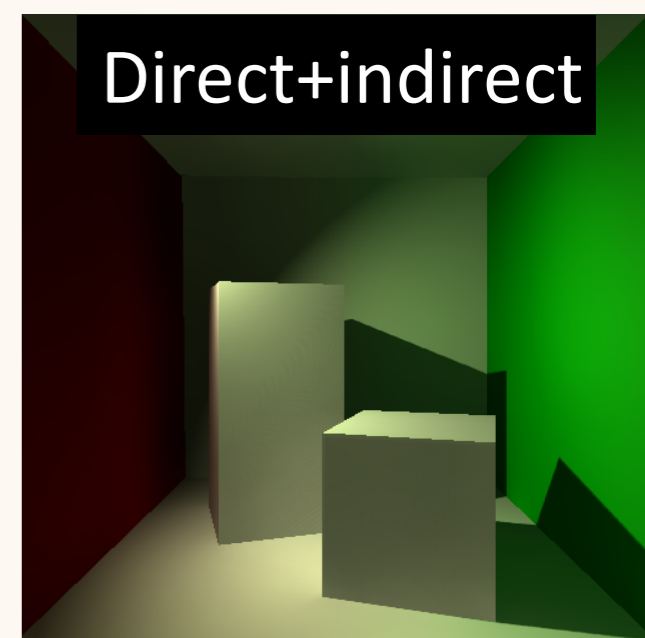
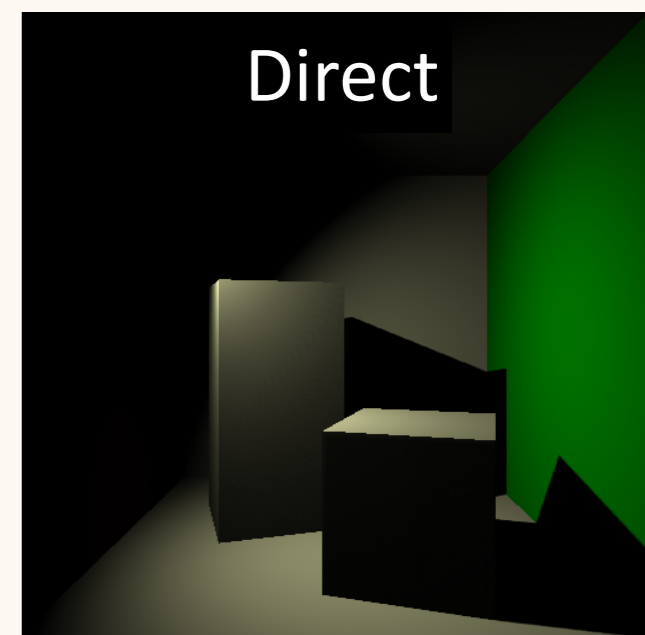
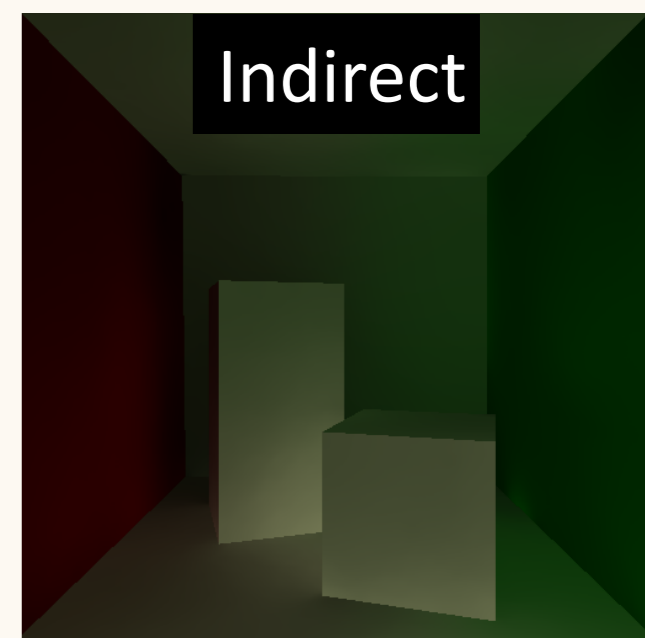
Rendering on GPU

Discussion and Conclusions

Application

Direct-to-Indirect PRT

- Moving camera and local light source
- Interactive indirect illumination
- Basic approach:
 - indirect from meshless basis
 - per-pixel direct lighting w/ shadow maps
 - precompute direct-to-indirect transport
 - follows [Kontkanen 06], [Hasan 06]



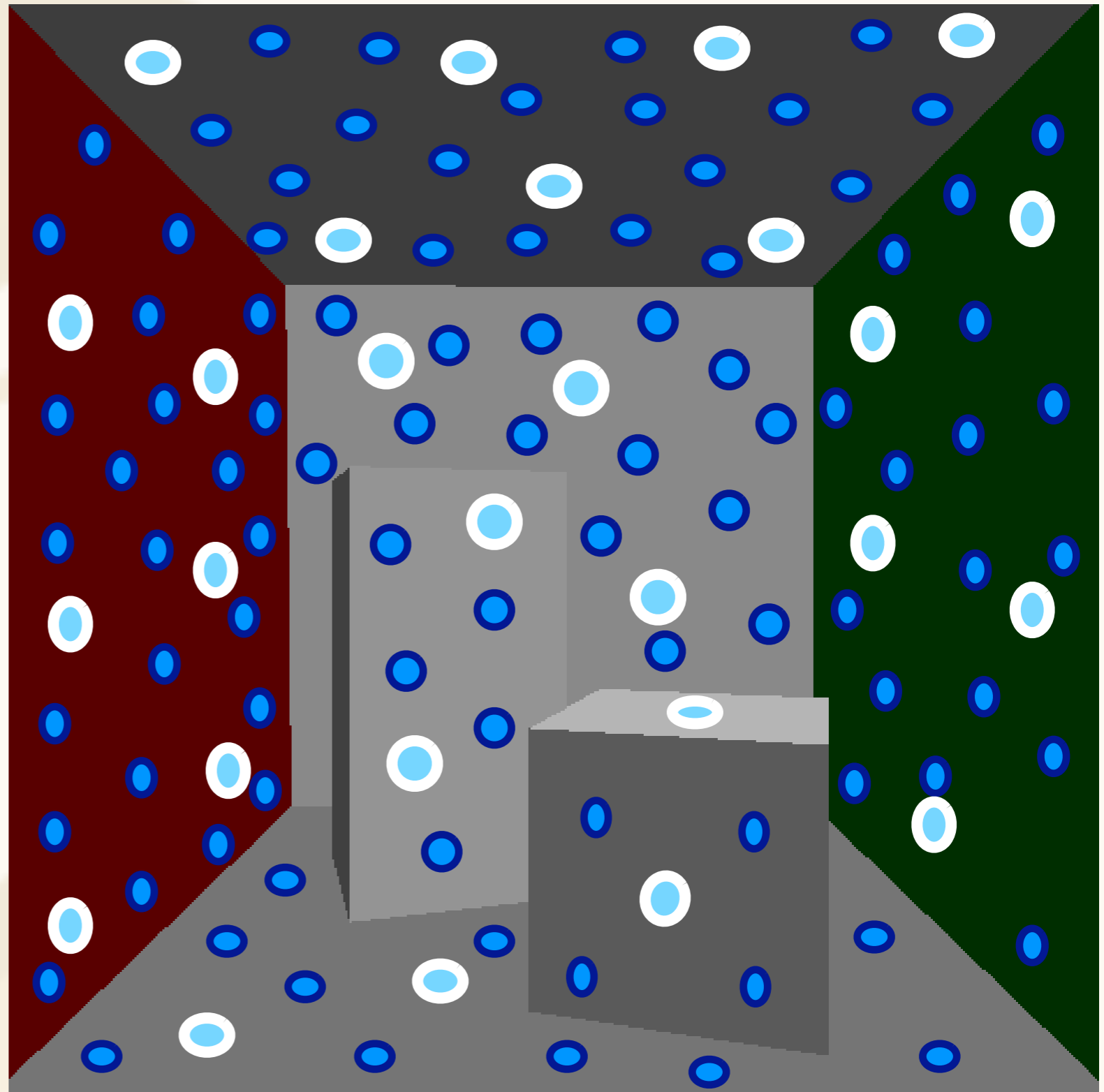
As an application of our basis, we describe a hierarchical direct-to-indirect PRT technique for complex geometry and local light sources, where we are able to move both the camera and a the local light source with interactive indirect illumination.

The basic idea is to render indirect illumination from the basis and compose that with per-pixel direct illumination rendered using traditional realtime techniques. The direct illumination is first projected into our basis. Then, a precomputed hierarchical light transport matrix is used to determine the indirect illumination given the direct illumination.

The basic premise is the same as in the work of Kontkanen and others and Hasan and others, but we are limited to neither simple geometry nor a fixed view.

Preprocessing

- compute transfer matrix
(light transport between
basis functions)

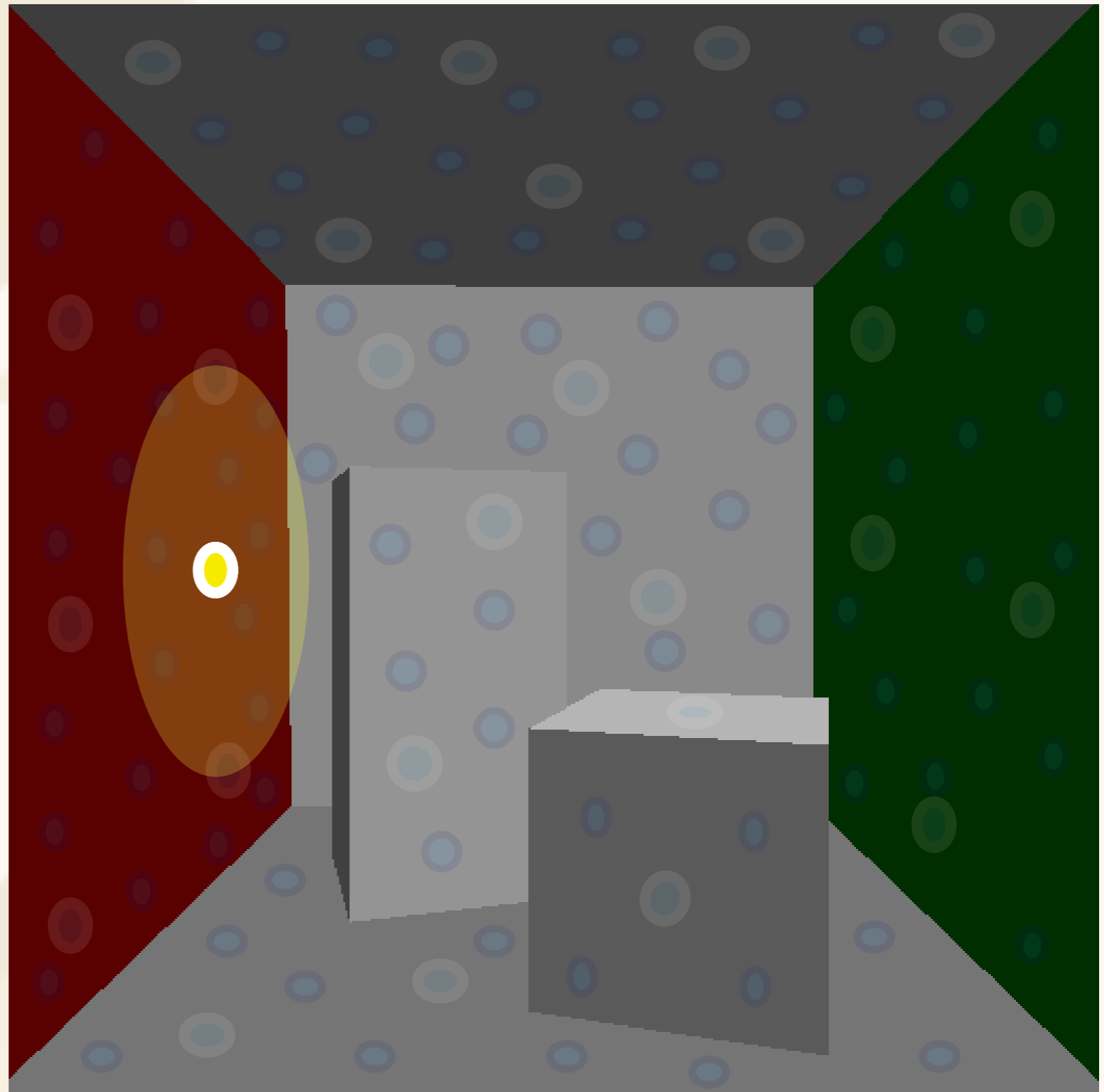


Let's start with the precomputation step, and let's look at a single basis function, denoted in yellow.

Now we ask the question: If this single basis function emits light at unit intensity, what is the resulting global illumination in the rest of the scene?

Preprocessing

- compute transfer matrix
(light transport between
basis functions)



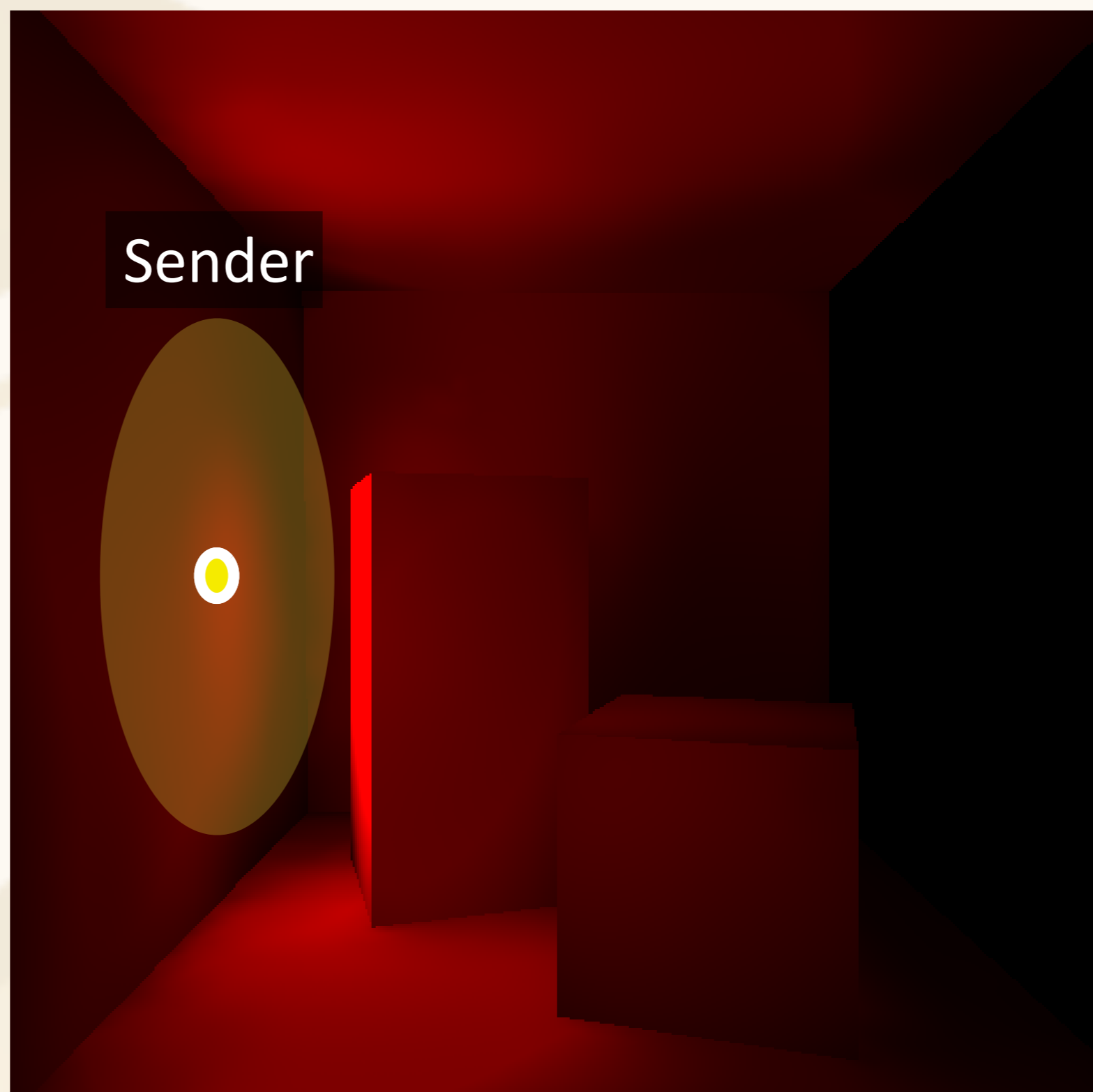
Let's start with the precomputation step, and let's look at a single basis function, denoted in yellow.

Now we ask the question: If this single basis function emits light at unit intensity, what is the resulting global illumination in the rest of the scene?

Preprocessing

- compute transfer matrix (light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Links from one sender basis function



That illumination is displayed here. Since the basis function resides on a red wall, the illumination it transmits is also red due to color bleeding. Now, this illumination is projected into the basis, meaning that we compute so-called links between the sender and the receiving functions.

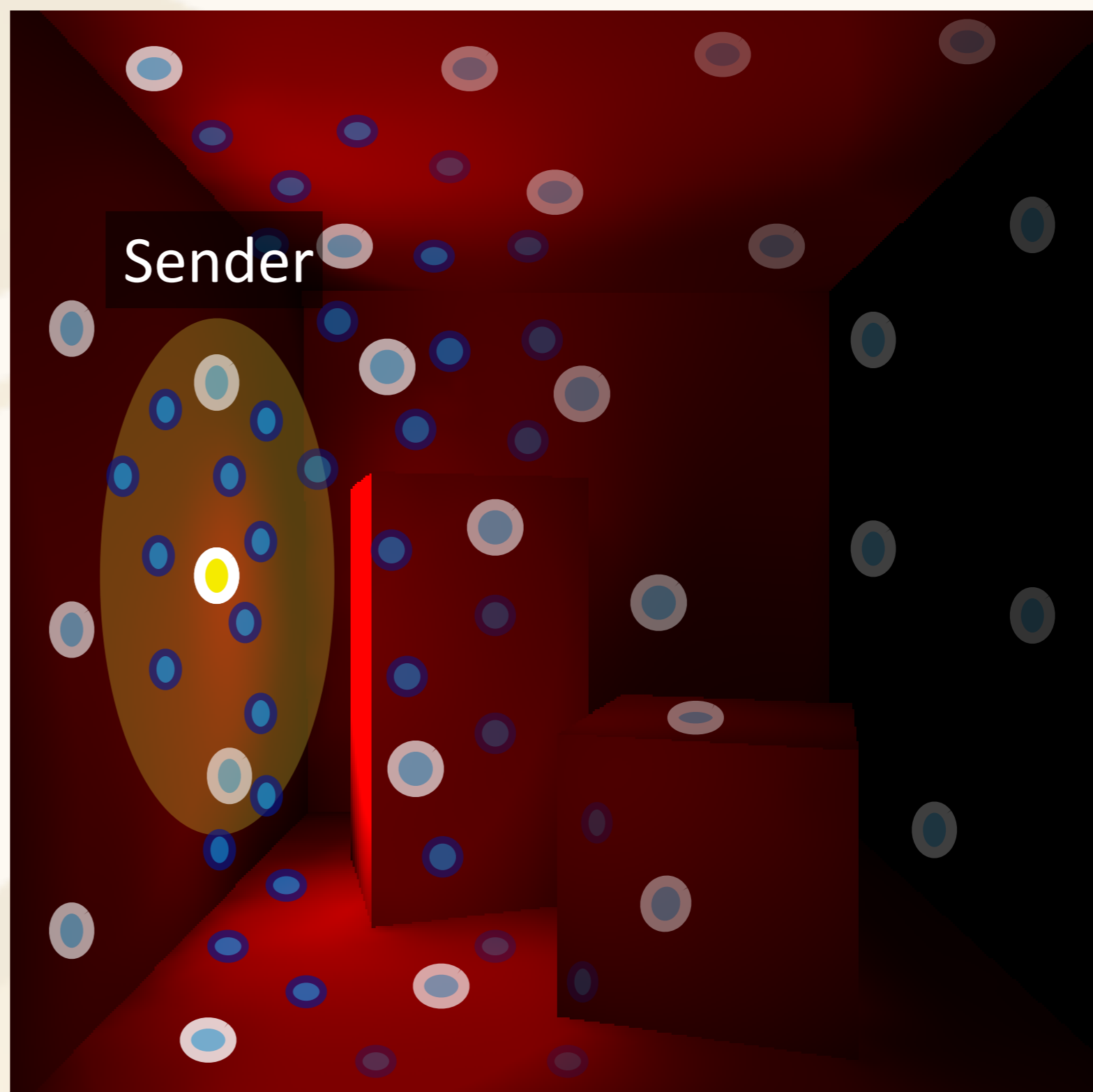
In the spirit of classical hierarchical radiosity, not all interactions need to be resolved to full accuracy, meaning for example that distant transfers can use a lower resolution than nearby ones. This is key to achieving efficient precomputation, since not all pairs of basis functions have to be considered. Unlike almost all previous PRT work, this can be seen as precomputation directly in compressed domain. Please see the paper for details.

Given the matrix formed by the links it is easy to determine the indirect illumination in the scene once we know the direct illumination: We just follow the links from senders to receivers and accumulate.

Preprocessing

- compute transfer matrix (light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Links from one sender basis function



That illumination is displayed here. Since the basis function resides on a red wall, the illumination it transmits is also red due to color bleeding. Now, this illumination is projected into the basis, meaning that we compute so-called links between the sender and the receiving functions.

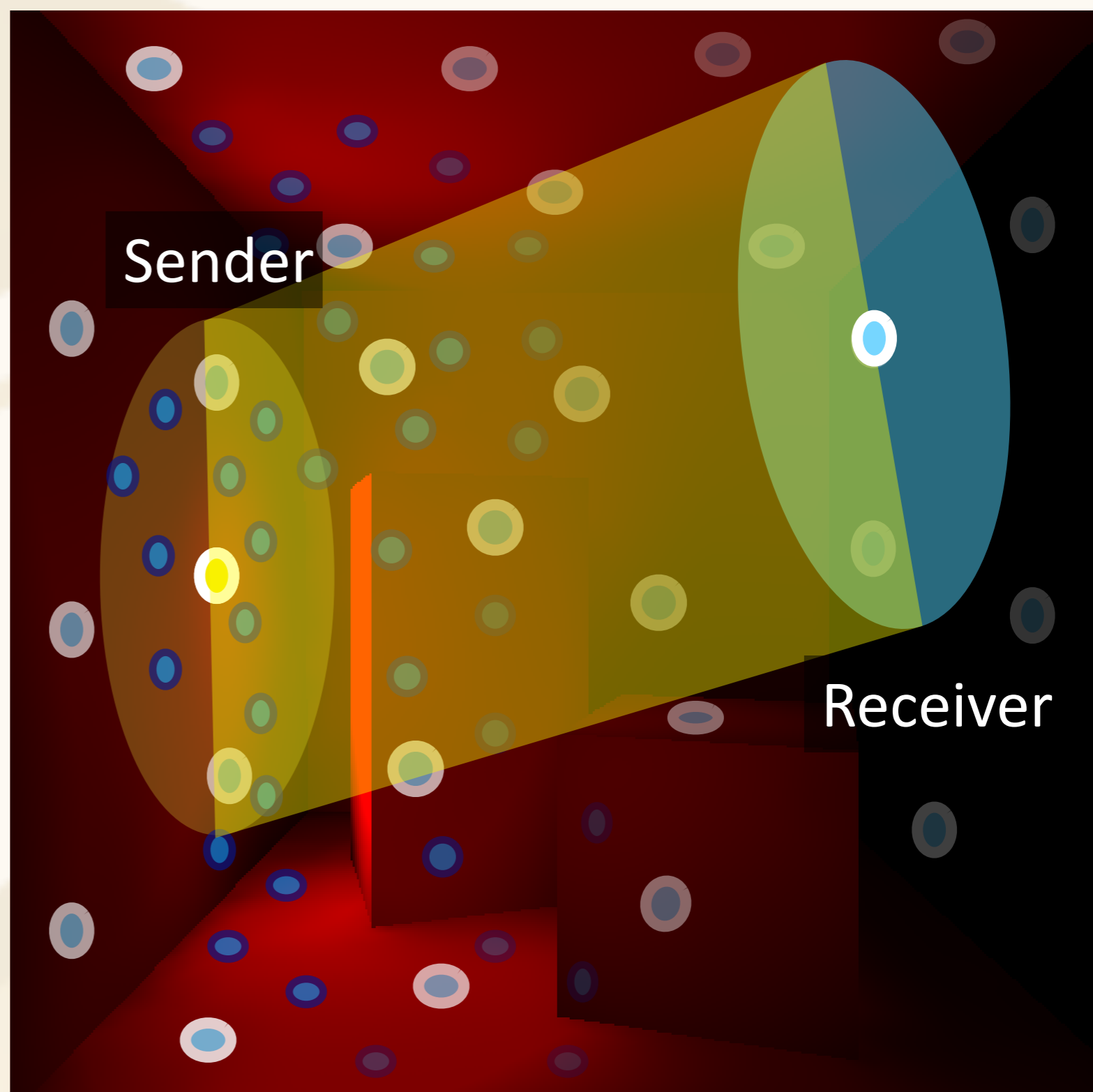
In the spirit of classical hierarchical radiosity, not all interactions need to be resolved to full accuracy, meaning for example that distant transfers can use a lower resolution than nearby ones. This is key to achieving efficient precomputation, since not all pairs of basis functions have to be considered. Unlike almost all previous PRT work, this can be seen as precomputation directly in compressed domain. Please see the paper for details.

Given the matrix formed by the links it is easy to determine the indirect illumination in the scene once we know the direct illumination: We just follow the links from senders to receivers and accumulate.

Preprocessing

- compute transfer matrix (light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Links from one sender basis function



That illumination is displayed here. Since the basis function resides on a red wall, the illumination it transmits is also red due to color bleeding. Now, this illumination is projected into the basis, meaning that we compute so-called links between the sender and the receiving functions.

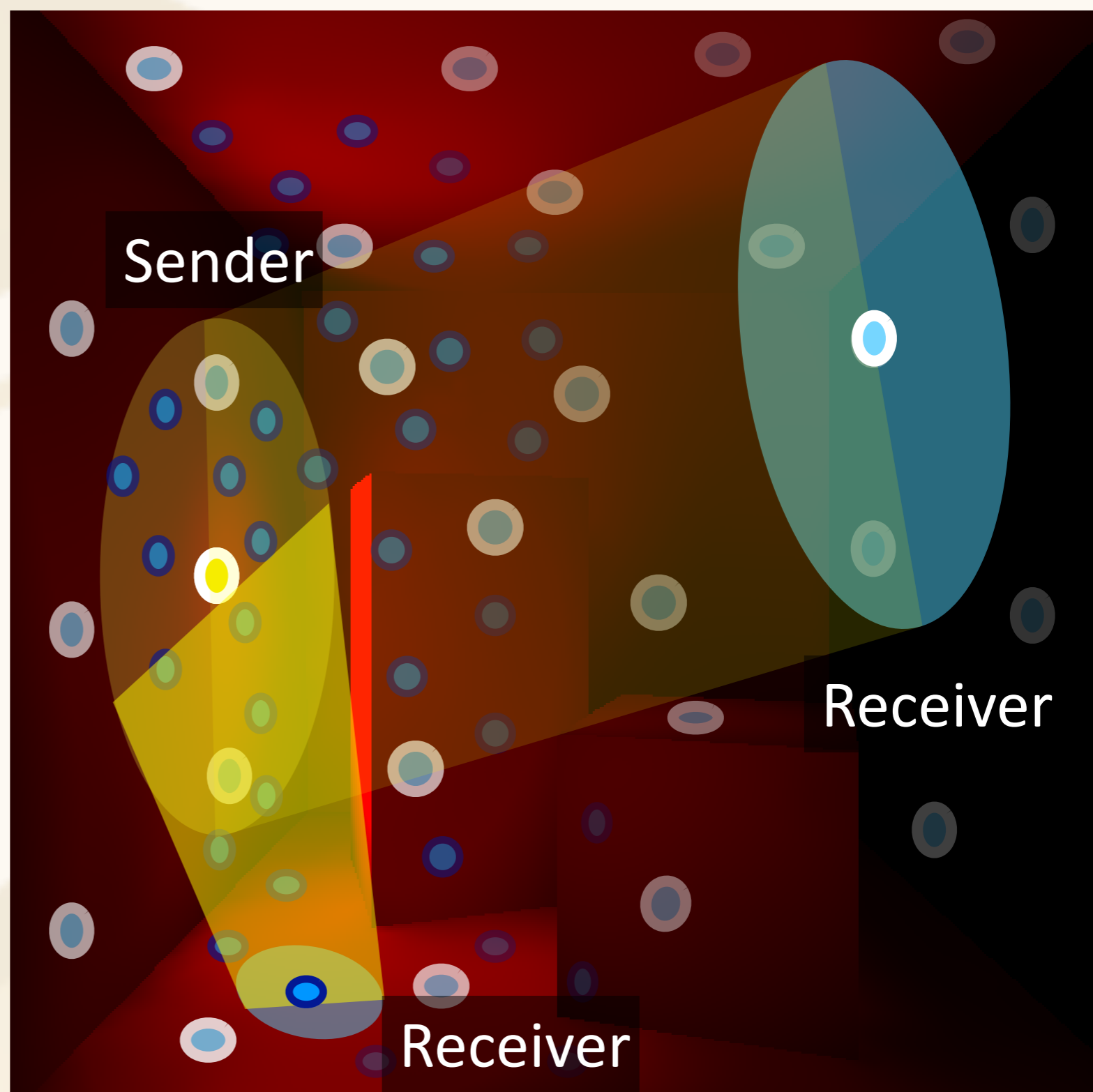
In the spirit of classical hierarchical radiosity, not all interactions need to be resolved to full accuracy, meaning for example that distant transfers can use a lower resolution than nearby ones. This is key to achieving efficient precomputation, since not all pairs of basis functions have to be considered. Unlike almost all previous PRT work, this can be seen as precomputation directly in compressed domain. Please see the paper for details.

Given the matrix formed by the links it is easy to determine the indirect illumination in the scene once we know the direct illumination: We just follow the links from senders to receivers and accumulate.

Preprocessing

- compute transfer matrix (light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Links from one sender basis function



That illumination is displayed here. Since the basis function resides on a red wall, the illumination it transmits is also red due to color bleeding. Now, this illumination is projected into the basis, meaning that we compute so-called links between the sender and the receiving functions.

In the spirit of classical hierarchical radiosity, not all interactions need to be resolved to full accuracy, meaning for example that distant transfers can use a lower resolution than nearby ones. This is key to achieving efficient precomputation, since not all pairs of basis functions have to be considered. Unlike almost all previous PRT work, this can be seen as precomputation directly in compressed domain. Please see the paper for details.

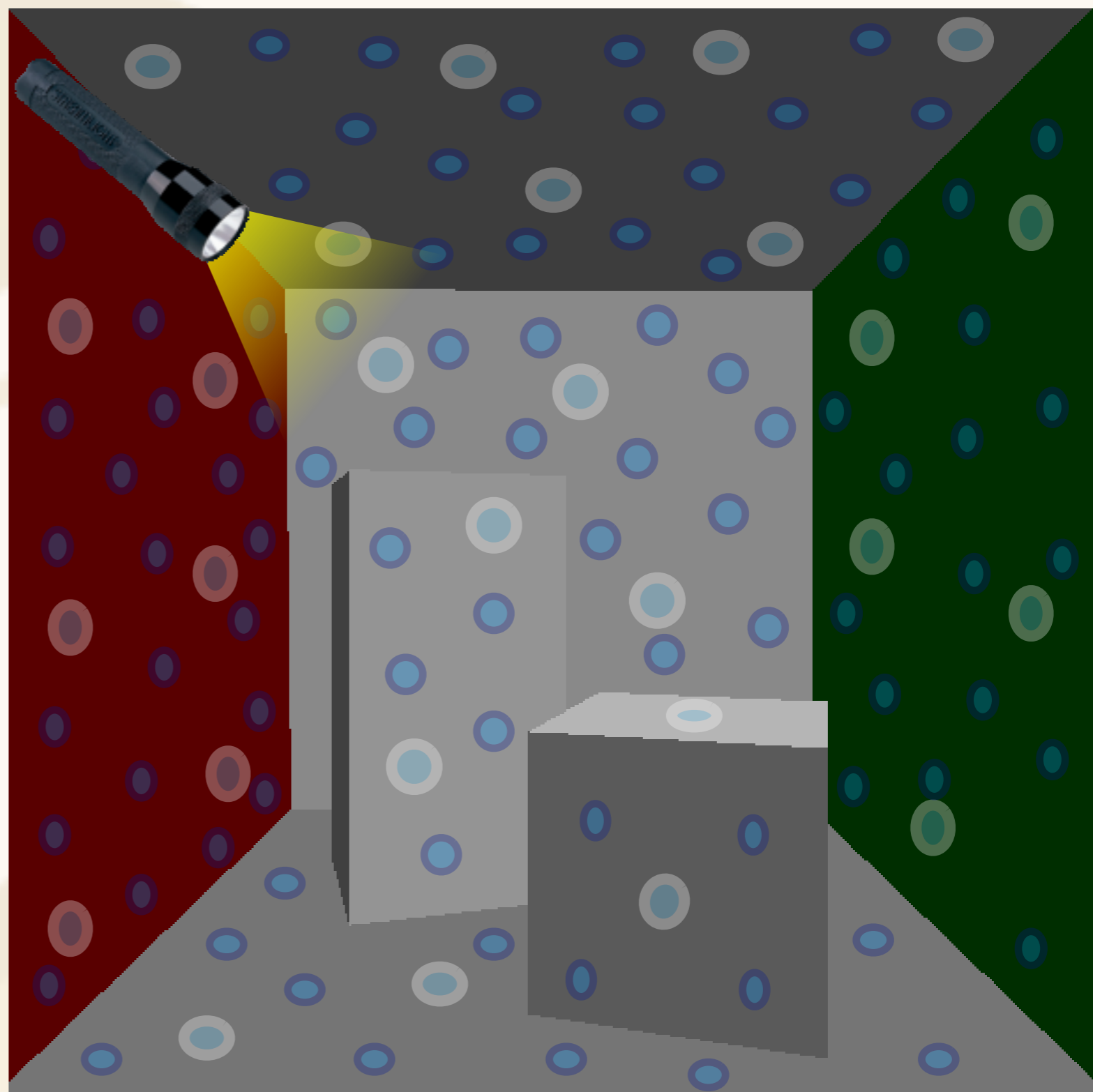
Given the matrix formed by the links it is easy to determine the indirect illumination in the scene once we know the direct illumination: We just follow the links from senders to receivers and accumulate.

Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- **Project direct illumination
(trace rays to samples)**



Now, the runtime usage of the matrix is simple.

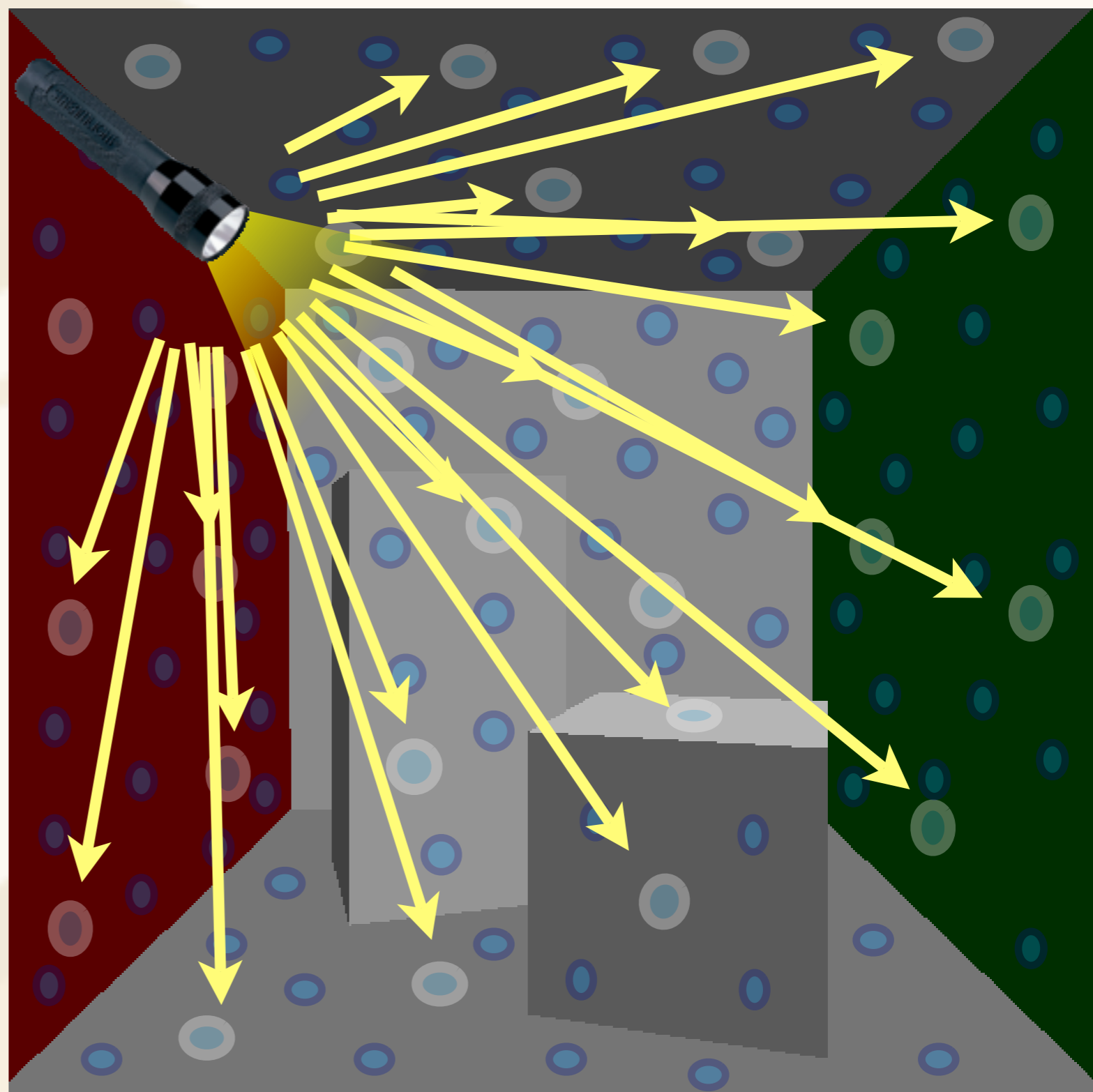
Each frame we compute the basis coefficients for direct illumination by casting rays from the light to the samples.

Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- **Project direct illumination
(trace rays to samples)**



Now, the runtime usage of the matrix is simple.

Each frame we compute the basis coefficients for direct illumination by casting rays from the light to the samples.

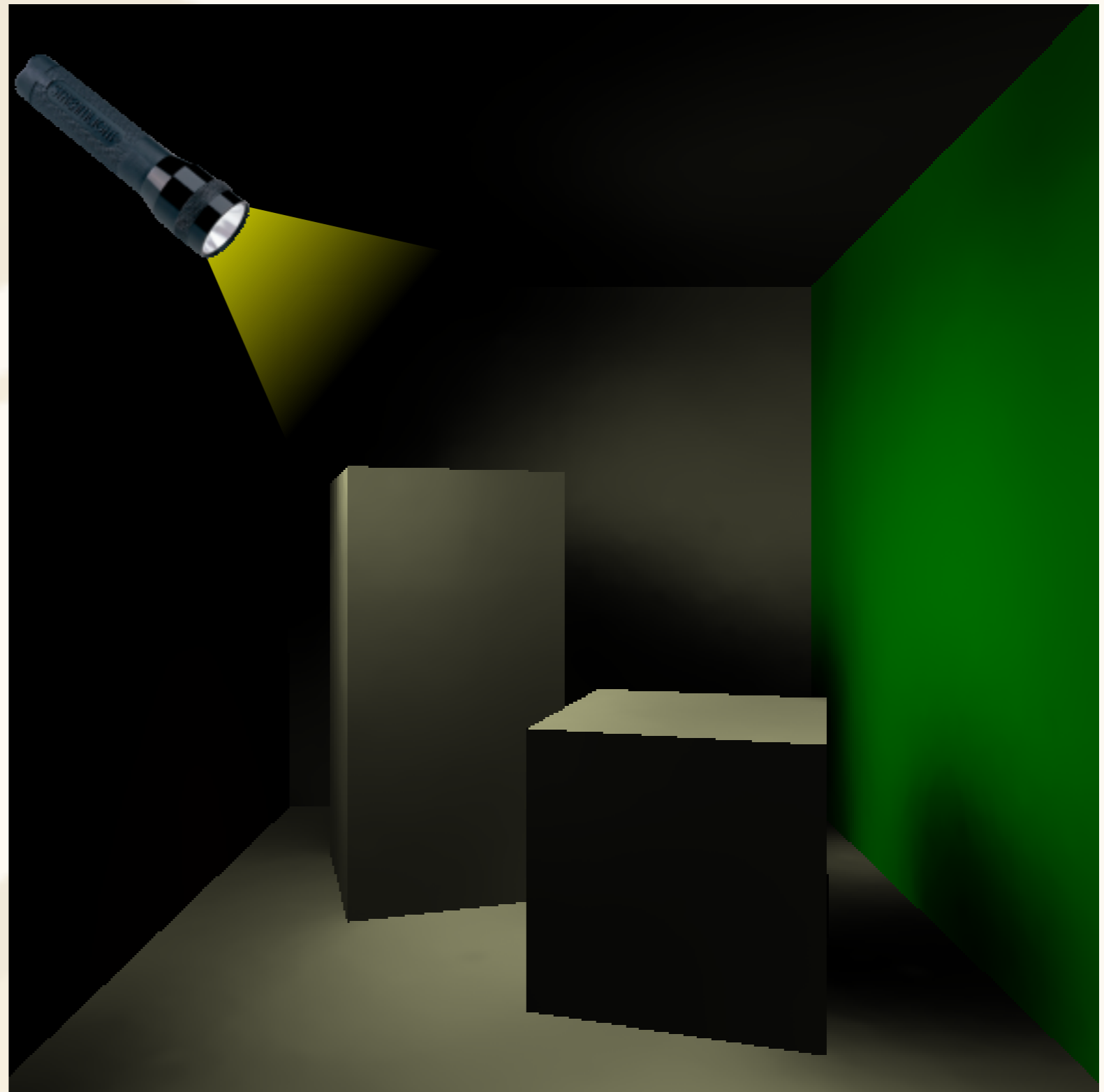
Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- **Project direct illumination
(trace rays to samples)**

**Approximate direct illumination
(never visualized directly)**



The result is a basis expansion for direct illumination, which is visualized here for didactic purposes only.

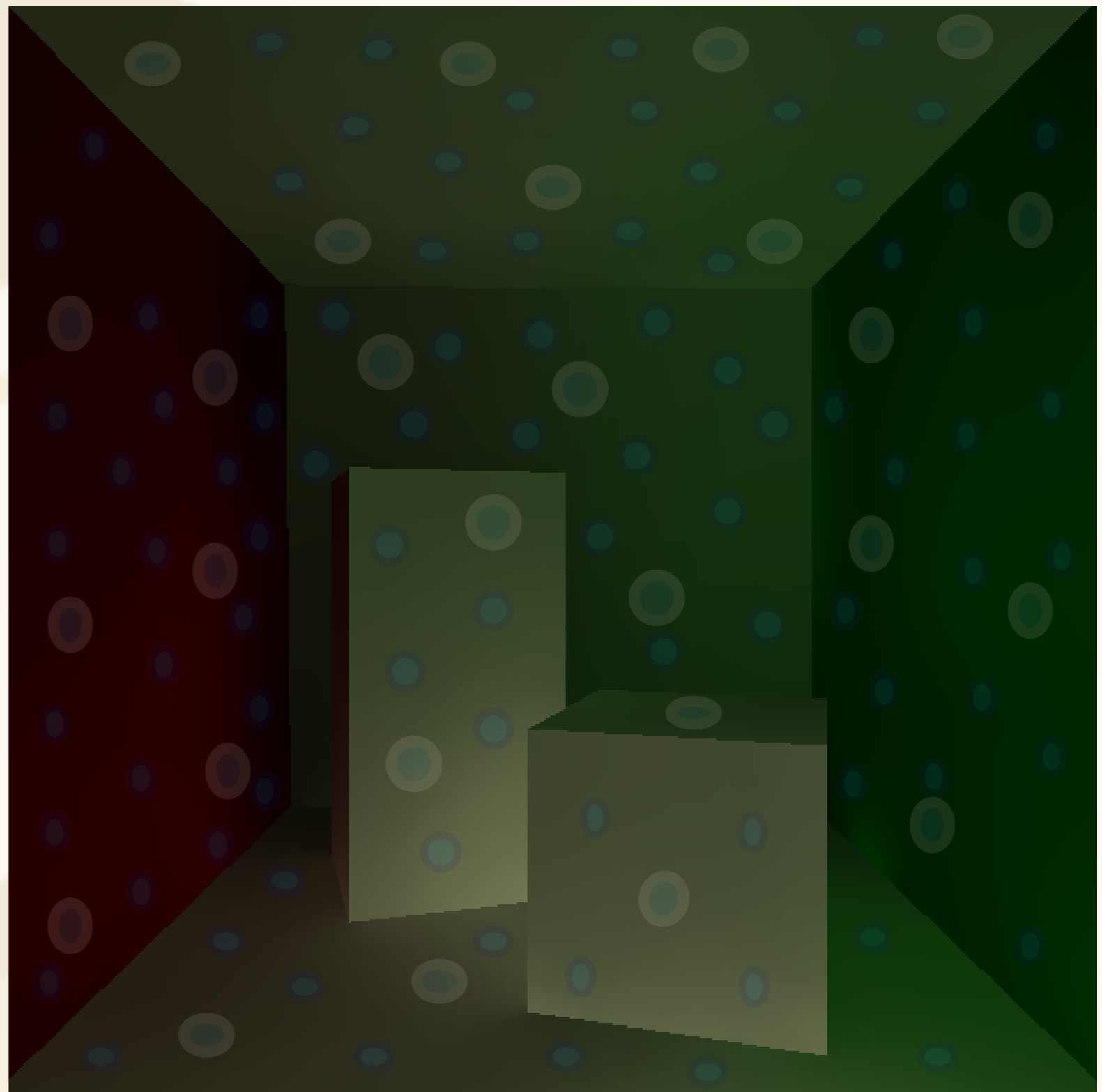
Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- Project direct illumination
(trace rays to samples)
- **Apply transport matrix**

Indirect Illumination



Now we merely multiply the direct illumination coefficient vector by the hierarchical sparse matrix which gives us a vector that describes indirect illumination.

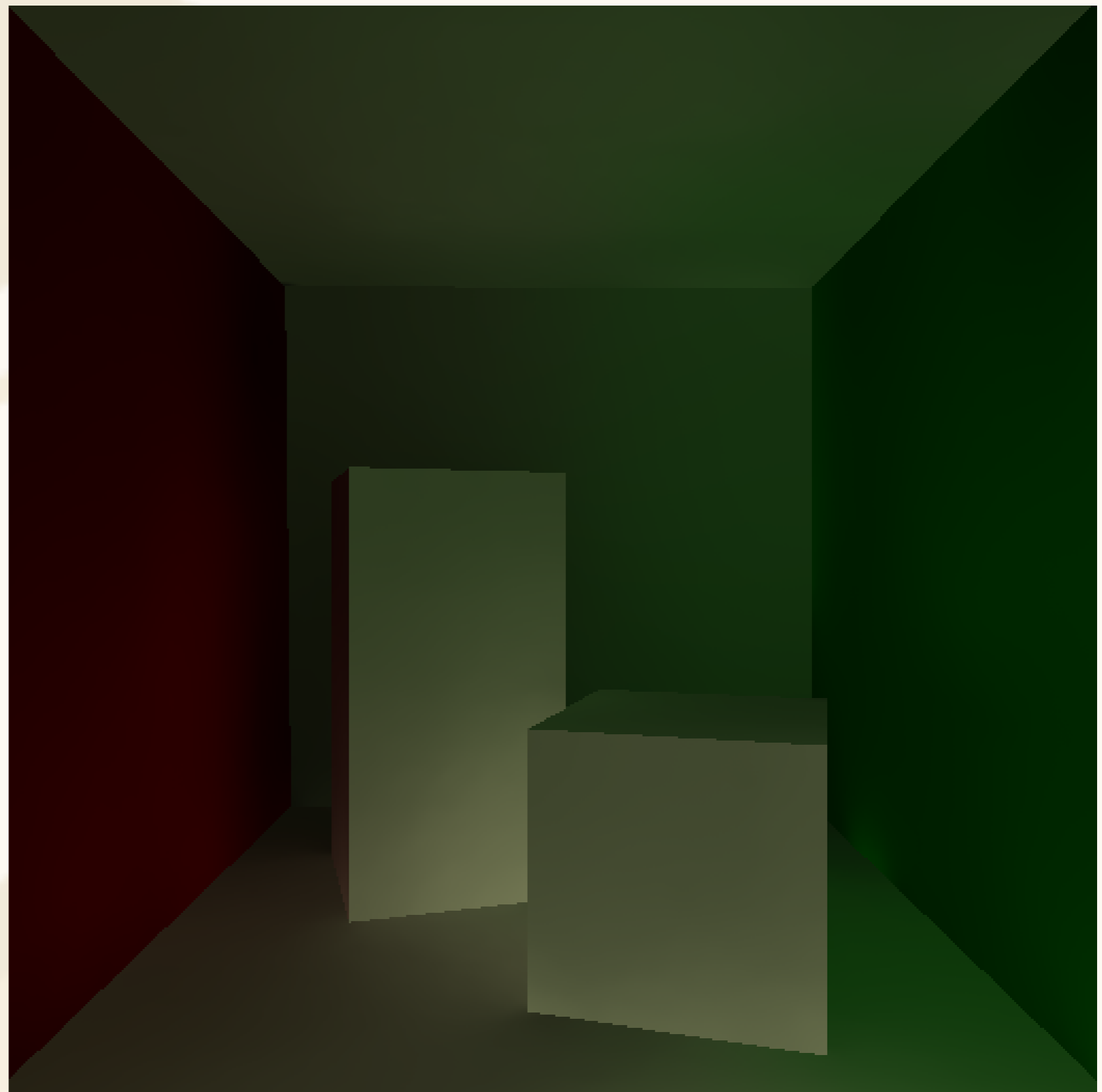
Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- Project direct illumination
(trace rays to samples)
- **Apply transport matrix**

Indirect Illumination



Now we merely multiply the direct illumination coefficient vector by the hierarchical sparse matrix which gives us a vector that describes indirect illumination.

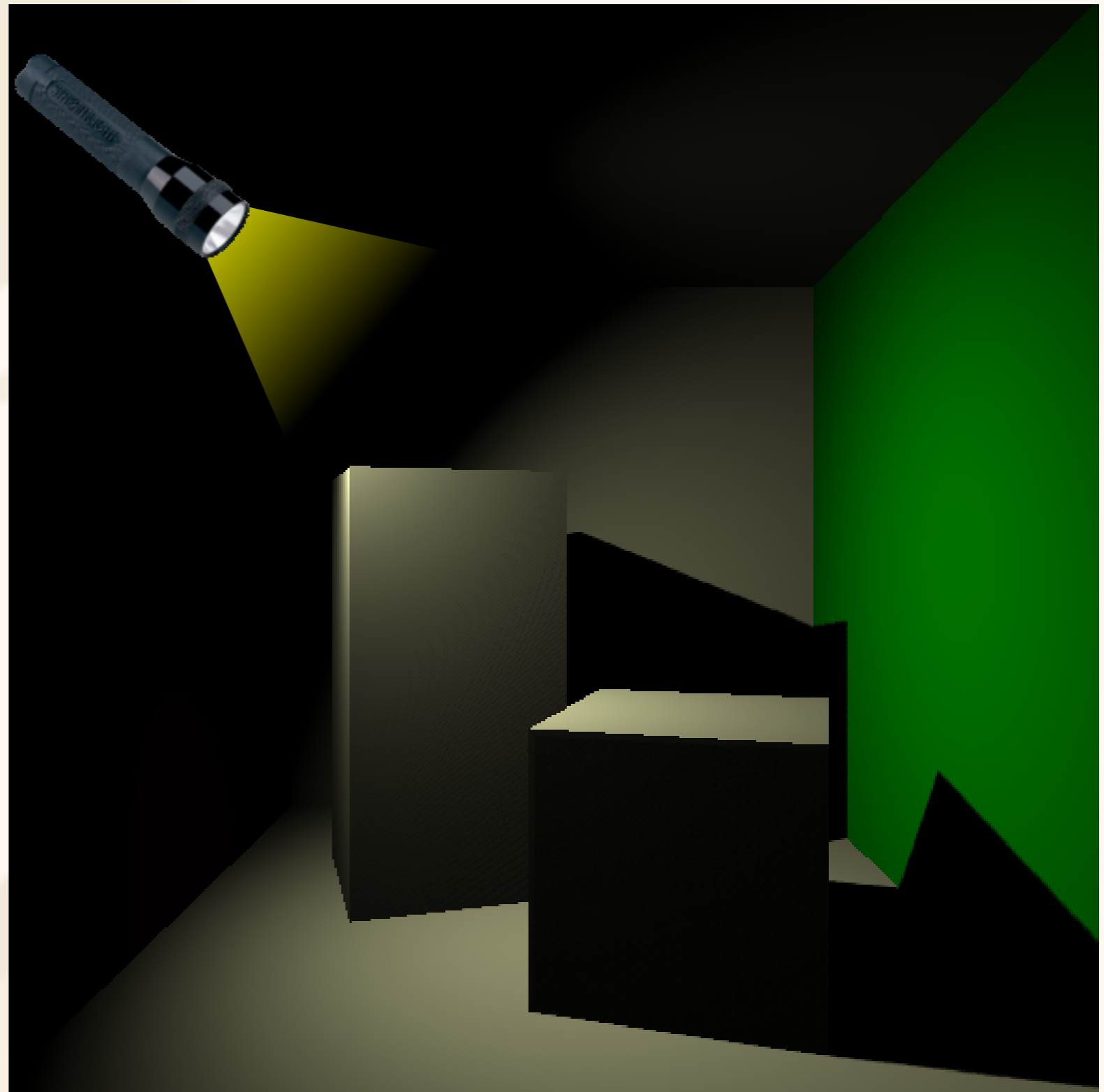
Preprocessing

- compute transfer matrix (light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- Project direct illumination (trace rays to samples)
- Apply transport matrix
- **Render per-pixel direct (shadow map)**

Per-Pixel Direct Illumination



Then, a per-pixel direct illumination image is rendered using shadow maps..

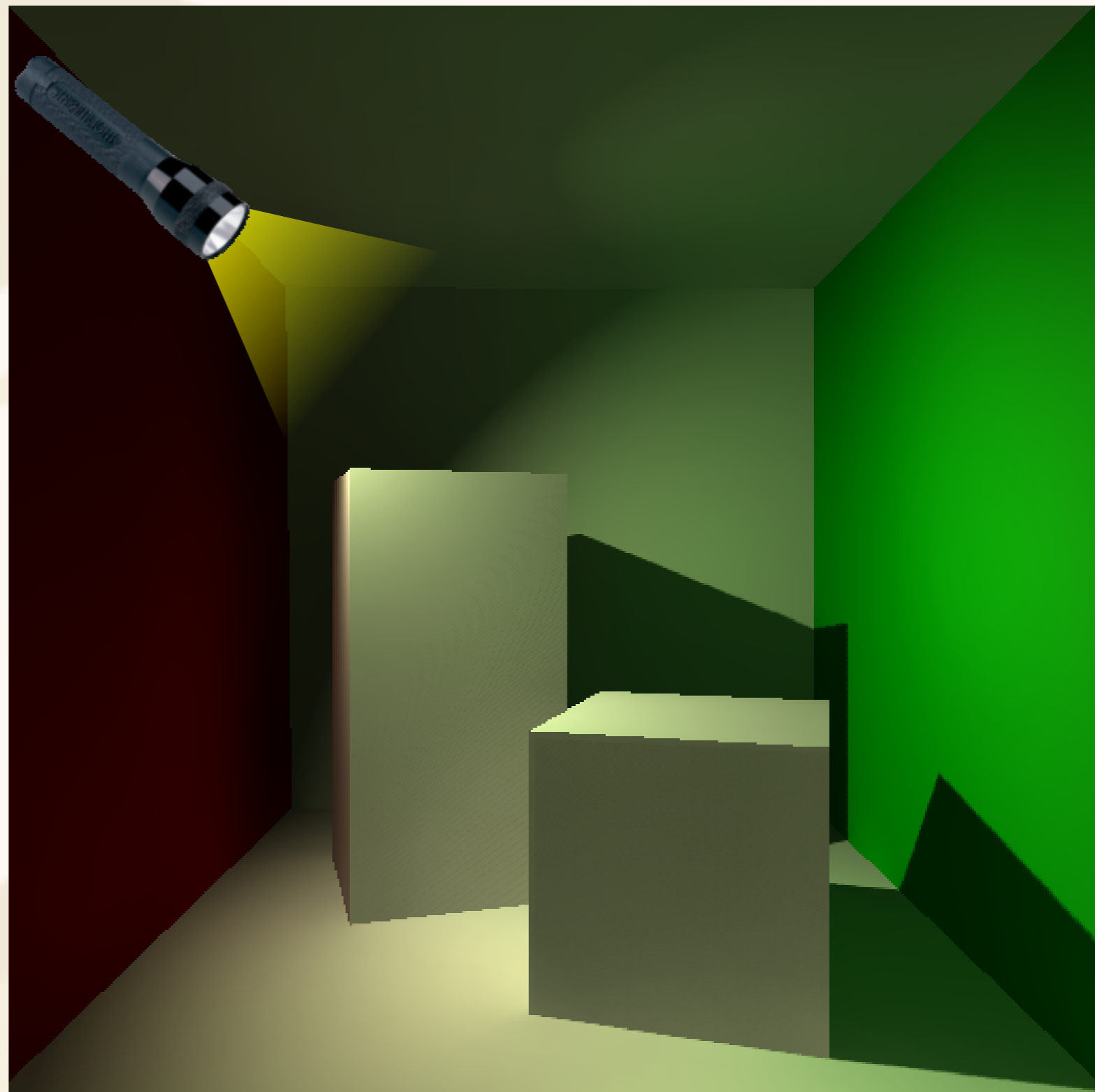
Preprocessing

- compute transfer matrix
(light transport between basis functions)
- sparsity due to hierarchy
- hierarchical refinement

Runtime (for each frame)

- Project direct illumination
(trace rays to samples)
- Apply transport matrix
- Render per-pixel direct
(shadow map)
- **Compose per-pixel direct
and indirect from basis**

Global Illumination (direct+indirect)



And finally composited with the indirect image. This yields a dynamic global illumination solution.

PRT Results Scenes

- **Sponza atrium (66 ktri)**
 - used widely in previous work
- **Great Hall (2.3 Mtri)**
 - difficult, intricate geometry
- **Kung Fu Panda (5.1 Mtri)**
 - courtesy of DreamWorks Animation
 - tessellated from actual production geometry



Let's look at some results. We used three scenes of increasing complexity. The Sponza atrium has been used in lots of previous work, while the latter two complex scenes contain both smooth geometry, detailed surfaces, and lots of topologically disconnected geometry such as cobblestones. They would be tough to parameterize with wavelets, and they contain so much geometry that non-hierarchical bases would result in really long precomputation times.

The 5 million triangle scene from DreamWorks Animation is tessellated from an actual movie set with no hand-tuning.

PRT Results

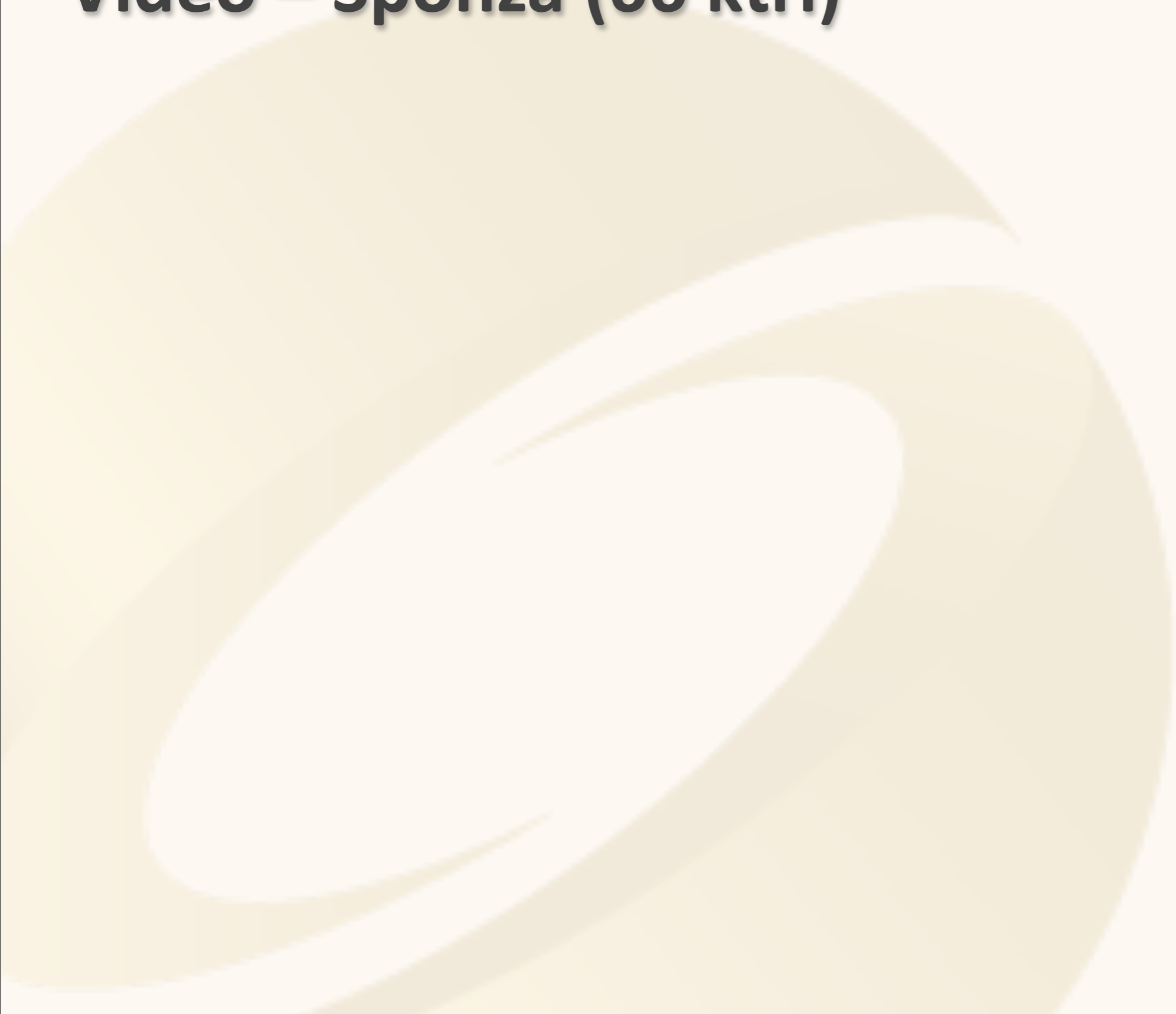
Statistics

- **Precomputation 27–36 min (single PC)**
 - 100x improvement on Sponza compared to [Kristensen 05]
- **Memory usage 78–120 MB**
- **Runtime performance**
 - moving light: 6–9 FPS
 - stationary light: 12–25 FPS

Thanks to the hierarchical precomputation algorithm, we achieve precomputation times in the order of half an hour on a single PC. On the Sponza scene this is two orders of magnitude faster than some prior relighting work for local light sources.

The performance varies between 6 and 9 FPS when the light is moving, and 12 and 25 FPS when the light is stationary but the camera is moving.

Video – Sponza (66 ktri)



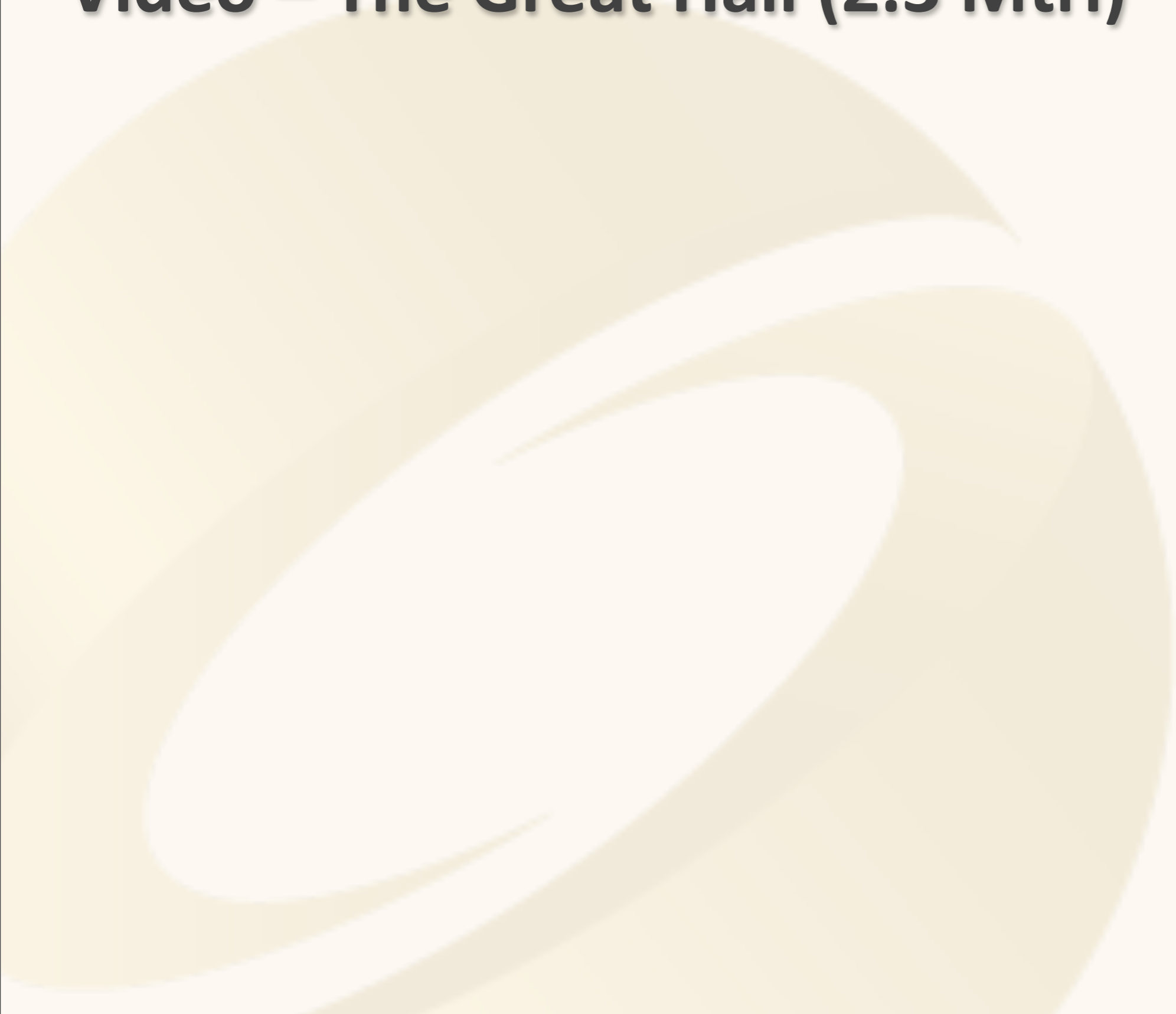


Note that only the hallway floor gets direct illumination; the ceiling receives only indirect lighting.



Note that only the hallway floor gets direct illumination; the ceiling receives only indirect lighting.

Video – The Great Hall (2.3 Mtri)





Direct and
indirect

Here we see the Great Hall model with a light moving in it. Note how the geometry contains both smooth parts and lots of detail, such as the cobblestones which are all modeled as individual objects.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

Rendering on GPU

Discussion and Conclusions

OK, now that we saw how to construct the basis, let's look at rendering.

Rendering on GPU

- **Render directly from meshless basis**
- **Resembles deferred splatting**
[Guennebaud 04, Gautron 05]
 - basis functions rendered as screen-aligned quads
- **Hierarchy causes overdraw**
 - flatten hierarchical solution to a nonhierarchical basis prior to display
 - could use any basis, e.g. vertices

We utilize deferred shading for rendering directly from the meshless basis. Our approach closely resembles previous deferred splatting techniques. Please see the paper for details.

When rendering from the hierarchical representation, each level of the hierarchy must be drawn separately. This increases overdraw and thus slows rendering down. We circumvent this by resampling the solution into a non-hierarchical meshless basis built from the same points before display. We however could use any basis, such as sampling at the vertices.

The paper describes further optimizations such as occlusion queries.

Overview and Previous Work

Reconstruction and Basis Functions

Constructing the Basis

Application: Direct-to-Indirect PRT

Rendering on GPU

Discussion and Conclusions

Limitations

- **Discontinuities like direct pointlight shadows can be problematic**
 - Use basis only for indirect illumination
- **Basis functions not necessarily restricted to a single surface patch**
 - Good!
 - But if leaking happens, finer levels must counter

As with all finite function bases, representing discontinuities such as shadows from pointlights can be difficult. This can result in some visible ringing. However, when representing only the smoother indirect illumination in the basis, such artifacts are avoided.

The basis functions are not limited to one surface primitive. This is a good thing, since patch or triangle boundaries do not cause discontinuities. However, sometimes this means that the basis function can leak for instance through a wall to an adjacent room. In such cases finer levels of the hierarchy must correct the leak.

Portability

- Rely only on ray tracing
- Meshless light transport algorithms are geometry-independent
- Example: Radiosity on meshes, quadric implicits and volumetric isosurface (see tech rep)



Let me further emphasize that none of the algorithms described above require anything of the geometry but point evaluations and ray tracing. This means that any light transport algorithm formulated in terms of our basis is entirely decoupled from the surface type. As an example, I'm showing a simple meshless hierarchical radiosity solution on a scene that contains a mesh, quadric implicit surfaces, and a volumetric isosurface of a bonsai tree.

For further applications of the basis, please see our tech report.

Summary

- **Meshless hierarchical basis**
 - Point samples, decoupled from geometry
 - No meshing, no parameterization
 - Enables hierarchical lighting algorithms on arbitrary surfaces
 - Many applications (see our tech rep)
- **Novel direct-to-indirect PRT algorithm**
 - Complex geometry, fast precomputation

In summary, we have described a meshless hierarchical function basis for light transport computations. It enables hierarchical coarse-to-fine illumination algorithms on arbitrary geometries, completely avoiding meshing and parameterization.

We applied our basis to a direct-to-indirect light transport algorithm, where we are able to demonstrate moving viewpoints, complex geometry and fast precomputation, a combination that has not been previously possible.

Acknowledgments

- **Colleagues for comments**

- **Scenes**

- DreamWorks Animation, Stephen Duck, Marko Dabrovic, Eric Tabellion, Andrew Pearce

- **Funding**

- National Technology Agency of Finland, Anima Vitae, AMD Research Finland, NVIDIA, Remedy Entertainment, Academy of Finland #108222, INRIA Associate Research Team ARTIS/MIT, and NSF CAREER 044756



The End — Thank You