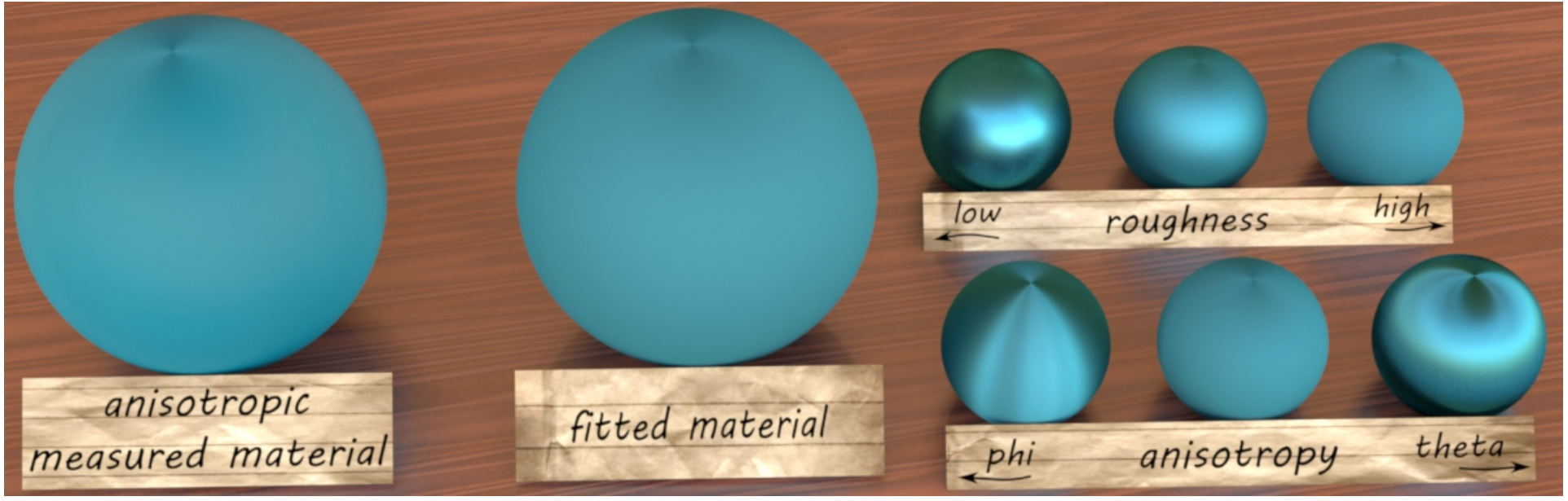
# Extracting Microfacet-based BRDF Parameters from Arbitrary Materials with Power Iterations




Jonathan Dupuy[1,2] Eric Heitz[3] Jean-Claude Iehl[1] Pierre Poulin[2] Victor Ostromoukhov[1]

[1] LIRIS, Université Lyon 1 [2] LIGUM, Université de Montréal [3] Karlsruhe Institute of Technology

# Talk's Topic: BRDFs

$$L_o = \int_{\Omega_+} L_i \cdot f_r \cdot \cos\theta_i \cdot d\omega_i$$

*And more particularly:*

Physically based microfacet BRDFs

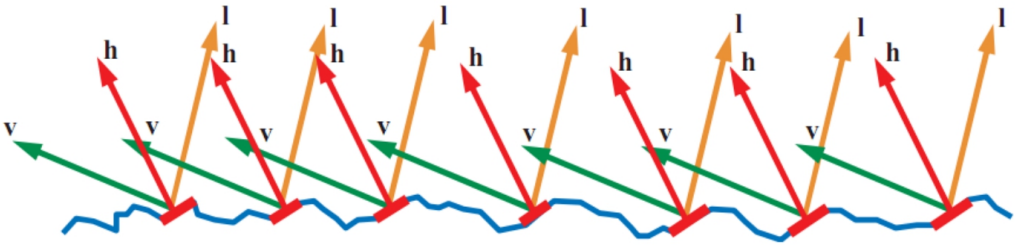
# Context


Real
(real materials)


Digital
(microfacet BRDFs)

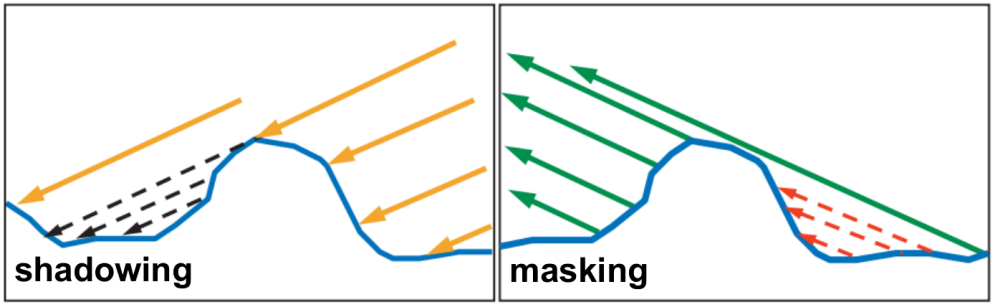
# Microfacet Theory

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



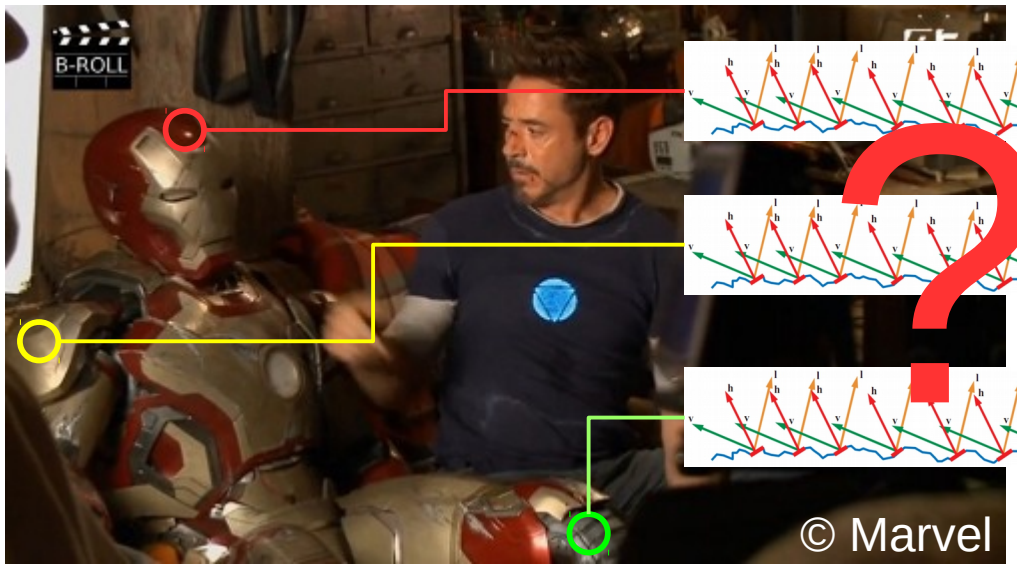

Intuition

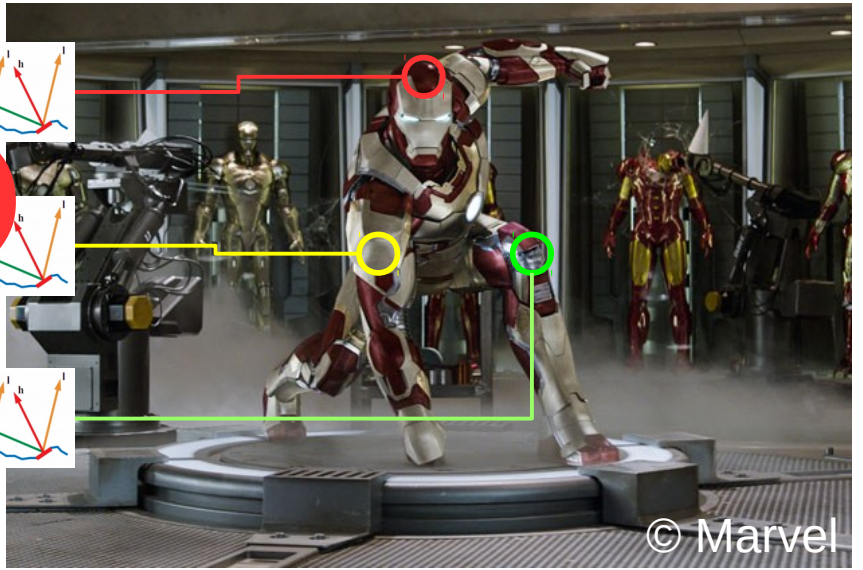
- Materials = microsurfaces
- Artists manipulate microsurface properties (NDF, roughness, etc.)




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 Fitting (Previous Work)

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

- Robustness / Speed
- Arbitrary metrics
- Reproducibility

[NDM05, BSH12, WZT*08]

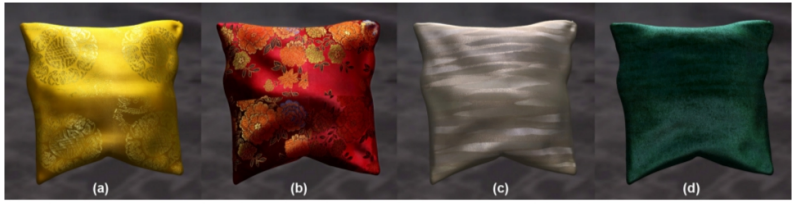
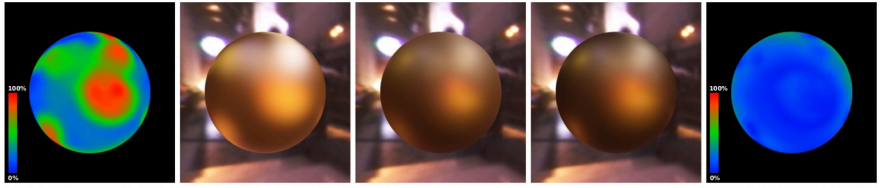



**Figure 1:** Fabric SVBRDFs from our algorithm mapped onto a pillow: (a) yellow satin, (b) red satin with colorful needlework, (c) wallpaper, (d) velvet.




(a) Beckmann Lab error (b) Beckmann distribution (c) Ground truth (d) SGD distribution (ours) (e) SGD Lab error (ours)

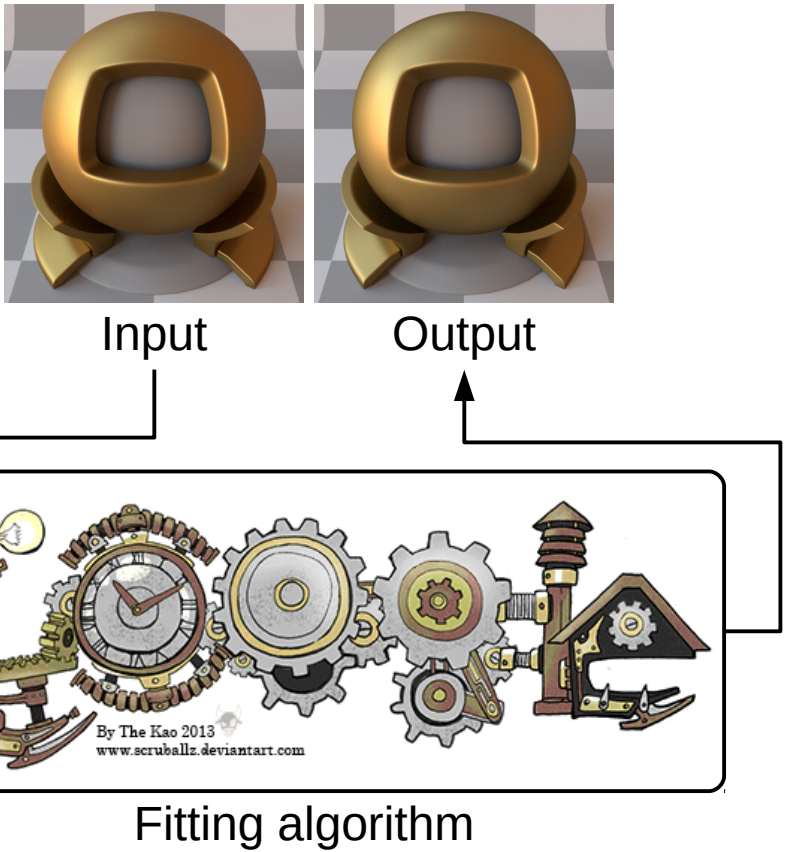
# Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

- Robustness / Speed
- Arbitrary metrics
- Reproducibility



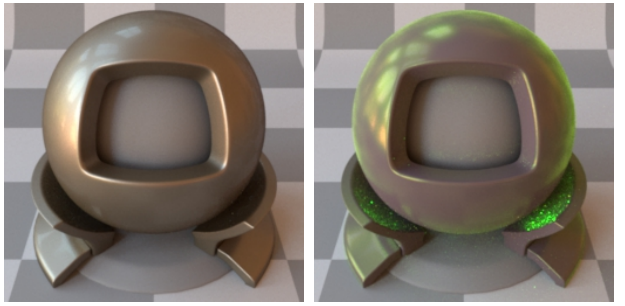
# Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

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



# Microfacet BRDF Fitting

Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

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

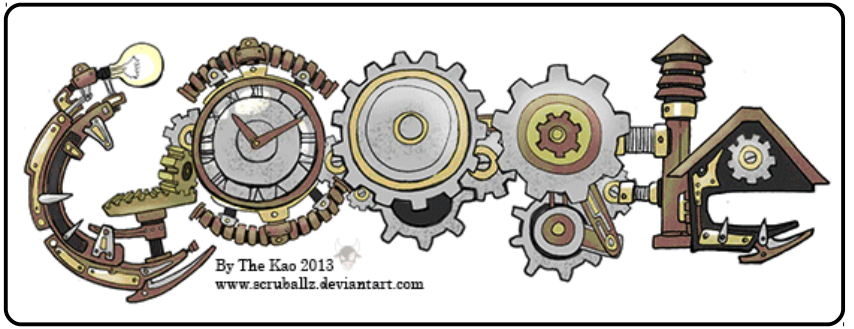
L1-norm ?

L∞-norm ?

L2-norm ?

Log space ?

Perceptual metric ?




Fitting algorithm

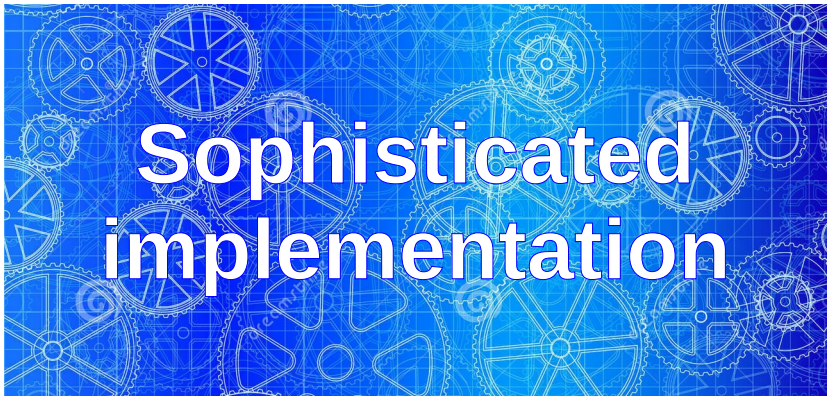
# Microfacet BRDF Fitting

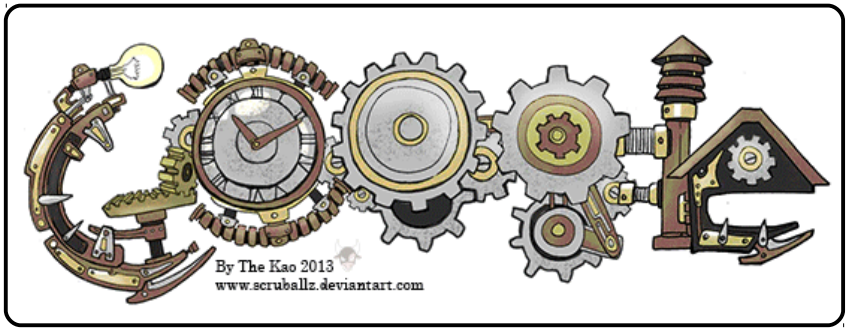
Approach

- Fitted microsurface
- Minimize fitting metric

Current limitations

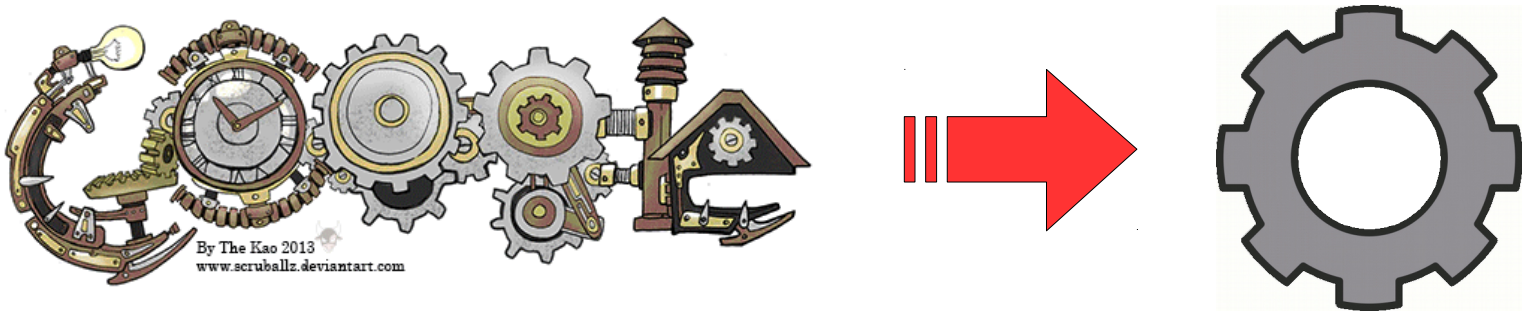
- Robustness / Speed
- Arbitrary metrics
- **Reproducibility**




Fitting algorithm

# Contribution



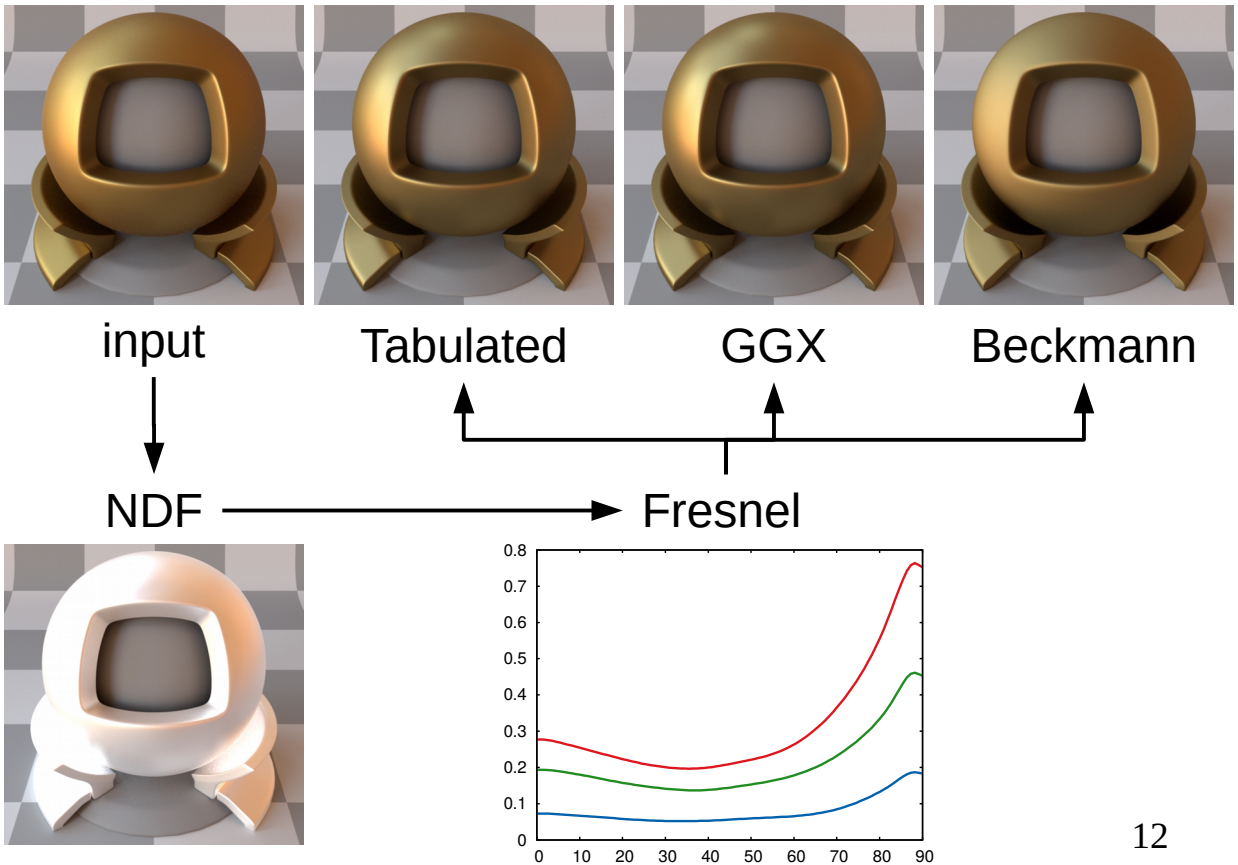
# Contribution

Overview

- Robust tabulation
- Robust parametric fits

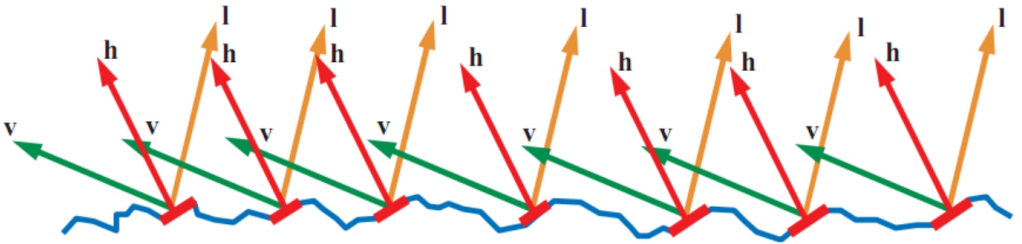
Properties

- Robustness
- Simplicity
- Speed
- Reproducibility



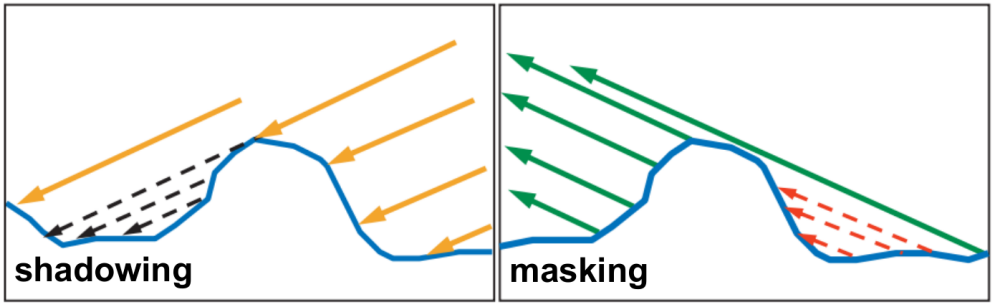
# Microfacet Theory

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




Intuition

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




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

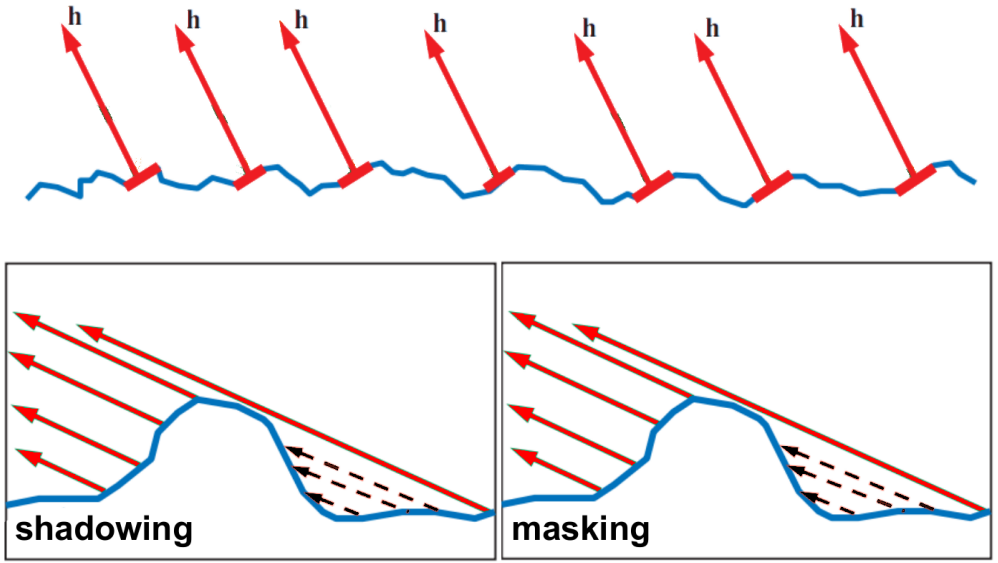
# Backscattering Equation

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)

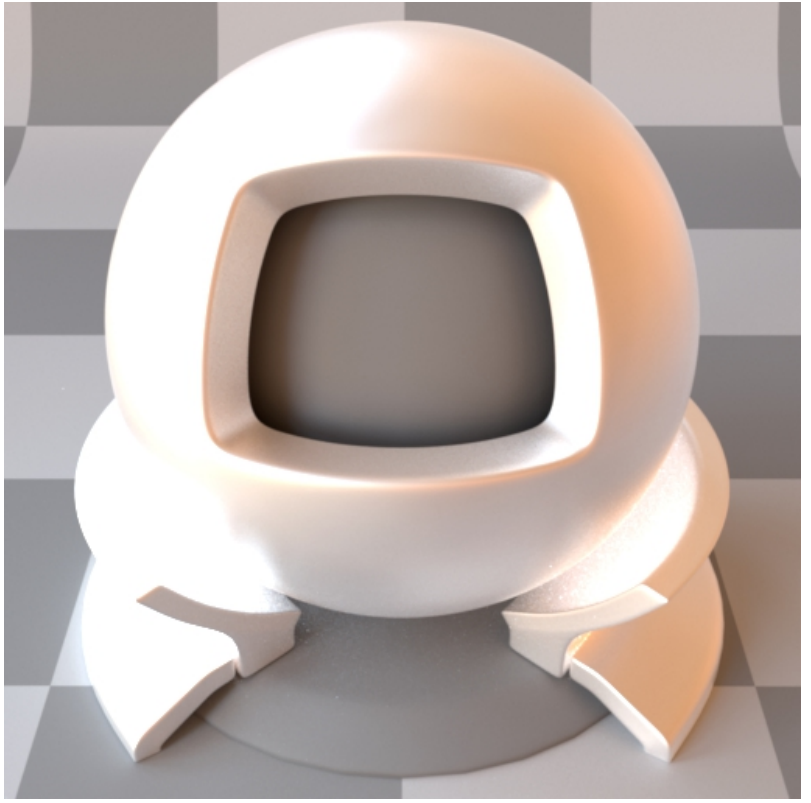
```
void tabular::compute_p22_smith(const brdf& brdf, int res)
{
    int cnt = res - 1;
    double dtheta = sqrt(M_PI * 0.5) / (double)cnt;
    matrix km(cnt);

    for (int i = 0; i < cnt; ++i) {
        double tmp = (double)i / (double)cnt;
        double theta = tmp * sqrt(M_PI * 0.5);
        double theta_o = theta * theta;
        double cos_theta_o = cos(theta_o);
        double tan_theta_o = tan(theta_o);
        vec3 fr = brdf.eval(dir(theta_o, 0.0), dir(theta_o, 0.0));
        double fr_i = fr.intensity();
        double kji_tmp = (dtheta * pow(cos_theta_o, 6.0)) * (8.0 * fr_i);

        for (int j = 0; j < cnt; ++j) {
            const double dphi_h = M_PI / 180.0;
            double tmp = (double)j / (double)cnt;
            double theta = tmp * sqrt(M_PI * 0.5);
            double theta_h = theta * theta;
            double cos_theta_h = cos(theta_h);
            double tan_theta_h = tan(theta_h);
            double tan_product = tan_theta_h * tan_theta_o;
            double nint = 0.0;

            for (double phi_h = 0.0; phi_h < 2.0 * M_PI; phi_h += dphi_h)
                nint += max(1.0, tan_product * cos(phi_h));
            nint *= dphi_h;

            km(j, i) = theta * kji_tmp * nint * tan_theta_h
                     / (cos_theta_h * cos_theta_h);
        }
    }

    // compute slope pdf
    km.eigenvector(m_p22, 4);
    m_p22.push_back(0);
}
```

# 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

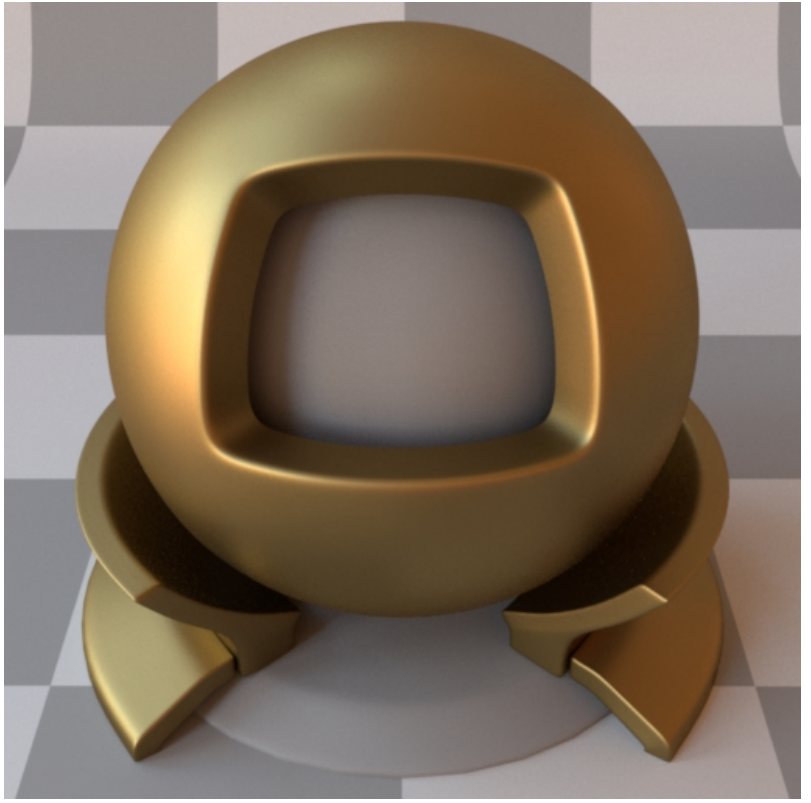


# Fresnel Extraction

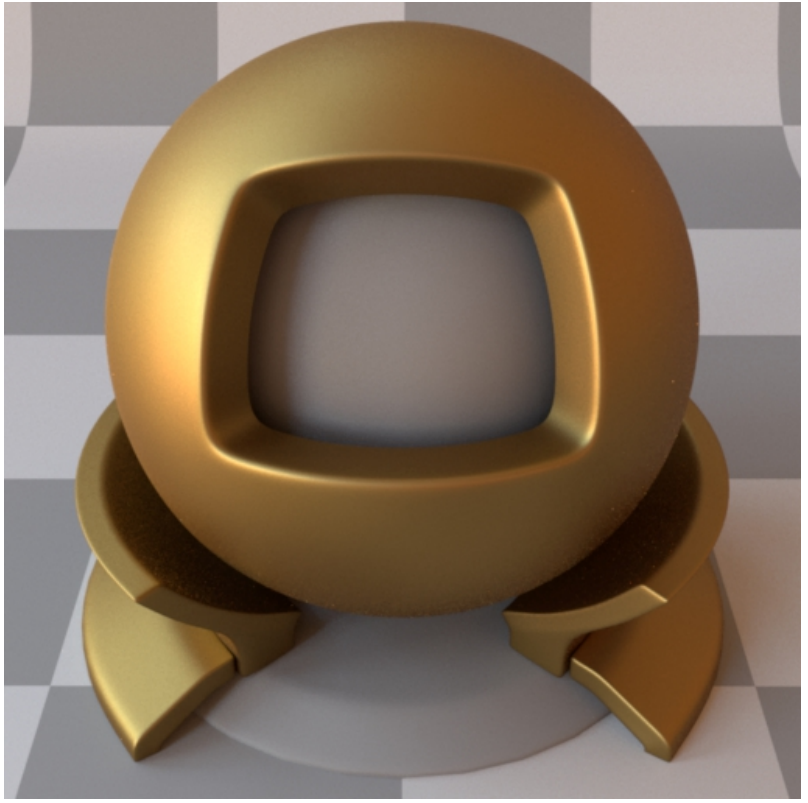
$$F(\theta_d) = \mathbb{E}\left[\frac{f_r}{f_{r,id}} \mid \mathbf{ih} = \cos\theta_d\right]$$

We compute an average response

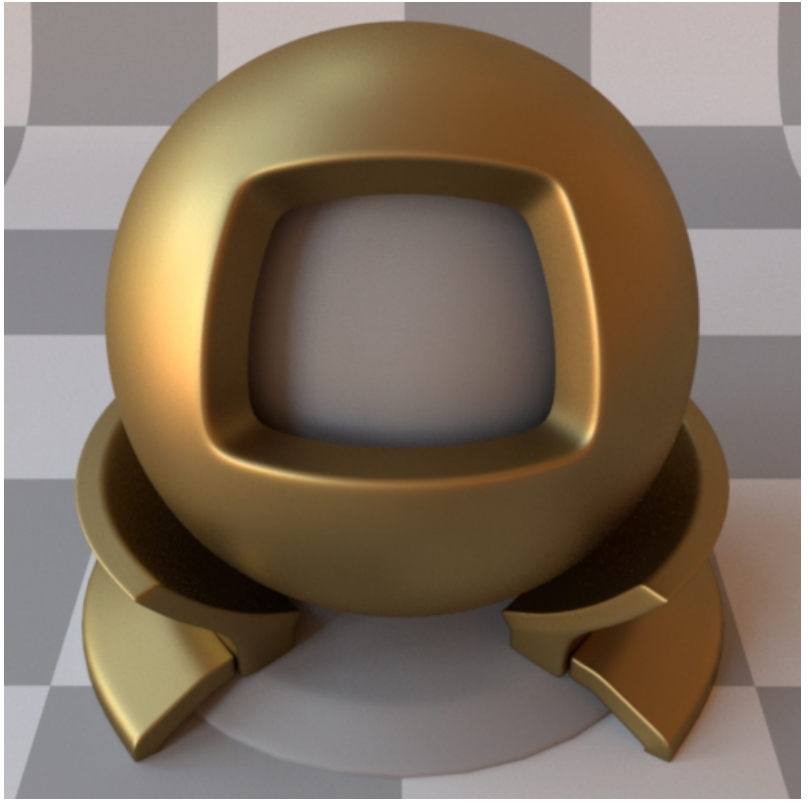
- Fully automatic
- Simple implementation
- Fast evaluation
- Works well in practice



# MERL gold-metallic-paint

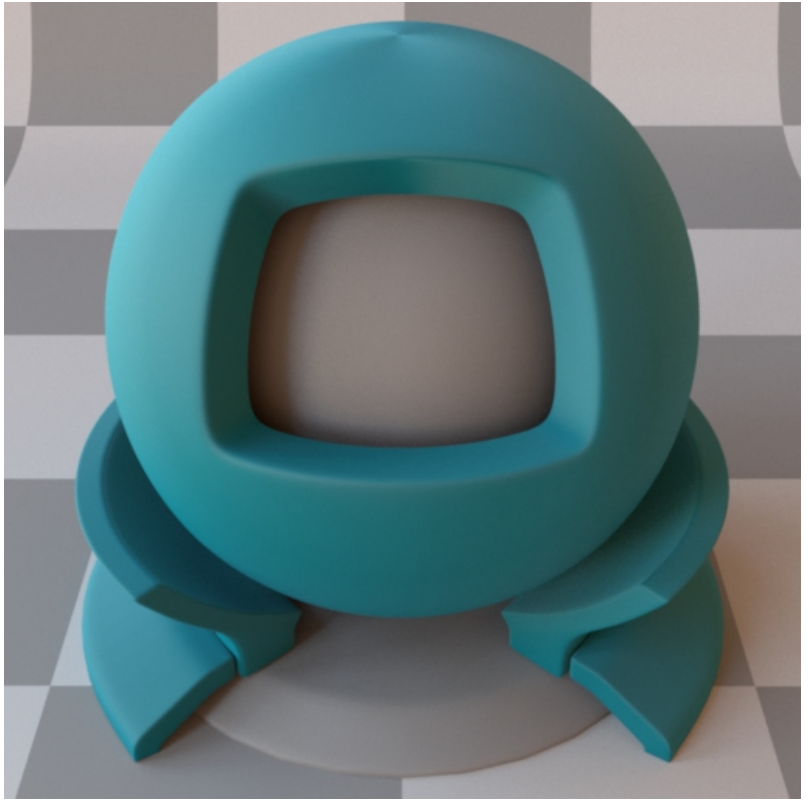


Acquired data

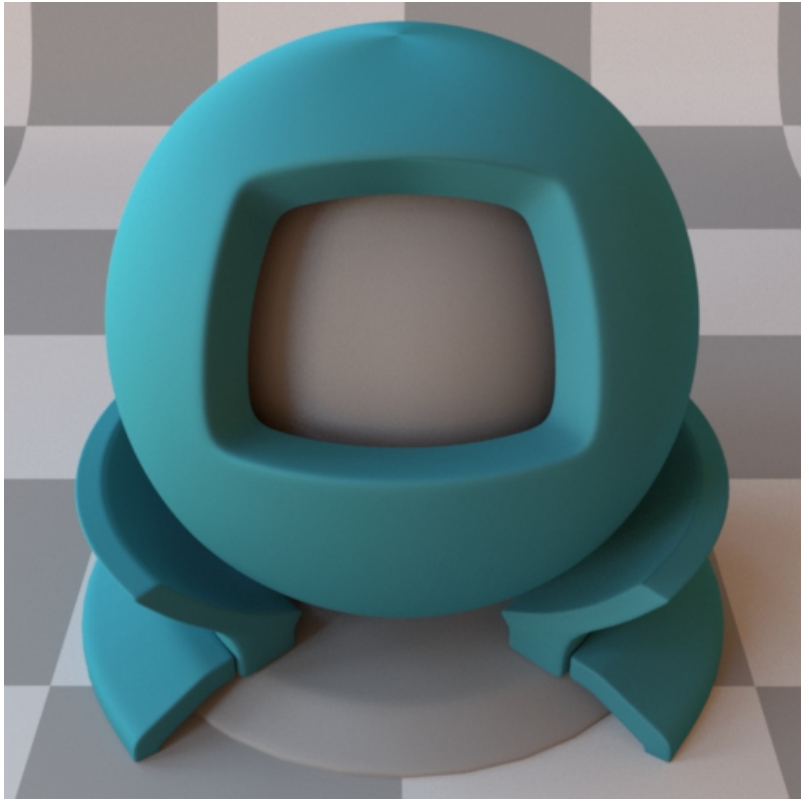


Ours: Tabulated

# UTIA m064-fabric099



Acquired data



Ours: Tabulated

# Parametric Fits

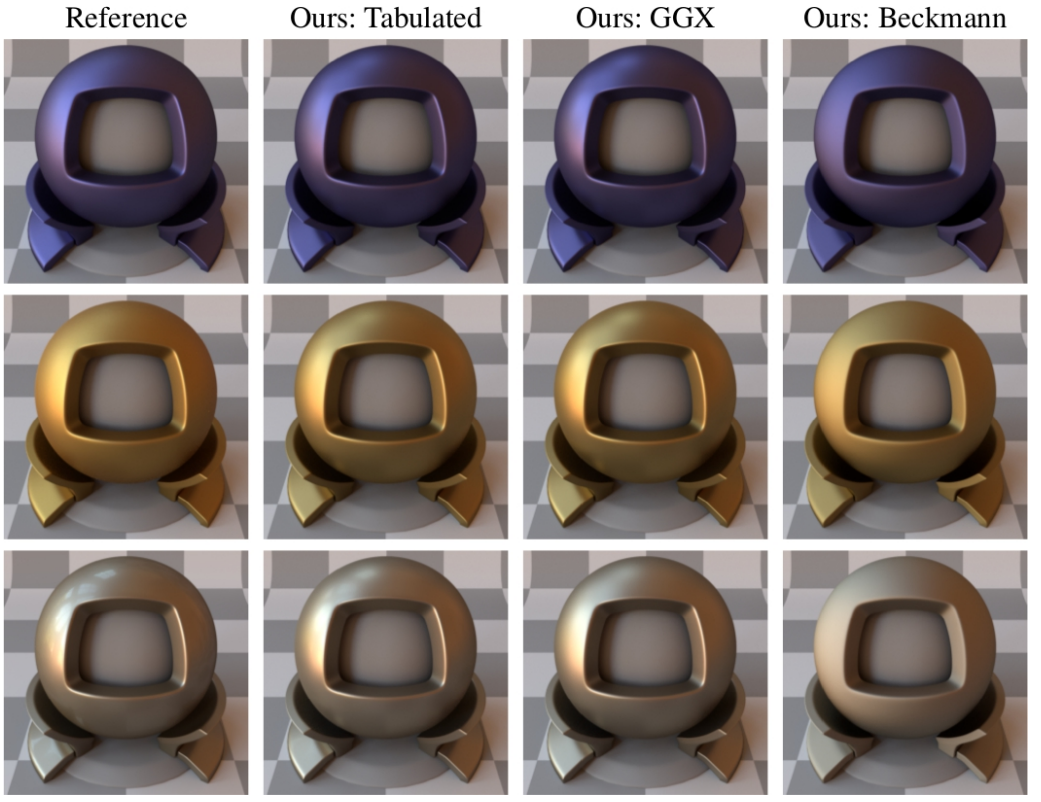
Beckmann roughness

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

GGX roughness

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

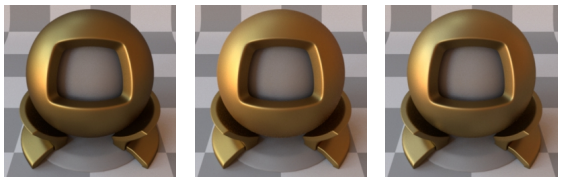
Our observation:
GGX > Beckmann



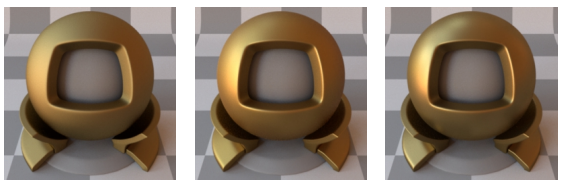
# Validation

**29 gold-metallic-paint**

Renderings

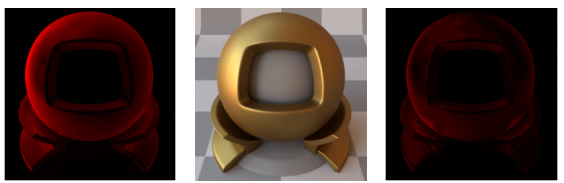


SGD | Reference | Ours: Tabulated



Ours: Gaussian | Reference | Ours: GGX

CIE dE00

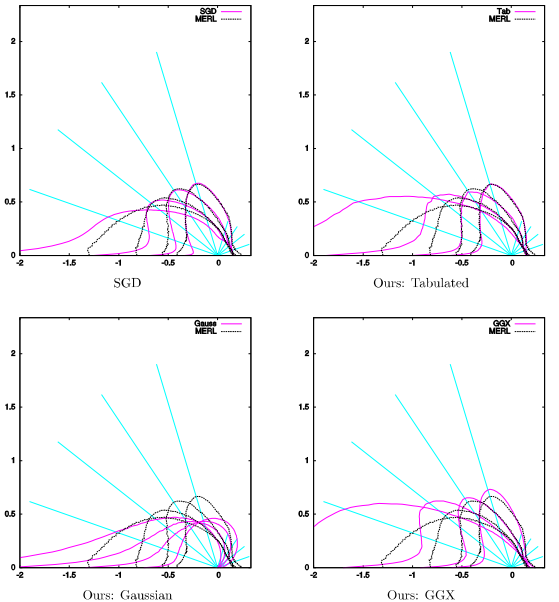


SGD | Reference | Ours: Tabulated

Ours: Gaussian | Reference | Ours: GGX



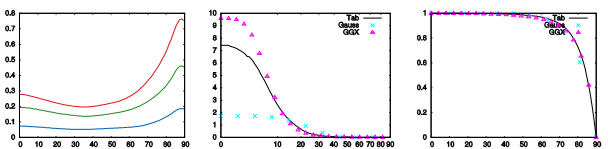
BRDF Lobes in the Incidence Plane (Cubic Root Applied)




SGD | Ours: Tabulated

Ours: Gaussian | Ours: GGX

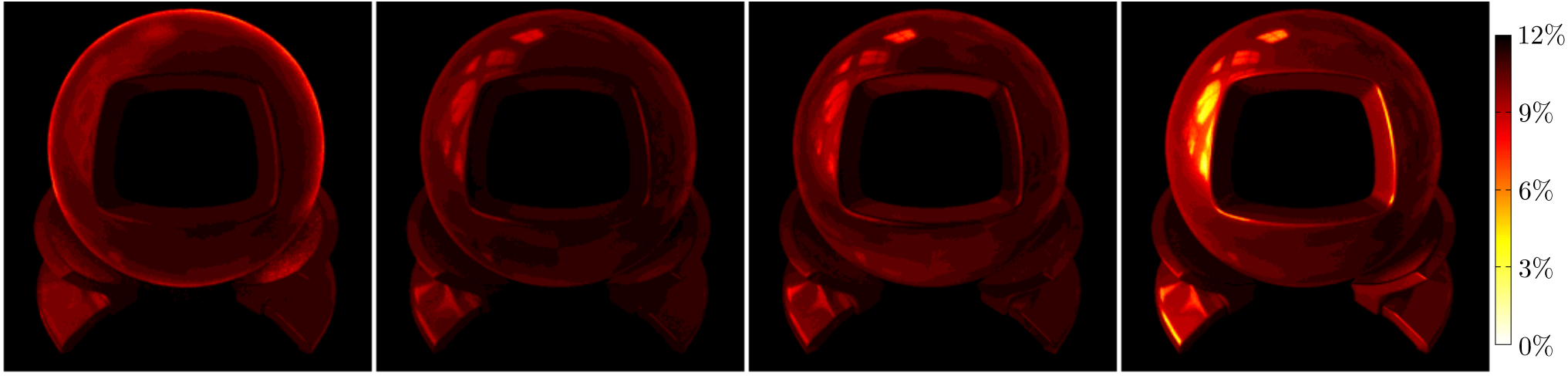
Fitted Microfacet Functions




Fresnel (R, G, B) | NDF ($D$) | Smith ($G_1$)

# Accuracy

Mean delta-E difference image on the MERL database




SGD [BSH12] | Ours: Tabulated | Ours: GGX | Ours: Beckmann

# 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

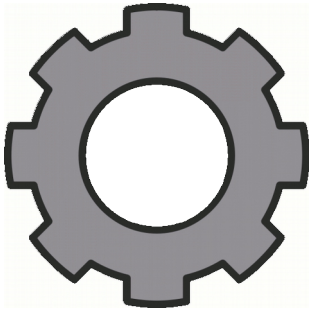
Novel fitting algorithm

- Robust tabulation
- Robust parametric fits

Key properties

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

```
void tabular::compute_p22_smith(const brdf& brdf, int res)
{
    int cnt = res - 1;
    double dtheta = sqrt(M_PI * 0.5) / (double)cnt;
    matrix km(cnt);

    for (int i = 0; i < cnt; ++i) {
        double tmp = (double)i / (double)cnt;
        double theta = tmp * sqrt(M_PI * 0.5);
        double theta_o = theta * theta;
        double cos_theta_o = cos(theta_o);
        double tan_theta_o = tan(theta_o);
        vec3 fr = brdf.eval(dir(theta_o, 0.0), dir(theta_o, 0.0));
        double fr_i = fr.intensity();
        double kji_tmp = (dtheta * pow(cos_theta_o, 6.0)) * (8.0 * fr_i);

        for (int j = 0; j < cnt; ++j) {
            const double dphi_h = M_PI / 180.0;
            double tmp = (double)j / (double)cnt;
            double theta = tmp * sqrt(M_PI * 0.5);
            double theta_h = theta * theta;
            double cos_theta_h = cos(theta_h);
            double tan_theta_h = tan(theta_h);
            double tan_product = tan_theta_h * tan_theta_o;
            double nint = 0.0;

            for (double phi_h = 0.0; phi_h < 2.0 * M_PI; phi_h += dphi_h)
                nint += max(1.0, tan_product * cos(phi_h));
            nint *= dphi_h;

            km(j, i) = theta * kji_tmp * nint * tan_theta_h
                     / (cos_theta_h * cos_theta_h);
        }
    }

    // compute slope pdf
    km.eigenvector(m_p22, 4);
    m_p22.push_back(0);
}
```



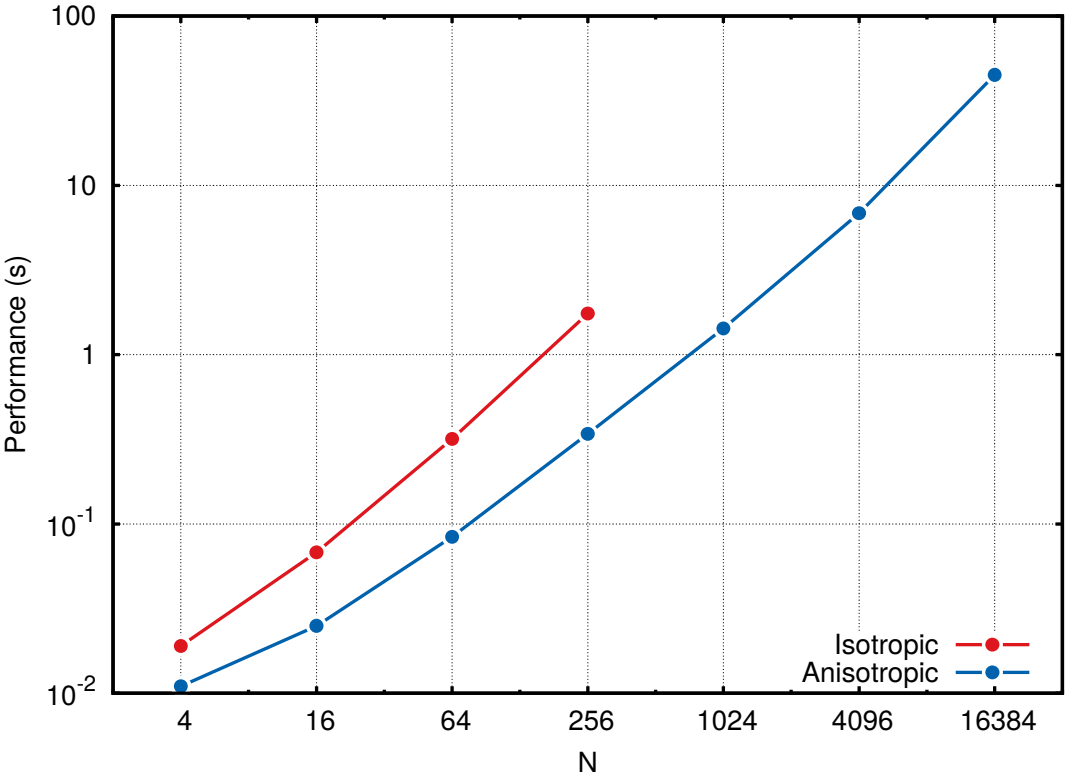
# Contribution

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

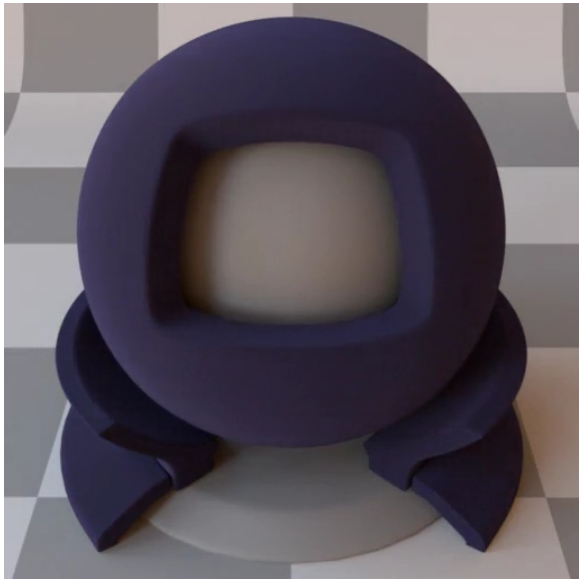
# Artistic Control

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

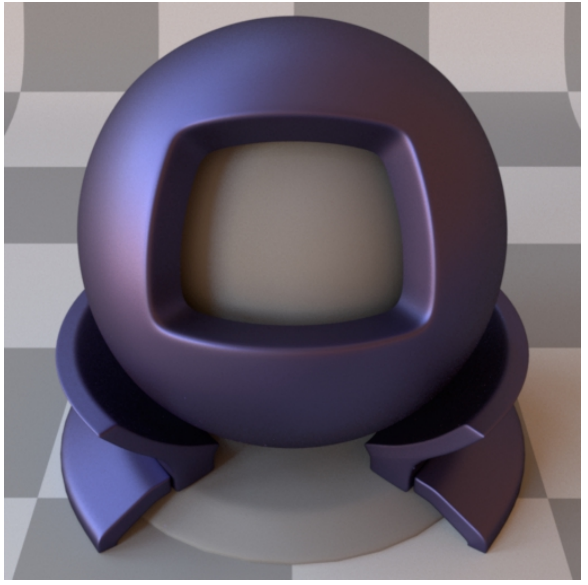
Tabulate the slope PDF

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

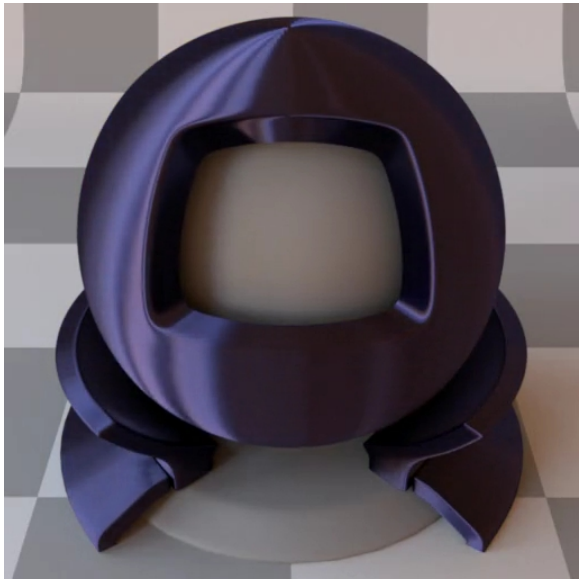
# Editing blue-metallic-paint



Isotropic stretch



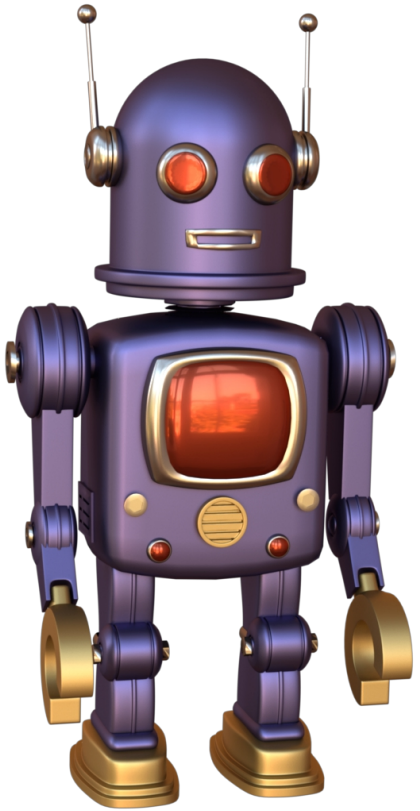
Acquired data



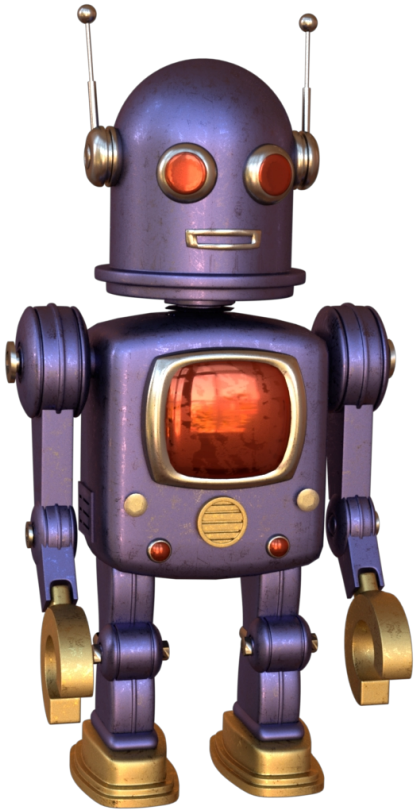
Anisotropic stretch

# Roughness Textures

+



=



Robot model courtesy of LAGOA

# Conclusion

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

SGD [BSH12] Acquired data Ours: Tabulated

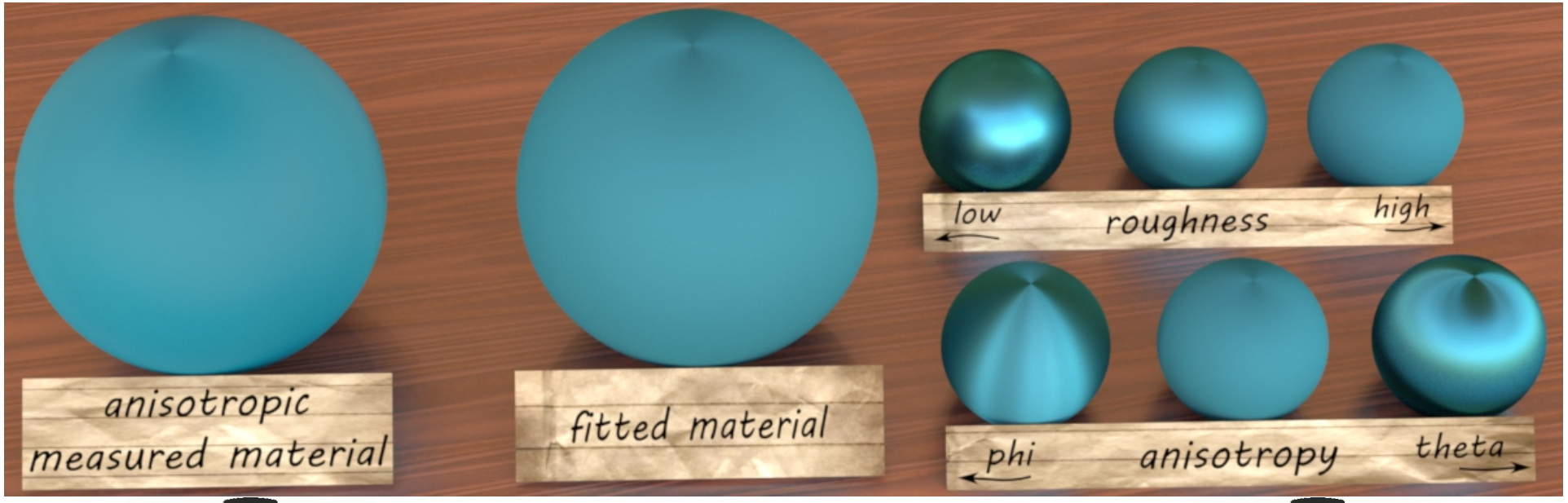
# Acknowledgments

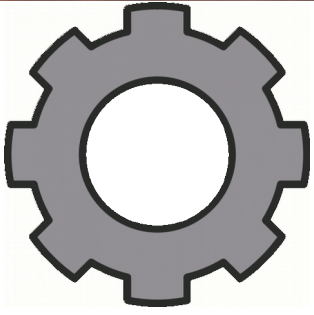

- **Laurent Pujo-Menjouet** and **Ionel Ciuperca** for their input on integral equations

- **Thiago Da Costa** and **Arno Zinke** for providing us with the LAGOA robot asset

- **Brent Burley** for discussing BRDF issues in production

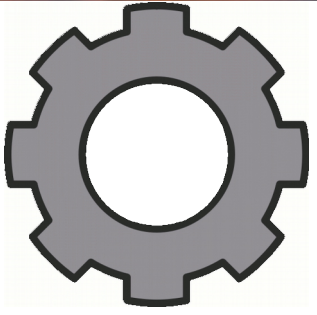
# Questions



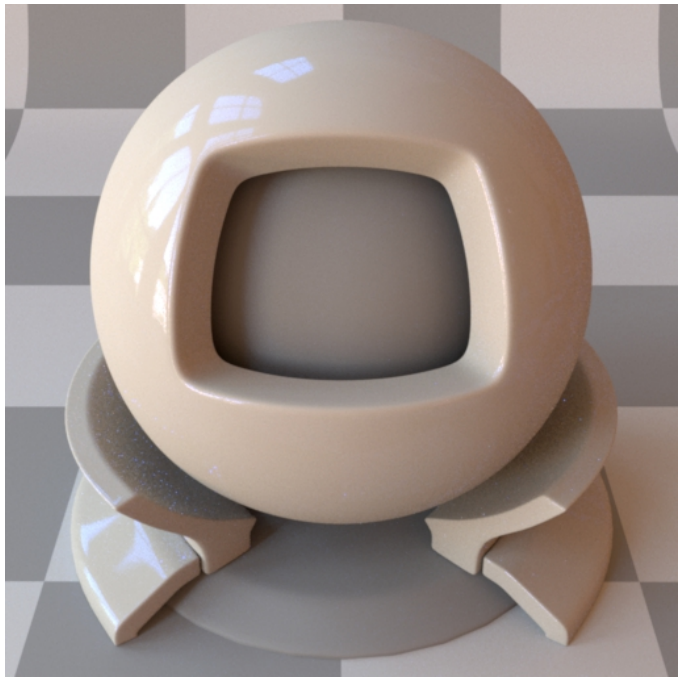



*Get our source code online:*

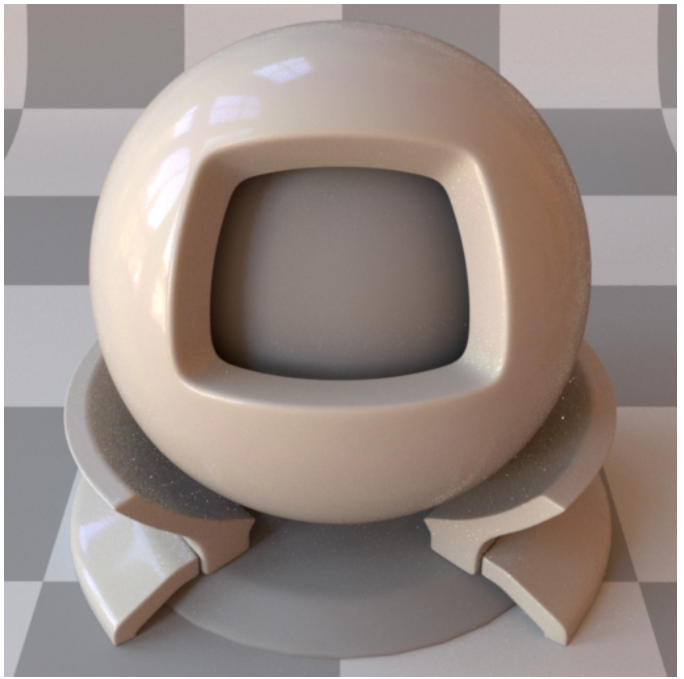
http://github.com/jdupuy/dj_brdf



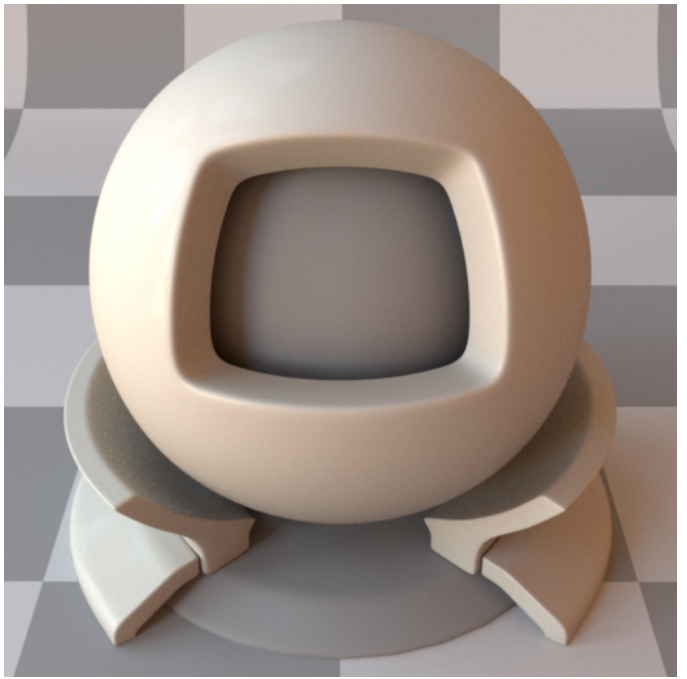
# Fail Case: Layers



SGD [BSH12]



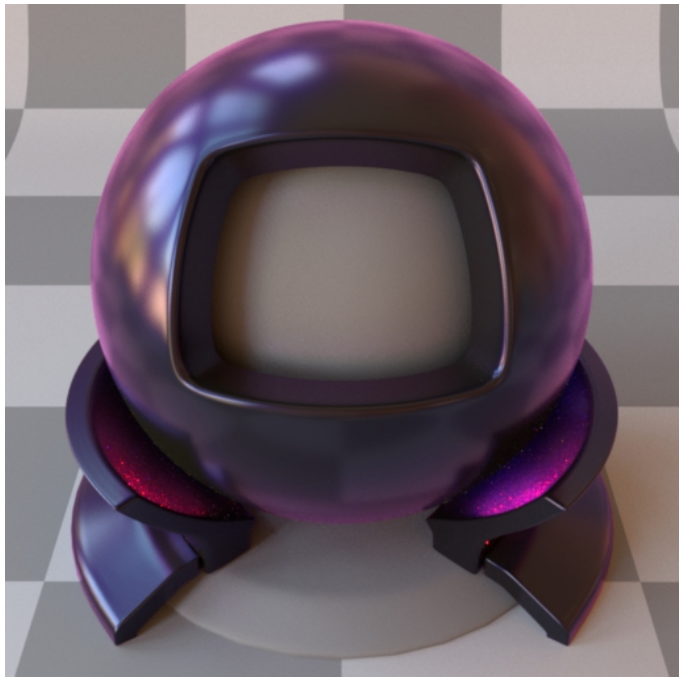
Acquired data



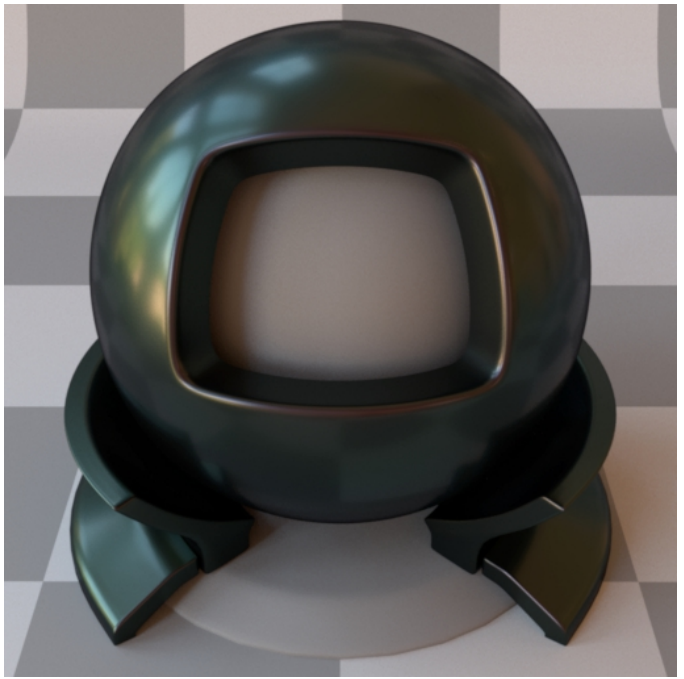
Ours: Tabulated

(MERL alumnia-oxide)

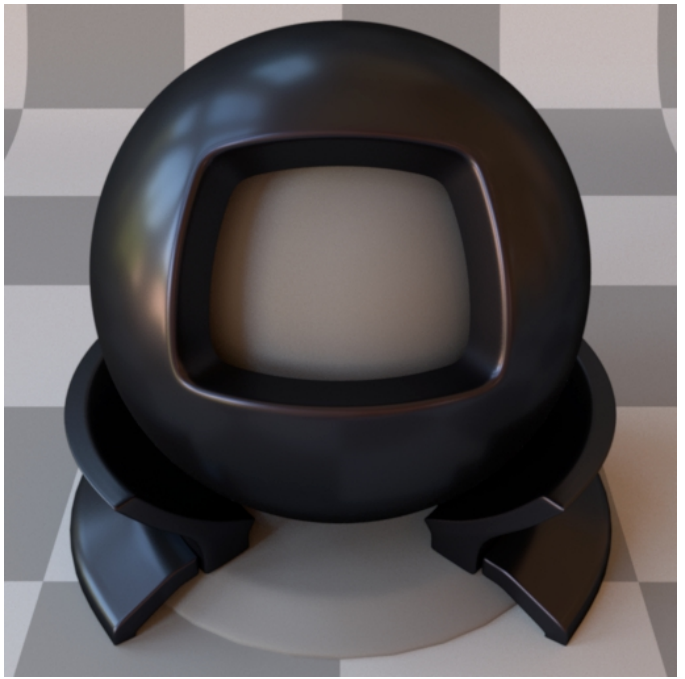
# Fail Case: Color Changing



SGD [BSH12]



Acquired data



Ours: Tabulated

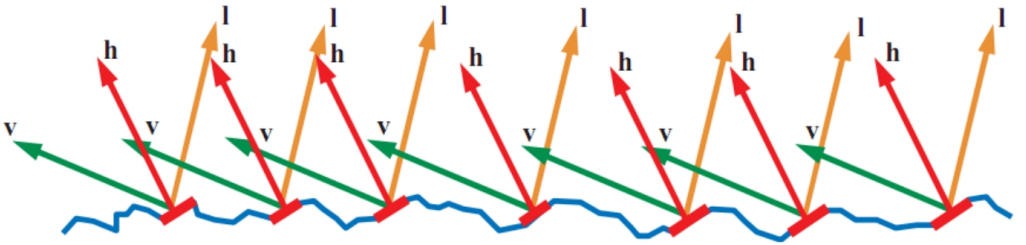
(MERL color-changing-paint1)

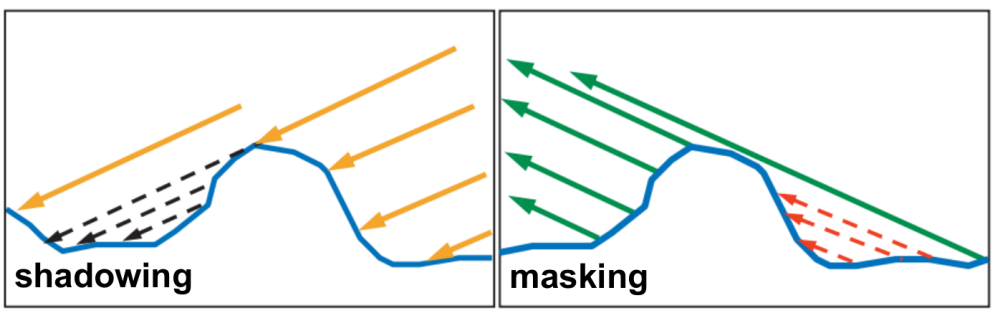
# Microfacet BRDFs

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

Modular components

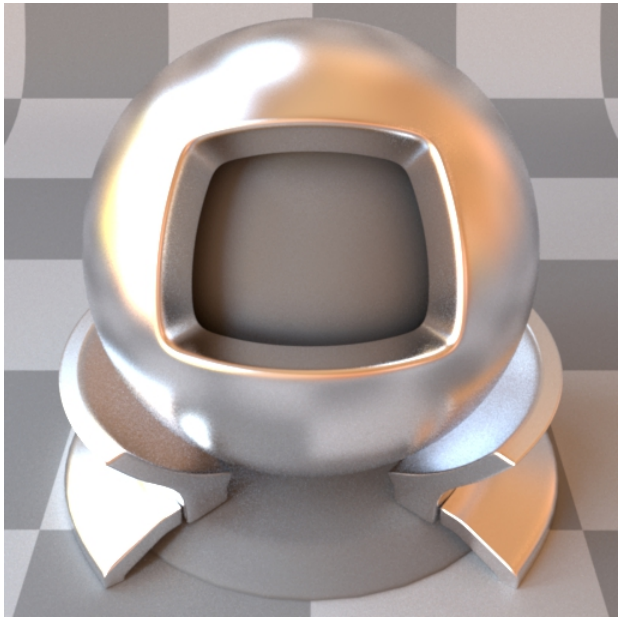
- Fresnel term $F$
- Distribution of normals $D$
- Roughness $\alpha$
- Geometric factor $G$



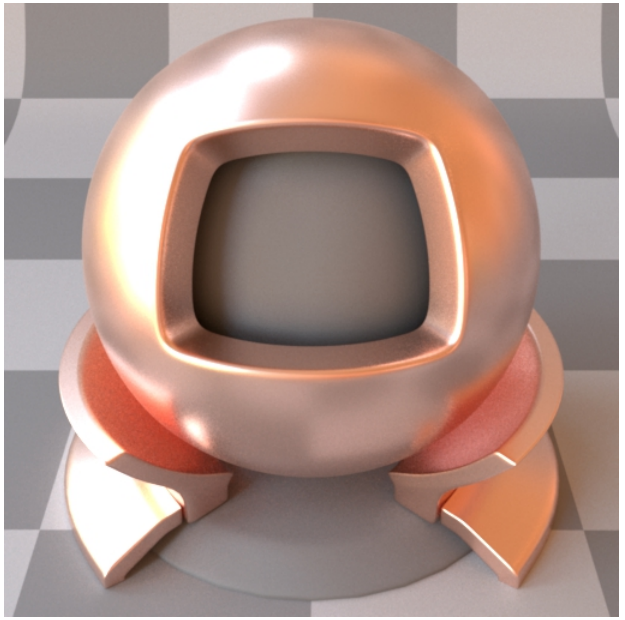




Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

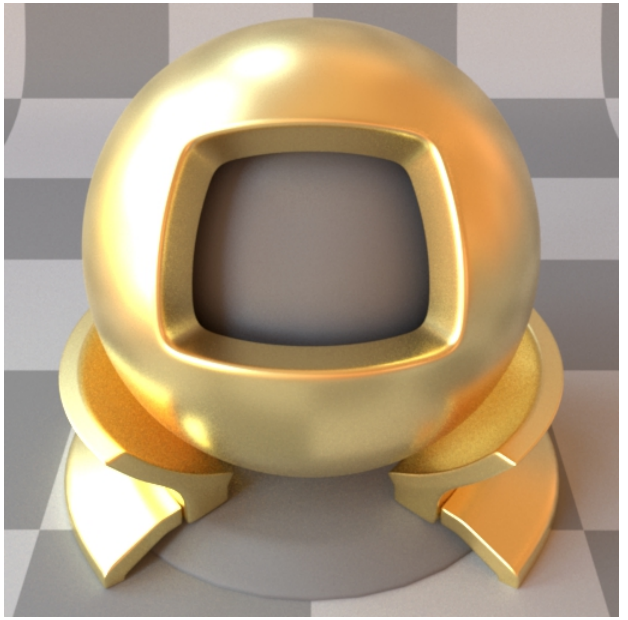
# Fresnel



Aluminum

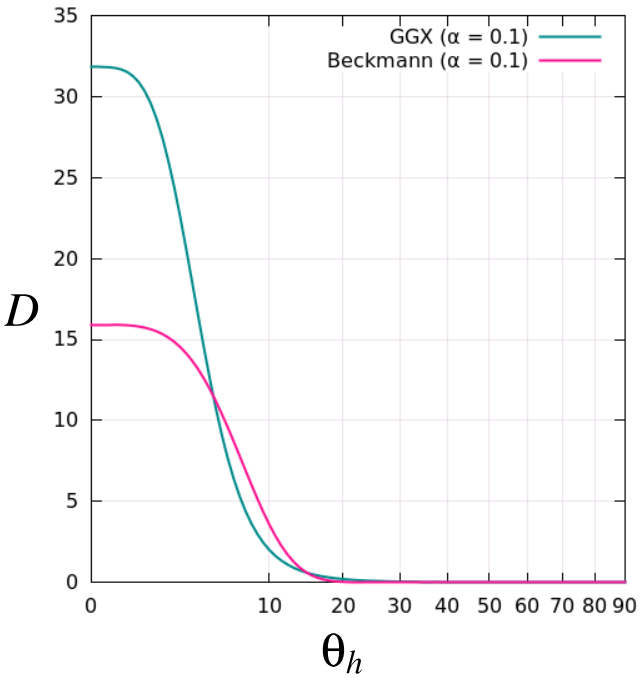


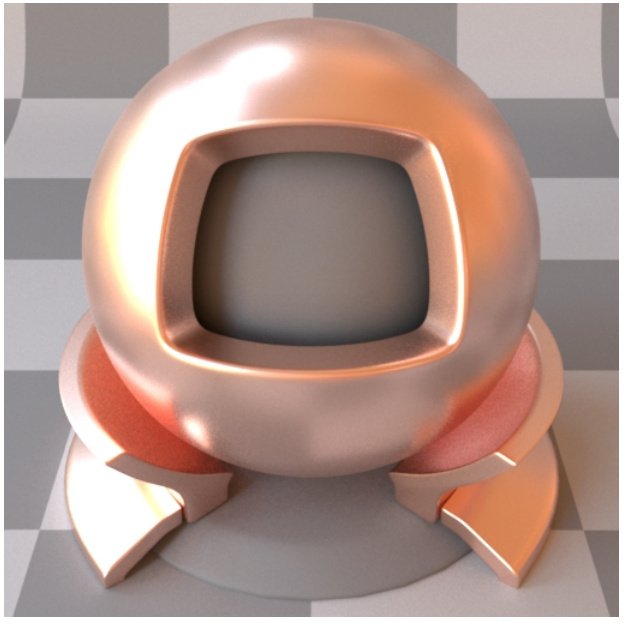
Copper



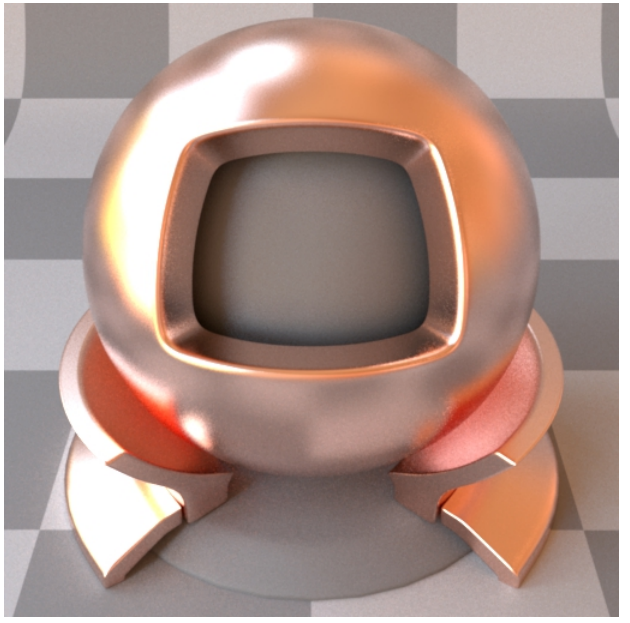
Gold

# Distribution of Normals



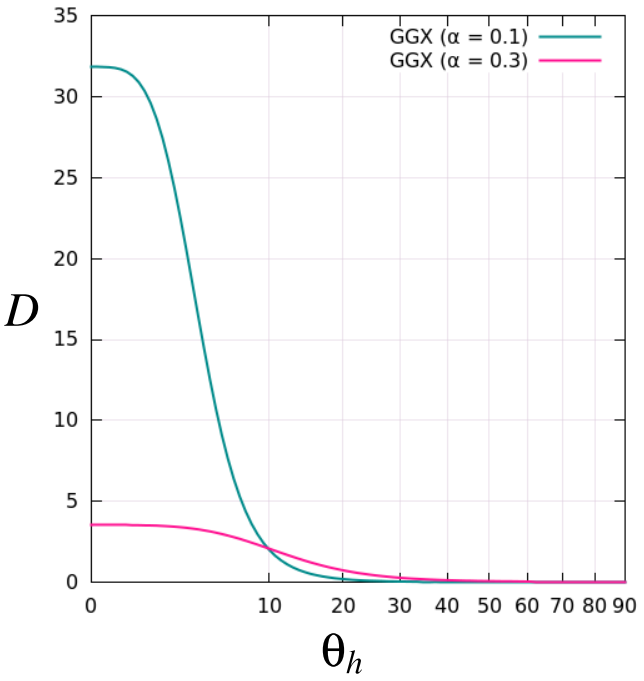



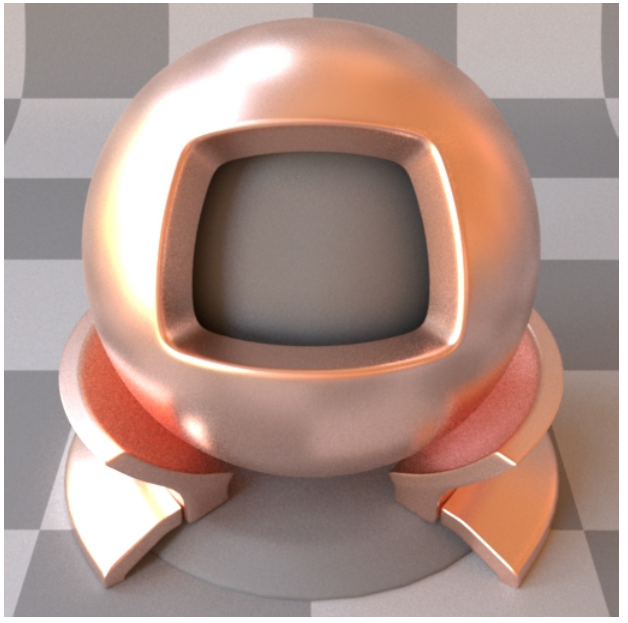
GGX



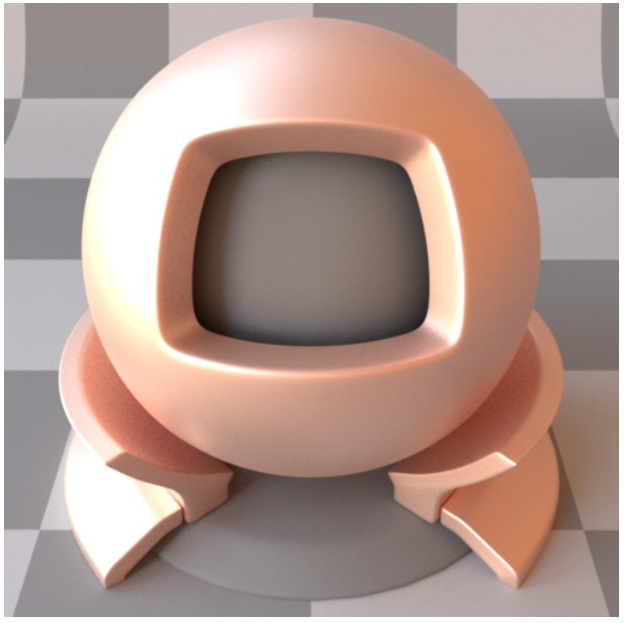
Beckmann

# Roughness



$\alpha = 0.1$



$\alpha = 0.3$

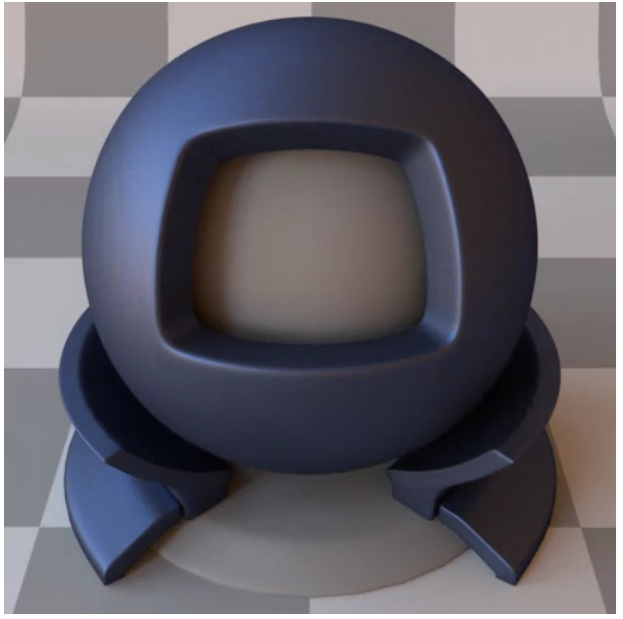
# Geometric Factor

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

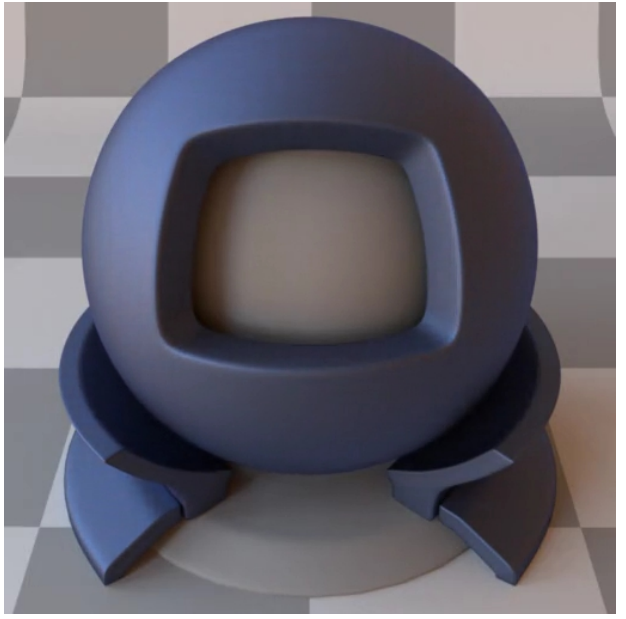
$$G_1(\mathbf{h},\mathbf{k}) = \min\left(1, 2\frac{\cos\theta_h \cos\theta_k}{\mathbf{kh}}\right)$$

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

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



Smith



V Cavities

# Inverting the BRDF

input (4D) output

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

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

# Too difficult :(

# Inverting the Smith Model

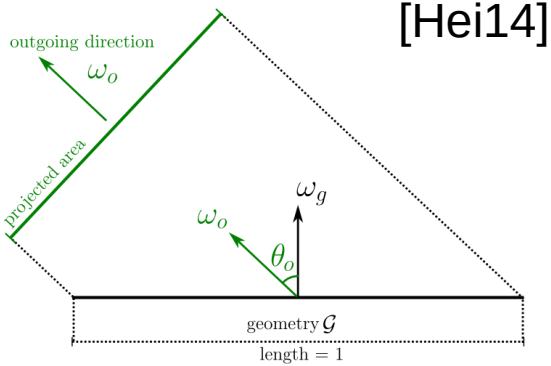
input output

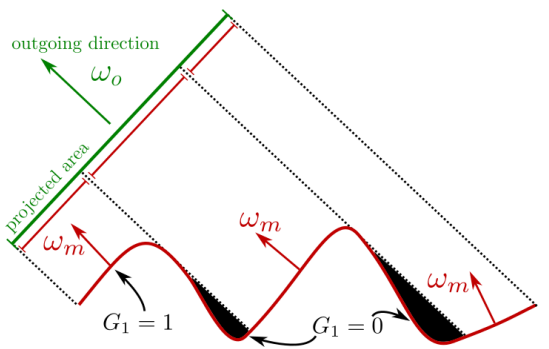
$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos\theta_i \cdot \cos\theta_o}$$

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

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

[Hei14]




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