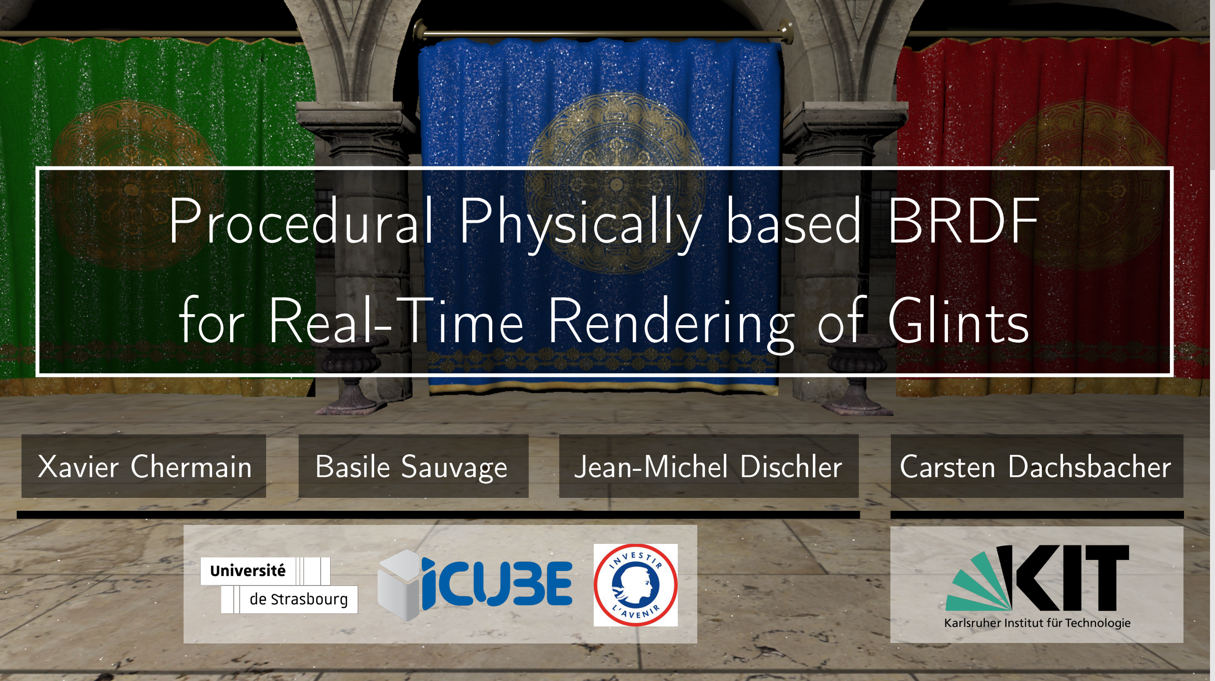
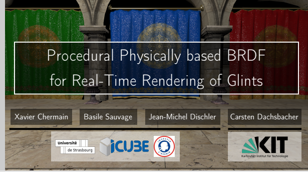


Procedural Physically based BRDF for Real-Time Rendering of Glints

Xavier Chermain Basile Sauvage Jean-Michel Dischler Carsten Dachsbacher

Université de Strasbourg
ICUBE
INVESTIR L'AVENIR
KIT Karlsruher Institut für Technologie

2021-12-04

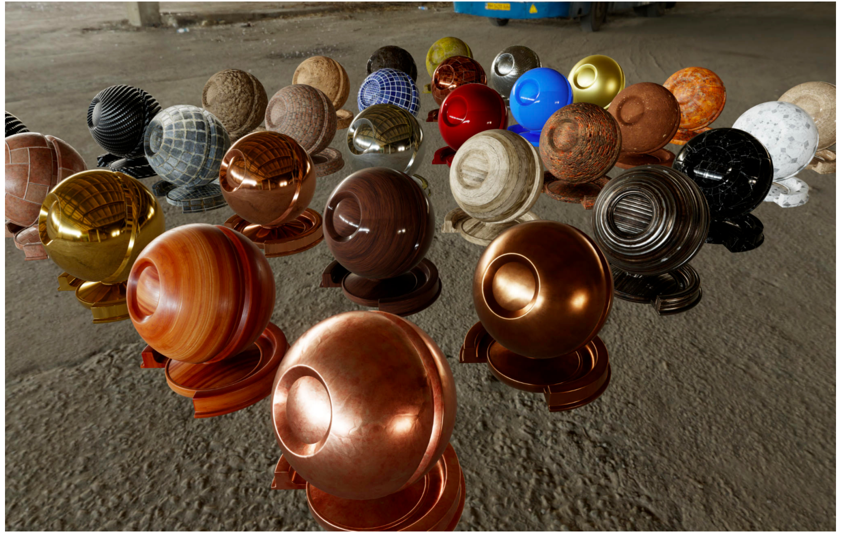


Procedural Physically based BRDF for Real-Time Rendering of Glints

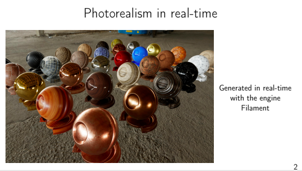
Xavier Chermain Basile Sauvage Jean-Michel Dischler Carsten Dachsbacher

Hello everyone, my name is Xavier Chermain. I am a postdoctoral researcher at the University of Strasbourg in France, and I will present our paper Procedural Physically based BRDF for Real-Time Rendering of Glints. This work is the result of a collaboration with the Karlsruhe Institute of Technology.

Photorealism in real-time



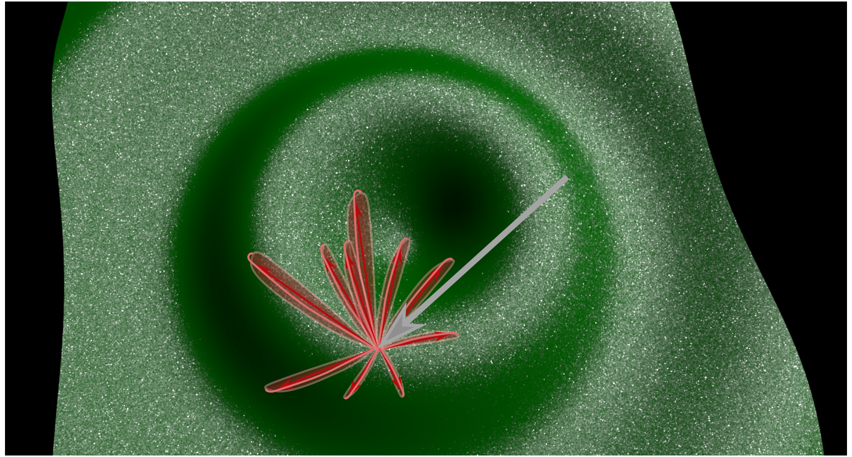
Generated in real-time with the engine Filament



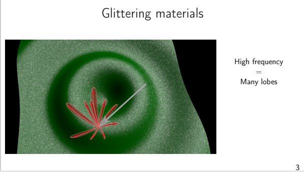
Photorealism in real-time
Generated in real-time with the engine Filament
2

Photorealism is more and more present in real-time rendering engines. For example, here, we have several different realistic materials that have been rendered in real-time with the Filament physically based engine.

Glittering materials



High frequency
=
Many lobes



Glittering materials

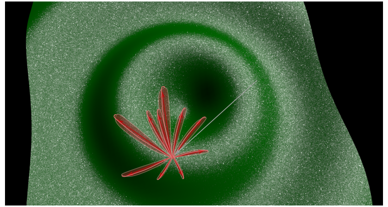
High frequency
=
Many lobes

3

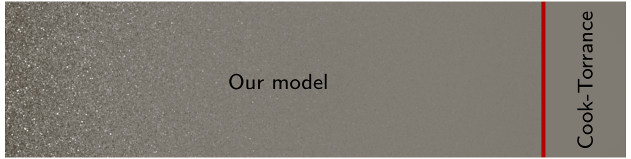
The race for photorealism pushes the modeling of complex materials, such as glittering materials. Glittering materials have reflectance distributions with high frequency. They have many lobes.

Contributions

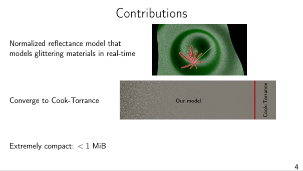
Normalized reflectance model that models glittering materials in real-time



Converge to Cook-Torrance

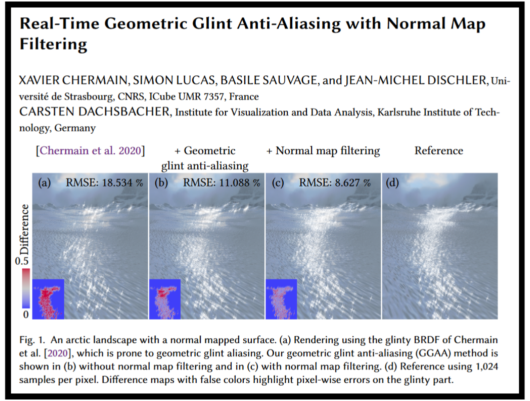


Extremely compact: < 1 MiB


Contributions
Normalized reflectance model that models glittering materials in real-time
Converge to Cook-Torrance
Our model
Cook-Torrance
Extremely compact: < 1 MiB
4

In our work, we propose an energy conserving reflectance model that models glittering materials in real-time. Our model converges to the standard Cook-Torrance model. Finally, the model is compact (less than 1 MiB of data).

Published extensions

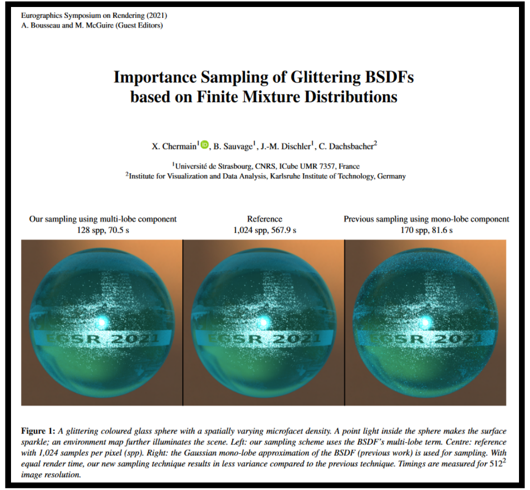


Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering

XAVIER CHERMAIN, SIMON LUCAS, BASILE SAUVAGE, and JEAN-MICHEL DISCHLER, Université de Strasbourg, CNRS, ICube UMR 7357, France
CARSTEN DACHSBACHER, Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology, Germany

[Chermain et al. 2020] | + Geometric glint anti-aliasing | + Normal map filtering | Reference

Fig. 1. An arctic landscape with a normal mapped surface. (a) Rendering using the glinty BRDF of Chermain et al. [2020], which is prone to geometric glint aliasing. Our geometric glint anti-aliasing (GGAA) method is shown in (b) without normal map filtering and in (c) with normal map filtering. (d) Reference using 1,024 samples per pixel. Difference maps with false colors highlight pixel-wise errors on the glinty part.



Eurographics Symposium on Rendering (2021)
A. Bousseau and M. McGuire (Guest Editors)

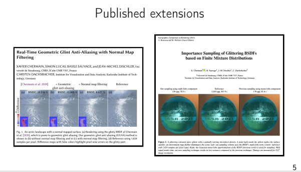
Importance Sampling of Glittering BSDFs based on Finite Mixture Distributions

X. Chermain[1], B. Sauvage[1], J.-M. Dischler[1], C. Dachsbacher[2]

[1] Université de Strasbourg, CNRS, ICube UMR 7357, France
[2] Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology, Germany

Our sampling using multi-lobe component 128 spp, 70.5 s | Reference 1,024 spp, 567.9 s | Previous sampling using mono-lobe component 170 spp, 81.6 s

Figure 1: *A glittering coloured glass sphere with a spatially varying microfacet density. A point light inside the sphere makes the surface sparkle; an environment map further illuminates the scene. Left: our sampling scheme uses the BSDF's multi-lobe term. Centre: reference with 1,024 samples per pixel (spp). Right: the Gaussian mono-lobe approximation of the BSDF (previous work) is used for sampling. With equal render time, our new sampling technique results in less variance compared to the previous technique. Timings are measured for* 512^2 *image resolution.*

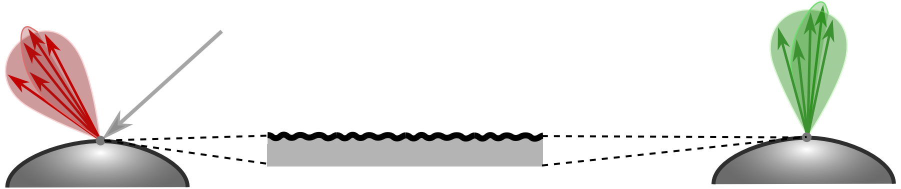


Published extensions
5

Note that we have improved our reflectance model in two subsequently published works. We have first tackled the issue of glint aliasing in real-time and normal map filtering with glints. Then, we have proposed an importance sampling scheme for our glittering reflectance model allowing efficient glint rendering in Monte Carlo renderers. However, in this presentation, I will focus on our original method accepted to Pacific Graphics 2020.

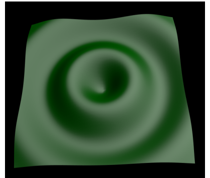
Related work: Single-lobe NDF

Reflectance model ⇐ microgeometry ⇒ Normal Distribution Function (NDF)

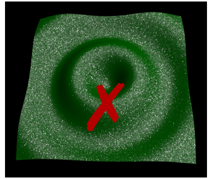


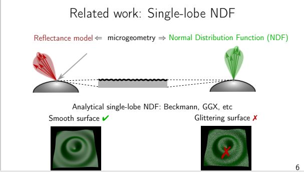
Analytical single-lobe NDF: Beckmann, GGX, etc

Smooth surface ✔



Glittering surface ✗





Related work: Single-lobe NDF

Reflectance model ⇐ microgeometry ⇒ Normal Distribution Function (NDF)

Analytical single-lobe NDF: Beckmann, GGX, etc

Smooth surface ✔

Glittering surface ✗

6

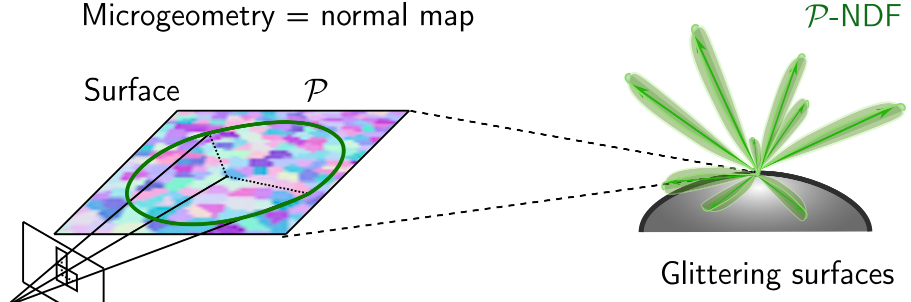
Reflectance models simulate the scattering of light in contact with microgeometry. It is widespread in rendering to model this microgeometry with a normal distribution function, abbreviated NDF. This model is much more compact than explicitly storing the microgeometry of the surface.

Two NDFs are commonly used: the Beckmann NDF, based on a Gaussian, and the GGX distribution, which is the NDF of an ellipsoid.

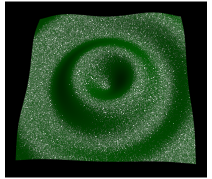
These NDFs are single-lobed and therefore do not allow to simulate glint.

Related work: normal map

Microgeometry = normal map



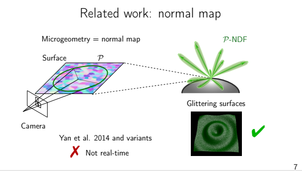
Glittering surfaces



✔

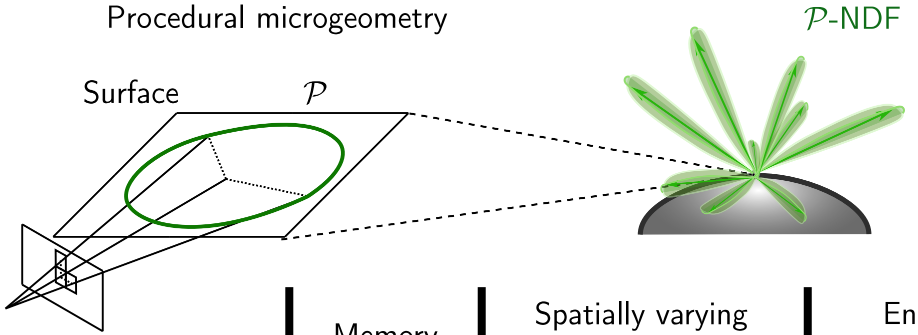
Yan et al. 2014 and variants

✗ Not real-time

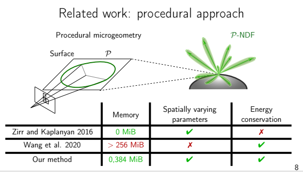

Related work: normal map
Microgeometry = normal map
$\mathcal{P}$-NDF
Surface
$\mathcal{P}$
Glittering surfaces
Camera
Yan et al. 2014 and variants
✗ Not real-time
✔
7

Glittering materials have NDFs that have hundreds of lobes, and these NDFs must spatially vary. An excellent way to get both properties is to use a normal map that represents the microgeometry. In this case, we need to compute the NDF in the pixel footprint P, which we call the P-NDF. This P-NDF is multi-level, high frequency, and spatially varying. Yan and colleagues propose an efficient method to compute the P-NDF, but their method is too expensive for real-time.

Related work: procedural approach



	Memory	Spatially varying parameters	Energy conservation
Zirr and Kaplanyan 2016	0 MiB	✔	✗
Wang et al. 2020	> 256 MiB	✗	✔
Our method	0,384 MiB	✔	✔


Related work: procedural approach

Procedural microgeometry $\mathcal{P}$-NDF Surface $\mathcal{P}$

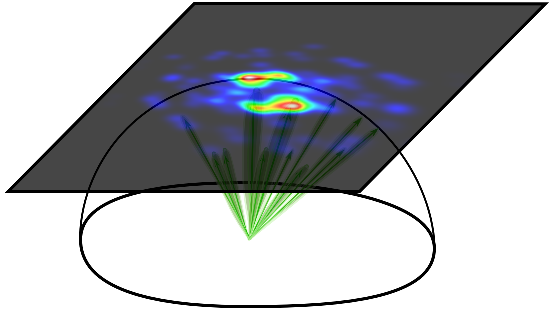
	Memory	Spatially varying parameters	Energy conservation
Zirr and Kaplanyan 2016	0 MiB	✔	✗
Wang et al. 2020	> 256 MiB	✗	✔
Our method	0,384 MiB	✔	✔

8

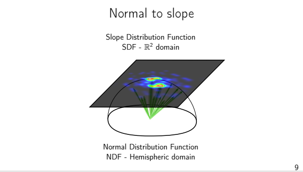
Zirr and Kaplanyan simulate glittering appearances in real-time with a procedural P-NDF. Their algorithm does not consume memory, and the model parameters can vary along the surface. However, their model does not conserve energy. Wang and colleagues propose an energy-conserving model, but the memory consumption exceeds 250 MiB, and the parameters cannot spatially vary. We also use a procedural approach, but our memory consumption is lightweight, our model parameters spatially vary, and our model conserves energy.

Normal to slope

Slope Distribution Function
SDF - $\mathbb{R}^2$ domain



Normal Distribution Function
NDF - Hemispheric domain



Normal to slope

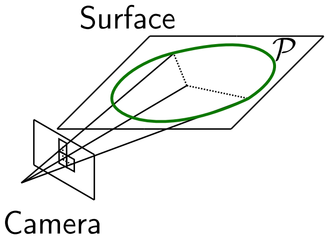
Slope Distribution Function
SDF - $\mathbb{R}^2$ domain

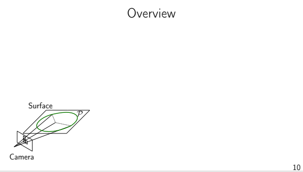
Normal Distribution Function
NDF - Hemispheric domain

9

Since the beginning, I have been talking about normal distributions, abbreviated NDF. Their domain is 3D; it is the domain of the hemisphere. To drop a dimension, we can work in the slope space, which is 2D. In this case, we have a Slope Distribution Function, abbreviated SDF. In the following, we will see how to generate procedural SDFs.

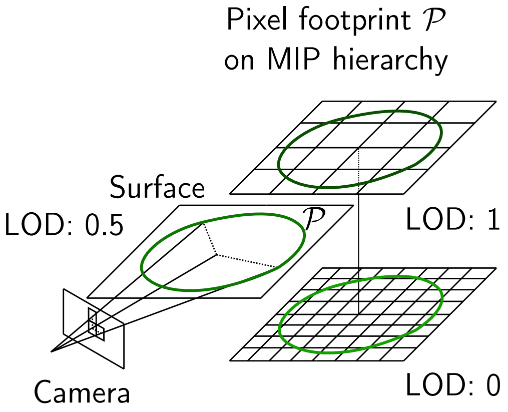
Overview

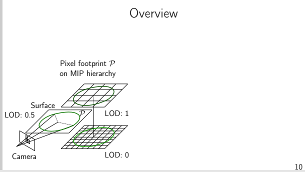




Now I'm going to give you an overview of our method. We have a virtual camera, and we want to calculate the amount of light energy that arrives on the middle pixel. This energy depends on the microgeometry, and therefore the SDFs contained within P.

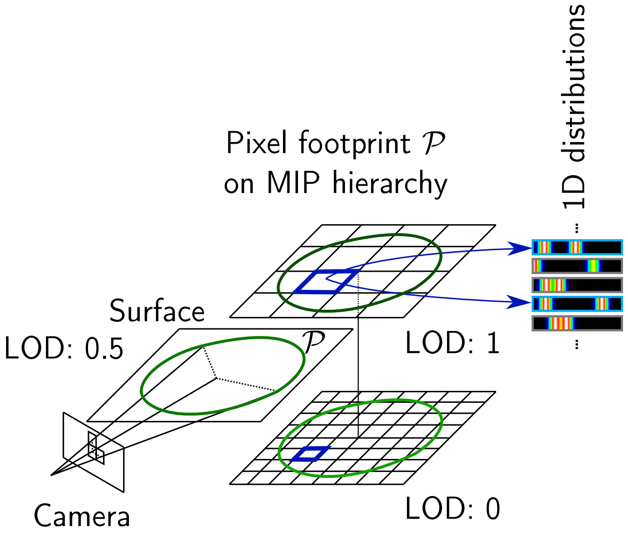
Overview

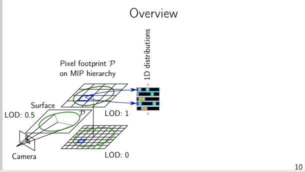




To bound the computation times, the surface is subdivided with cells, and cells are in a MIP hierarchy. The size of P allows us to access a continuous level of detail, abbreviated LOD. Here, the LOD is 0.5. The two adjacent discrete LODs always have a constant number of cells within the pixel footprint P.

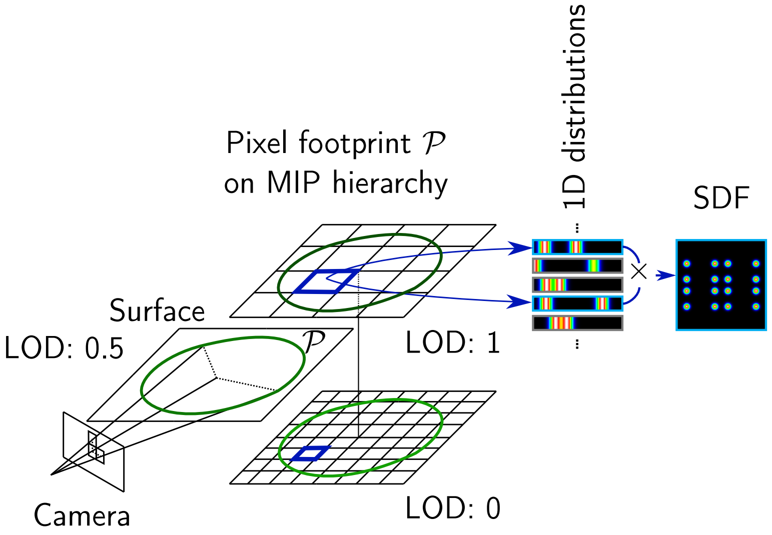
Overview

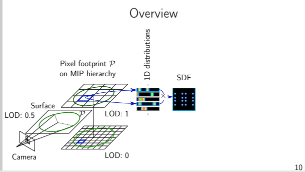



Overview
Pixel footprint $\mathcal{P}$ on MIP hierarchy
1D distributions
Surface
LOD: 0.5
LOD: 1
Camera
LOD: 0
10

We procedurally construct a SDF for each cell by randomly selecting two 1D distributions contained in a dictionary.

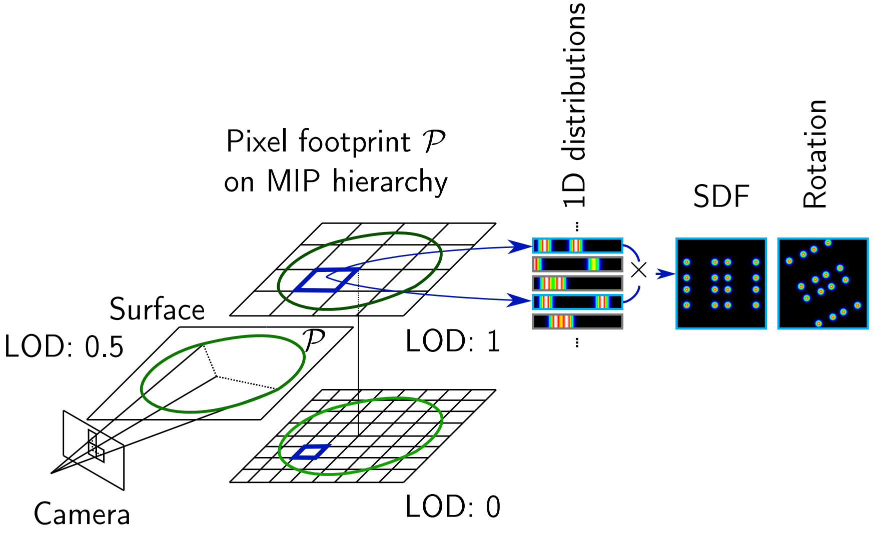
Overview



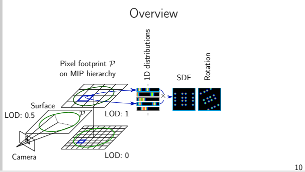


The tensor product of these two 1D distributions gives a 2D distribution with several lobes.

Overview

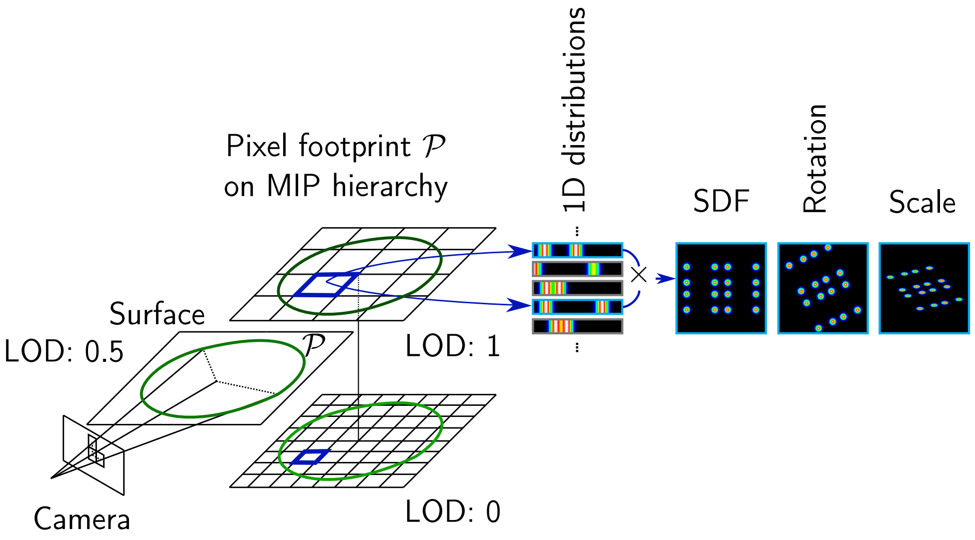


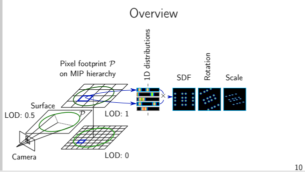
└Overview



To avoid lobe alignments, we apply a random rotation.

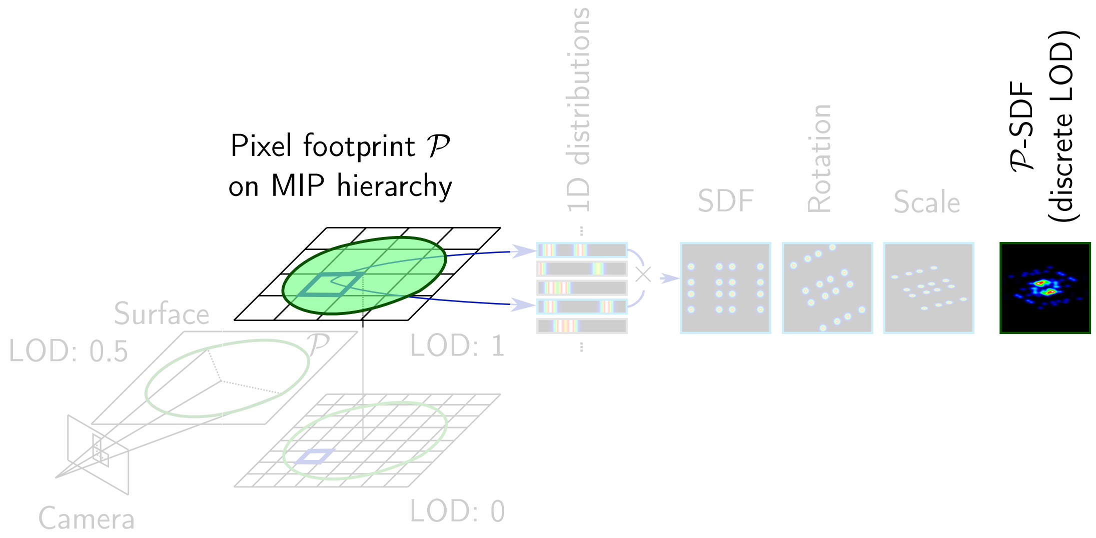
Overview



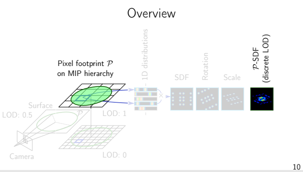


To control the roughness of the material, we apply a scale.

Overview

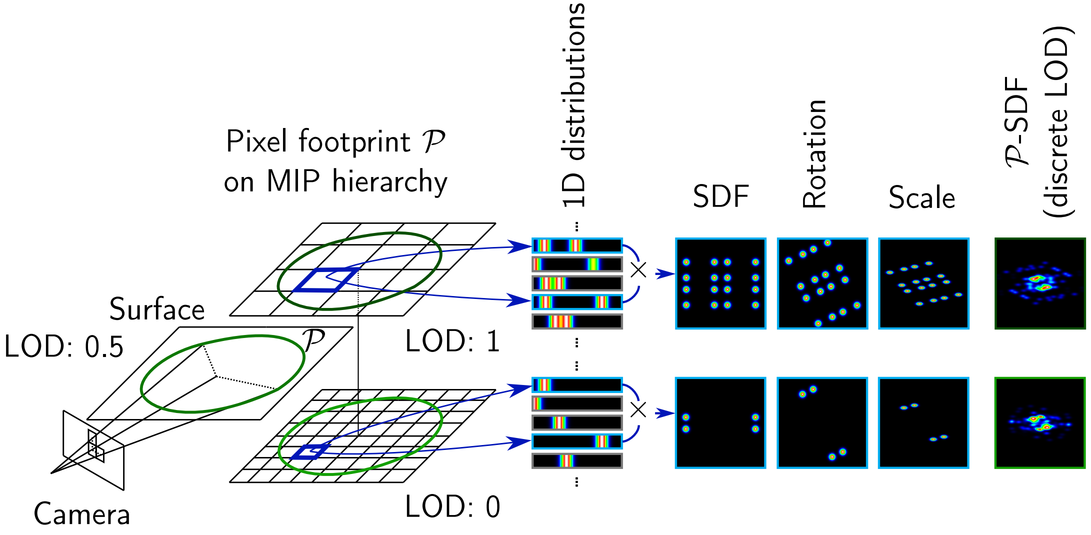


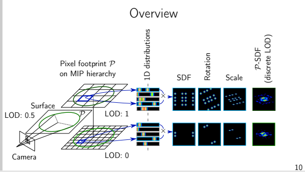
└Overview



The average of the cell distributions gives a P-SDF at a discrete LOD.

Overview

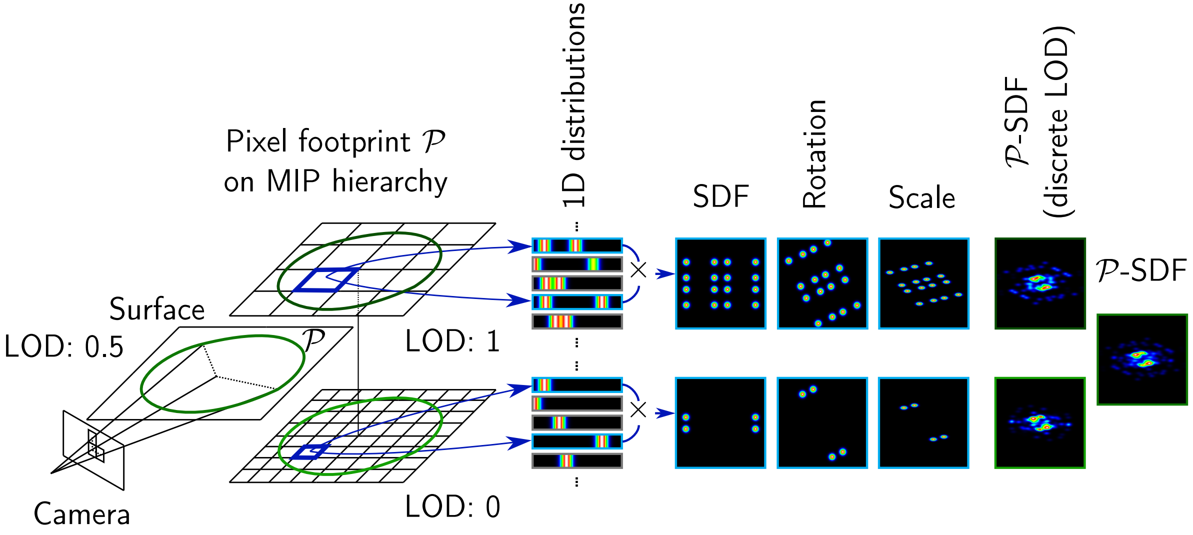


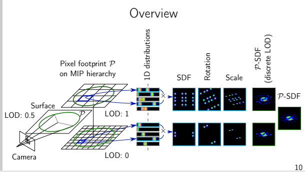


Overview
Pixel footprint $\mathcal{P}$ on MIP hierarchy
1D distributions
SDF
Rotation
Scale
$\mathcal{P}$-SDF (discrete LOD)
Surface
LOD: 0.5
LOD: 1
LOD: 0
Camera
10

We do the same for the lower discrete LOD.

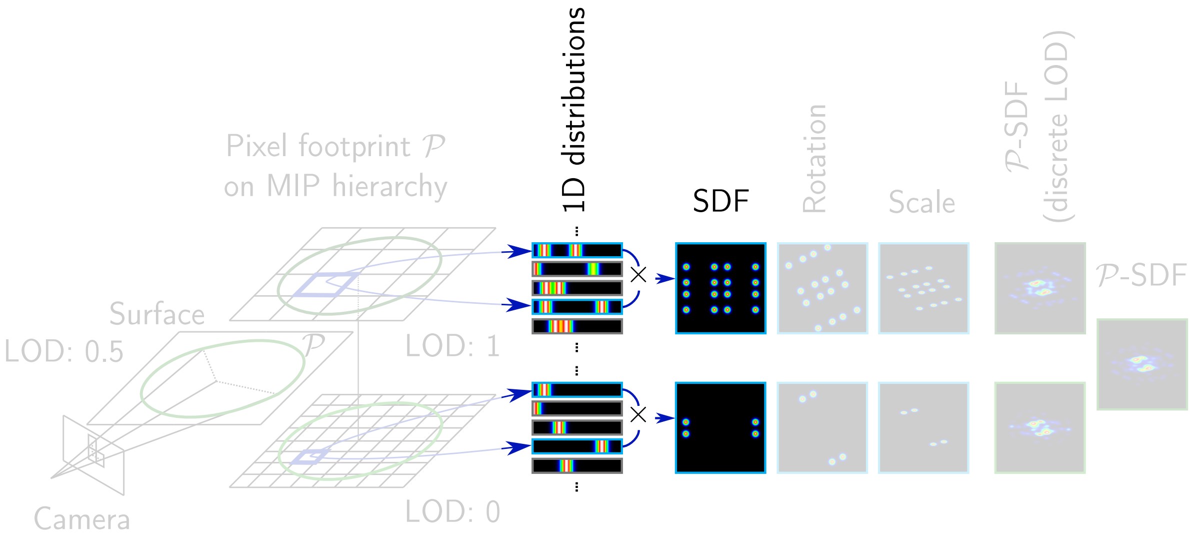
Overview





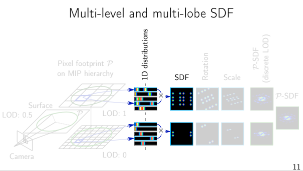
To have smooth transitions when zooming, we linearly interpolate the P-SDFs at the adjacent discrete LODs. The resulting P-SDF is used to calculate the reflectance of the material.

Multi-level and multi-lobe SDF



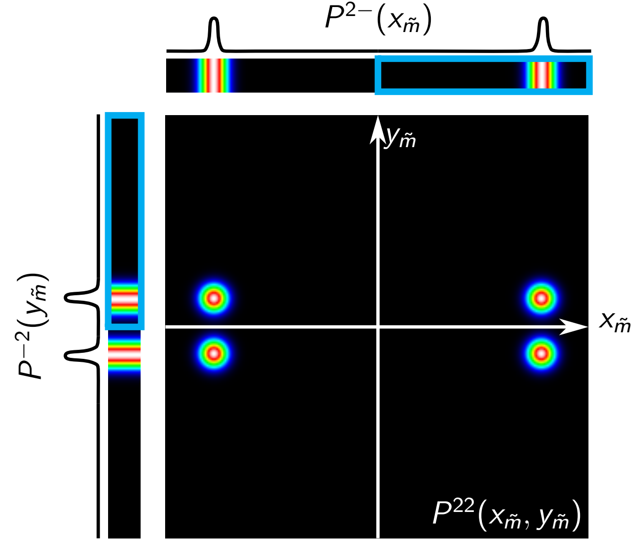
2021-12-04

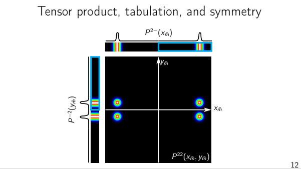
└Multi-level and multi-lobe SDF



I will now talk about the procedural generation of high-frequency multi-level SDFs from a set of 1D distributions.

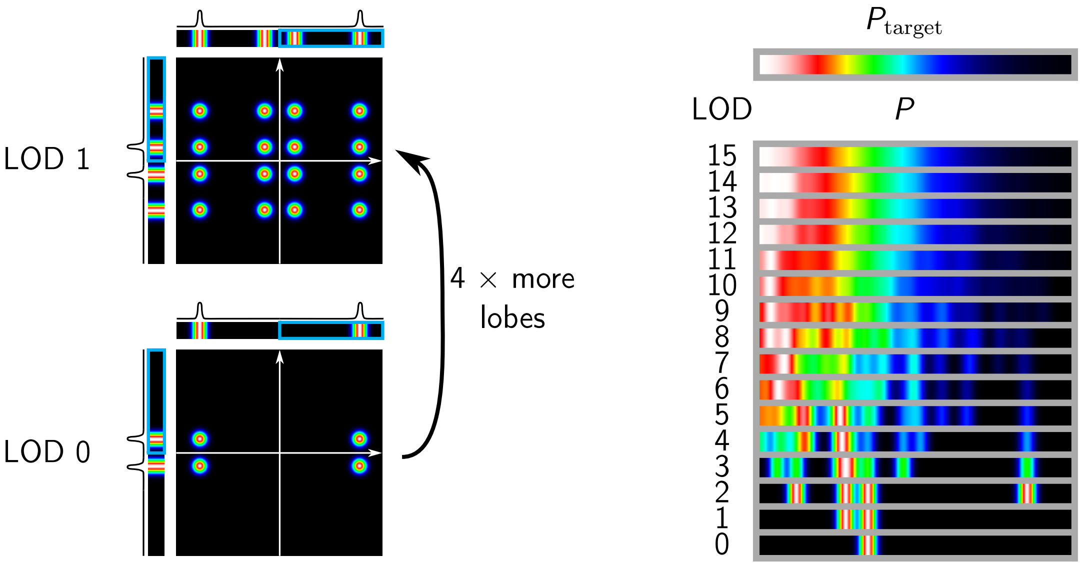
Tensor product, tabulation, and symmetry

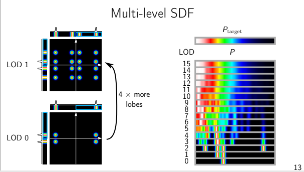



Tensor product, tabulation, and symmetry
12

In our method, we define our 2D SDF as the tensor product of two 1D SDFs. These 1D distributions will be tabulated and stored in memory. In order to have symmetric distributions and divide the storage by two, the 1D distributions are even functions.

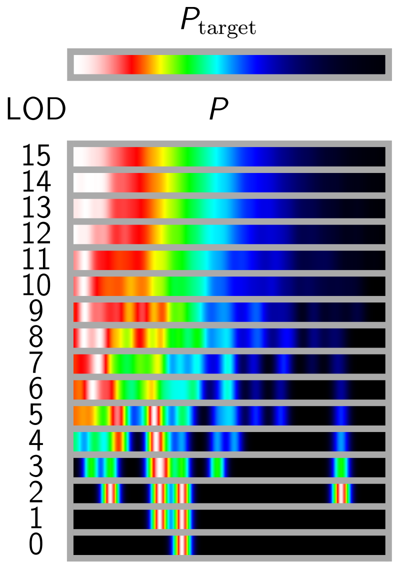
Multi-level SDF

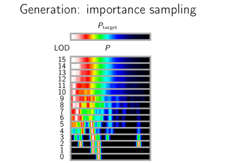




Since we are using a MIP hierarchy, the SDFs are multi-level. The minimum number of lobe at the lowest level is one in 1D, and therefore four in 2D. At the highest level, the distribution is a Gaussian, because the Gaussian is separable. The number of lobes is multiplied by two in 1D to go from level l to level l+1. In 2D, the number of lobes is therefore multiplied by four.

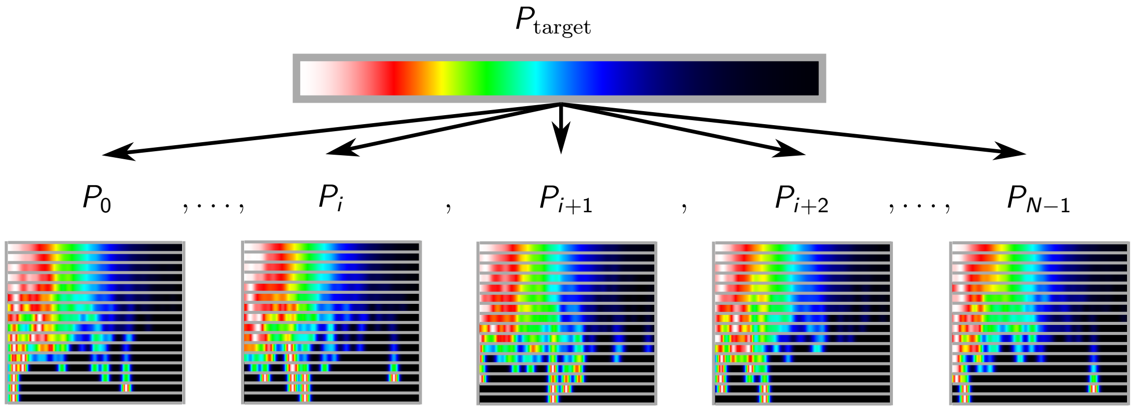
Generation: importance sampling





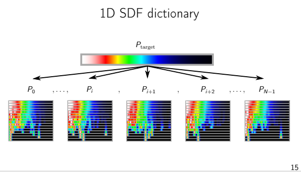
To generate multi-level SDFs that converge to a target distribution, we importance sample the target distribution to place tiny lobes. The aggregation of the lobes allows converging to the target distribution.

1D SDF dictionary



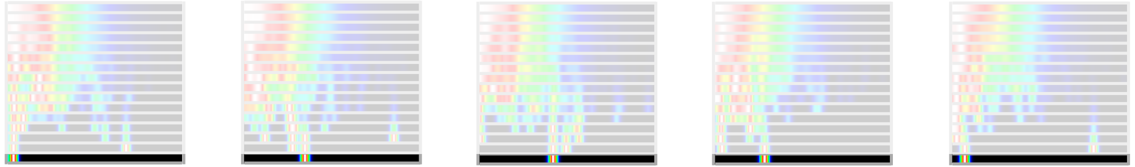
P_{target}

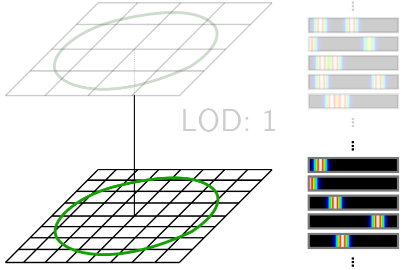
$P_0 \quad , \dots , \quad P_i \quad , \quad P_{i+1} \quad , \quad P_{i+2} \quad , \dots , \quad P_{N-1}$

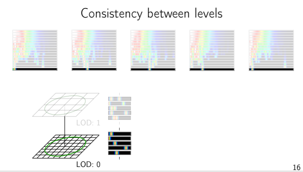


We are running N times the generation algorithm to have N different 1D multi-level distributions. This set is called the dictionary.

Consistency between levels

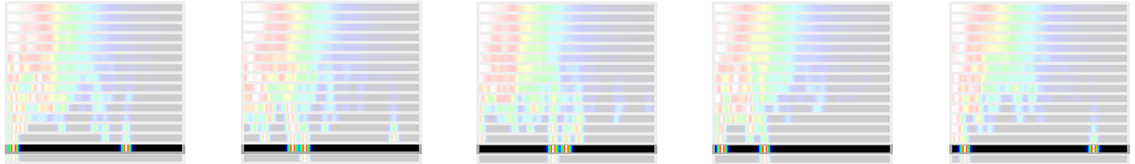


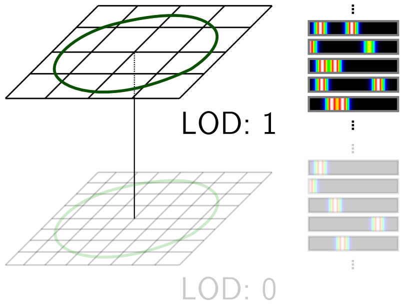


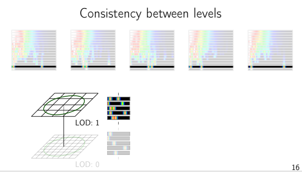


So, the dictionary is multi-level. The distributions of level 0 are used by the cells of LOD 0.

Consistency between levels

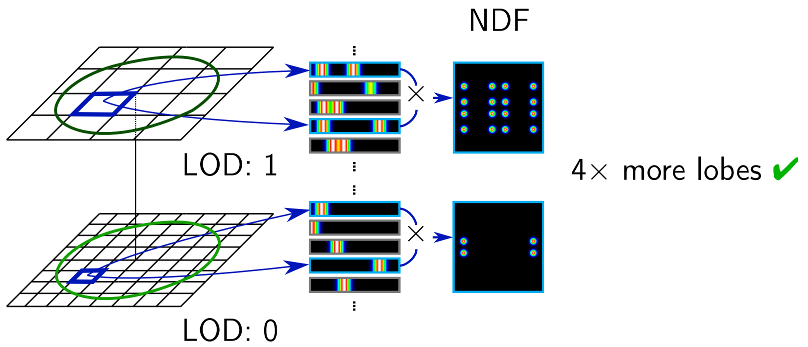


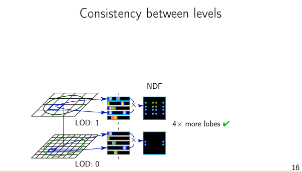




The distributions of level 1 are used by the cells of LOD 1.

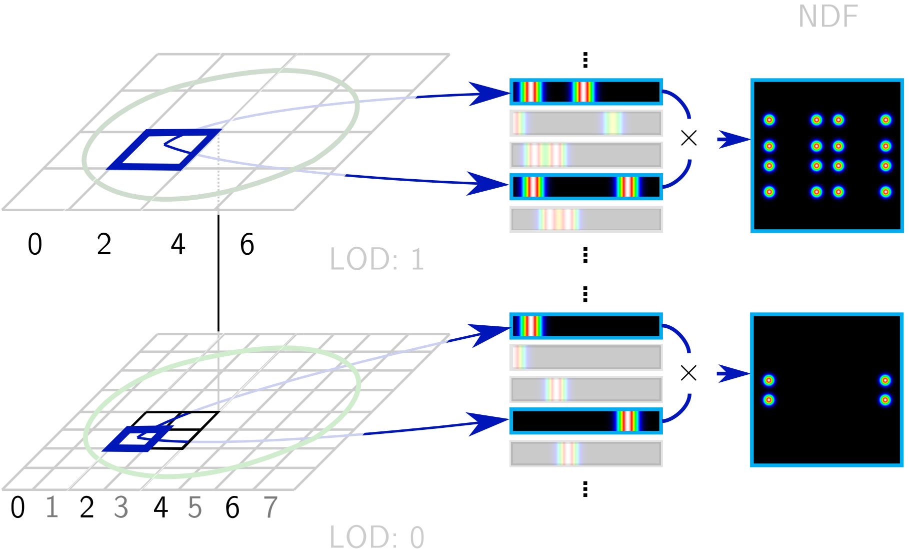
Consistency between levels

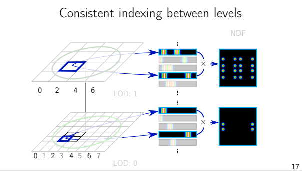




Note that with our dictionary, the number of lobes is multiplied by four when going from LOD l to LOD l+1. This result is expected as we use a MIP hierarchy.

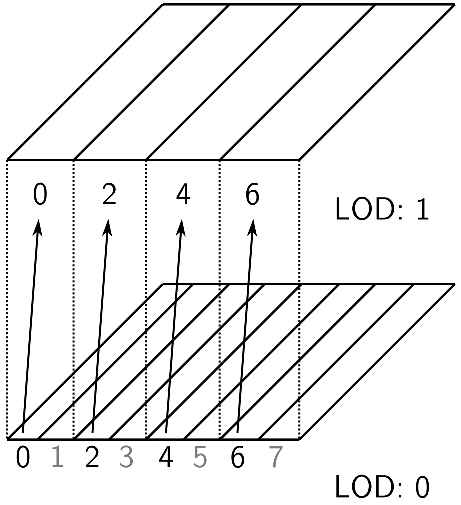
Consistent indexing between levels

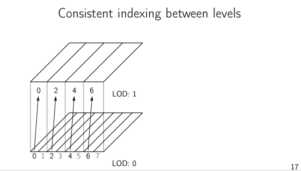



Consistent indexing between levels

We also expect that the lobes of a level l are also present at a level $l + 1$ to ensure consistency between LOD. To partially accomplish this, the indexing of the cells is consistent between levels.

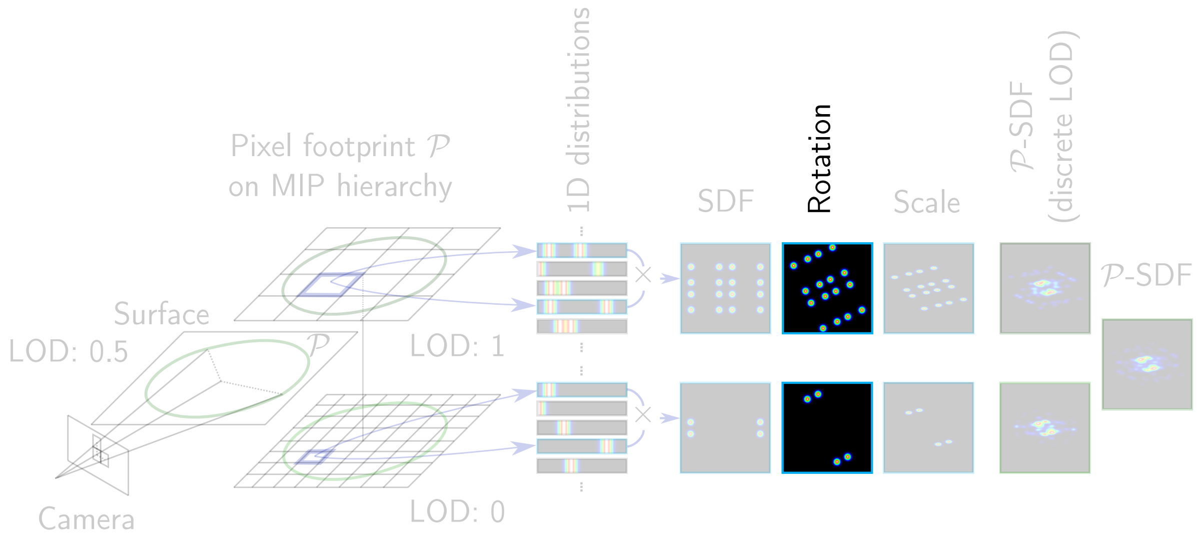
Consistent indexing between levels

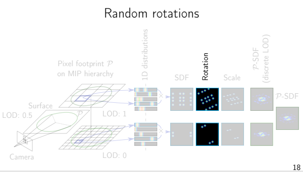




Cell indices are used to randomly select distributions, so it is crucial to raise some indices of LOD l to LOD l+1, as in this case. So, at LOD 1, cell indices are increased by two. At LOD 2, cell indices are increased by four, and so on.

Random rotations

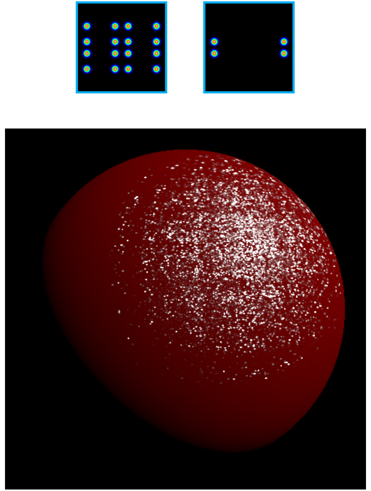



Random rotations
18

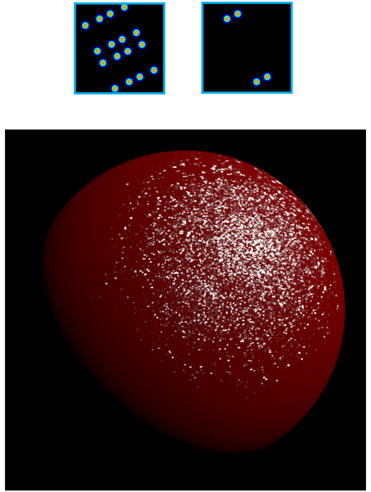
Now I'll show you what happens if we don't apply random rotations to the SDFs.

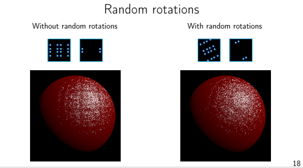
Random rotations

Without random rotations



With random rotations





Random rotations

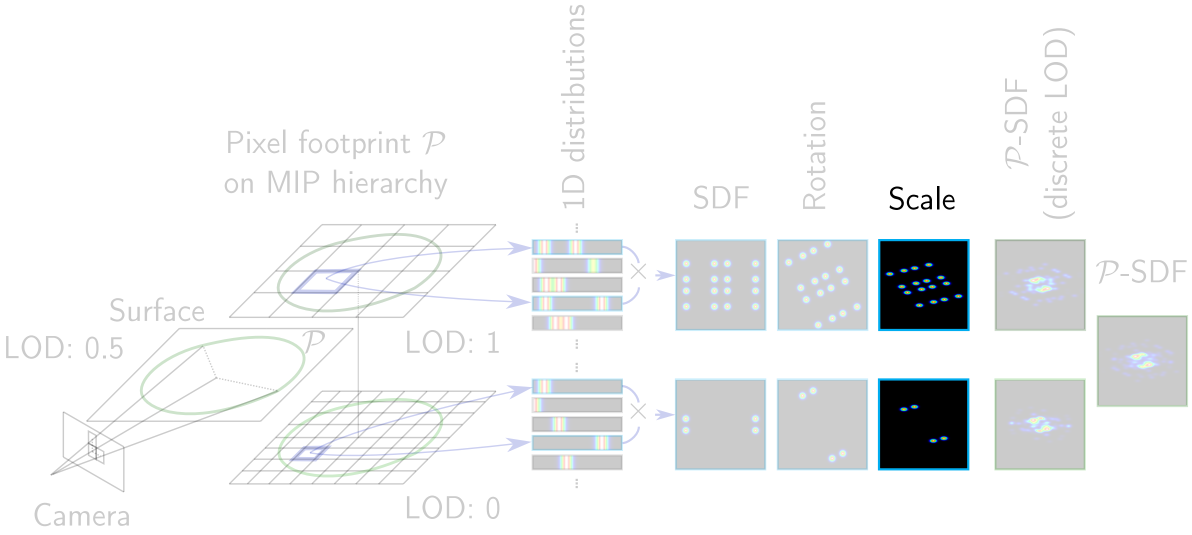
Without random rotations

With random rotations

18

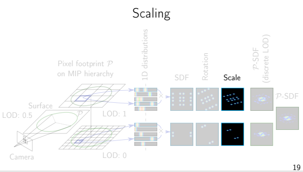
We have glint alignments with few distributions in the dictionary, as the lobes are all aligned in the x and y directions. Random rotations per cell remove these alignments, which improves the rendering quality.

Scaling



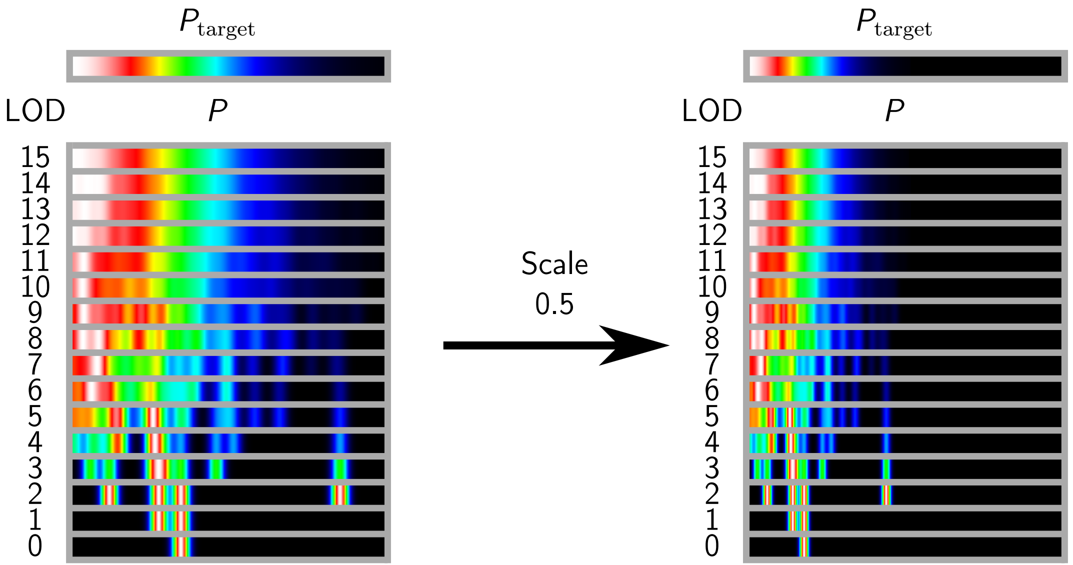
2021-12-04

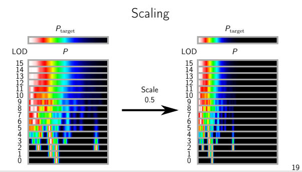
└Spatially varying multi-level SDF



Now I’ll talk about scaling.

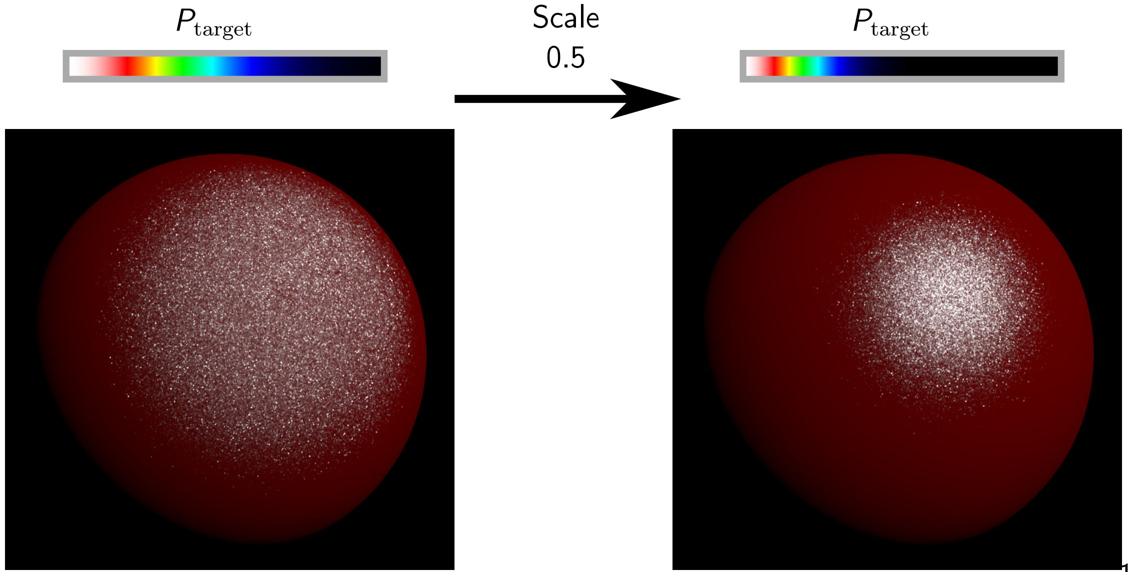
Scaling

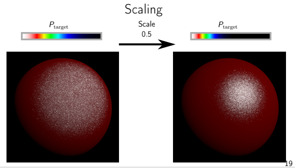




Scaling allows controlling the variance of the target distribution, which allows controlling the roughness of the material without having to regenerate and store another dictionary.

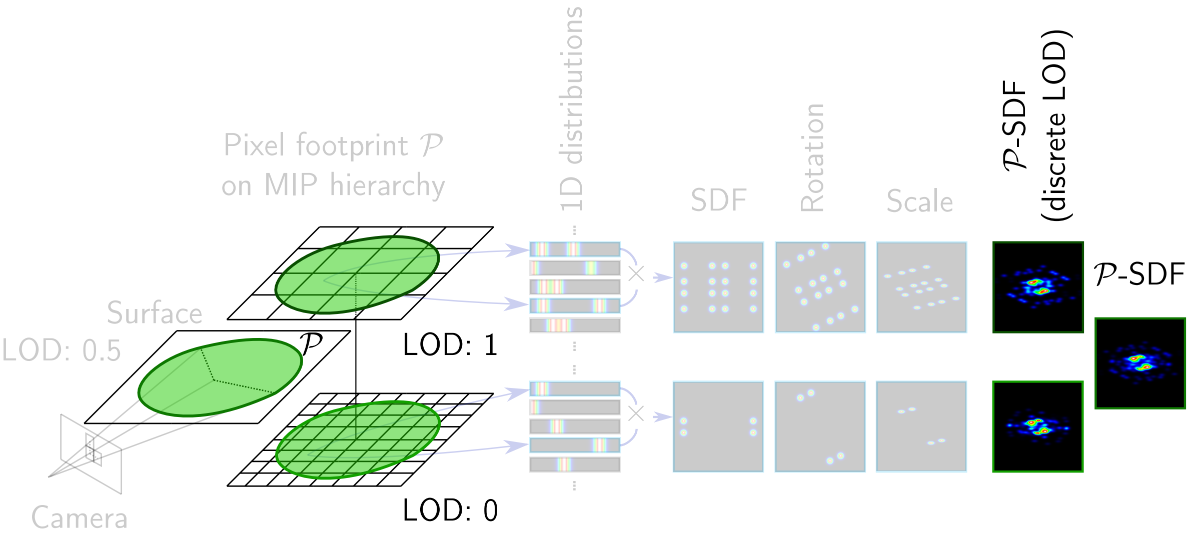
Scaling

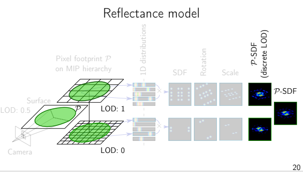




Controlling the roughness allows you to control the area covered by glint. Here, a scaling of 0.5 allows reducing by two the area covered by glint.

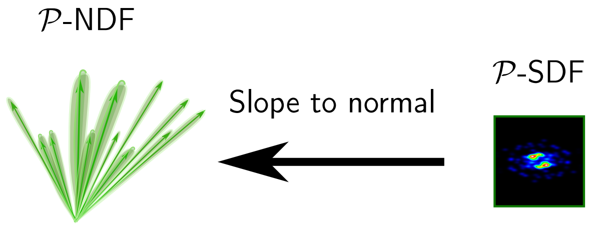
Reflectance model

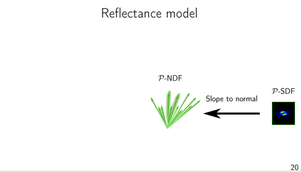




After averaging cell dependant SDFs, and interpolating the two P-SDFs at adjacent discrete LODs, we end up with a P-SDF.

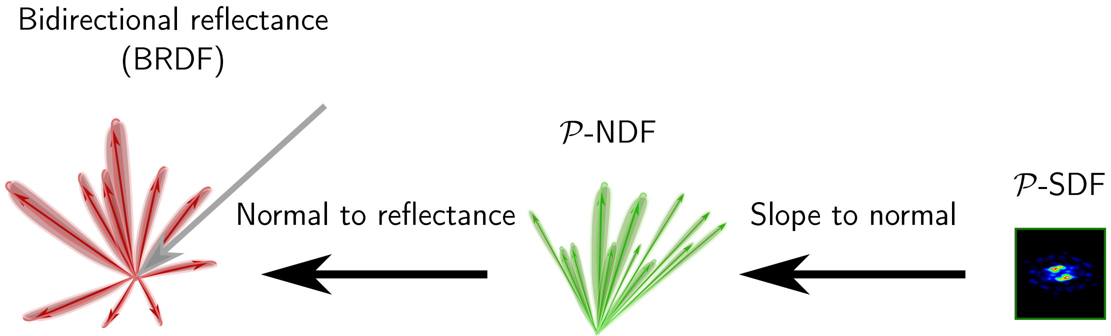
Reflectance model





The P-SDF is converted to P-NDF using the slope-to-normal transformation.

Reflectance model



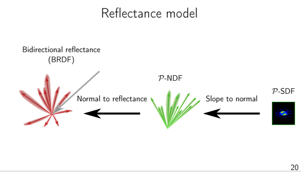
Bidirectional reflectance (BRDF)

Normal to reflectance

$\mathcal{P}$-NDF

Slope to normal

$\mathcal{P}$-SDF



Reflectance model

Bidirectional reflectance (BRDF)

Normal to reflectance

$\mathcal{P}$-NDF

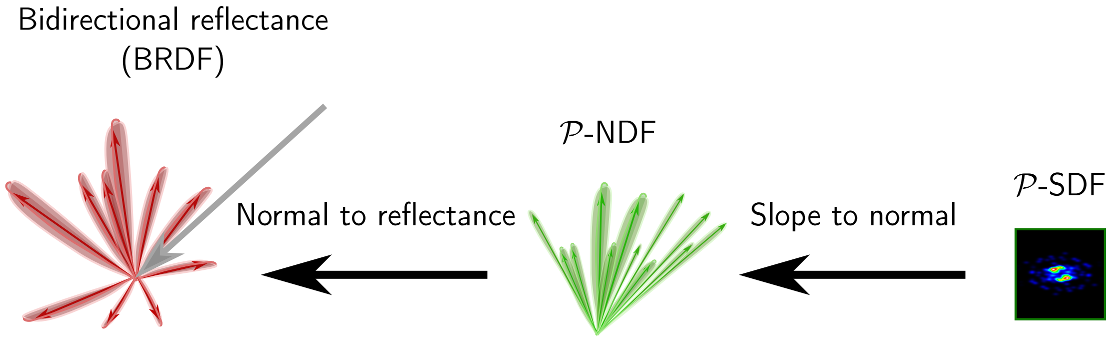
Slope to normal

$\mathcal{P}$-SDF

20

Then the distribution of normals is converted to bidirectional reflectance, that is, to BRDF.

Reflectance model



Bidirectional reflectance (BRDF)

$\mathcal{P}$-NDF

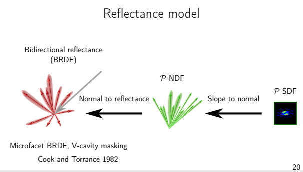
$\mathcal{P}$-SDF

Normal to reflectance

Slope to normal

Microfacet BRDF, V-cavity masking

Cook and Torrance 1982



Reflectance model

Bidirectional reflectance (BRDF)

$\mathcal{P}$-NDF

Normal to reflectance

Slope to normal

$\mathcal{P}$-SDF

Microfacet BRDF, V-cavity masking

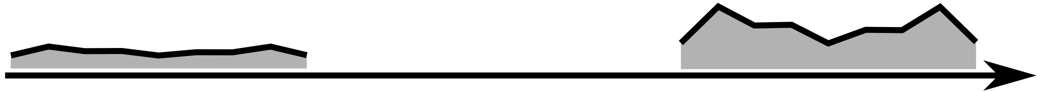
Cook and Torrance 1982

20

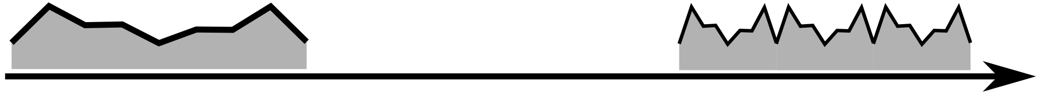
We use a microfacet BRDF with V-cavity masking. So, we fit into the Cook and Torrance model.

User parameters

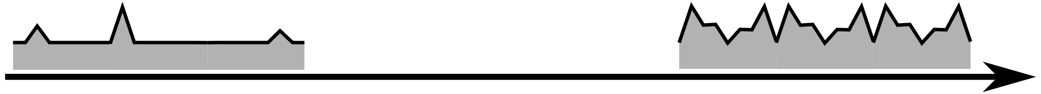
Roughness

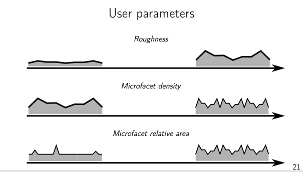


Microfacet density



Microfacet relative area





I will now discuss the user parameters of our BRDF. As with the Cook and Torrance model, we can control the roughness of the material. The rougher the material, the steeper the slopes that make up its micro-geometry. (Window switch). Here, I lower the roughness, and therefore the area where the glints are is smaller. (Window switch) Unlike the Cook Torrance model, we can also control the density of microfacets per square meter. The higher the density, the more microfacets per square meter. (Window switch) We converge to a Cook and Torrance specular lobe if I increase this parameter. If I lower it, we start to see individual microfacets. (Window switch) Finally, we also propose to control the relative area of the microfacets. This parameter is a mask, and if its value is low, the glints are scattered. (Window switch). So, if I lower the microfacet relative area, we get scattered glints.

Results: Sponza

Video

└Results: Sponza

Results: Sponza

Video

Now I'll show you some animations. Here we have the Sponza scene, where the stones have glittering minerals, visible only if we are close to the material. We also have glitter curtains with a much higher relative microfacet area than the stone.

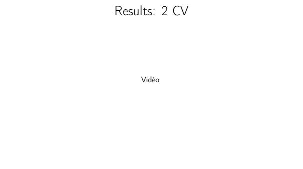
Results: 2 CV

Vidéo

2021-12-04

└Renderings

└Results: 2 CV



Results: 2 CV

Vidéo

Here we have a deux chevaux. We use our glittering BRDF to model metallic paint. Here, the glitters are randomly colored to simulate iridescence cheaply.

Results: Varnished parquet

Video

└Results: Varnished parquet

Results: Varnished parquet

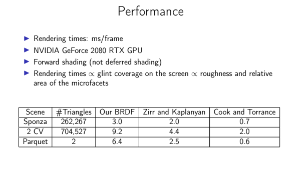
Video

Since our algorithm is procedural and a dictionary can be used for several roughnesses, our method is compatible with roughness maps. A roughness map allows modulating the shape of the specular lobes on a surface, as here with parquet having a spatially varying roughness.

Performance

- Rendering times: ms/frame
- NVIDIA GeForce 2080 RTX GPU
- Forward shading (not deferred shading)
- Rendering times $\propto$ glint coverage on the screen $\propto$ roughness and relative area of the microfacets

Scene	#Triangles	Our BRDF	Zirr and Kaplanyan	Cook and Torrance
Sponza	262,267	3.0	2.0	0.7
2 CV	704,527	9.2	4.4	2.0
Parquet	2	6.4	2.5	0.6



Performance

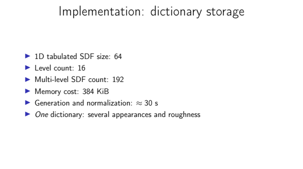
- Rendering times: ms/frame
- NVIDIA GeForce 2080 RTX GPU
- Forward shading (not deferred shading)
- Rendering times $\propto$ glint coverage on the screen $\propto$ roughness and relative area of the microfacets

Scene	#Triangles	Our BRDF	Zirr and Kaplanyan	Cook and Torrance
Sponza	262,267	3.0	2.0	0.7
2 CV	704,527	9.2	4.4	2.0
Parquet	2	6.4	2.5	0.6

Now I will talk about performance. The rendering time is proportional to the glint coverage on the screen, which is proportional to the roughness and the relative area of the microfacets. The Sponza scene, takes on average 3 ms to render an image with our method. It takes 2 ms to render an image with the non-physical-based glittering BRDF of Zirr and Kaplanyan. It takes 0.7 ms with the non-glittering BRDF of Cook and Torrance. If we look at the performance for the other scenes, we have a rise of computation times between 50 % and 156 % compared to Zirr and Kaplanyan.

Implementation: dictionary storage

- 1D tabulated SDF size: 64
- Level count: 16
- Multi-level SDF count: 192
- Memory cost: 384 KiB
- Generation and normalization: $\approx$ 30 s
- *One* dictionary: several appearances and roughness



Implementation: dictionary storage

- 1D tabulated SDF size: 64
- Level count: 16
- Multi-level SDF count: 192
- Memory cost: 384 KiB
- Generation and normalization: $\approx$ 30 s
- *One* dictionary: several appearances and roughness

Now I will address the subject of the dictionary storage. Each 1D distribution is stored in an array of size 64. There are 16 distribution levels and 192 multi-scale distributions in the dictionary. The memory cost is therefore 384 KiB. Generating and normalizing the distributions takes approximately 30 s. The generation is done only once because once a dictionary is generated, it allows for multiple appearances and roughnesses.

Implementations

- OpenGL:
  `https://github.com/ASTex-ICube/real_time_glint`
- WebGL:
  `http://igg.unistra.fr/People/reproctex/Demos/Real_Time_Glint/`
- Shadertoy:
  `https://www.shadertoy.com/view/wstcRH`
- pbrt-v3:
  `https://github.com/ASTex-ICube/importance_sampling_glint`
- Dictionary generator:
  `https://github.com/ASTex-ICube/real_time_glint_dictgenerator`

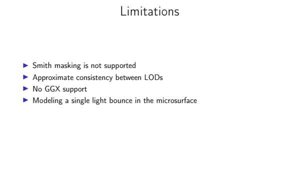
Implementations

- OpenGL:
  https://github.com/ASTex-ICube/real_time_glint
- WebGL:
  http://igg.unistra.fr/People/reproctex/Demos/Real_Time_Glint/
- Shadertoy:
  https://www.shadertoy.com/view/wstcRH
- pbrt-v3:
  https://github.com/ASTex-ICube/importance_sampling_glint
- Dictionary generator:
  https://github.com/ASTex-ICube/real_time_glint_dictgenerator

Our algorithm is relatively simple. We implemented it with OpenGL, WebGL, Shadertoy, and the pbrt-v3 renderer. The code of the dictionary generator is also available.

Limitations

- Smith masking is not supported
- Approximate consistency between LODs
- No GGX support
- Modeling a single light bounce in the microsurface



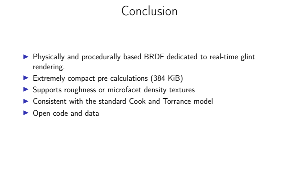
Limitations

- Smith masking is not supported
- Approximate consistency between LODs
- No GGX support
- Modeling a single light bounce in the microsurface

I will now talk about the limitations. Currently, Smith masking is not handled. There is also approximate lobe consistency between LODs. There is no convergence to the GGX distribution, and we only model a light bounce in the microsurface.

Conclusion

- Physically and procedurally based BRDF dedicated to real-time glint rendering.
- Extremely compact pre-calculations (384 KiB)
- Supports roughness or microfacet density textures
- Consistent with the standard Cook and Torrance model
- Open code and data



Conclusion

- Physically and procedurally based BRDF dedicated to real-time glint rendering.
- Extremely compact pre-calculations (384 KiB)
- Supports roughness or microfacet density textures
- Consistent with the standard Cook and Torrance model
- Open code and data

To summarize, in this work, we proposed a physically based and procedural BRDF dedicated to real-time glint rendering. The pre-computations are highly compact. Our method is compatible with roughness textures and microfacet density textures. Our model is consistent with the standard model of Cook and Torrance. Finally, our code and data are open. Merci, thank you for your attention. If you have questions, don't hesitate.