MSc CGVI Final Project

Relighting for 2D Copy & Paste

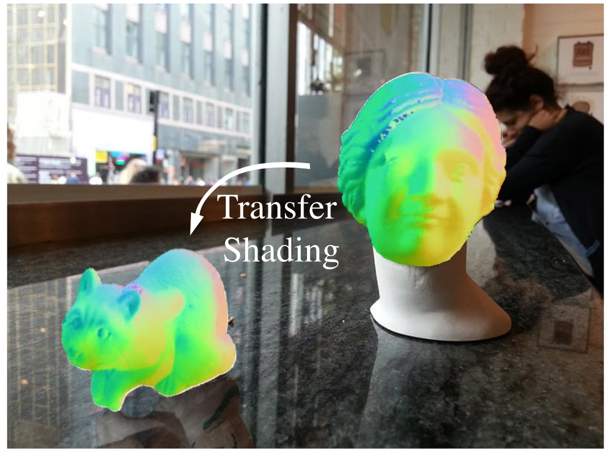
Lukas Murmann
l.murmann.12@ucl.ac.uk

Thesis Supervisor: Jan Kautz

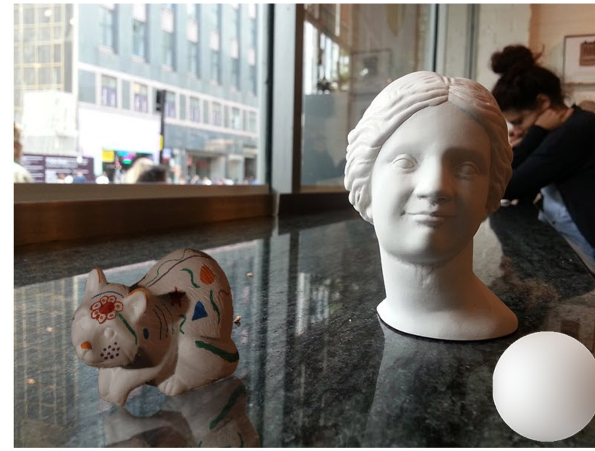
13 September 2013

Relighting for 2D Copy & Paste Introduction

After Copy & Paste: Raccoon seems out of place.



- Estimate normals
- Estimate shading on roman statue
- Transfer shading



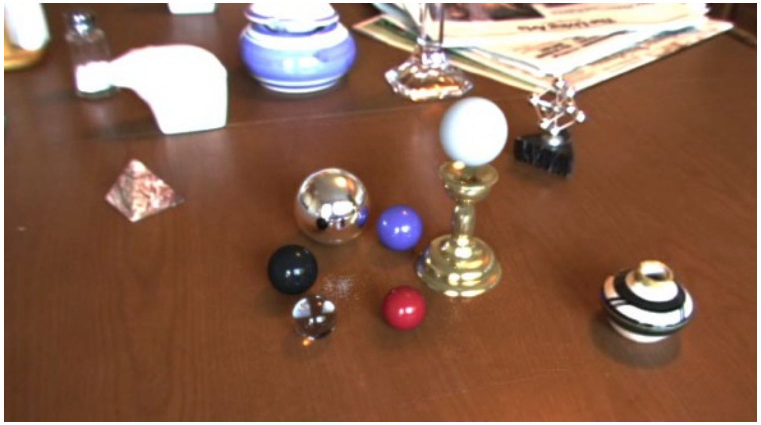
The result looks more plausible.

Almost no user interaction. Suitable for unskilled user.

More Applications: Insert 3D Models, Detect Tampering

Related Work

Rendering Synthetic Objects into Real Scenes [Debevec 1998]

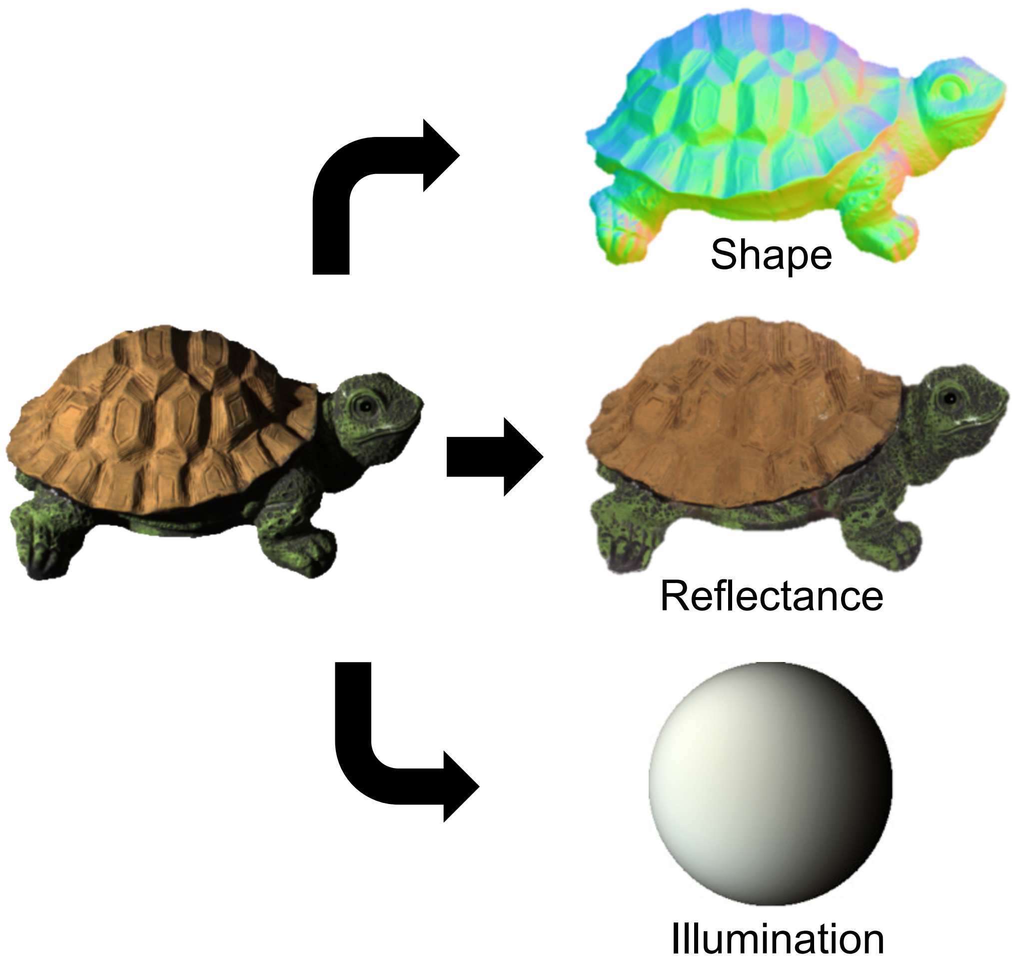


This paper introduced idea of “rendering into photographs”.

Difference to our work:

- **Geometry:** We do not render 3D models, but 2D images.
- **Lighting:** We estimate shading from image, Debevec captures with light probe.

SIRFS (Shape, Illumination, and Reflectance from Shading) [Barron and Malik 2013]



Differences to our work:

- SIRFS focuses on Computer Vision; we focus on Computer Graphics applications.
- SIRFS is one monolithic optimization; we perform each step independently.
- Our algorithm runs factor 5 faster.

Relighting Method

The Shading Equation

General Reflectance Equation

$$L(\vec{x},\vec{\omega}_o) = \int_{\Omega_{\mathcal{H}}} f_r(\vec{x},\vec{\omega}_i,\vec{\omega}_o)\, L(\vec{x},\vec{\omega}_i)\, \vec{n}\cdot\vec{\omega}_i \, \mathrm{d}\vec{\omega}_i$$

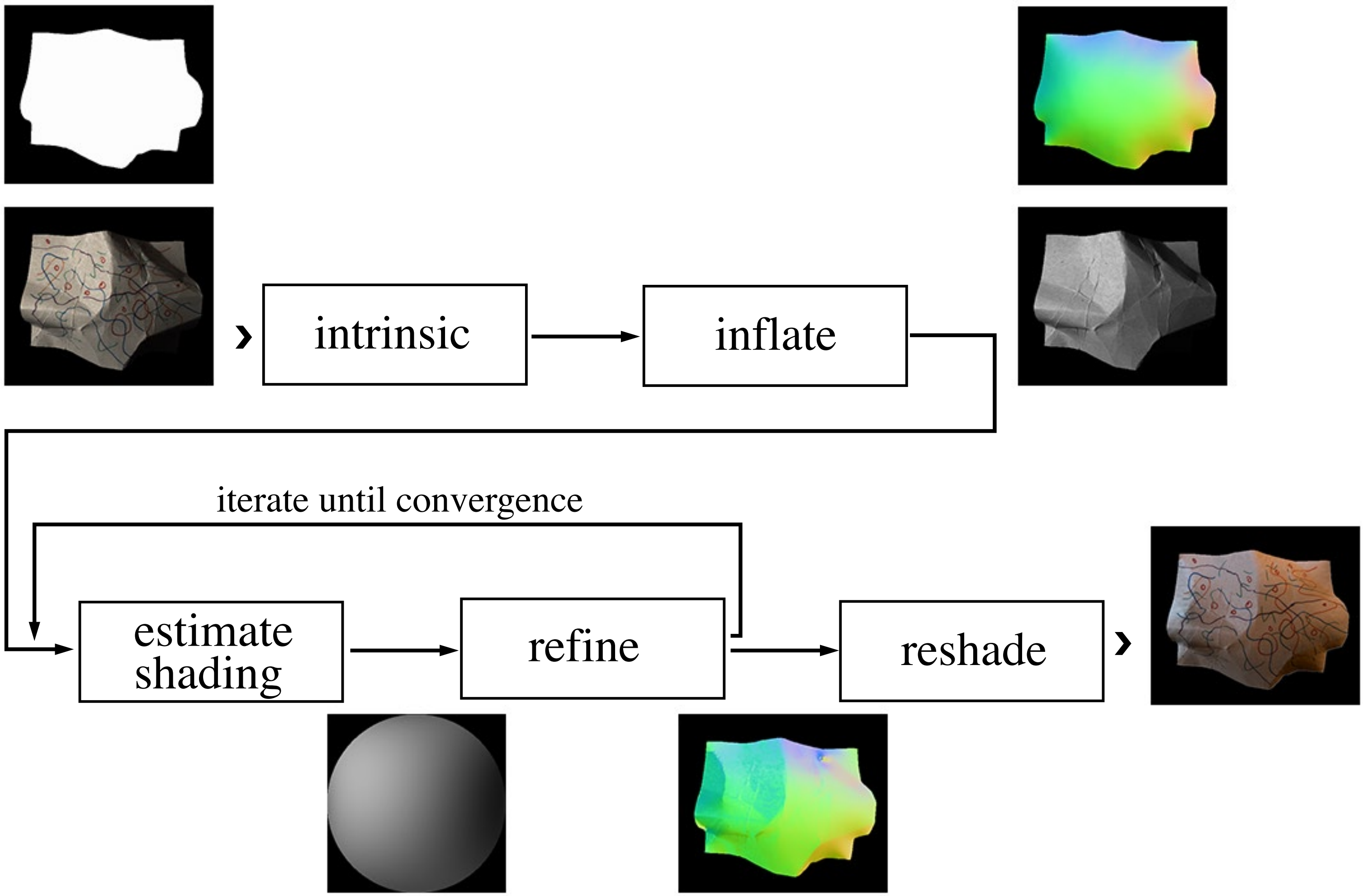
becomes with assumptions

- Spatially constant illumination
- Orthographic projection
- Can factor BRDF

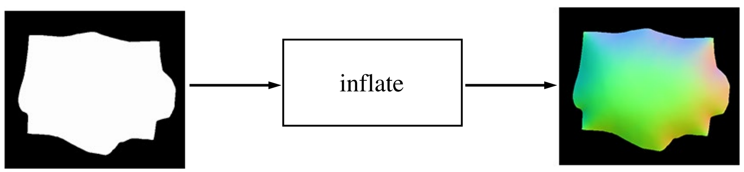
$$\begin{aligned} L(\vec{x},\vec{n}) &= \rho(\vec{x}) \int_{\Omega_{\mathcal{H}}} f_r'(\vec{\omega}_i)\, L(\vec{\omega}_i)\;\; \vec{\omega}_i\cdot\vec{n}\, \mathrm{d}\vec{\omega}_i \\ &= \rho(\vec{x})\mathcal{S}(\vec{n}) \end{aligned}$$

… with Reflectance Image $\rho(\vec{x})$
Shading Sphere $\mathcal{S}(\vec{n})$
and Geometry $\vec{n}$

Relighting Pipeline Overview



Inflation Based on Variational Mesh Editing [Botsch et al. 2004]



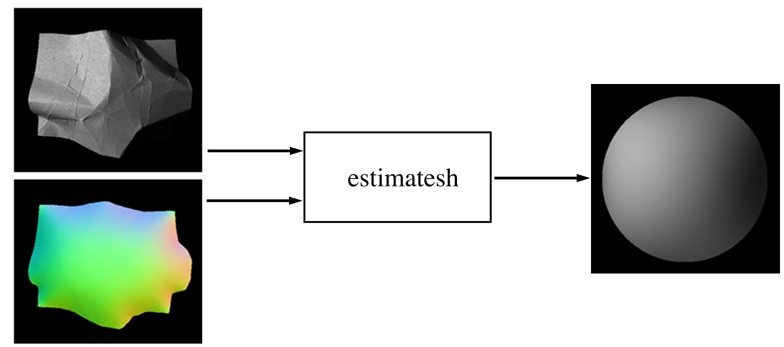
Model as PDE
$$\Delta x = \kappa, \quad v_i \in \Omega \setminus \partial\Omega$$
$$x = 0, \quad v_i \in \partial\Omega$$

and solve sparse linear system

$$\begin{array}{cc}
\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & & & \Delta & & \\ & & & & \Delta & & \\ & & & & \Delta & & \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \kappa \\ \kappa \\ \kappa \end{bmatrix}
& \begin{array}{l} \text{Boundary Vertices} \\ \\ \\ \\ \text{Free Vertices} \end{array}
\end{array}$$

with Laplace-Beltrami weights Δ.

Estimate Shading with Linear Regression



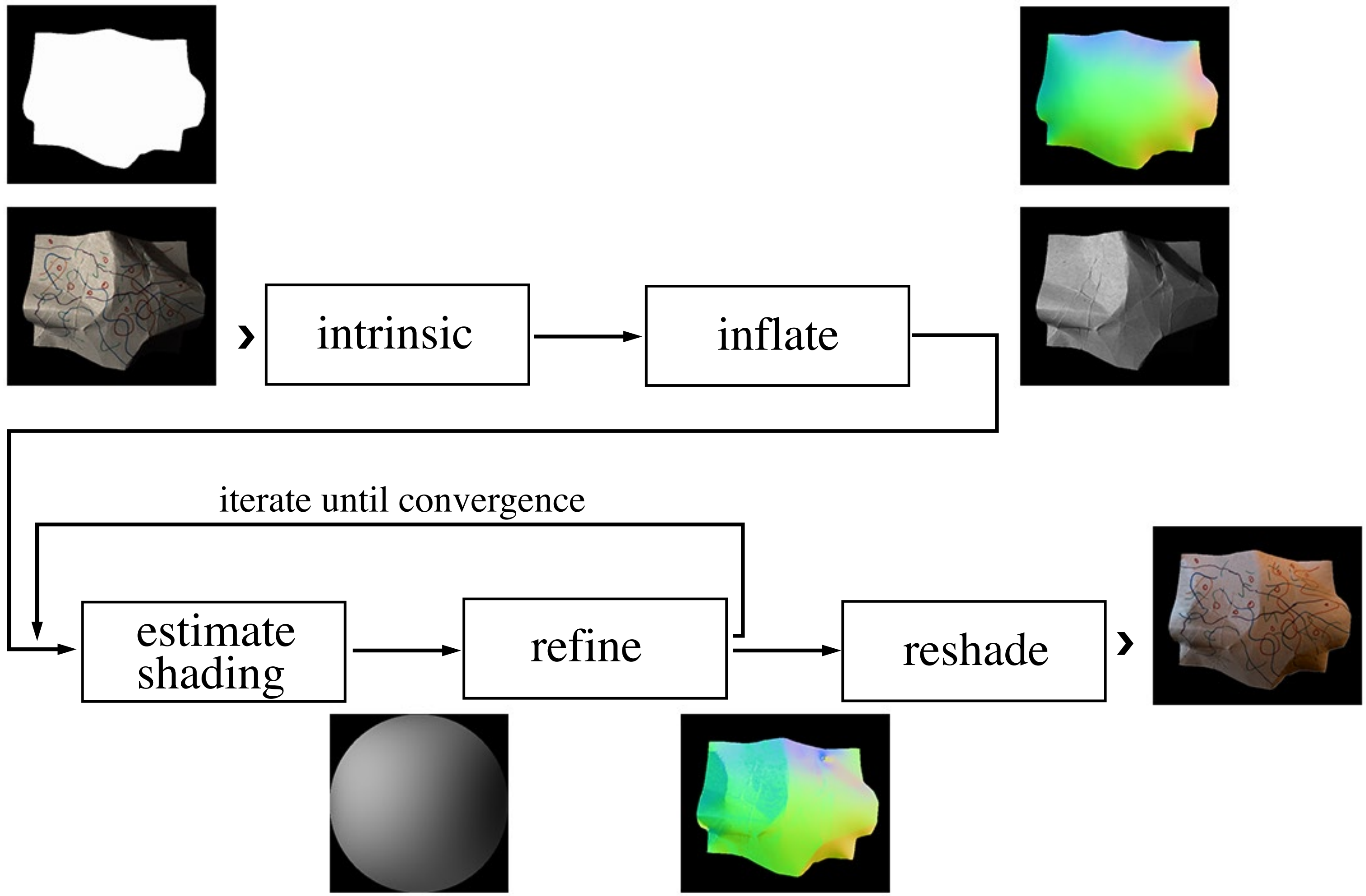
Fit weighting coefficients of spherical harmonic basis.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_i \left(\mathcal{S}(\vec{n}_i, \boldsymbol{\theta}) - s_i\right)^2$$

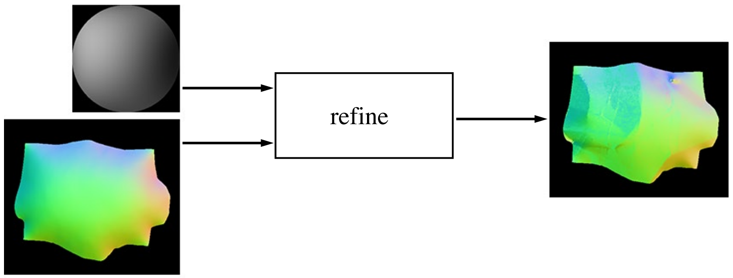
with shading sphere

$$\mathcal{S}(\vec{n}_i, \boldsymbol{\theta}) = \sum_{l=0}^{L} \sum_{m=-l}^{l} Y_{lm}(\vec{n}_i)\theta_{lm}$$
$$= \boldsymbol{Y}(\vec{n}_i)^T \boldsymbol{\theta}$$

Relighting Pipeline Overview



Refine Normals in Nonlinear Optimization



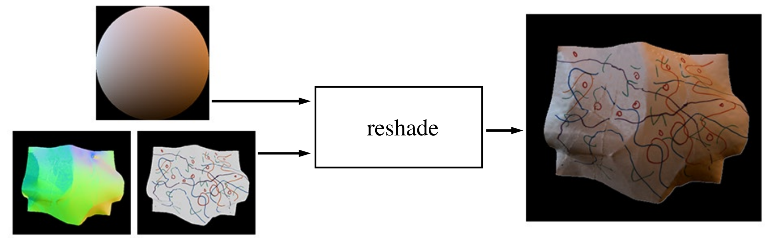
Refine each normal direction $\vec{n}_i$ s.t.

$$\vec{n}_i = \arg\min_{\vec{n}_i} \left| \mathcal{S}(\vec{n}_i) - s_i \right|$$

Hard Problem:

- We need nonlinear optimization
- Will only find local minimum
- Much more unknowns than in shading estimation.

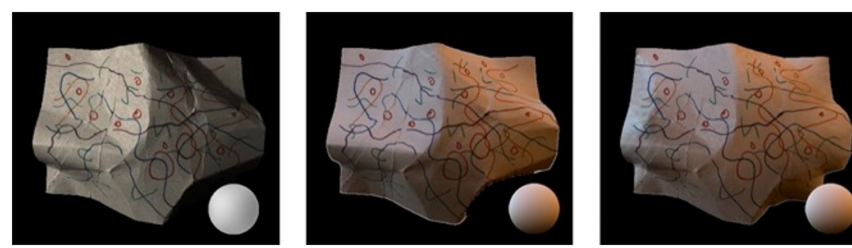
Relight object with Second Shading Sphere



Final steps

- Import second shading sphere
- Here, Pisa [Debevec 1998] environment.
- Evaluate shading equation $\rho(\vec{x})\mathcal{S}(\vec{n})$

Result



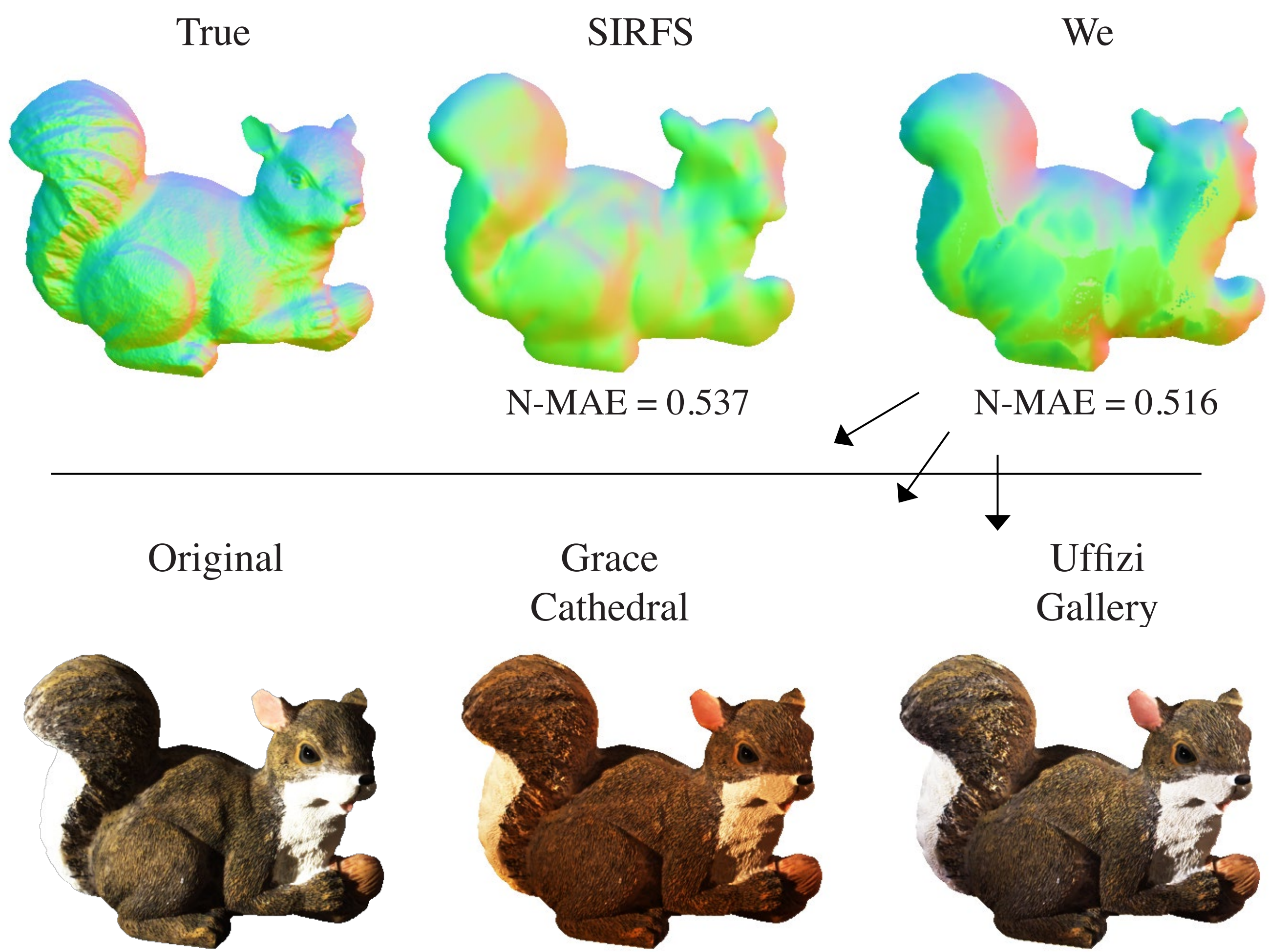
Original True Our Solution

Results

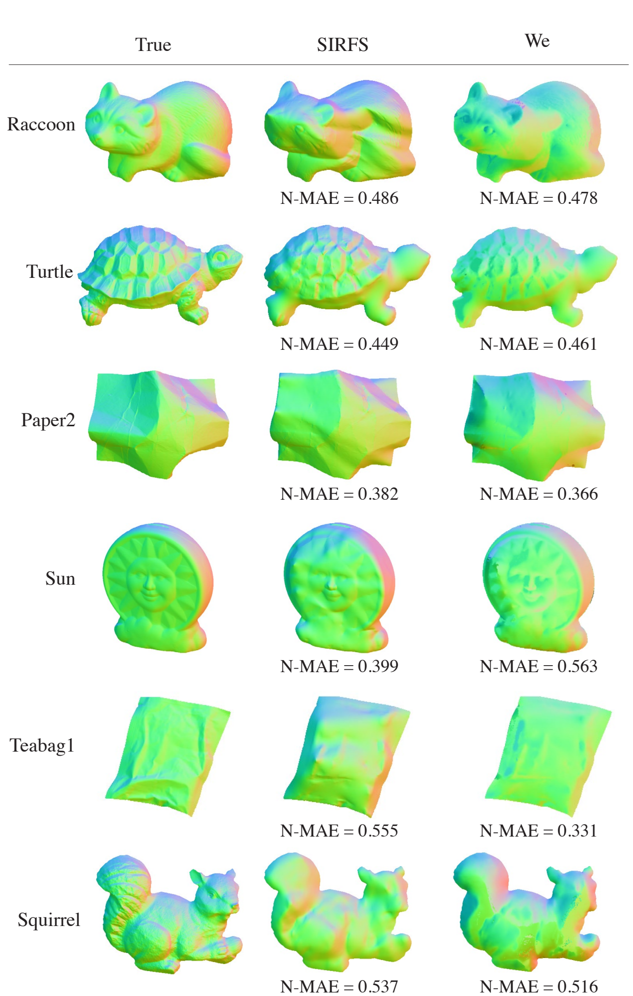
Relighting the MIT Data Set – Benchmark Results

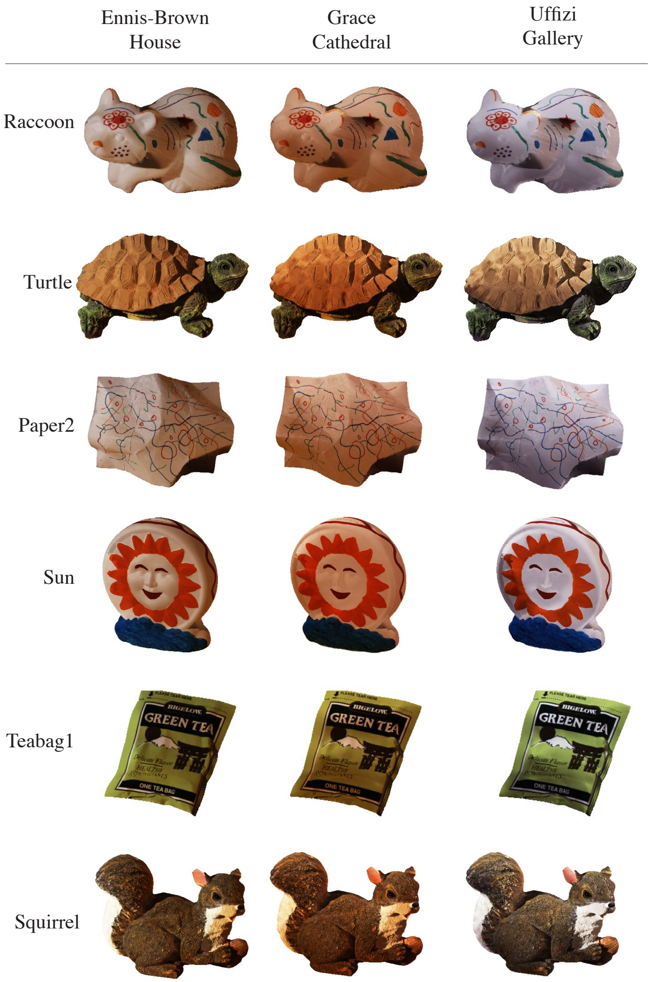
Normal Estimates
and
Mean Absolute Error

Relighting our
Geometry Estimate



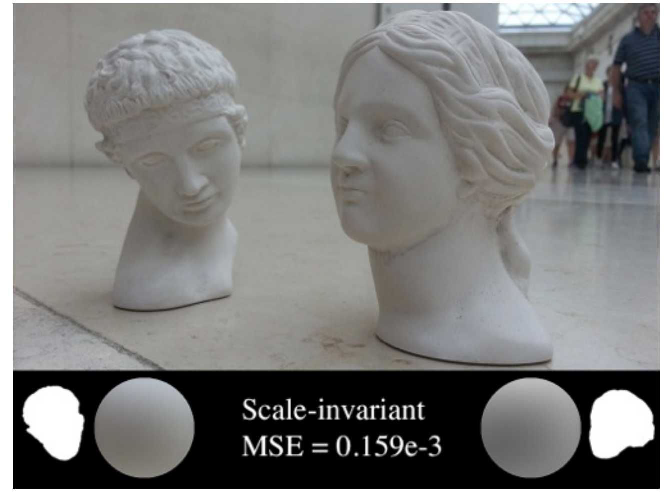
See Report for More Results





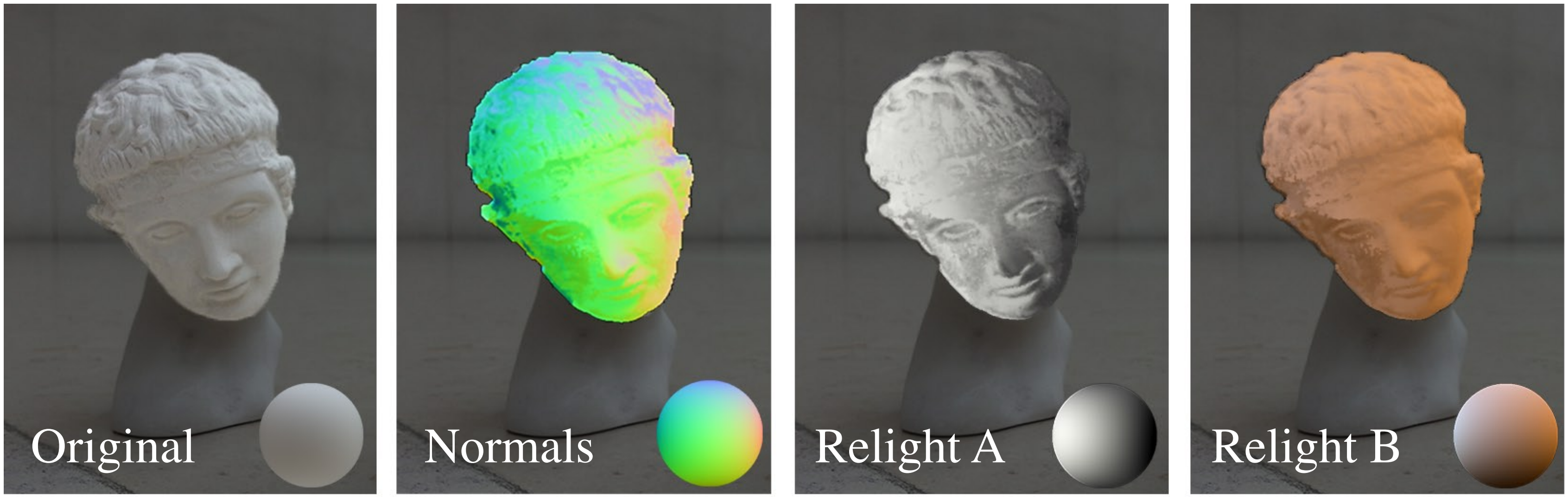
Rendering 3D Models

Tampering Detection



Conclusion

Discussion



Comparison to SIRFS

- Geometry estimates as good as state-of-the-art (SIRFS).
- Our algorithm is simpler.
- Our algorithm is faster.

Next Steps

- Normal refinement is hard.
- Normal discontinuities are often distracting.
- TODO: Smarter prior for normal field (Integrability, Smoothness).

Q&A

Bibliography

[Barron2013] Jonathan Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. Technical Report UCB/EECS-2013-117, EECS, UC Berkeley, May 2013. 12, 25, 37, 40, 46

[Botsch2004] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, August 2004. 14, 59

[Curless1996] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM. 37

[Debevec1998] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA, 1998. ACM. 5, 6, 7, 9, 36

Backup

Minimizers in Normal Refinement Lie on Isocontour

Every direction on isocontour is local minimizer

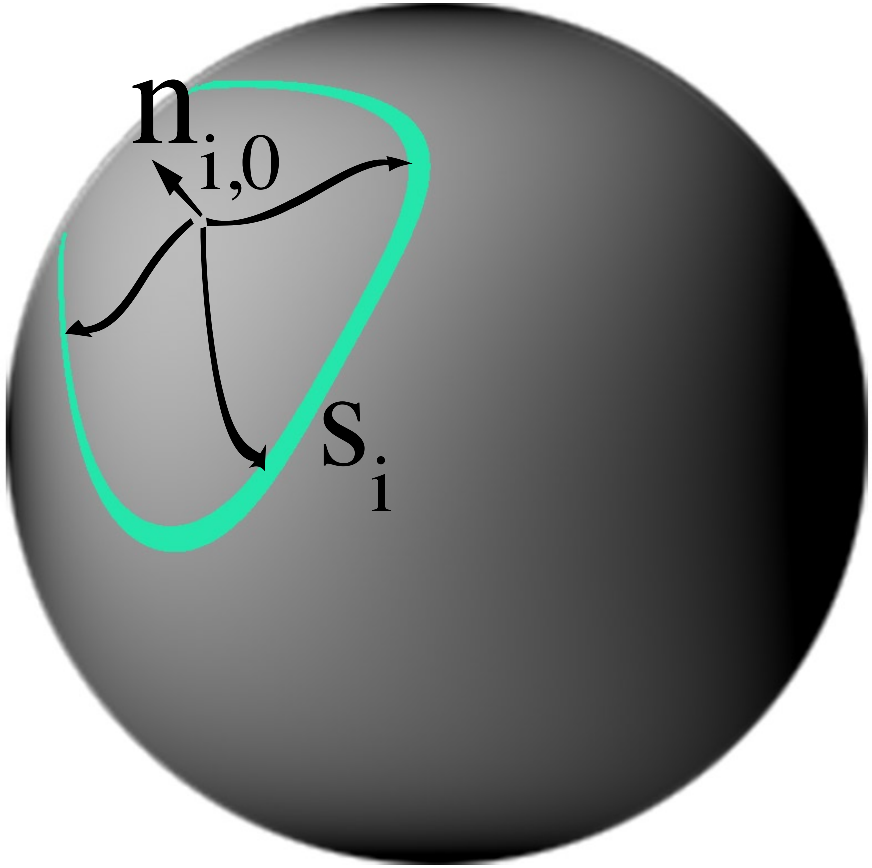
We **need additional penalty term.**

Current penalty:

- Angular distance to n_0

Smarter prior:

- Integrability
- Smoothness

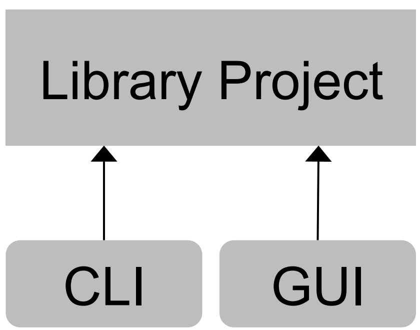


Command Line

```
$ estimatesh [-L<n>]
         -n <normals.png>
         -s <shading.png>
         -o <sh_coeffs.mat>

$ refine –c <sh_coeffs.mat>
         -n <normals.png>
         -s <shading.png>
         -o <refined.png>
$ reshade …
```

Interactive GUI



Timing for Turtle Test Case

SIRFS: 4m40s

```
err =

        grosse: 0.0682
       shading: 0.0286
   reflectance: 0.0155
        normal: 0.4112
         light: 0.0042
        height: 17.0583
           avg: 0.0980

>> state

state =

             height: [300x467 double]
              light: [9x3 double]
             normal: [300x467x3 double]
            shading: [300x467x3 double]
        reflectance: [300x467x3 double]
         final_loss: 44.7568
         solve_time: 281.9241
    reflectance_exp: [300x467x3 double]
    reflectance_max: [300x467x3 double]
```

We: 55s

```
Inflate & Shading  10s
First Refine       25s
Second Refine      20s
---
                   55s
```

Note: Neither our code nor SIRFS is optimized for performance.

Tolerances in optimization have impact on convergence speed of both algorithms.