RENDERING SPECULAR EFFECTS

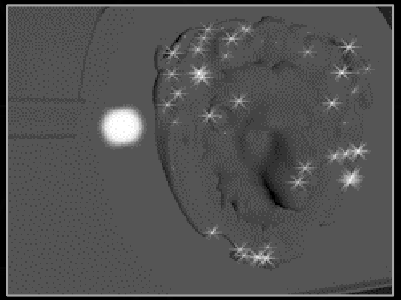
Anton S. Kaplanyan

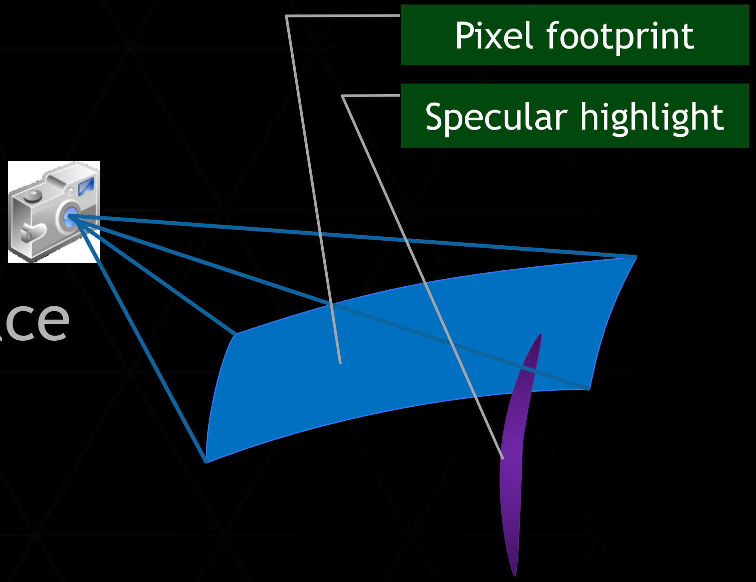
1 August 2016

NVIDIA

SPECULAR ALIASING

- Specular highlight can be small
  - Gets brighter when gets smaller
  - Specular microdetails sparkle
- Specular aliasing
  - Emphasized on curved geometry
  - Does not go away even after many samples
- Pixel footprint spans a large area on the surface
  - Stretched on curved surfaces





RENDERING SPECULAR

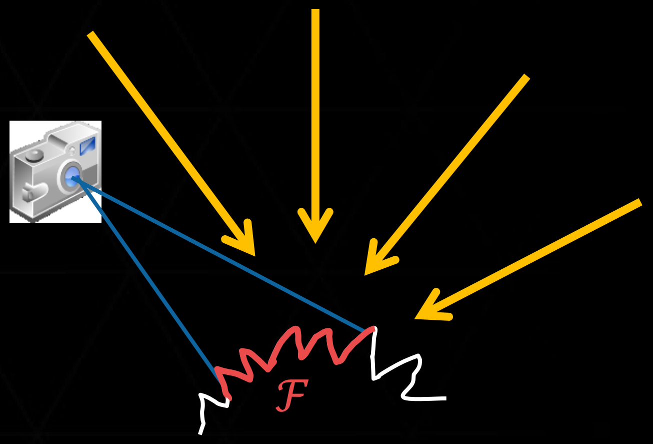
- Rendering equation: scattering at surface point x

$$L_o(\mathrm{x}) = \int_\Omega f_r(\mathrm{x}, \omega_i, \omega_o) L_i(\mathrm{x})\, \mathrm{d}\omega_i^\perp$$

- Flux incident at the image pixel is

$$\mathrm{I_j} = \int_\mathcal{F} W_j\, L_o(\mathrm{x})\, G\, \mathrm{dx}$$

- Want to integrate over the pixel footprint $\mathcal{F}$



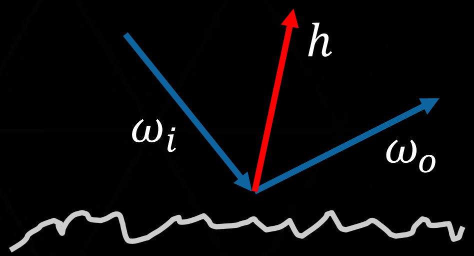
MICROFACET BSDF

- The BSDF $f_r(\mathrm{x}, \omega_i, \omega_o)$ is a scattering function

$$h = \frac{\omega_i + \omega_o}{||\omega_i + \omega_o||}$$

- Cook-Torrance microfacet BSDF is commonly used

$$f_r(\mathrm{x}, \omega_i, \omega_o) = \frac{G(\omega_i, \omega_o) F(\omega_i, \omega_o) D(h)}{4|\mathrm{n} \cdot \omega_i||\mathrm{n} \cdot \omega_o|}$$

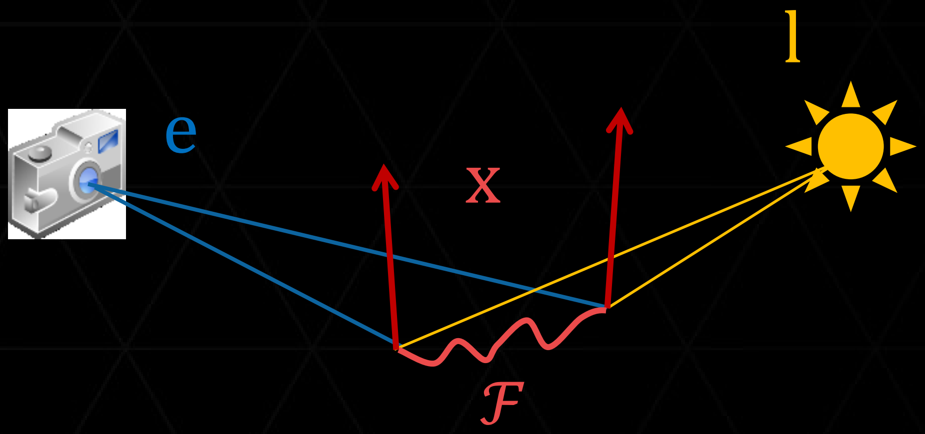


- Shadowing-masking and Fresnel (G and F) are [0;1] bounded
- The *Normal Distribution Function* (NDF)
  - Density of actively reflecting microfacets given h
  - Unbounded values!

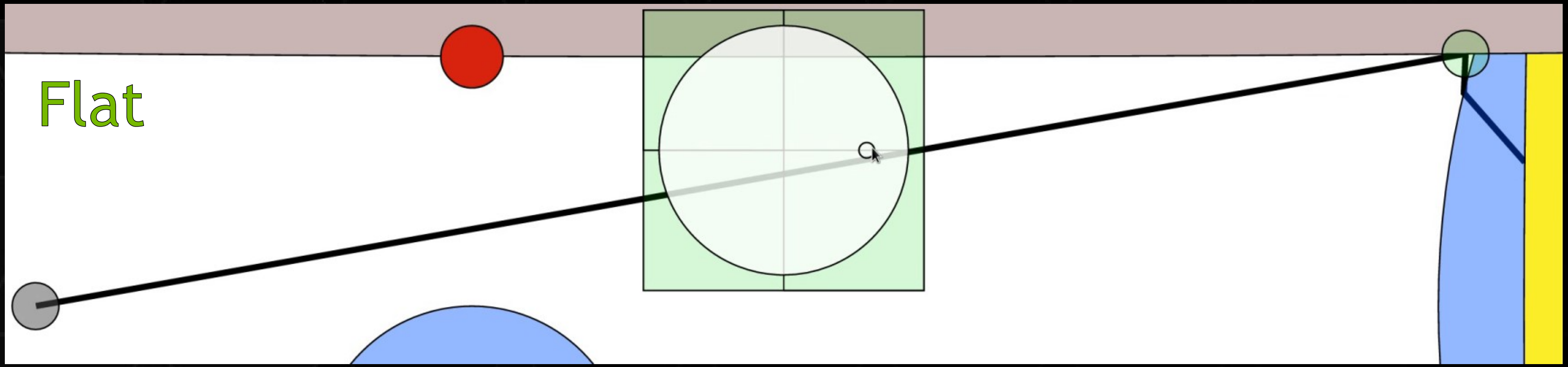
THREE-POINTS TRANSPORT

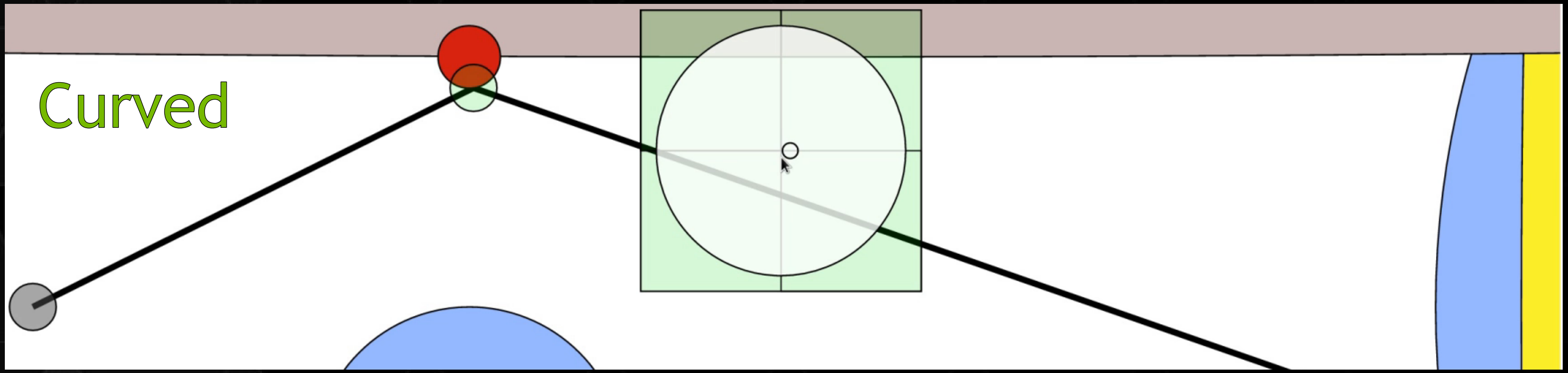
- The half vector depends on three adjacent vertices: $\mathrm{x}, \mathrm{l}, \mathrm{e}$
- The three-point transport with integration of NDF is

$$\mathrm{I_j} \approx C_2 \int_{\mathcal{F}} D\big(h(\mathrm{x}, \mathrm{l}, \mathrm{e})\big)\mathrm{dx}$$



VARIATION OF HALF VECTOR



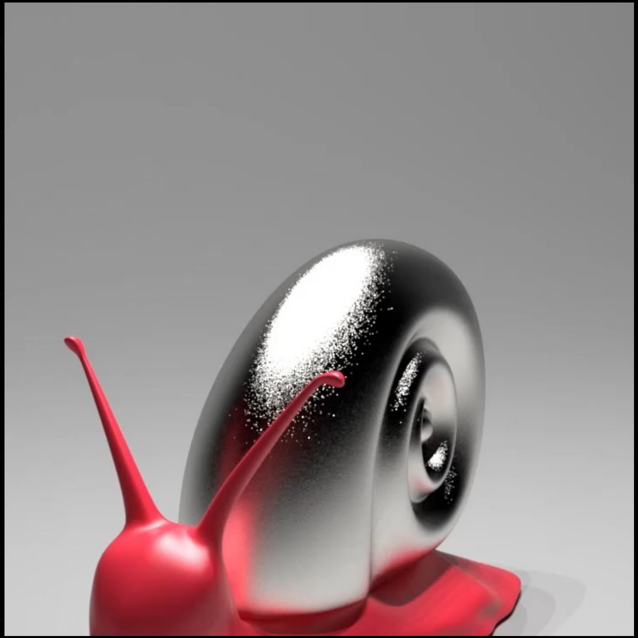


Materials with Specular Microdetails

Joint work with Tobias Zirr

CONCEPT OF A MICRODETAIL

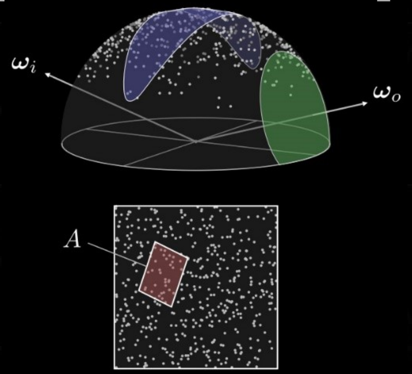
- Correlated clusters
  - Correlation both in NDF and on surface
  - Isotropic or anisotropic
  - Model with nested distributions
  - Glints, grooves in brushed metal, etc.



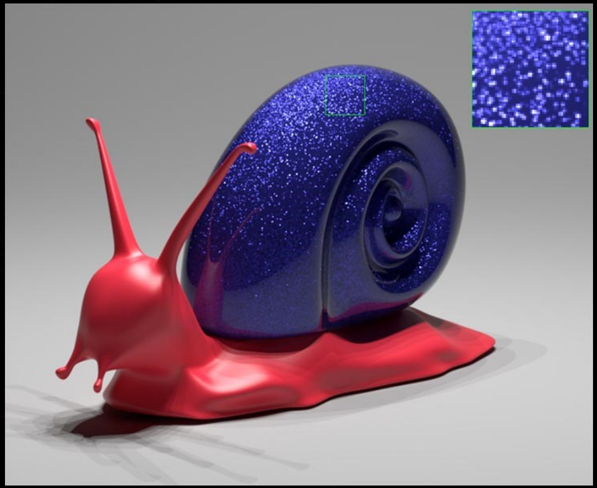
Results from [Yan et al.14]

PREVIOUS WORK

- *Discrete Stochastic Microfacet Models* [Jakob et al.14]
  - Hierarchical search
  - In spatial and half vector (slope) domain



- *Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces* [Yan et al.14]
  - Hierarchical pruning of a filtered micronormal map
  - In half vector domain, parallel light and eye rays

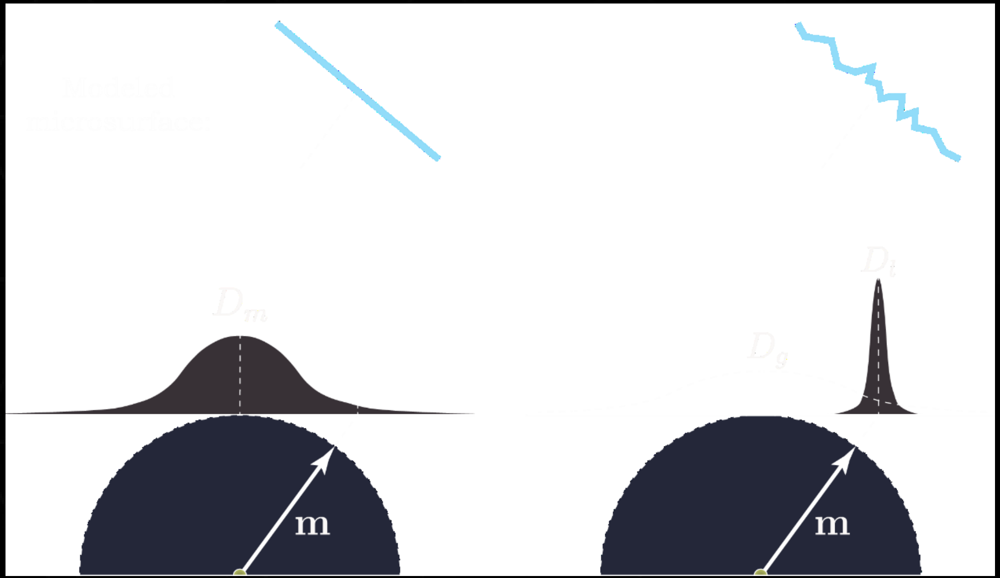


PREVIOUS WORK: REAL-TIME

- *Sparkly but not too Sparkly! A Stable and Robust Procedural Sparkle Effect*
  - SIGGRAPH AiRT’15, EGSR’16 (Studio Gobo)
  - 3D grid of sparkle kernels
    - Based on “Gettin’ procedural” [Shopf10]
  - Sparse sparkles & glints
- *Labs R&D: Rendering Techniques in Rise of the Tomb Raider*
  - SIGGRAPH AiRT’15 (Eidos Montreal)
  - Simple procedural noise for sparkles

BISCALE NDF MODEL

- Nested NDF distribution
  - D_m selects a slope of a single microdetail
  - D_l defines shape of a microdetail
- D_g is a resulting global NDF
  - Convolution of D_m and D_l



Select microdetail from D_m

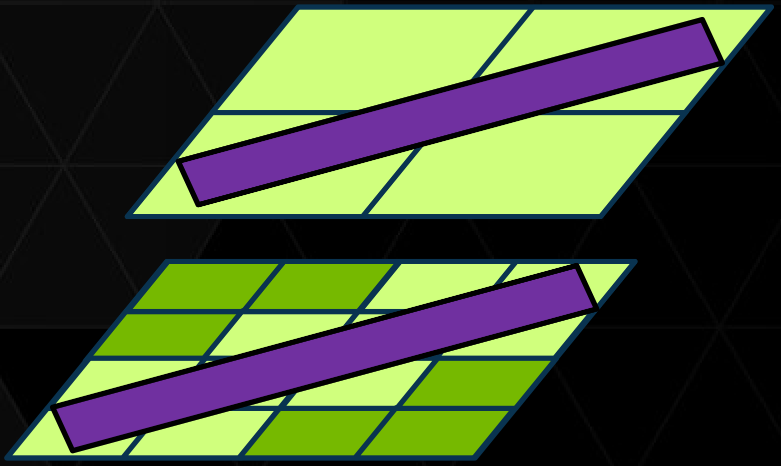
Select final slope from D_l

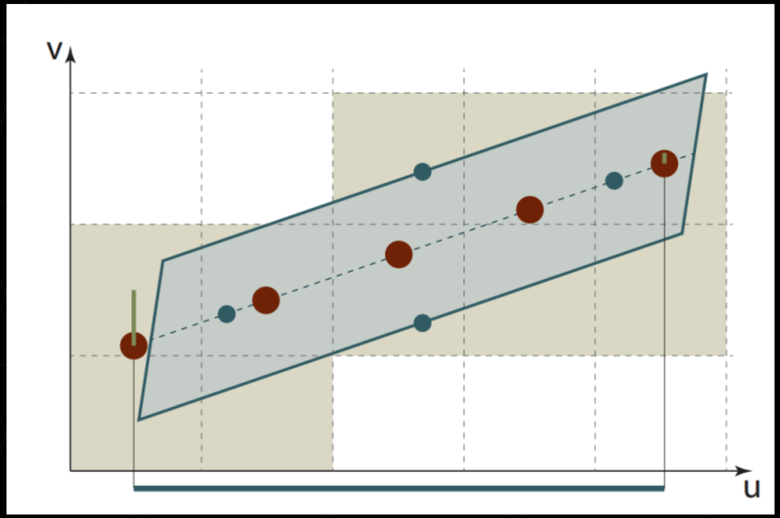
AUTHORING WITH BISCALE NDFS

- Powerful artistic control:
  - Local roughness D_l controls detail appearance
  - Global roughness D_g controls distant appearance

COHERENT STOCHASTIC PROCESS

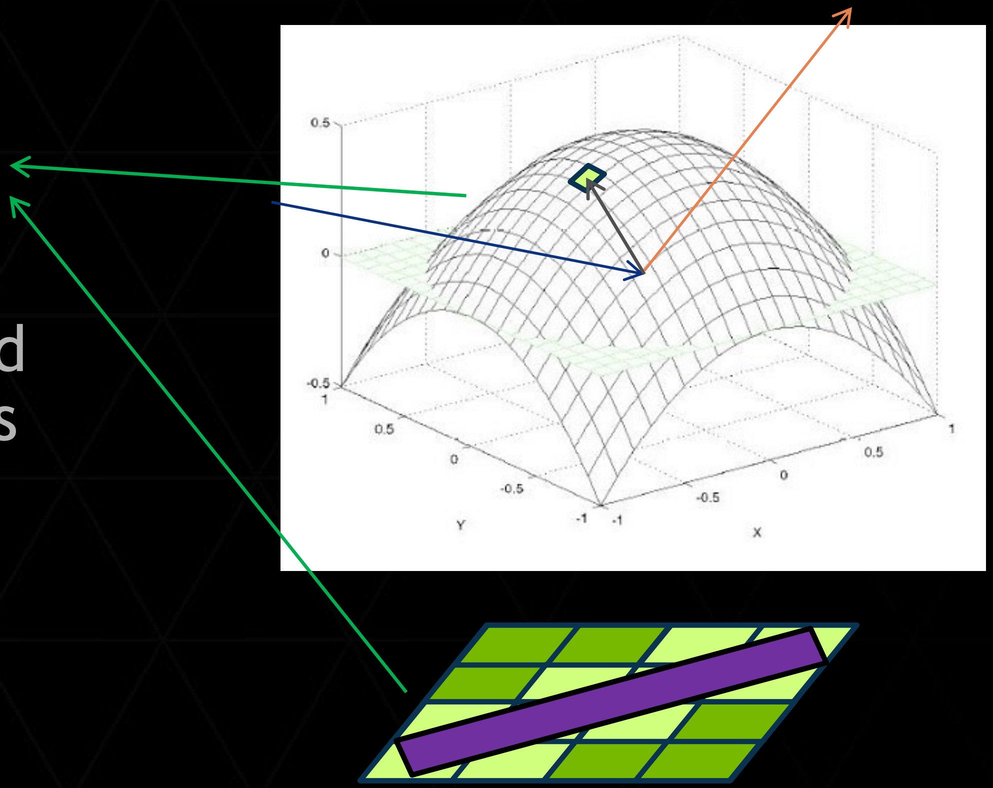
- ***Stable texture-space*** power-of-two grids and anisotropic filtering:
  - One binomial draw per grid cell
  - Trilinear interpolation





VIEW DEPENDENCY (SHIMMERING)

- Search space 4D: Also need subdivision of microdetail orientations
- Paraboloid half vector grid
- Seed binomial using a 4D index
- Perturb half vector partitioning using texture grid index to avoid simultaneous change of sparkles



PERFORMANCE

- GeForce GTX 980, 1080p
- Maximum anisotropy: 16x

Scene	Polys	Isotropic footprint, ms	Grazing angle, ms
Full-screen pass	2	0.9	2.9
Snow	32k	2.5	4.0
Dress	100k	1.4	4.4
Car (grooves)	570k	2.5	3.9
Crytek Sponza	262k	3.0	5.9

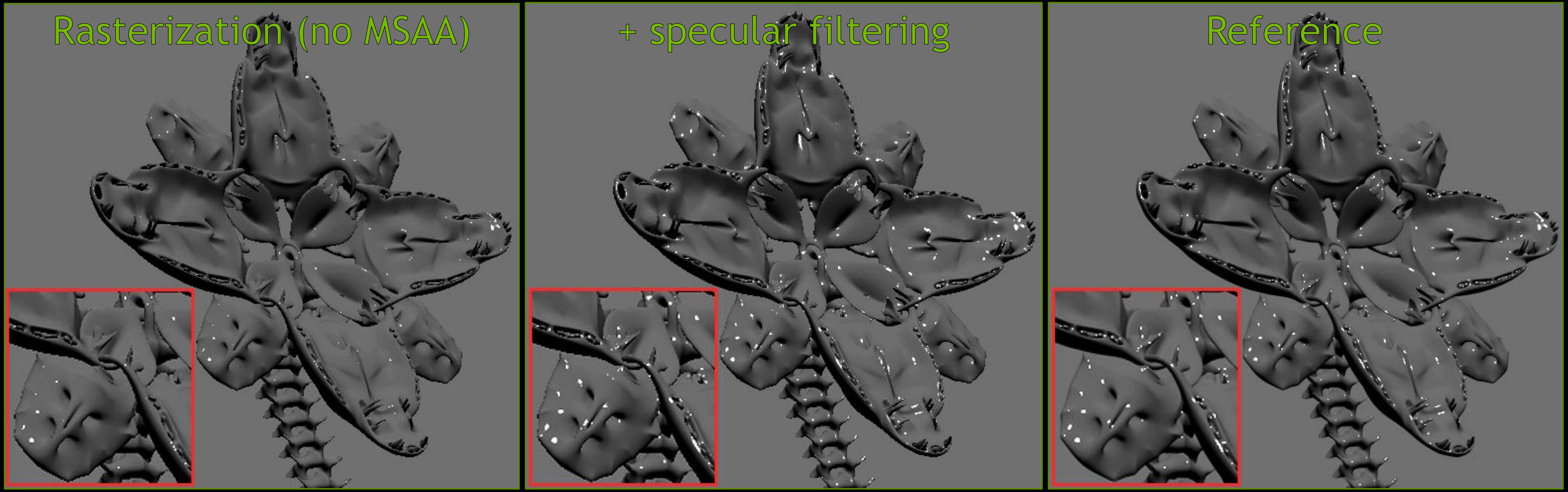
- ALU variance:
  - 8-64 cells to shade, 412 static instructions, 204 within a loop for one cell
  - No texture fetches

EXAMPLE CODE & RESULTS

- Example available online: https://www.shadertoy.com/view/ldVGRh

Specular Antialiasing

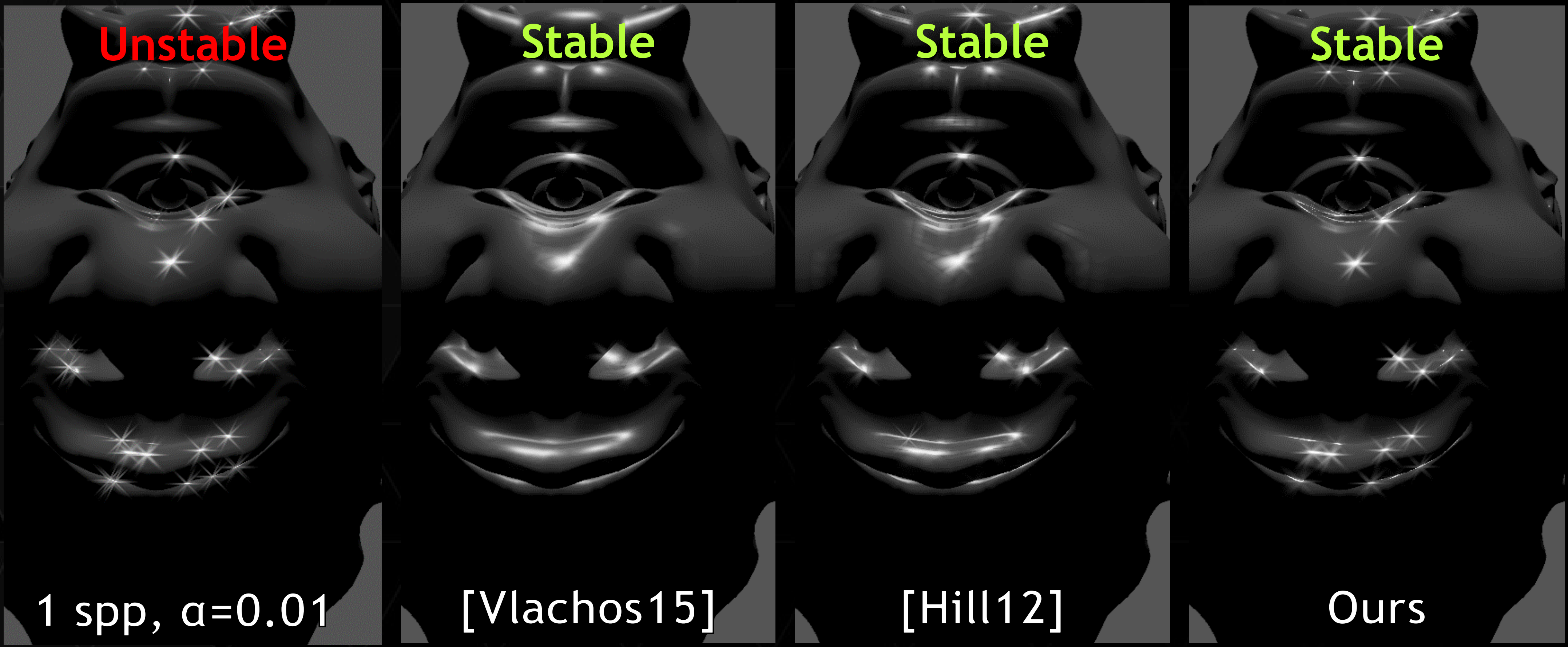
Joint work with Stephen Hill, Anjul Patney, and Aaron Lefohn



METHOD SUMMARY (SPOILERS!)

- Accurate and stable filtering of specular highlight on the pixel footprint
- Simple, real-time, robust
- Compatible with common real-time methods
  - Deferred shading
  - Normal maps and filtering thereof (e.g., LEAN/CLEAN, vMF)
  - Support for Beckmann and GGX NDF models
- Requires models with high-quality shading normal

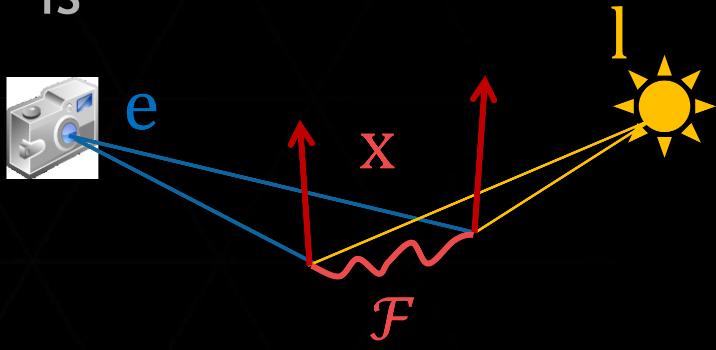
PREVIOUS WORK



THREE-POINTS TRANSPORT

- The half vector depends on three adjacent vertices: $\mathrm{x}, \mathrm{l}, \mathrm{e}$
- The three-point transport with integration of NDF is

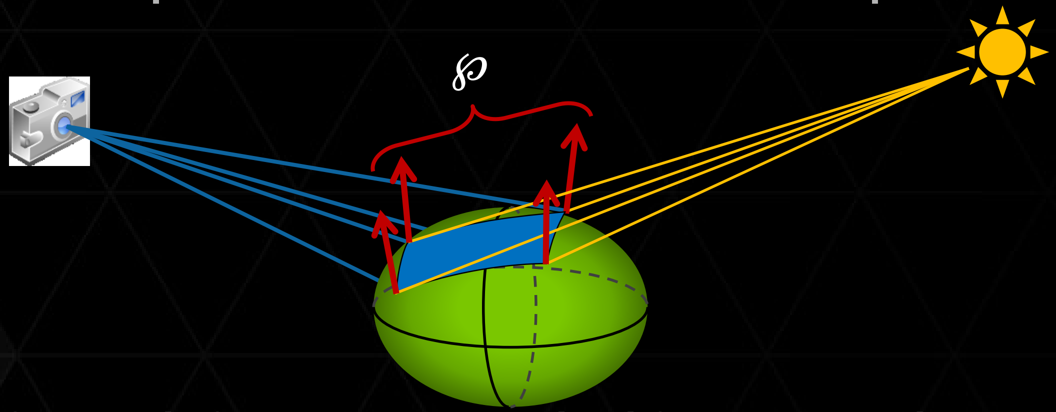
$$\mathrm{I_j} \approx C_2 \int_{\mathcal{F}} D\big(h(\mathrm{x}, \mathrm{l}, \mathrm{e})\big) d\mathrm{x}$$



- How does a half-vector h change w.r.t. $\mathrm{x}/\mathrm{l}/\mathrm{e}$?
  - Use first-order 2x2 derivative matrix M of h w.r.t. $\mathrm{x}/\mathrm{l}/\mathrm{e}$ [Jakob12]

TRANSFORMING THE PIXEL FOOTPRINT

- Obtain variation of slopes due to finite area of pixel footprint



- M is a mapping of pixel footprint to half-vector domain!
  - First transform ray differentials into Δx on surface
  - Then multiply by matrix M to get vectors Δh of $\wp$ in slope domain

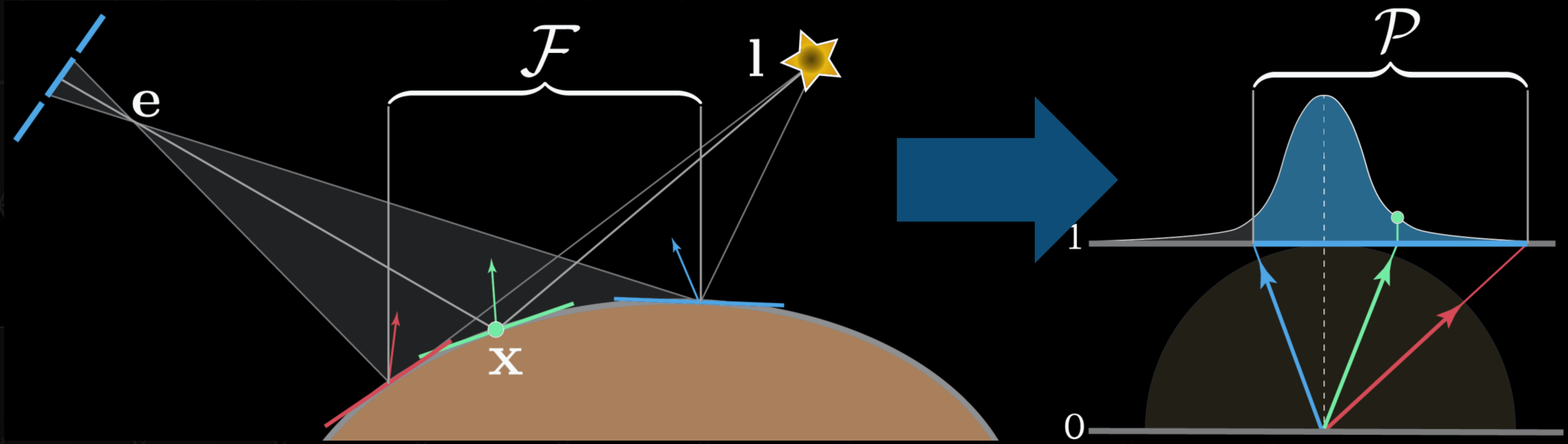
$$\wp \approx M(\mathcal{F})$$

- NDF filtering is then a 2D integration over the region $\wp$!

CHANGE OF DOMAIN

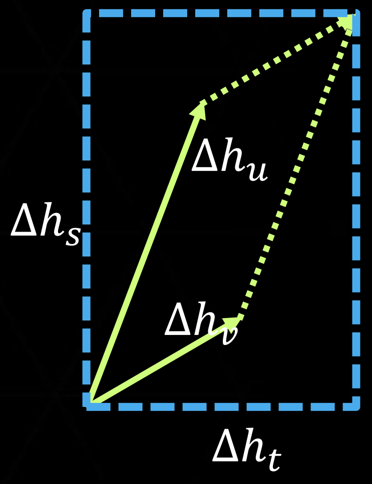
We can filter the NDF based on the pixel footprint $\mathcal{F}$

$$\int_{\mathcal{F}} D(h(\mathrm{x}))\mathrm{dx} = \int_{\wp} D(h) \left|\frac{\mathrm{d}h}{\mathrm{dx}}\right| \mathrm{d}h \approx \frac{|\mathcal{F}|}{|\wp|} \int_{\wp} D(h)\,\mathrm{d}h$$



PRACTICAL NDF FILTERING

- How to compute M?
- Benefit from quad shading on GPU!
  - Use **`ddx/ddy`** to obtain the final value with finite differencing
  - Matrix $\wp = \mathrm{d}h/\mathrm{d}uv = M(\mathcal{F})$ is first-order change of h induced by pixel footprint
  - Implicitly accounts for surface curvature with derivative of shading normal
- Robust temporal stability
  - Use an axis-aligned rectangle bounding the parallelogram
  - Aligned along s and t axes of the slope domain
  - Overfilters the NDF



ALGORITHM

Compute half-vector in slope domain

```
void shade()
{
...
      // Compute plane-plane half vector in local shading frame (hpp)
      vec3 hppWS = hWS / dot(hWS, shadingNormal);
      vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
```

ALGORITHM

Compute half-vector in slope domain → Compute its ddx/ddy derivatives

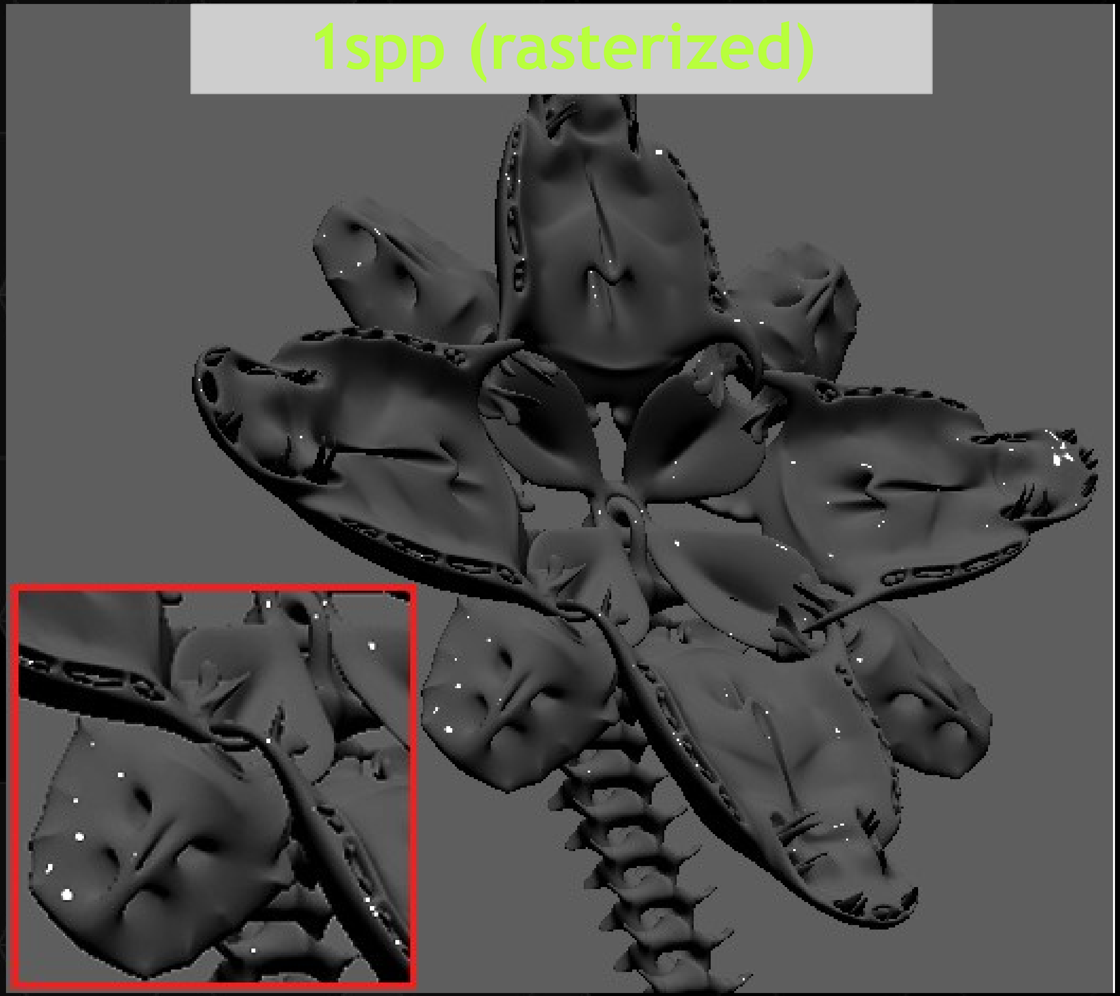
```
void shade()
{
...
      // Compute plane-plane half vector (hpp)
      vec3 hppWS = hWS / dot(hWS, shadingNormal);
      vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
      // Use ddx/ddy, thanks to quad shading!
      mat2 dhduv = mat2(dFdx(hpp), dFdy(hpp));
      // Compute filtering rectangular region
     vec2 rectFp = min((abs(dhduv[0]) + abs(dhduv[1])) * 0.5f, 0.7f);
```

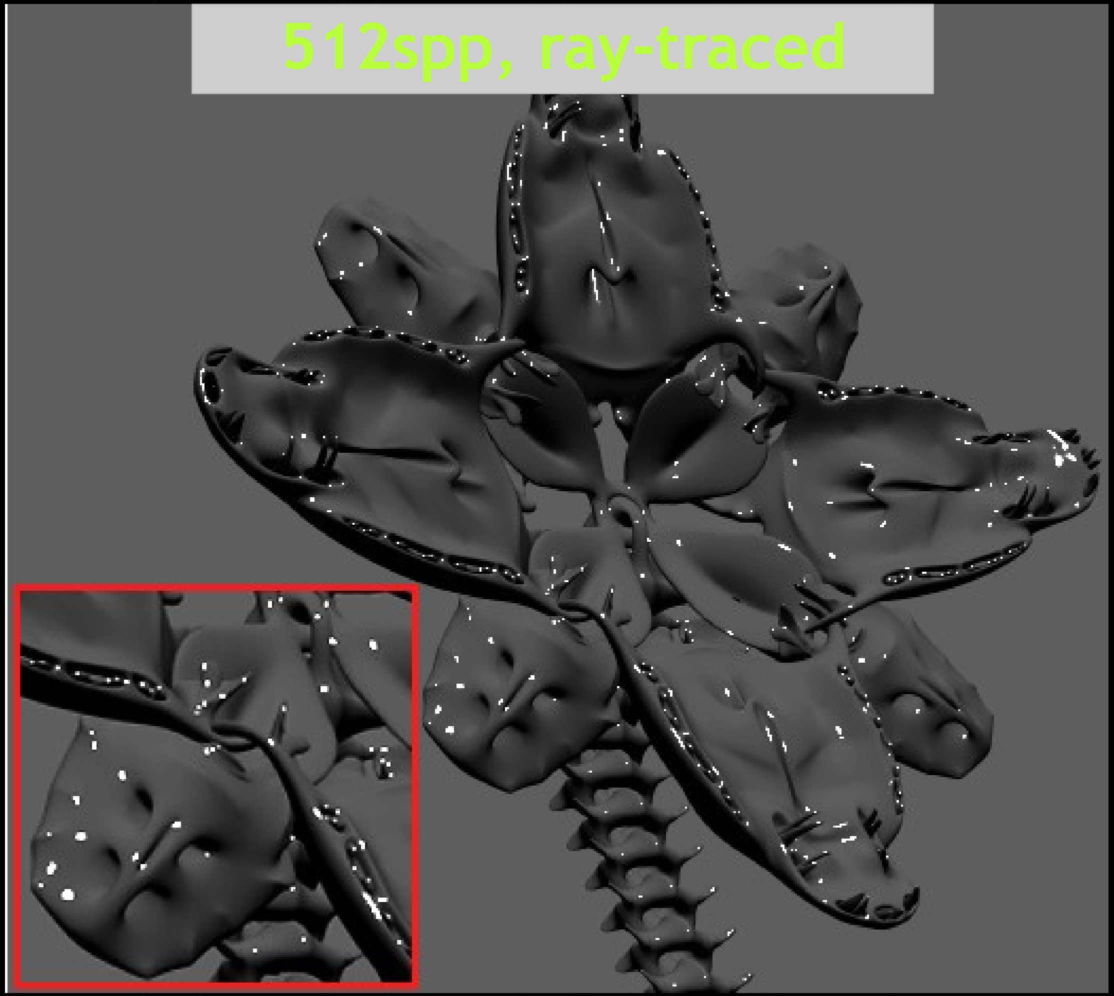
ALGORITHM

Compute half-vector in slope domain → Compute its ddx/ddy derivatives and rectangle → Integrate the NDF using resulting rectangle

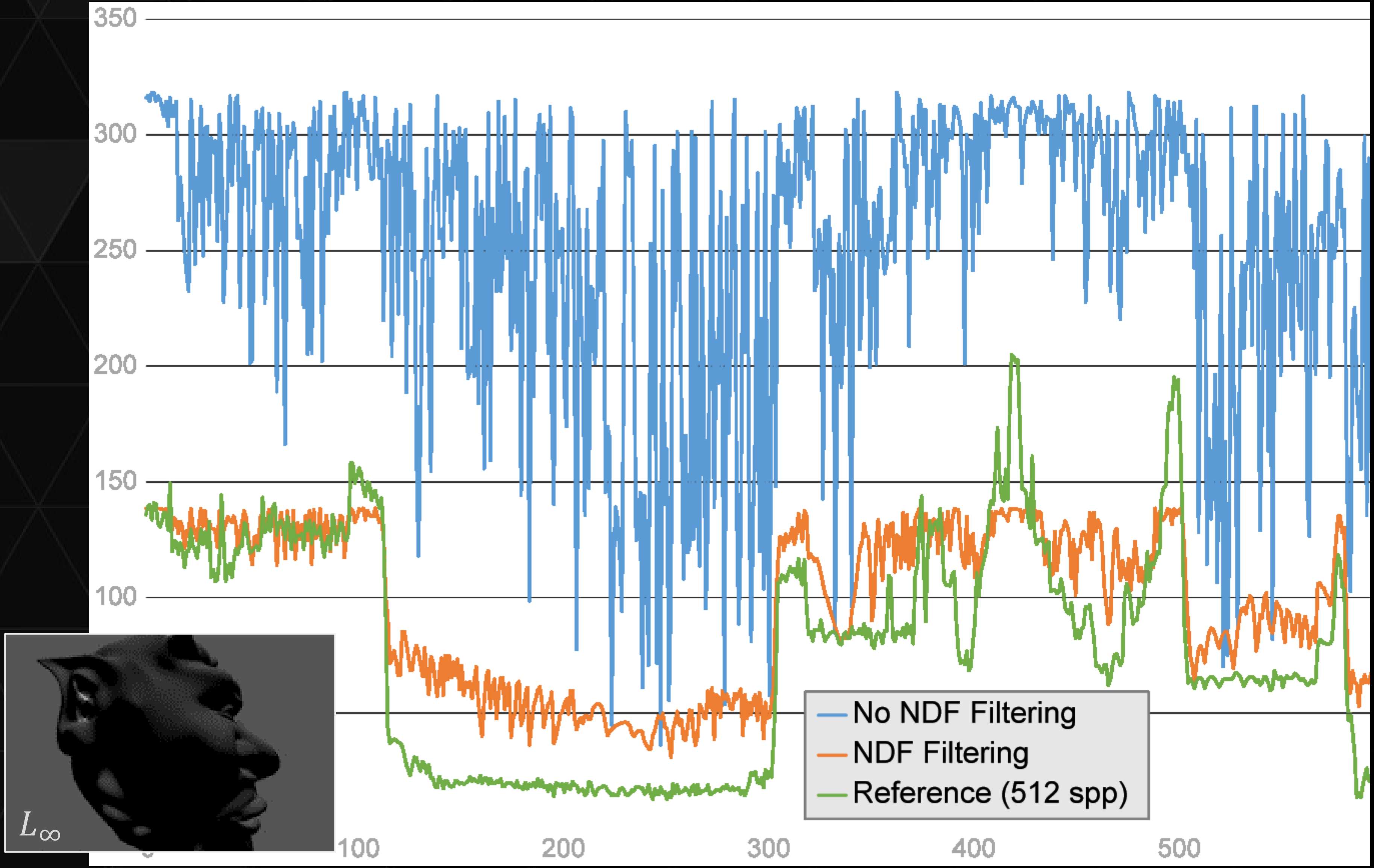
```
void shade()
{
...
      // Compute plane-plane half vector (hpp)
      vec3 hppWS = hWS / dot(hWS, shadingNormal);
      vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
      // Use ddx/ddy, thanks to quad shading!
      mat2 dhduv = mat2(dFdx(hpp), dFdy(hpp));
      // Compute filtering rectangular region
     vec2 rectFp = min((abs(dhduv[0]) + abs(dhduv[1])) * 0.5f, 0.7f);
     // Covariance matrix of pixel filter's Gaussian (remapped in roughness units)
     vec2 covMx = rectFp * rectFp * 2.f;
     roughness = sqrt(roughness*roughness + covMx);          // Beckmann proxy convolution (for GGX)
```

RESULTS





RESULTS


350
300
250
200
150
100
100
200
300
400
500
No NDF Filtering
NDF Filtering
Reference (512 spp)
L_∞

CONCLUSION

- NDF filtering for stable specular shading
  - Integrate highlight across pixel footprint on a local shading quadric
  - Preserve highlight energy, find difficult and small highlights
  - Compatible and orthogonal to other methods, simple and readily usable
- <u>Limitations</u>
- Addresses only *shading* aliasing
  - No improvements for geometric aliasing
  - Can still alias with high-frequency bumpy geometry
- Relies on *properly modeled* shading normals

TAKE HOME MESSAGE

- Filtering of diffuse illumination is well-established
  - Filter on surface, e.g., texture filtering
- When filtering specular & glossy, always consider a 3-point transport
- Specular constraint lives in local shading frame
  - Half vector depends on all three vertices of the path
  - Depends on the curvature of the surface

Thank you!

Q&A