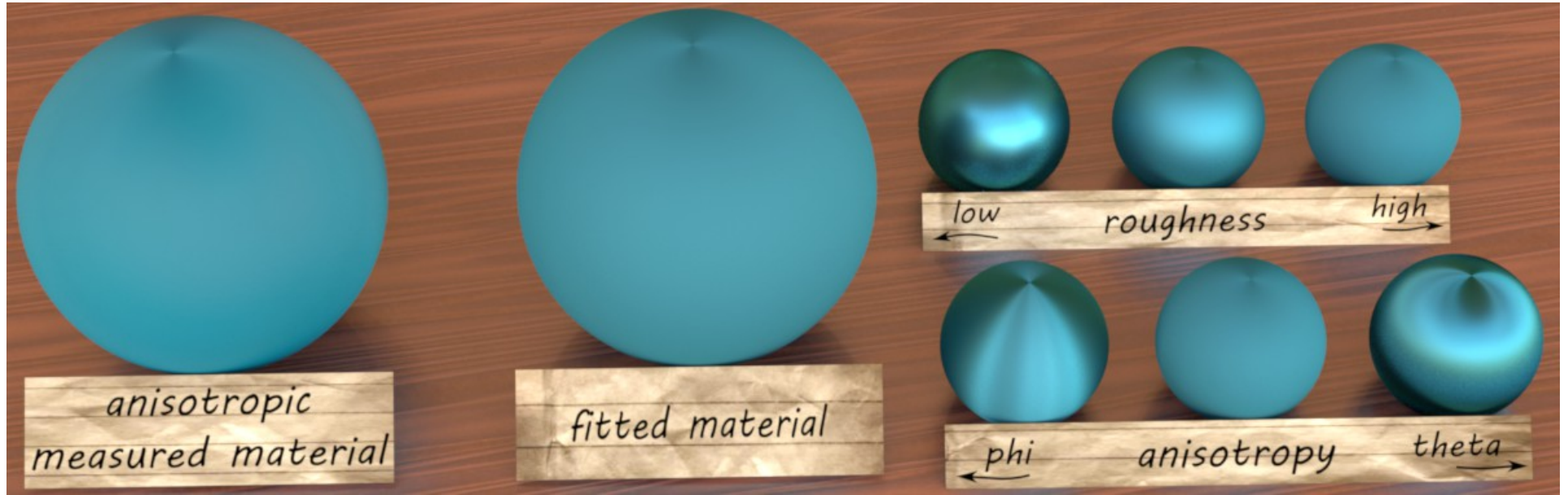


Extracting Microfacet-based BRDF Parameters from Arbitrary Materials with Power Iterations



Jonathan Dupuy^{1,2} Eric Heitz³ Jean-Claude Iehl¹ Pierre Poulin² Victor Ostromoukhov¹

¹ LIRIS, Université Lyon 1

² LIGUM, Université de Montréal

³ Karlsruhe Institute of Technology

Talk's Topic: BRDFs

$$L_o = \int_{\Omega_+} L_i \cdot \mathbf{f}_r \cdot \cos \theta_i \cdot d\omega_i$$

And more particularly:

Physically based microfacet BRDFs



Context



Real
(real materials)



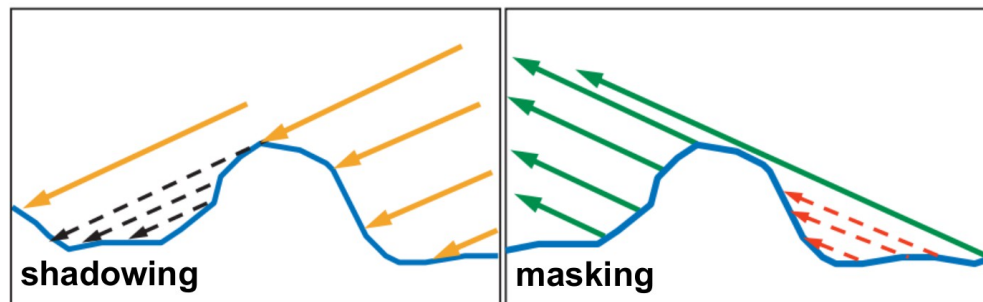
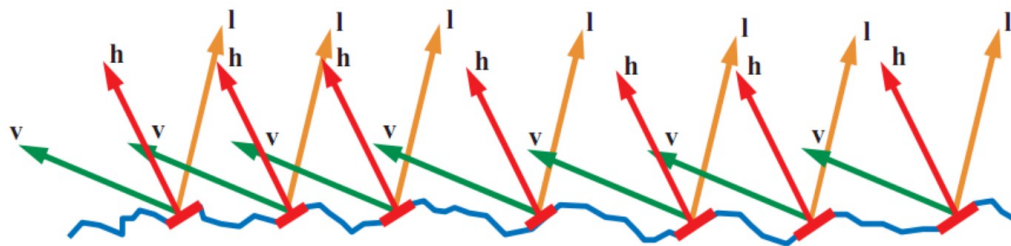
Digital
(microfacet BRDFs)

Microfacet Theory

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

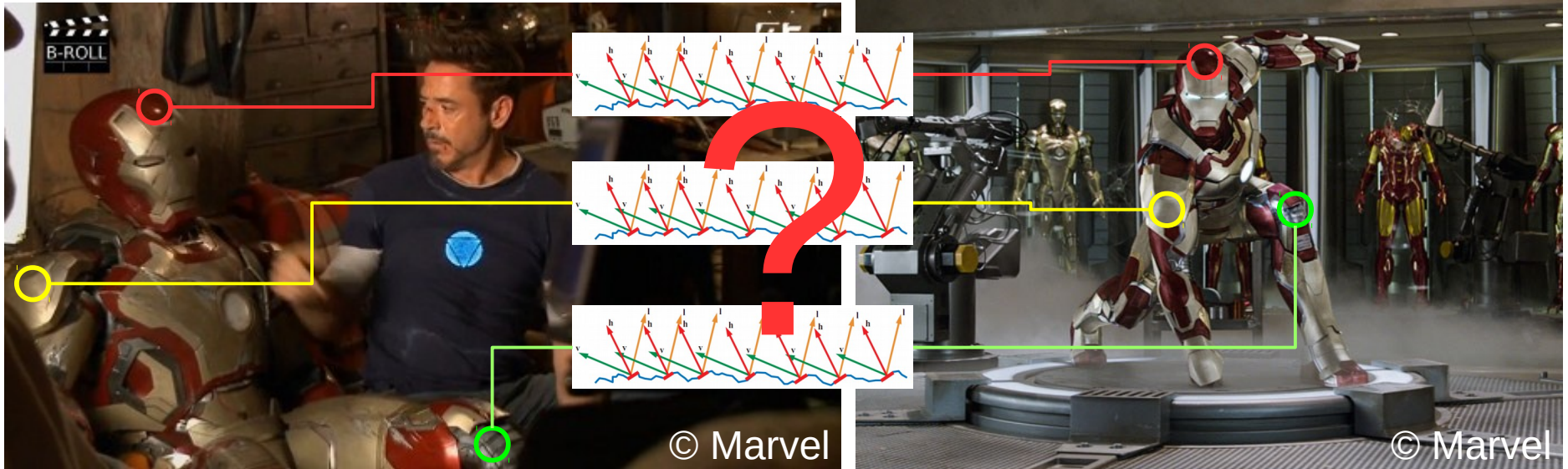
Intuition

- Materials = microsurfaces
- Artists manipulate microsurface properties (NDF, roughness, etc.)



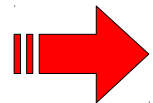
Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

Problem



Real
(real materials)

Digital
(microfacet BRDFs)



Can we retrieve the microsurface of real materials ?

Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

- Robustness / Speed
- Arbitrary metrics
- Reproducibility

[NDM05, LKYU12, BSH12, WZT*08]

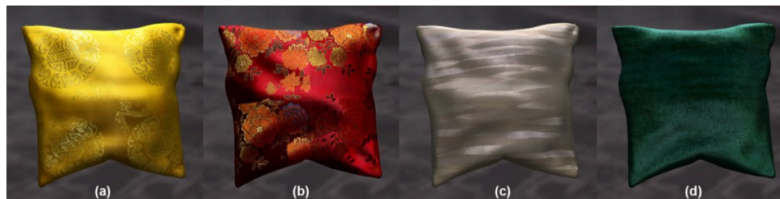
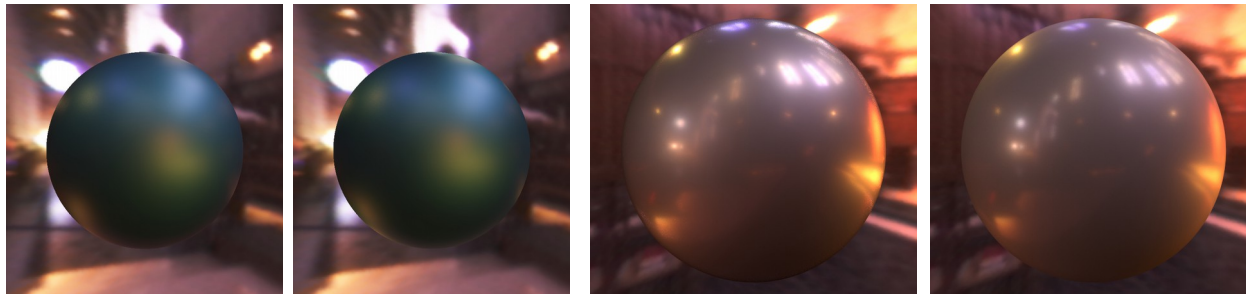


Figure 1: Fabric SVBRDFs from our algorithm mapped onto a pillow: (a) yellow satin, (b) red satin with colorful needlework, (c) wallpaper, (d) velvet.



Acquired

[BSH12]

Acquired

[LKYU12]

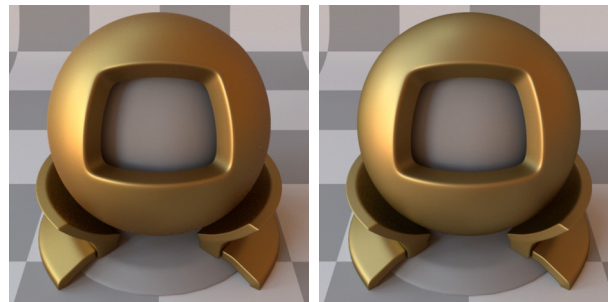
Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

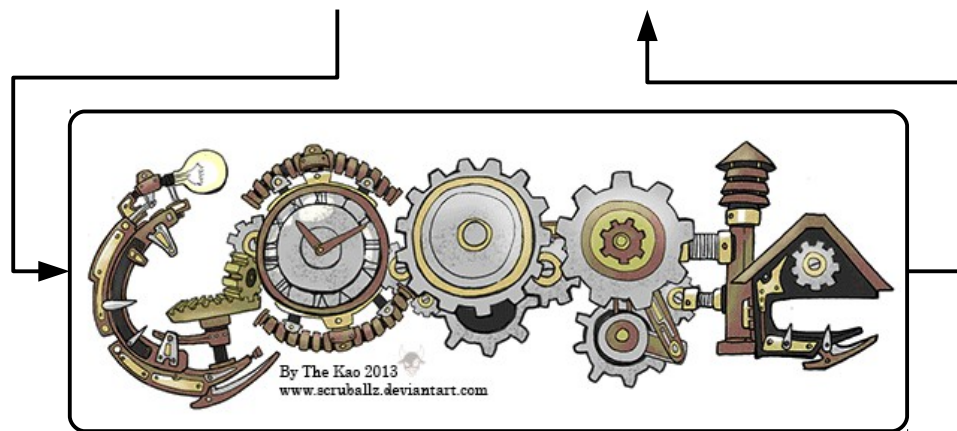
Current limitations

- Robustness / Speed
- Arbitrary metrics
- Reproducibility



Input

Output



Fitting algorithm

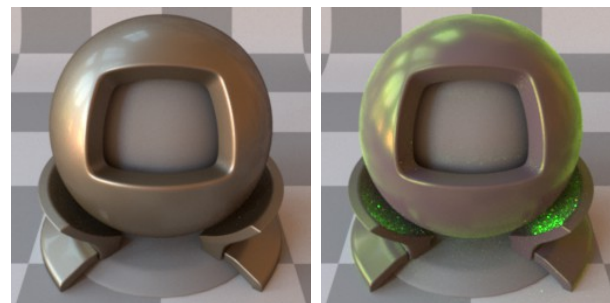
Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

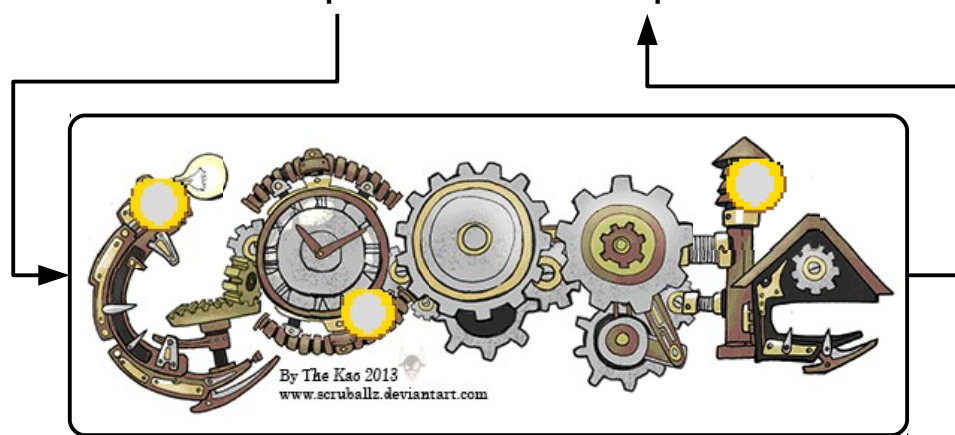
Current limitations

- **Robustness / Speed**
- Arbitrary metrics
- Reproducibility



Input

Output



Fitting algorithm

Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

- Robustness / Speed
- **Arbitrary metrics**
- Reproducibility

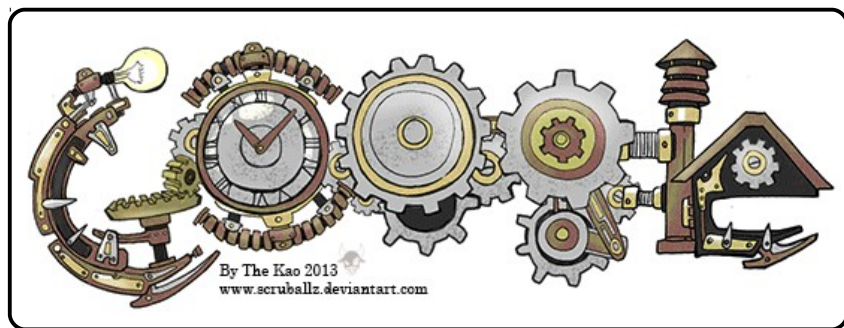
L1-norm ?

L^∞ -norm ?

L2-norm ?

Log space ?

Perceptual metric ?



Fitting algorithm

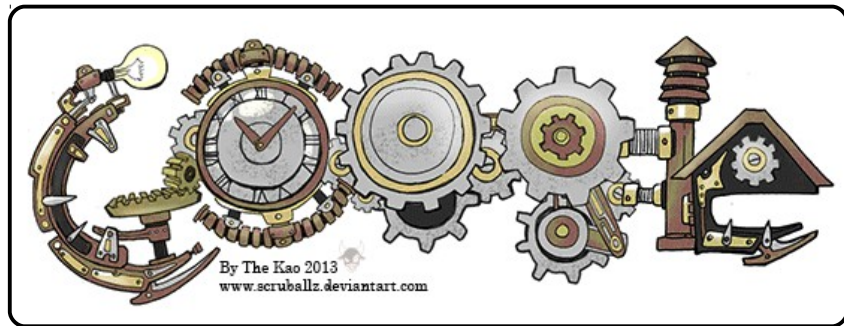
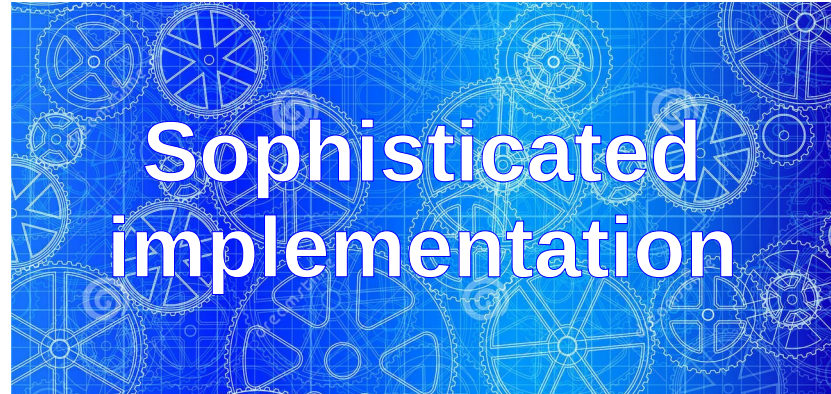
Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

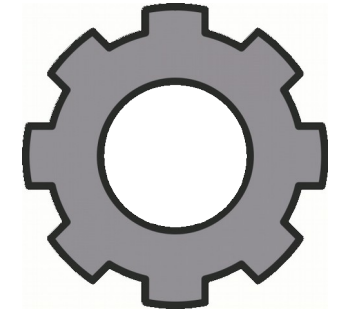
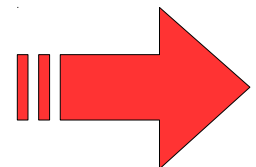
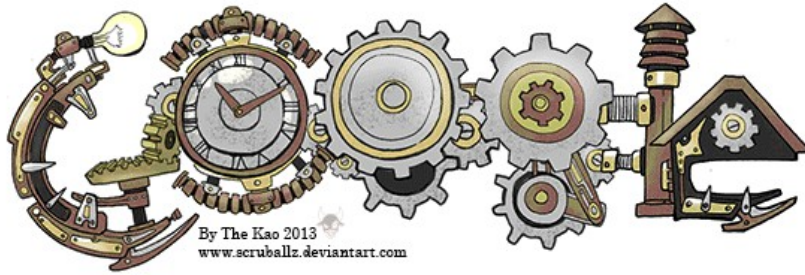
Current limitations

- Robustness / Speed
- Arbitrary metrics
- **Reproducibility**



Fitting algorithm

Contribution



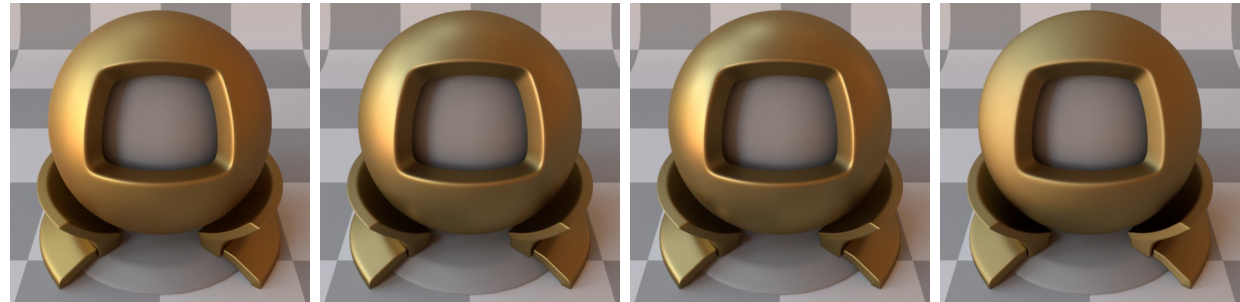
Contribution

Overview

- Robust tabulation
- Robust parametric fits

Properties

- Robustness
- Simplicity
- Speed
- Reproducibility



input

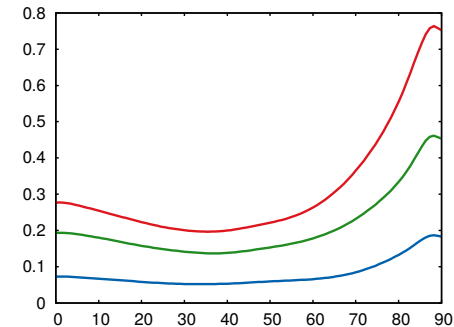
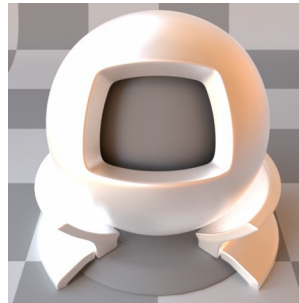
Tabulated

GGX

Beckmann

NDF

Fresnel

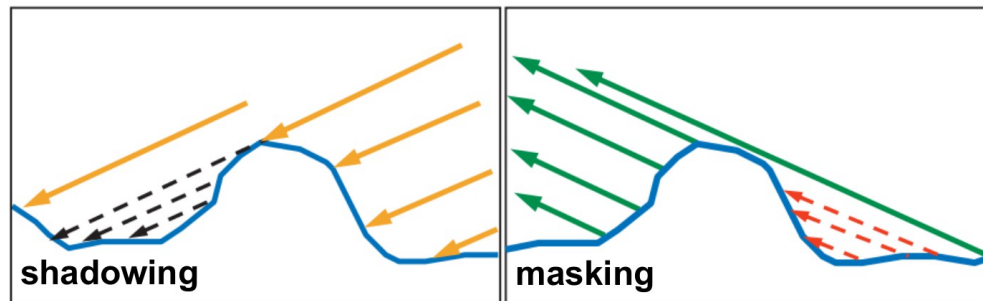
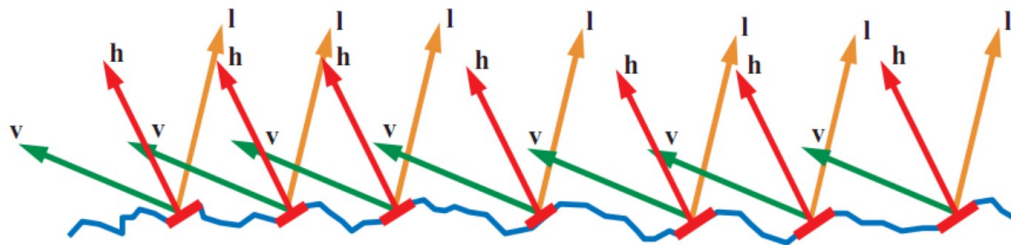


Microfacet Theory

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

Intuition

- Materials = microsurfaces
- Artists manipulate microsurface properties (NDF, roughness, etc.)

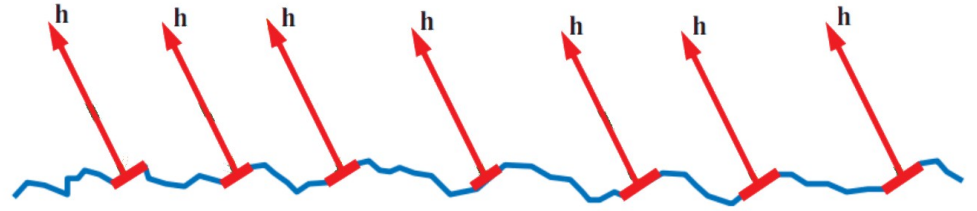


Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

Backscattering Equation

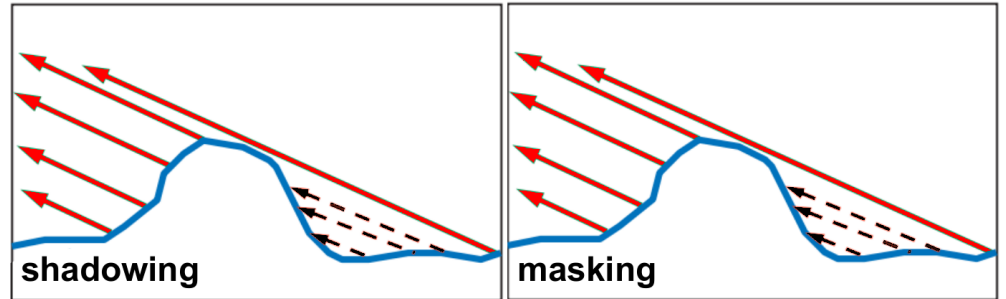
Mathematical (previous work)

$$f_r = \frac{F_0 \cdot D \cdot G}{4 \cdot \cos^2 \theta_o}$$



Physical (our work)

$$f_r = \frac{F_0 \cdot D \cdot G_1}{4 \cdot \cos^2 \theta_o}$$



Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

NDF Extraction

Physical equation

$$f_r = \frac{F_0 \cdot D \cdot G_1}{4 \cdot \cos^2 \theta_o}$$

Inverted equation

$$F_0 D(\mathbf{o}) = \int_{\Omega_+} K(\mathbf{o}, \mathbf{h}) D(\mathbf{h}) d\omega_h$$

(Fredholm equation of the second kind)

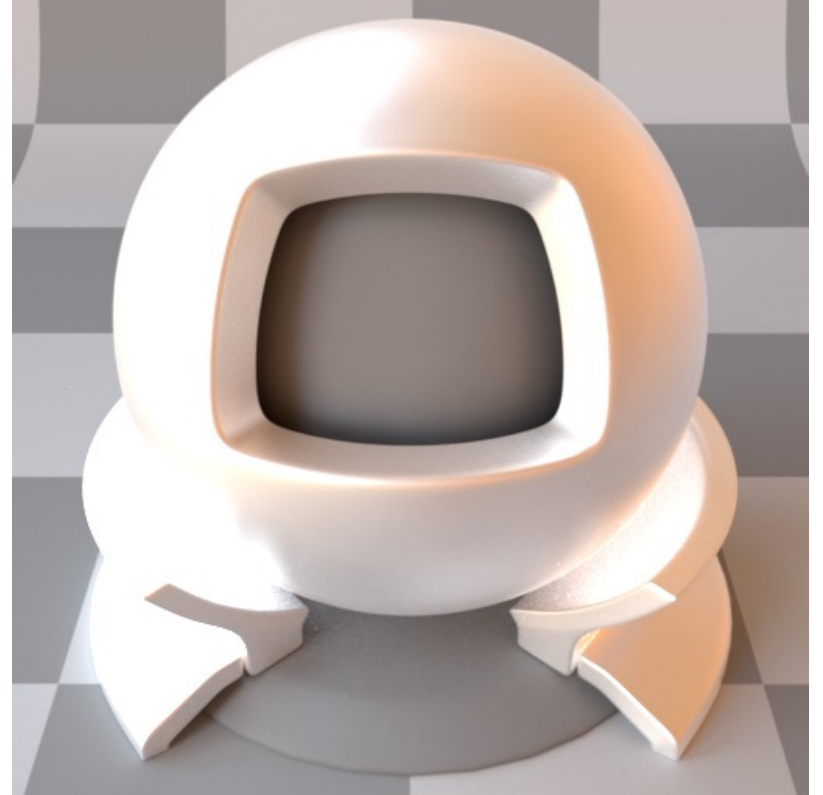
```
1 void tabular::compute_p22_smith(const brdf& brdf, int res)
2 {
3   int cnt = res - 1;
4   double dtheta = sqrt(M_PI * 0.5) / (double)cnt;
5   matrix km(cnt);
6
7   for (int i = 0; i < cnt; ++i) {
8     double tmp = (double)i / (double)cnt;
9     double theta = tmp * sqrt(M_PI * 0.5);
10    double theta_o = theta * theta;
11    double cos_theta_o = cos(theta_o);
12    double tan_theta_o = tan(theta_o);
13    vec3 fr = brdf.eval(dir(theta_o, 0.0), dir(theta_o, 0.0));
14    double fr_i = fr.intensity();
15    double kji_tmp = (dtheta * pow(cos_theta_o, 6.0)) * (8.0 * fr_i);
16
17    for (int j = 0; j < cnt; ++j) {
18      const double dphi_h = M_PI / 180.0;
19      double tmp = (double)j / (double)cnt;
20      double theta = tmp * sqrt(M_PI * 0.5);
21      double theta_h = theta * theta;
22      double cos_theta_h = cos(theta_h);
23      double tan_theta_h = tan(theta_h);
24      double tan_product = tan_theta_h * tan_theta_o;
25      double nint = 0.0;
26
27      for (double phi_h = 0.0; phi_h < 2.0 * M_PI; phi_h += dphi_h)
28        nint += max(1.0, tan_product * cos(phi_h));
29      nint *= dphi_h;
30
31      km(j, i) = theta * kji_tmp * nint * tan_theta_h
32      ..... / (cos_theta_h * cos_theta_o);
33    }
34  }
35
36  // compute slope.pdf
37  km.eigenvector(m_p22, 4);
38  m_p22.push_back(0);
39 }
```

Ideal Mirrors

$$f_{r,id} = \frac{D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

Special microfacet BRDF

- Fresnel term is 1
- Independent of wavelength



Fresnel Extraction

$$F(\theta_d) = \mathbb{E} \left[\frac{f_r}{f_{r,id}} \mid \mathbf{ih} = \cos \theta_d \right]$$

We compute an average response

- Fully automatic
- Simple implementation
- Fast evaluation
- Works well in practice



MERL gold-metallic-paint

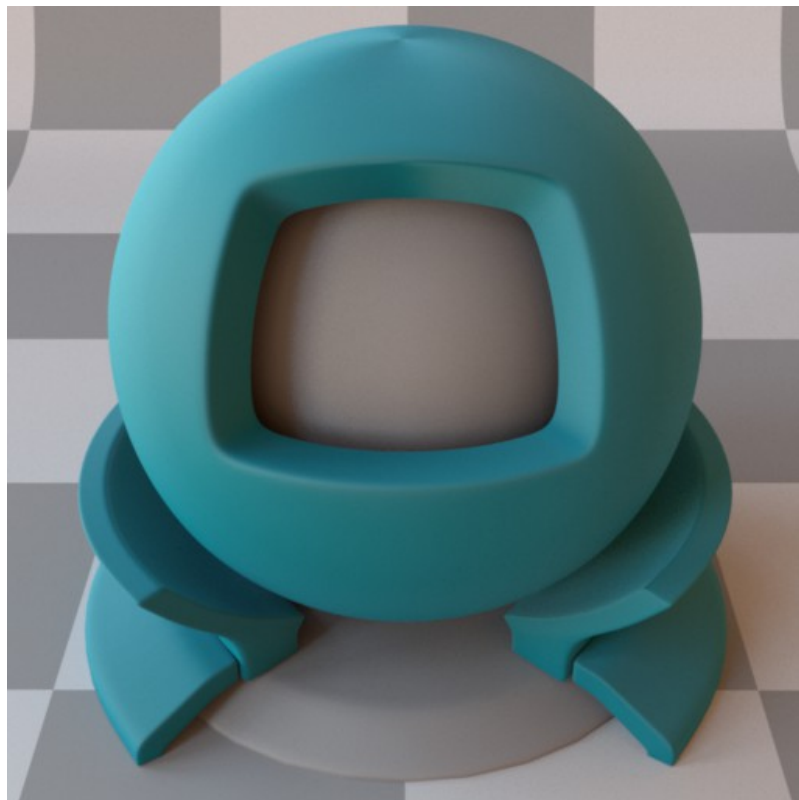


Acquired data

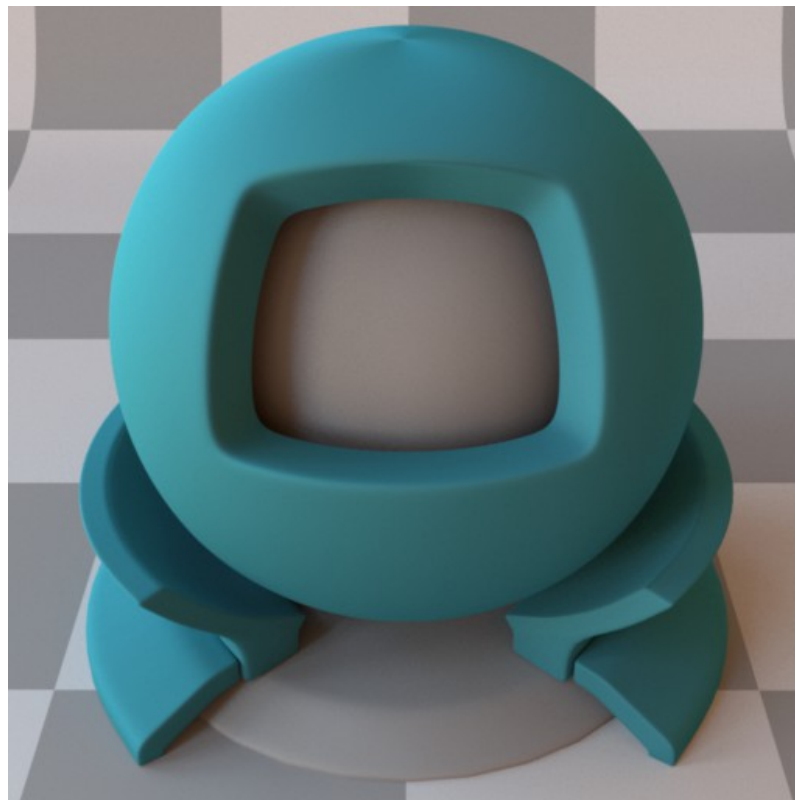


Ours: Tabulated

UTIA m064-fabric099



Acquired data



Ours: Tabulated

Parametric Fits

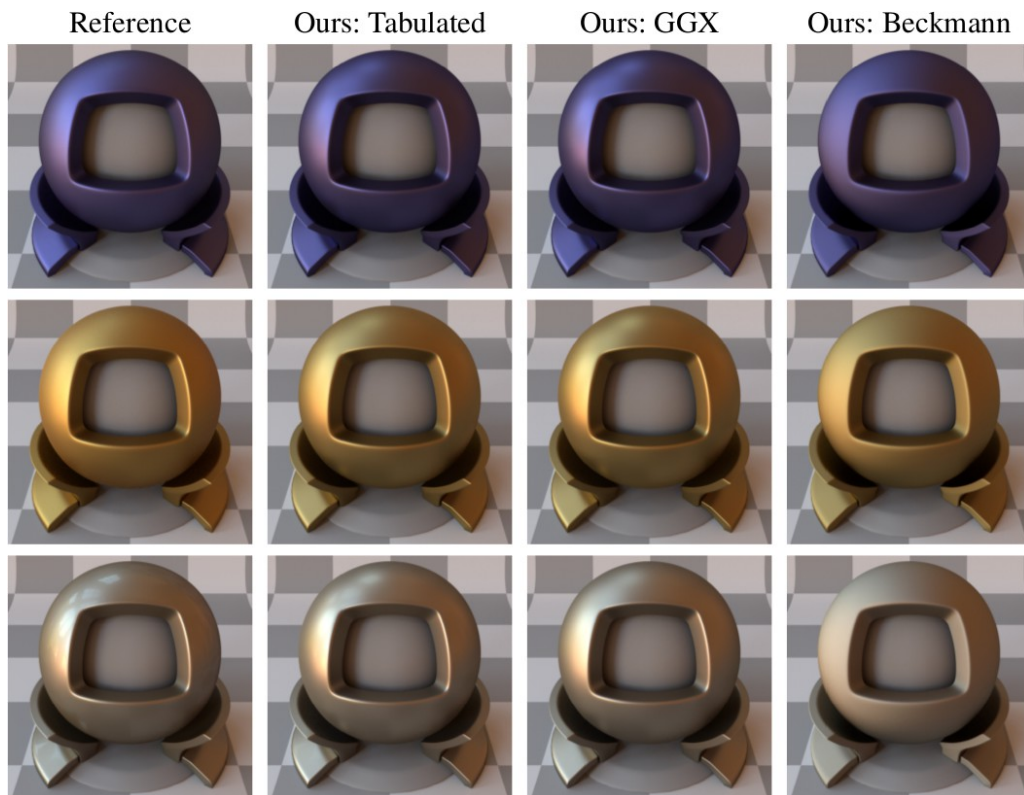
Beckmann roughness

$$2\alpha_G^2 = \int_{\mathbb{R}^2} \tilde{x}_h^2 P(\tilde{\mathbf{h}}) d\tilde{\mathbf{h}}$$

GGX roughness

$$\alpha_X = \int_{\mathbb{R}^2} |\tilde{x}_h| P(\tilde{\mathbf{h}}) d\tilde{\mathbf{h}}$$

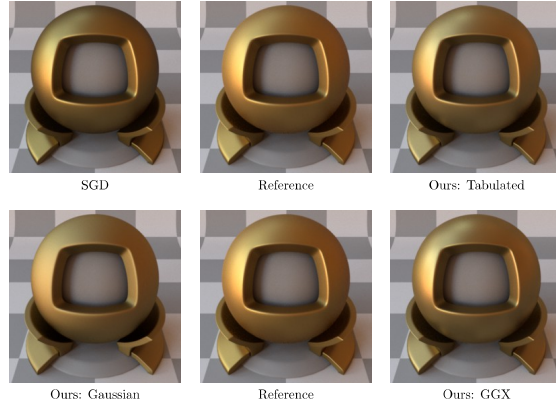
Our observation:
GGX > Beckmann



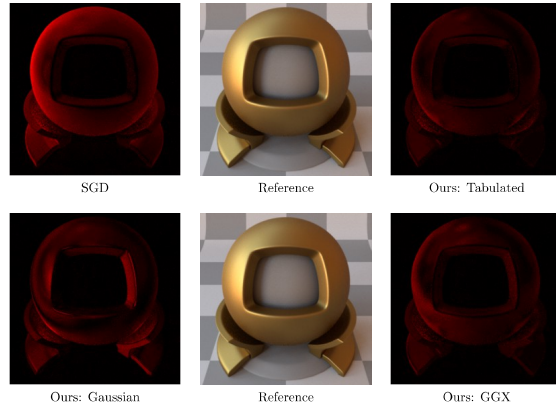
Validation

29 gold-metallic-paint

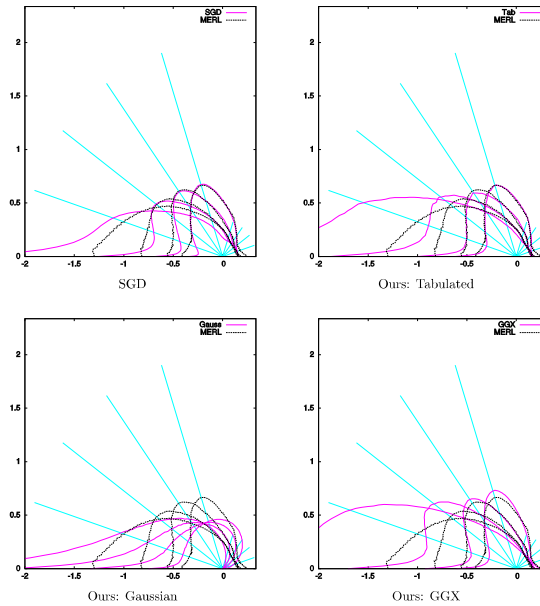
Renderings



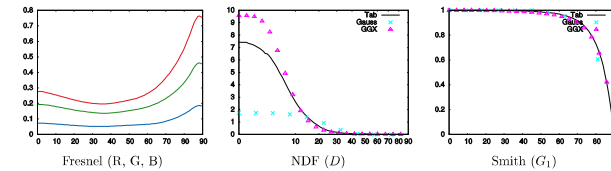
CIE dE00



BRDF Lobes in the Incidence Plane (Cubic Root Applied)

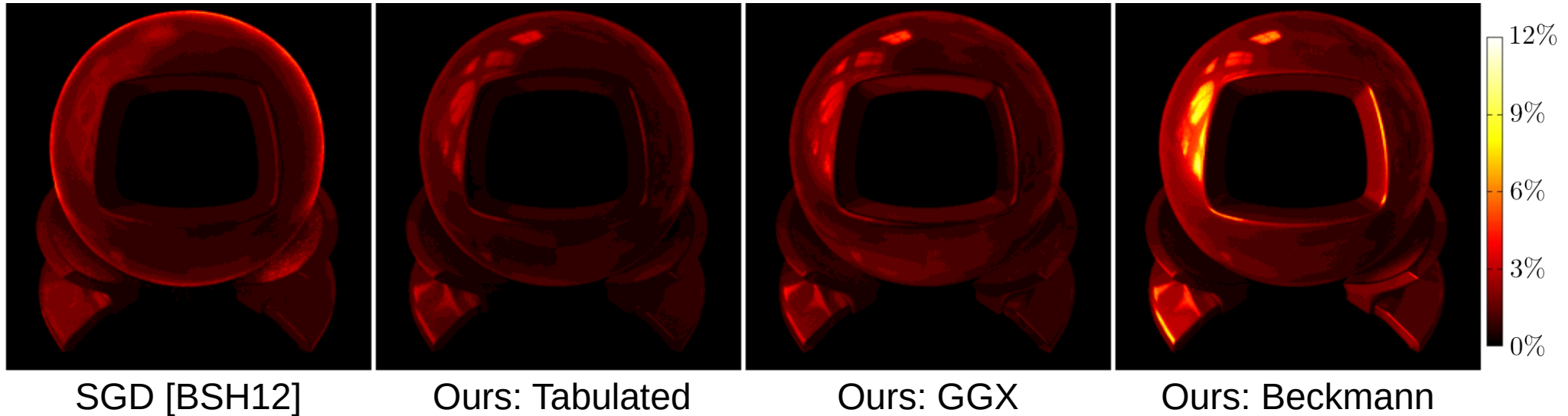


Fitted Microfacet Functions



Accuracy

Mean delta-E difference image on the MERL database



Contribution

Overview

- Robust tabulation
- Robust parametric fits

Key properties

- **Robustness**
- Simplicity
- Speed
- Reproducibility

$$F_0 D(\mathbf{o}) = \int_{\Omega_+} K(\mathbf{o}, \mathbf{h}) D(\mathbf{h}) d\omega_h$$

$$\Rightarrow F_0 \cdot \mathbf{d} = \mathbf{K} \cdot \mathbf{d}$$

NDF coefficients

nonnegative matrix

Perron-Frobenius theorem
the solution is always the eigenvector
with the largest magnitude

Contribution

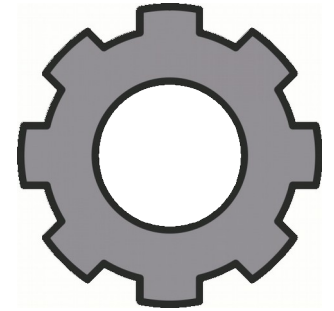
Novel fitting algorithm

- Robust tabulation
- Robust parametric fits

Key properties

- Robustness
- **Simplicity**
- Speed
- Reproducibility

```
1 void tabular::compute_p22_smith(const brdf& brdf, int res)
2 {
3     int cnt = res - 1;
4     double dtheta = sqrt(M_PI * 0.5) / (double)cnt;
5     matrix km(cnt);
6
7     for (int i = 0; i < cnt; ++i) {
8         double tmp = (double)i / (double)cnt;
9         double theta = tmp * sqrt(M_PI * 0.5);
10        double theta_o = theta * theta;
11        double cos_theta_o = cos(theta_o);
12        double tan_theta_o = tan(theta_o);
13        vec3 fr = brdf.eval(dir(theta_o, 0.0), dir(theta_o, 0.0));
14        double fr_i = fr.intensity();
15        double kji_tmp = (dtheta * pow(cos_theta_o, 6.0)) * (8.0 * fr_i);
16
17        for (int j = 0; j < cnt; ++j) {
18            const double dphi_h = M_PI / 180.0;
19            double tmp = (double)j / (double)cnt;
20            double theta = tmp * sqrt(M_PI * 0.5);
21            double theta_h = theta * theta;
22            double cos_theta_h = cos(theta_h);
23            double tan_theta_h = tan(theta_h);
24            double tan_product = tan_theta_h * tan_theta_o;
25            double nint = 0.0;
26
27            for (double phi_h = 0.0; phi_h < 2.0 * M_PI; phi_h += dphi_h)
28                nint += max(1.0, tan_product * cos(phi_h));
29            nint *= dphi_h;
30
31            km(j, i) = theta * kji_tmp * nint * tan_theta_h
32            ..... / (cos_theta_h * cos_theta_o);
33        }
34    }
35
36    // compute slope.pdf
37    km.eigenvector(m_p22, 4);
38    m_p22.push_back(0);
39 }
```



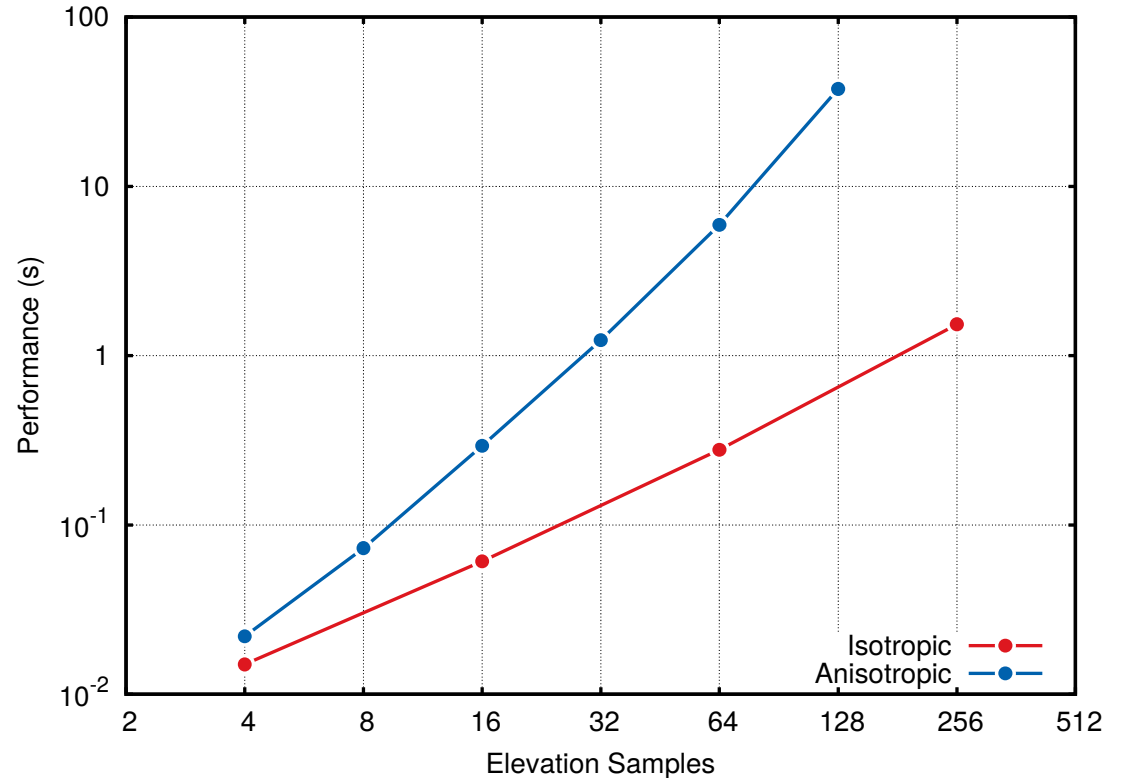
Contribution

Novel fitting algorithm

- Robust tabulation
- Robust parametric fits

Key properties

- Robustness
- Simplicity
- **Speed**
- Reproducibility



(Intel i5-2500 @ 3.30 GHz)

Contribution

Novel fitting algorithm

- Robust tabulation
- Robust parametric fits

Key properties

- Robustness
- Simplicity
- Speed
- **Reproducibility**



Source code:

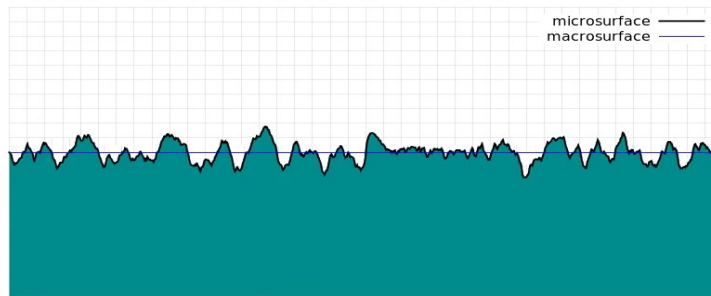
http://github.com/jdupuy/dj_brdf

Artistic Control

$$D = P \left(\frac{\tilde{x}_h}{\alpha_x}, \frac{\tilde{y}_h}{\alpha_y} \right) \frac{\sec^4 \theta_h}{\alpha_x \alpha_y}$$

Tabulate the slope PDF

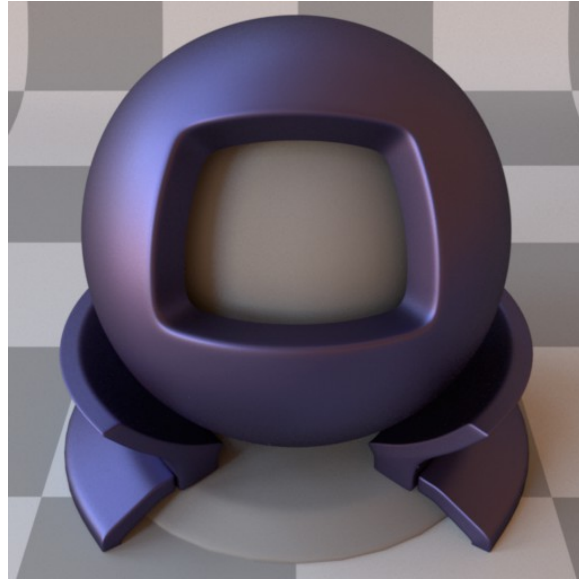
- Roughness $\propto \text{stretch}^{-1}$
- Efficient BRDF evaluation
- Efficient BRDF sampling



Editing blue-metallic-paint



Isotropic stretch

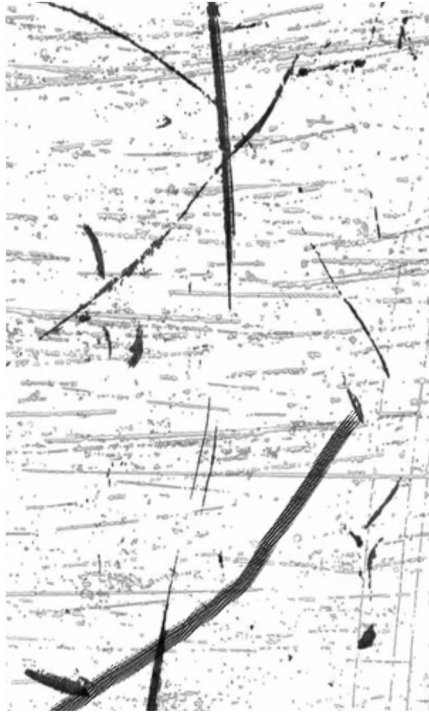


Acquired data

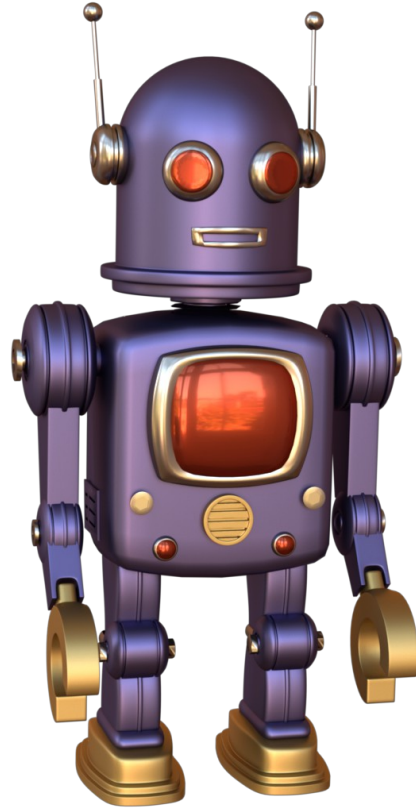


Anisotropic stretch

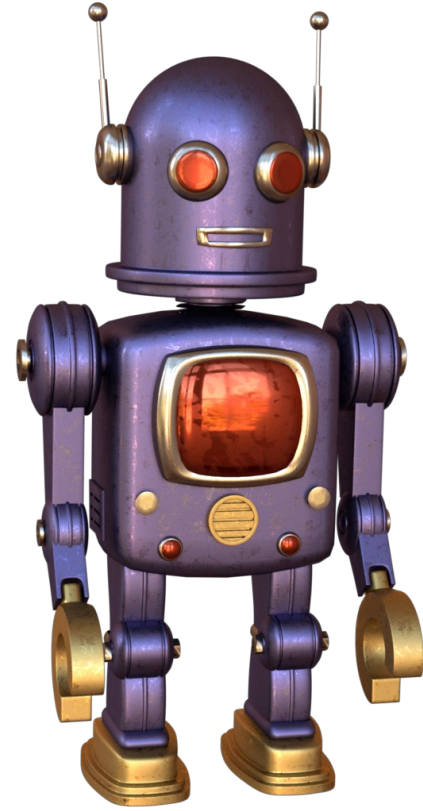
Roughness Textures



+

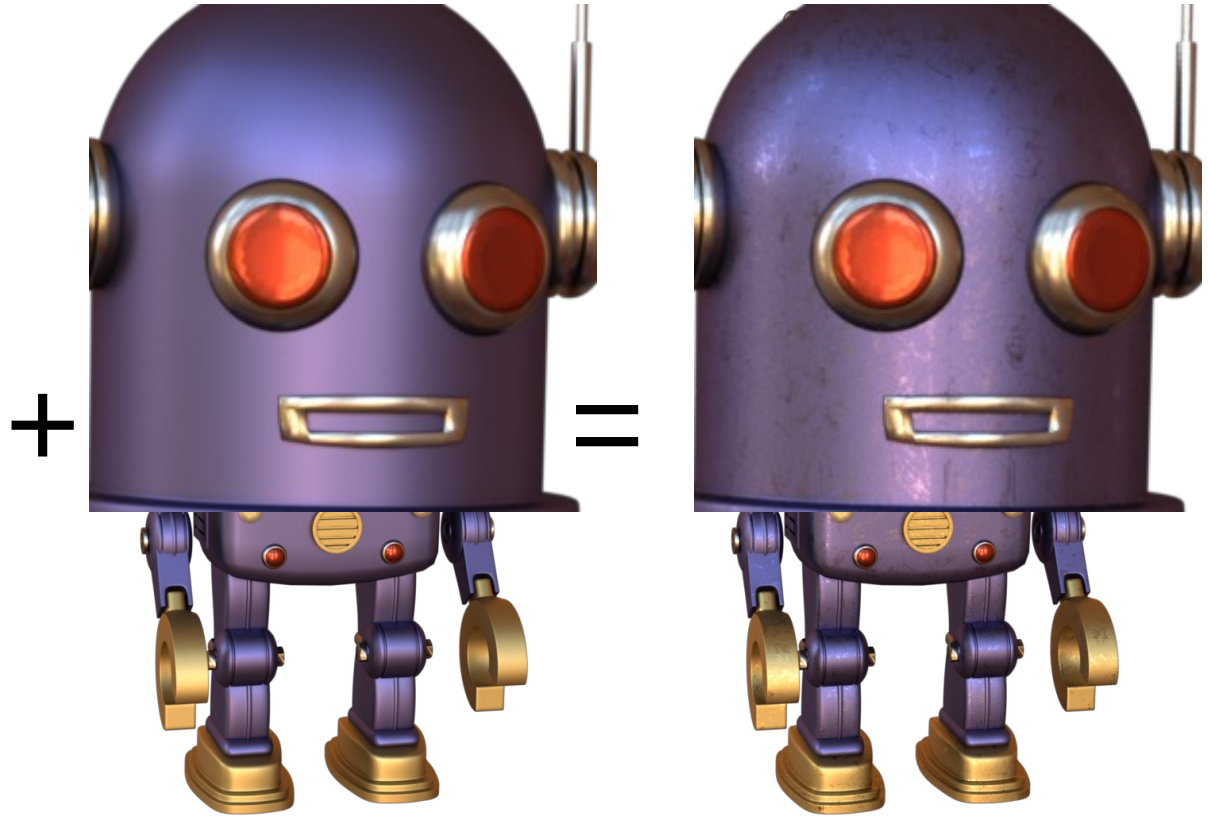
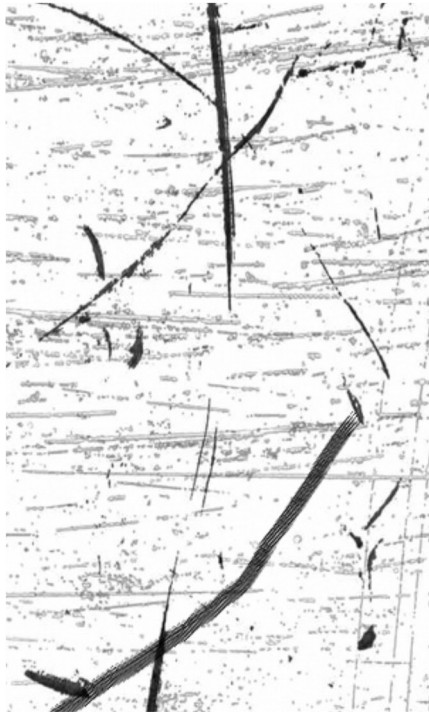


=



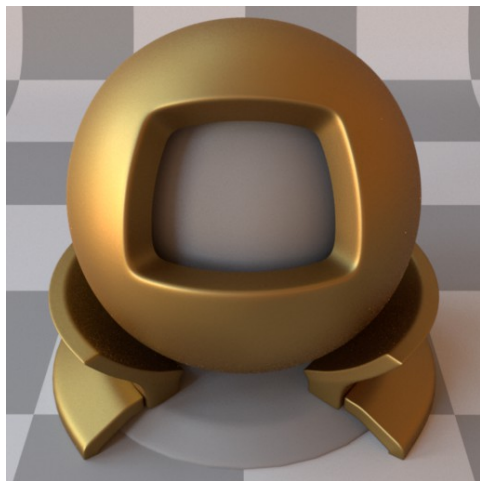
Robot model courtesy of LAGOA

Roughness Textures



Robot model courtesy of LAGOA

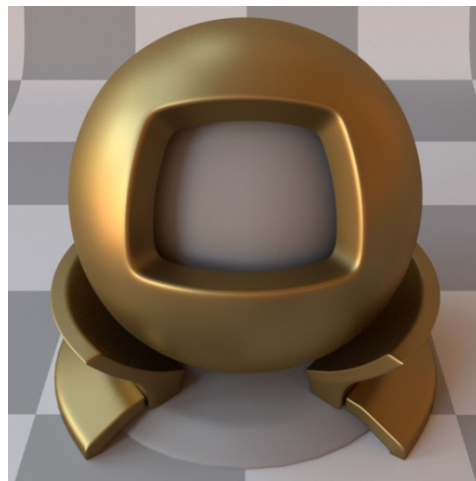
Memory Consumption



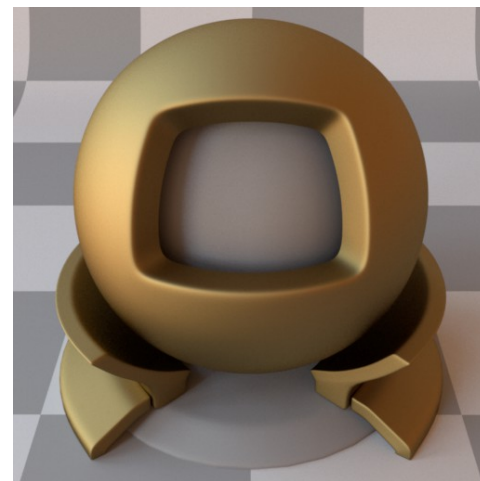
Acquired Data
35 MB



Ours: Tabulated
2.1 KB



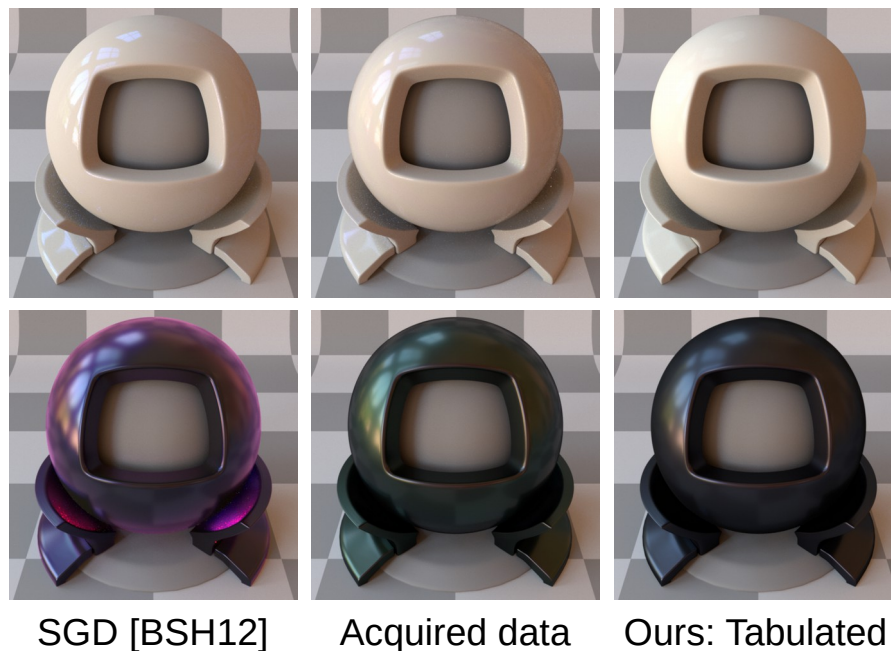
Ours: GGX
1 KB



Ours: Beckmann
1 KB

Conclusion

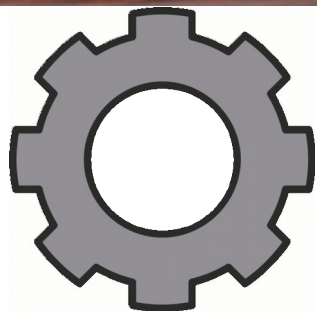
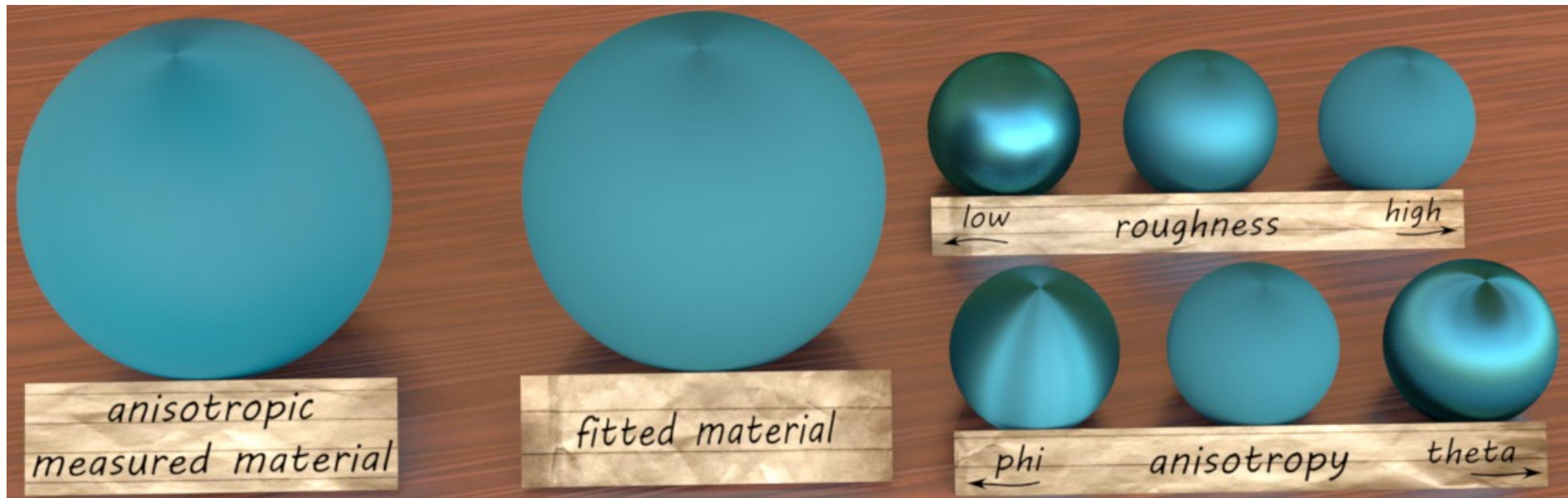
- Simple fitting technique
 - Robust
 - Fast
 - Reproducible
- Future work
 - Extend microfacet models (multiple bounces, layers, ...)
 - Invert BTDF, BSSRDF, and SVBRDF



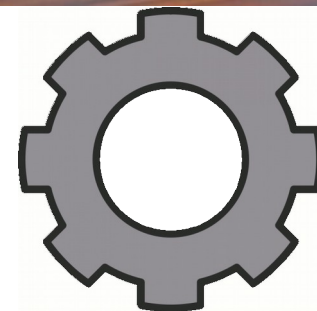
Acknowledgments

- **Laurent Pujo-Menjouet** and **Ionel Ciuperca** for their input on integral equations
- **Thiago Da Costa** and **Arno Zinke** for providing us with the LAGOA robot asset
- **Brent Burley** for discussing BRDF authoring in production

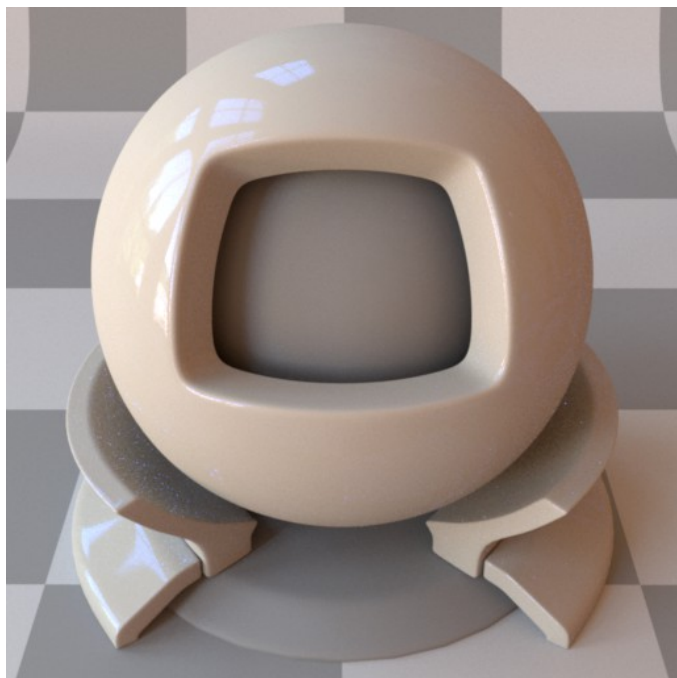
Questions



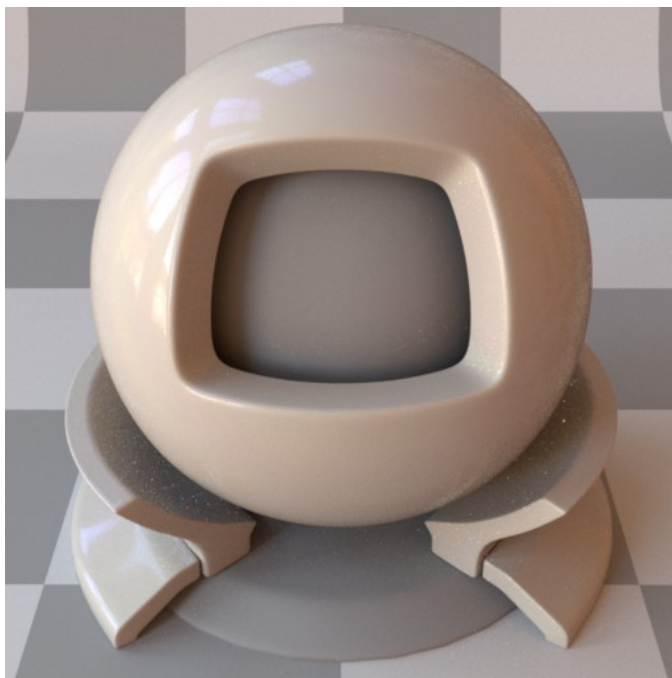
Get our source code online:
http://github.com/jdupuy/dj_brdf



Fail Case: Layers

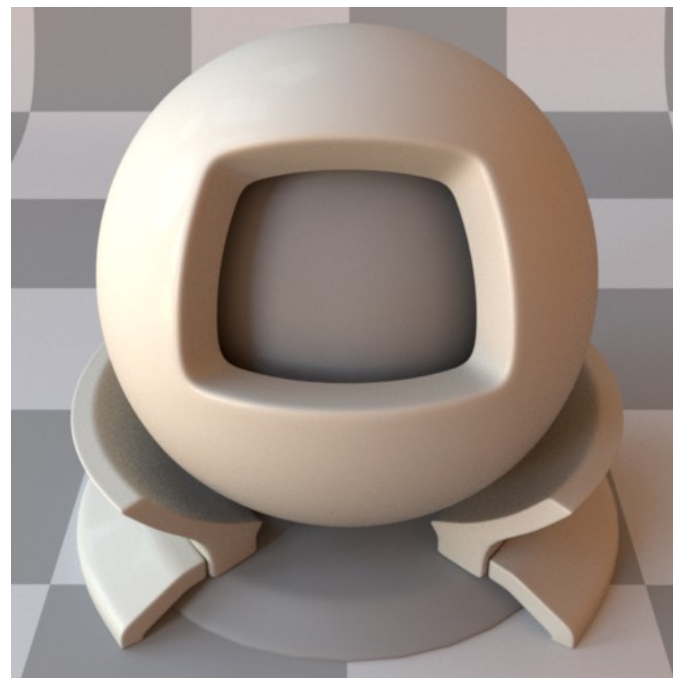


SGD [BSH12]



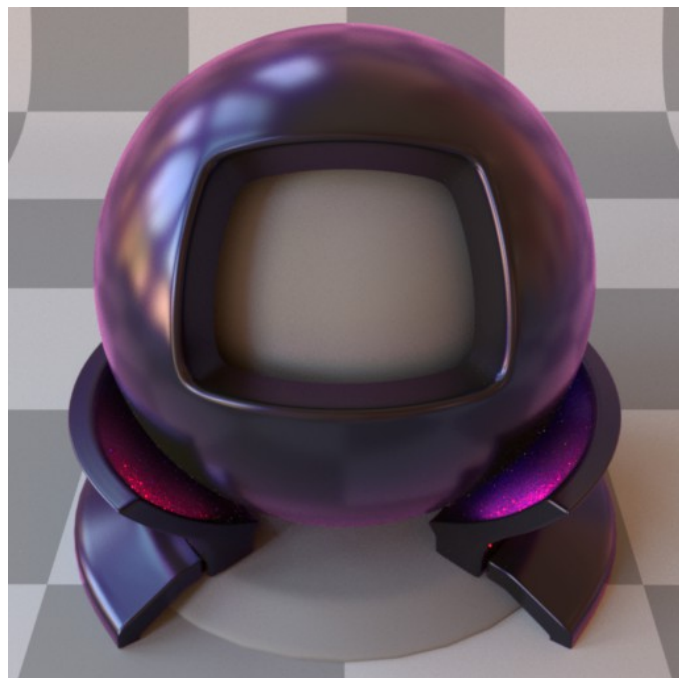
Acquired data

(MERL alumnia-oxide)

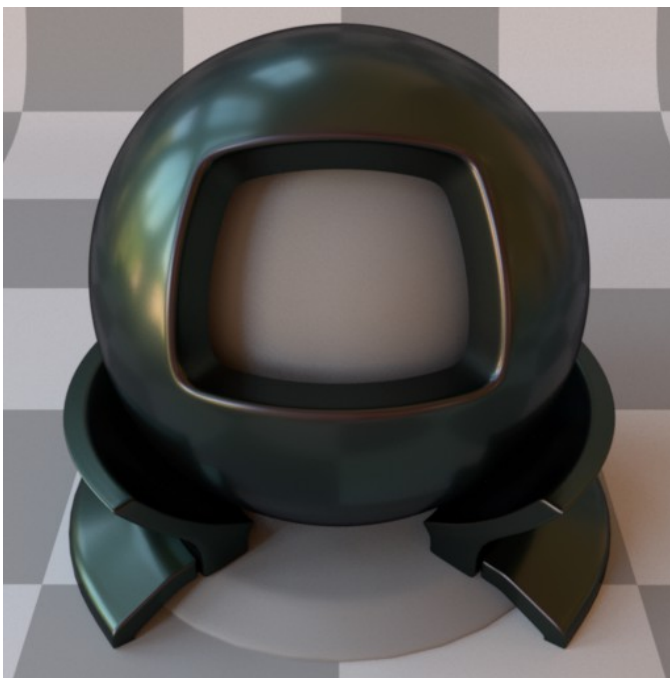


Ours: Tabulated

Fail Case: Color Changing



SGD [BSH12]



Acquired data

(MERL color-changing-paint1)



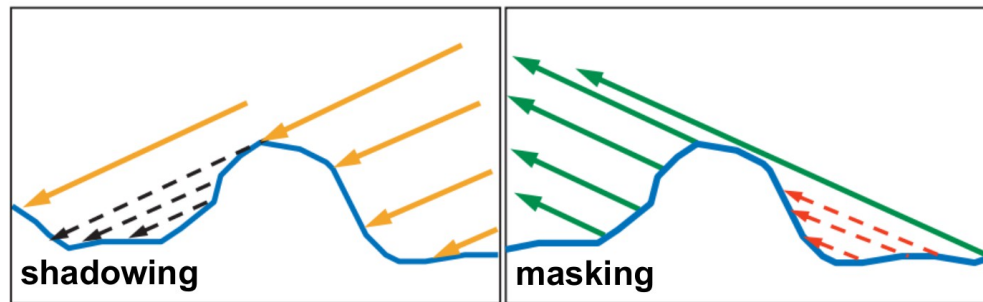
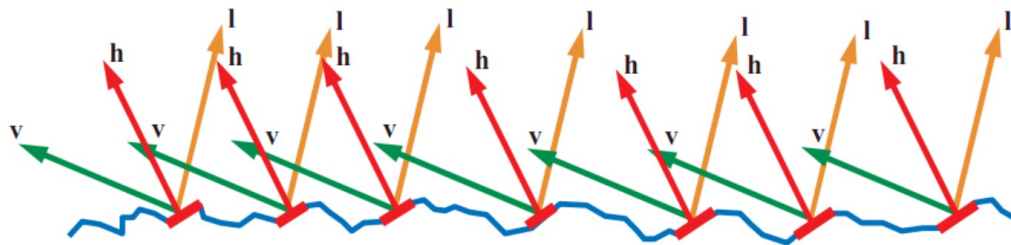
Ours: Tabulated

Microfacet BRDFs

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

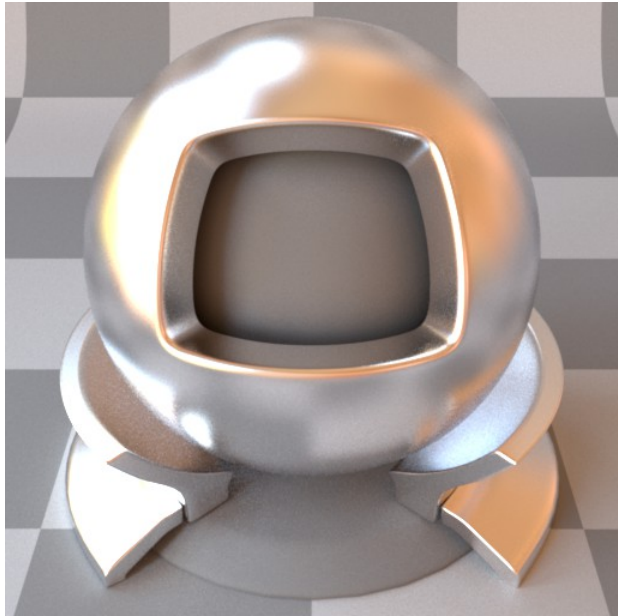
Modular components

- Fresnel term F
- Distribution of normals D
- Roughness α
- Geometric factor G

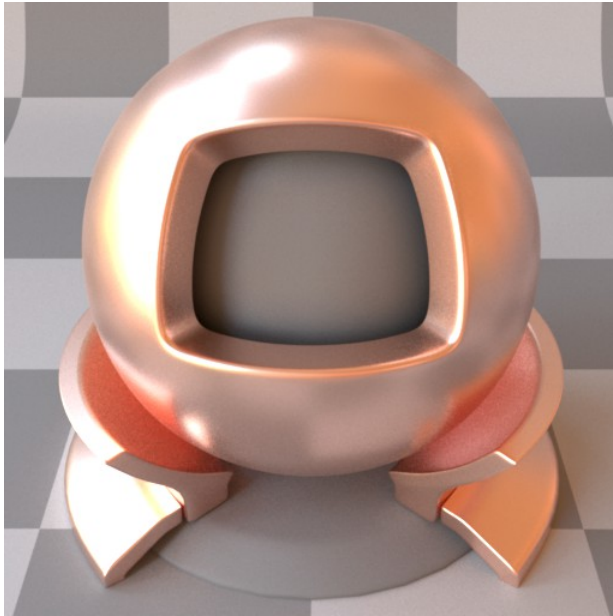


Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

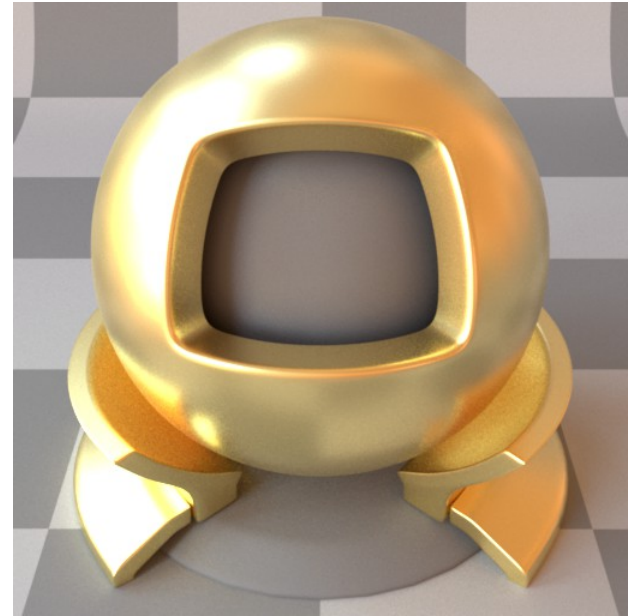
Fresnel



Aluminum

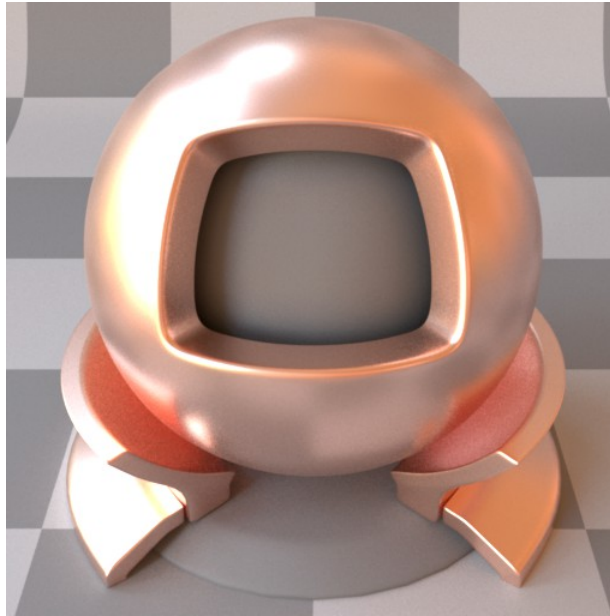
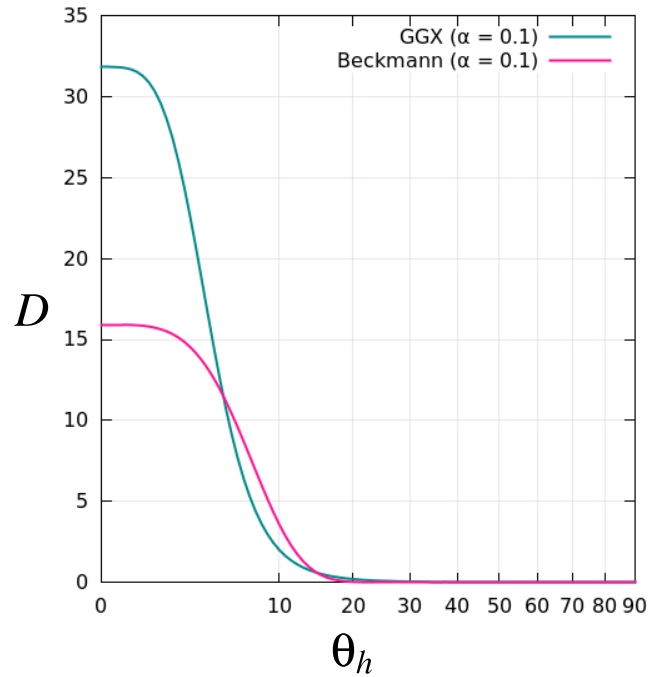


Copper

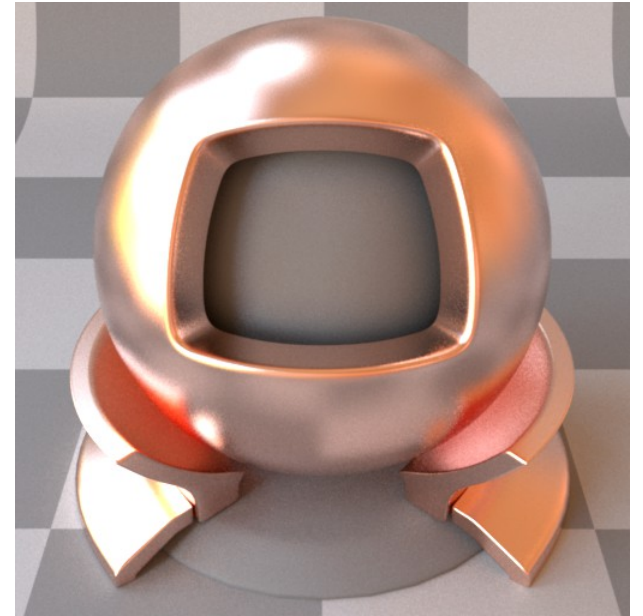


Gold

Distribution of Normals

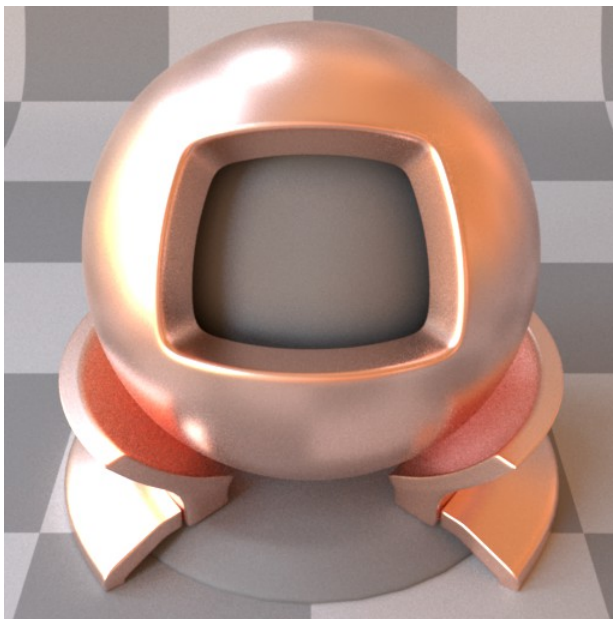
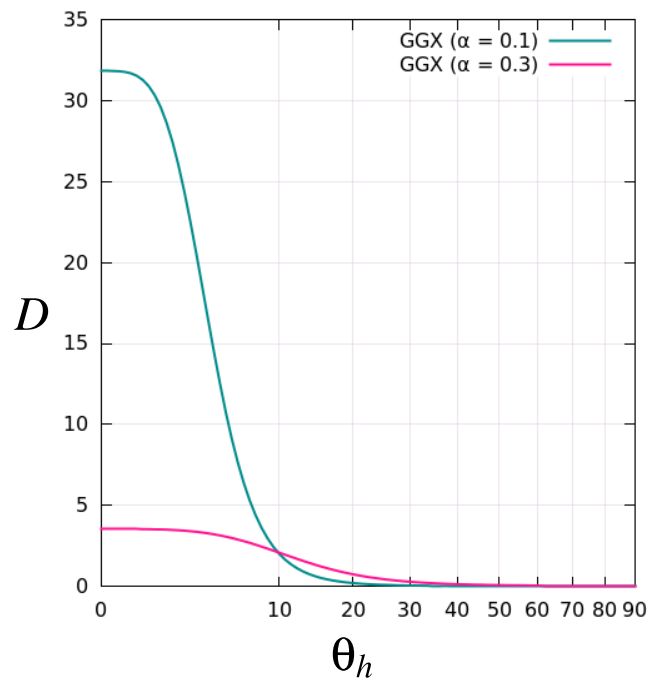


GGX

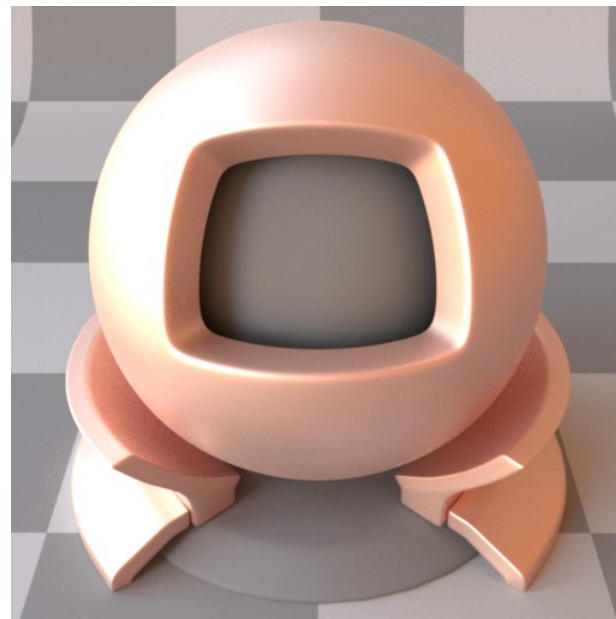


Beckmann

Roughness



$\alpha = 0.1$



$\alpha = 0.3$

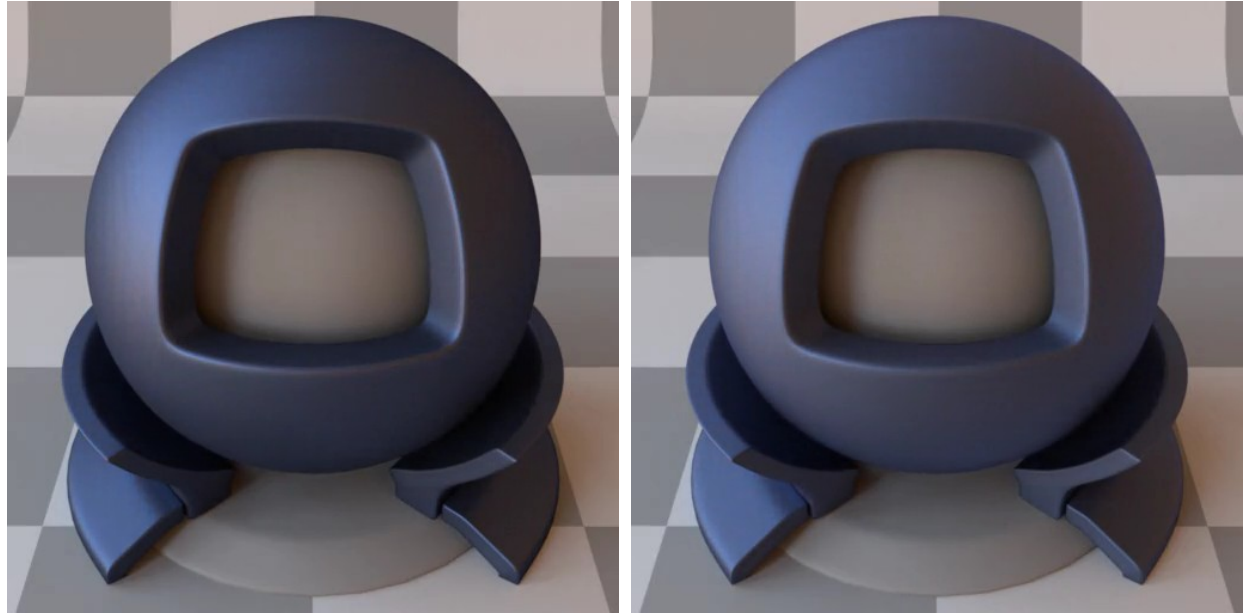
Geometric Factor

$$G = \min (G_1(\mathbf{h}, \mathbf{i}), G_1(\mathbf{h}, \mathbf{o}))$$

$$G_1(\mathbf{h}, \mathbf{k}) = \min \left(1, 2 \frac{\cos \theta_h \cos \theta_k}{\mathbf{kh}} \right)$$

$$G = \frac{G_1(\mathbf{i})G_1(\mathbf{o})}{G_1(\mathbf{i}) + G_1(\mathbf{o}) - G_1(\mathbf{i})G_1(\mathbf{o})}$$

$$G_1(\mathbf{k}) = \frac{\cos \theta_k}{\int_{\Omega_+} \mathbf{kh} D(\mathbf{h}) d\omega_h}$$



Smith

V Cavities

Inverting the BRDF

$$f_r(\mathbf{i}, \mathbf{o}) = \frac{\text{input (4D)} \quad \text{output} \quad F \cdot D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

Given an **input material**, can we retrieve a **Fresnel term**, a **distribution of normals**, and a **geometric factor** ?

Too difficult :(

Inverting the Smith Model

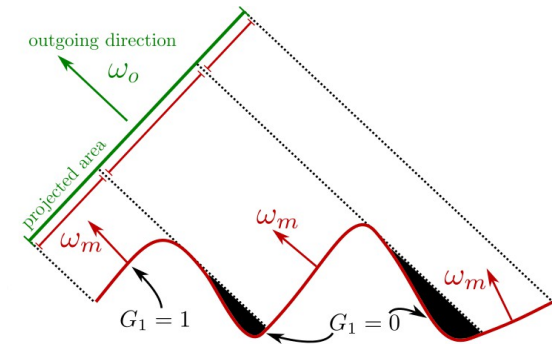
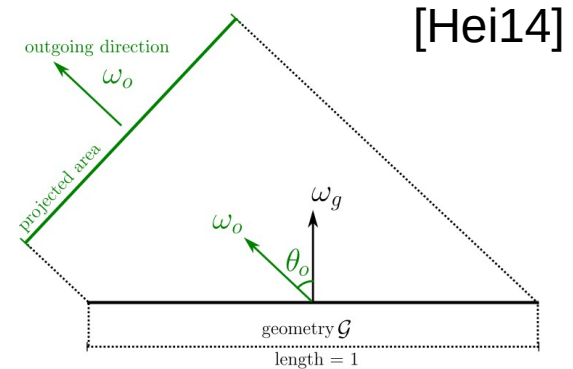
input

output

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos \theta_i \cdot \cos \theta_o}$$

$$G = \frac{G_1(\mathbf{i})G_1(\mathbf{o})}{G_1(\mathbf{i}) + G_1(\mathbf{o}) - G_1(\mathbf{i})G_1(\mathbf{o})}$$

$$G_1(\mathbf{k}) = \frac{\cos \theta_k}{\int_{\Omega_+} \mathbf{k} \mathbf{h} D(\mathbf{h}) d\omega_h}$$



Given an **input material**, can we retrieve a **Fresnel term** and a **distribution of normals** ?