Rendering for Data Driven Computational Imaging

Tristan Swedish
Camera Culture Group, MIT Media Lab

ciml.media.mit.edu

Data-Driven Computational Imaging

Time	Title	Presenter
08:30 – 08:50	Introduction to Computational Imaging	Guy Satat (MIT)
08:50 - 09:15	Data-Driven Computational Imaging Survey	Tristan Swedish (MIT)
09:15 – 10:00	Data-Driven Non-line-of-sight Imaging and 3D Reconstruction	Guy Satat (MIT)
10:00 – 10:20	Break	
10:20 – 11:00	Rendering and Simulation for Data-Driven Computational Imaging	Tristan Swedish (MIT)
11:00 – 12:00	Visual Sensing Using Machine Learning	Vivek Boominathan (Rice), Ashok Veeraraghavan (Rice)

Graphics in Computational Imaging

$$I_o(x, y) = \int_{\Omega_{\theta,\phi}} \int_{\lambda} \int_{\rho} \int_{t} \sum_{n} I_i(x, y, \theta, \phi, \lambda, \rho, t, n)$$

- The Plenoptic Function

1. **Generating training and test data**
    a. Examples in the literature
    b. Graphics 101
    c. Practical considerations

2. **Runtime solution (analysis by synthesis)**
    a. Differentiable Rendering
    b. Ongoing research / examples

Why Render?

- Large amounts of real data with sufficient variation is difficult to collect
- Rare events may be under-represented in dataset
- Leverage well understood imaging physics as prior knowledge
- Introduce interpretable constraints on the underlying world representation

What can we render?

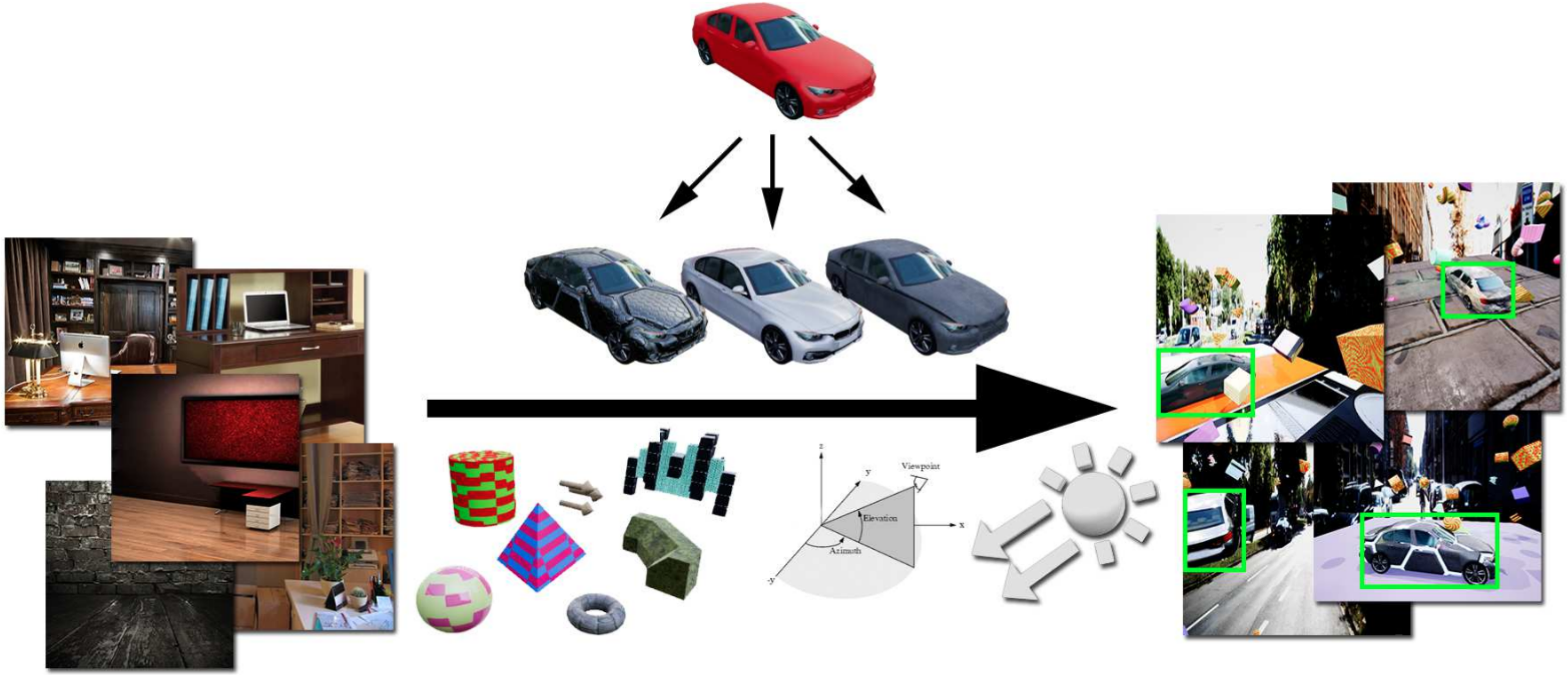
Optical Phenomena (Computational Imaging):

- “Photorealistic” Geometrical Optics
    - Path-tracing
    - Physically Based Materials
    - Camera/Film Models
- Volumetric Scattering (“Participating Media”)
- Transient Images: Time of Flight

Other Physics:

Seismic, Acoustic, E&M: Wave equation solvers or approximation

Example: Domain Randomization

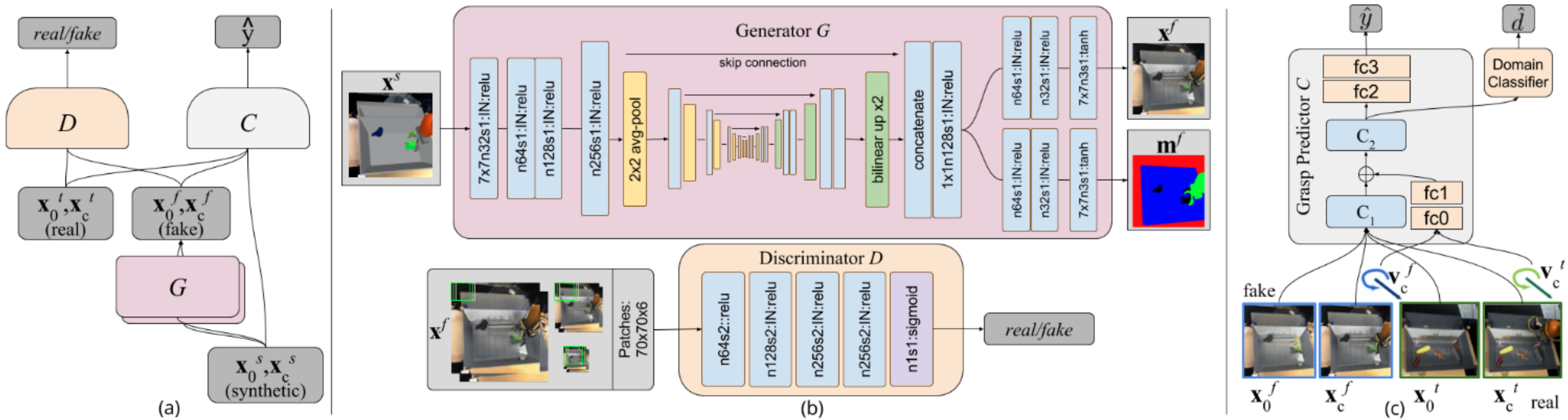


Tremblay, Prakash, Acuna, Brophy. *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*, CVPR Workshops, 2018.

Example: Domain Randomization + Photorealistic

J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox & S. Birchfield. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. Proceedings of The 2nd Conference on Robot Learning, in PMLR, 2018.

Example: Using Simulation and Domain Adaptation for Deep Robotic Grasping



Synthetic ⟶ Real

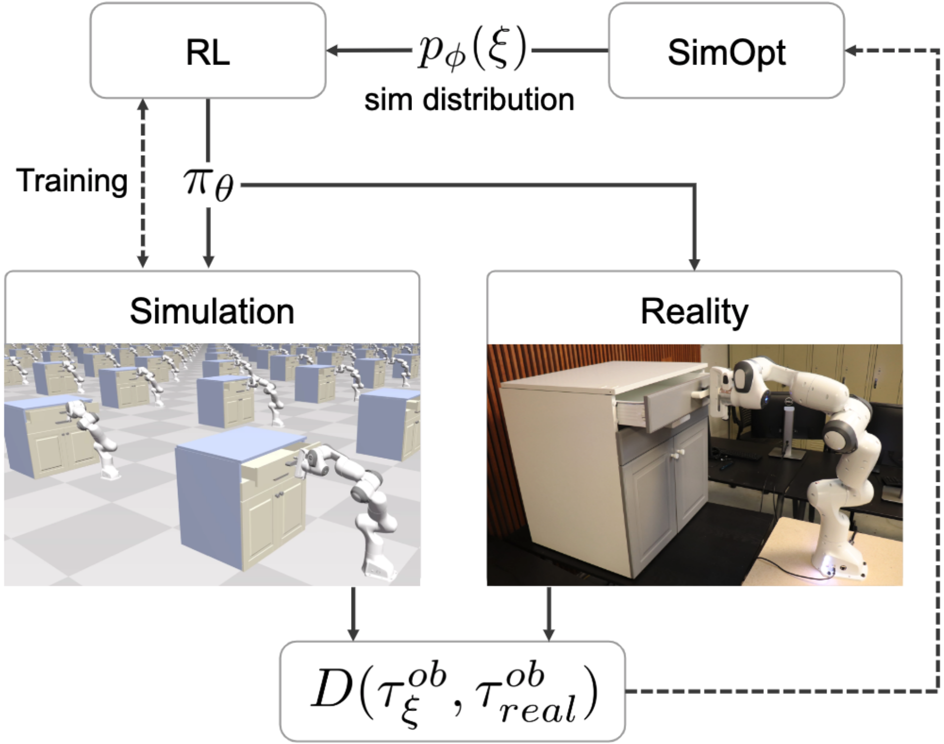
K. Bousmalis, A. Irpan, P. Wohlhart, et al. *Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping*. ICRA, 2018.

Example: Synthetic Data with Domain Adaptation for Monocular Depth

Real ⟶ Synthetic

A. Atapour-Abarghouei and T. Breckton. *Real-Time Monocular Depth Estimation using Synthetic Data with Domain Adaptation via Image Style Transfer*. CVPR, 2018.

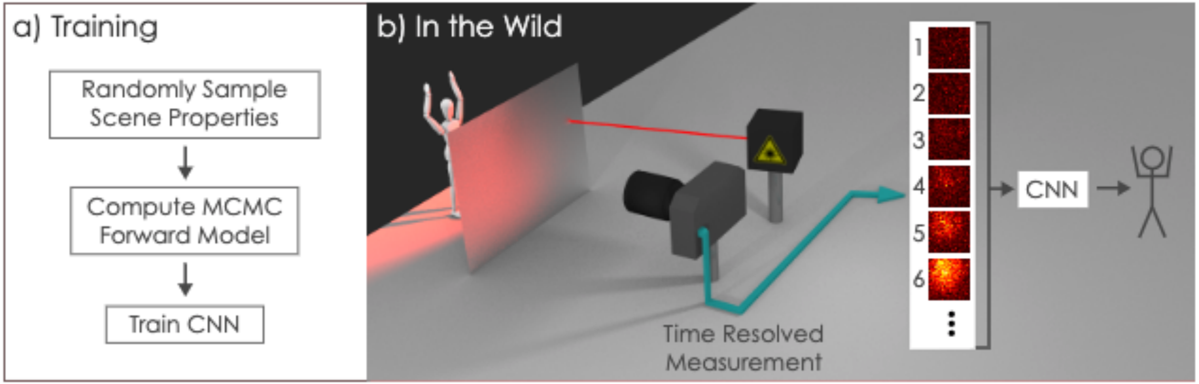
Example: Adapting Simulation Randomization

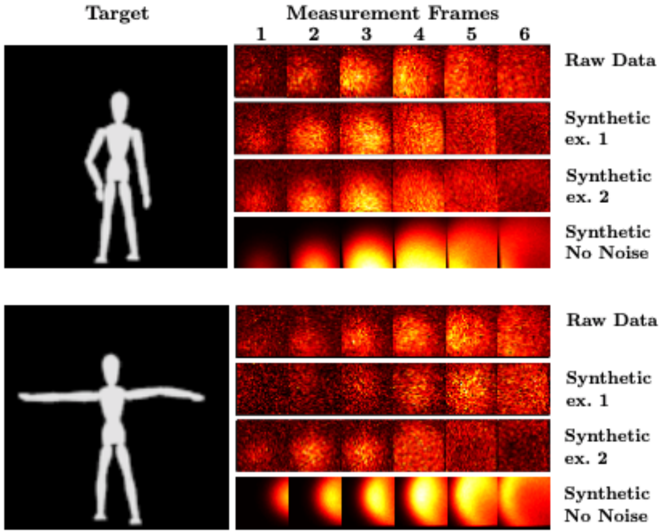


*__Note__: Physics Simulation rather than synthetic input image.

Y. Chebotar et al. *Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience*. ICRA, 2019.

Example: Classification through Scattering





Guy Satat, Matthew Tancik, Otkrist Gupta, Barmak Heshmat, and Ramesh Raskar. *Object classification through scattering media with deep learning on time resolved measurement*. Optics Express, 2017.

Example: Classification through Scattering

Calibration parameters	
Laser	
- Incident position	$L_P \sim U(-4, 4)cm$
Diffuser	
- Scattering profile	$D_D \sim N(0, \sigma)$, $\sigma \sim U(0.8, 1.2)rad$
Camera	
- Position	$C_P \sim U(-1.5, 1.5)cm$
- Time resolution	$C_{TR} \sim N(0, \sigma)$, $\sigma \sim 56 + U(-5, 5)ps$
- Time jitter	$C_{TS} \sim U(0, 3 * 56)ps$
- Field of view	$C_{FV} \sim U(0.1, 0.2)rad$
- Homography	Normal distributions
Noise	
- Dark count	$N_{DC} \sim U(3000, 9000)$ photons
Target parameters	
- Position	$T_{P_{x,y,z}} \sim U(-4, 4)cm$
- Scale	$T_S \sim U(18, 30)cm$

Guy Satat, Matthew Tancik, Otkrist Gupta, Barmak Heshmat, and Ramesh Raskar. *Object classification through scattering media with deep learning on time resolved measurement*. Optics Express, 2017.

Industry

DeepVisionData
synthetictrainingdata.com

Self-driving Car Simulators

CARLA

NVIDIA DRIVE Constellation

Popular Rendering Tools

General Purpose

- Path Tracing: Mitsuba / PBRT, Blender Cycles
- Raster / Realtime: Unreal Engine, Unity3D

Frameworks and Pipelines for Reinforcement Learning and Robotics

- Gazebo Sim (OSRF / ROS) - Also for interactive / physics simulation

- OpenAI Gym (various renderers)
- Games (e.g. GTA5)

Mitsuba

mitsuba-renderer.org

Blender Cycles

Check! - Ron Shaver
https://www.artstation.com/artwork/r3E8e

Gazebo Sim

http://gazebosim.org/

Renderers in Film Production

Generally, the trend for photorealism is to use physically based rendering

ACM Transactions on Graphics (TOG) - Special Issue On Production Rendering, 2018

Manuka
L. Fascione, et al. *Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production.*

Hyperion
B. Burley, et al. *The Design and Evolution of Disney's Hyperion Renderer.*

Arnold/Sony Imageworks Arnold
I. Georgiev, et al. *Arnold: A Brute-Force Production Path Tracer.*
C. Kulla, et al. *Sony Pictures Imageworks Arnold.*

RenderMan
P. Christensen, et al. *RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering.*

Arnold: Brute-force Production Path Tracer

“Maya” copyright SSE, VFX by The Mill, 2015.

I. Georgiev, et al. *Arnold: A Brute-Force Production Path Tracer*, ACM Transactions on Graphics (TOG), 2018.

Graphics 101

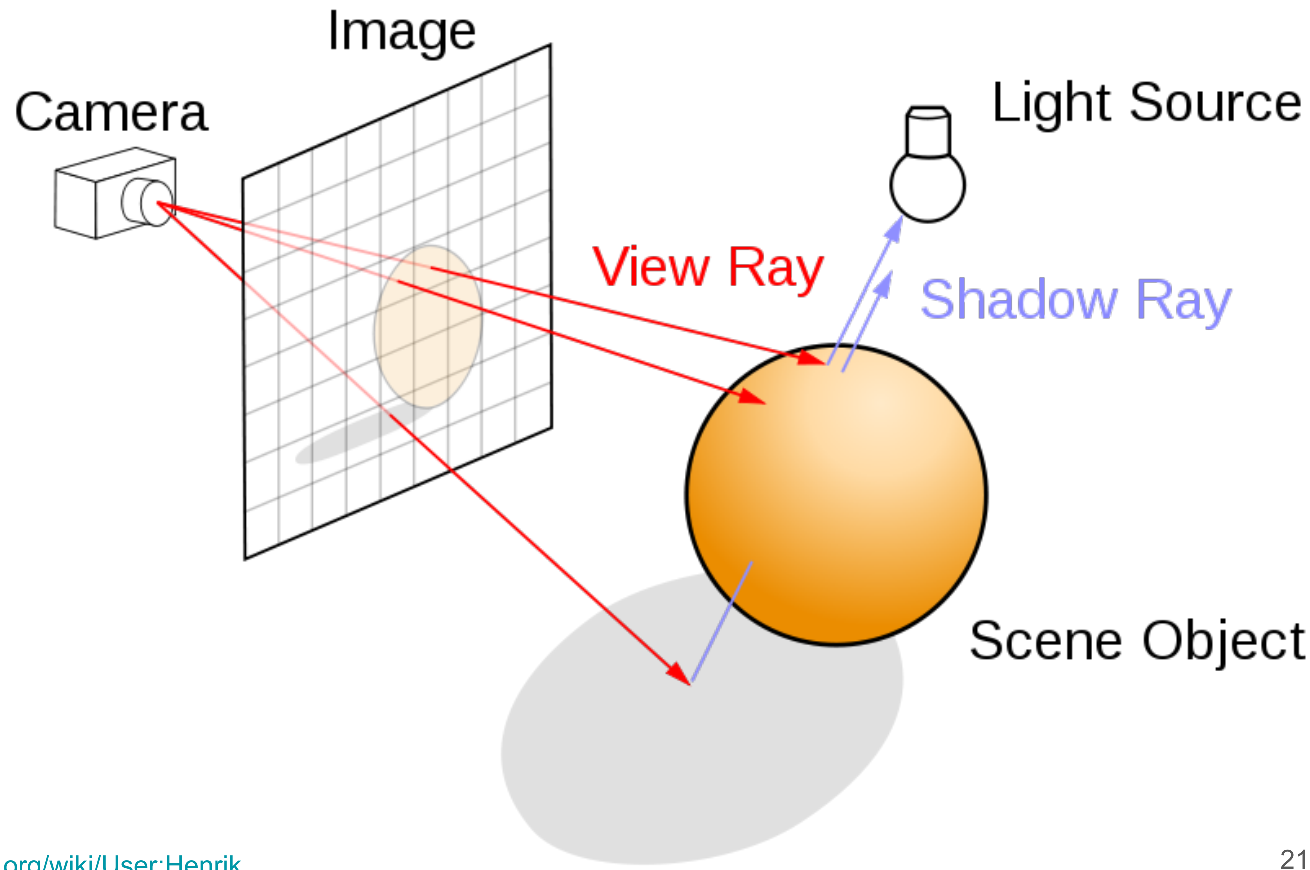


Image by Henrik: https://commons.wikimedia.org/wiki/User:Henrik

Graphics 101: Rendering Equation and Plenoptic Function

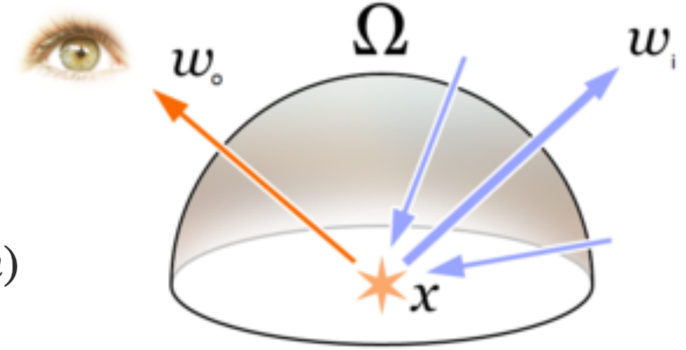
The Rendering Equation

$$L_{\mathrm{o}}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;=\; L_e(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;+\; \int_{\Omega} f_r(\mathbf{x},\, \omega_{\mathrm{i}},\, \omega_{\mathrm{o}},\, \lambda,\, t)\, L_{\mathrm{i}}(\mathbf{x},\, \omega_{\mathrm{i}},\, \lambda,\, t)\, (\omega_{\mathrm{i}} \,\cdot\, \mathbf{n})\, \mathrm{d}\,\omega_{\mathrm{i}}$$

Rendering: ***How to integrate?***

The Plenoptic Function

$$I_o(x,y) = \int_{\Omega_{\theta,\phi}} \int_{\lambda} \int_{\rho} \int_{t} \sum_{n} I_i(x,y,\theta,\phi,\lambda,\rho,t,n)$$



Computational Imaging: ***Use knowledge of optical transport***

Wikimedia: Timrb

Graphics 101: Rendering Approximations

Rasterization: Real-time rendering

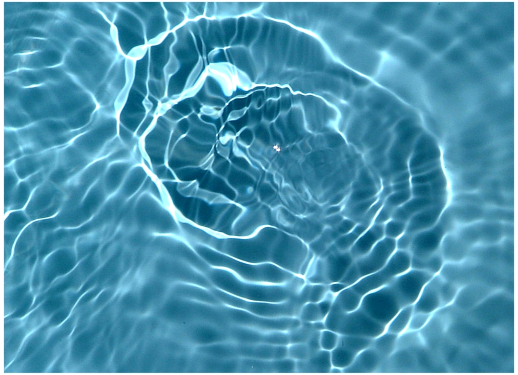
- Almost all games / interactive applications

Radiosity: Global Illumination

- Finite Element method to calculate steady state global illumination (diffuse paths)

Photon Mapping: Global Illumination / Caustics

- Popular for special cases, approximates rendering equation (but can be biased)
- Decouple illumination and geometry terms of the rendering equation



http://archvizcamp.com/vray-pool-water-caustics/

Path-Tracing: Sampling method to estimate integral in rendering equation

- Physically accurate, but long rendering times
- Necessary to simulate more advanced camera distortions or special optical configurations

Graphics 101: Photorealistic Rendering

Trend: the graphics industry is moving to more physically-based technologies to ensure consistency and streamline asset creation.

Physically Based Materials: Material BRDF

Geometry: Triangle Mesh

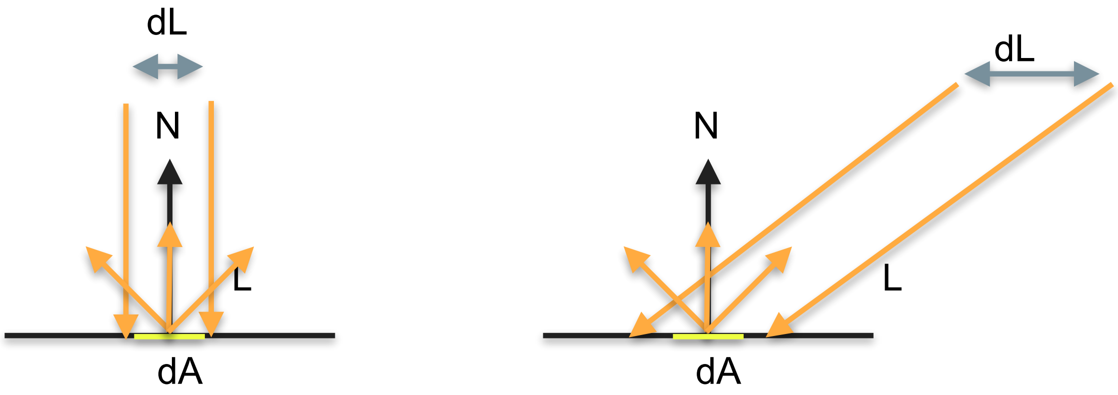
Path-tracing: Rendering Equation Integration

Camera Modeling: noise, lens distortion, depth of field

LuxCoreRender
San Pedro Bedroom by Charles Nandeya Ehouman (Sharlybg)

Graphics 101: Lambertian Shading

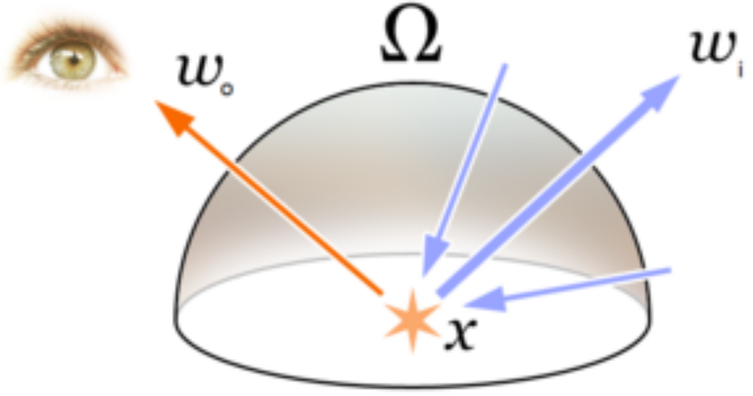
Lambert Cosine Law



$$\mathbf{L} \cdot \mathbf{N} = |N||L| \cos \alpha = \cos \alpha$$

Graphics 101: Materials

BRDF : Diffuse + Specular
BSDF: BRDF + BTDF (transmission)



Wikimedia: Timrb

Cook-Torrance Specular BRDF:

D - Distribution Function

F - Fresnel Function

G - Geometry Function

$$f_r(\mathbf{i},\mathbf{o},\mathbf{n}) = \frac{F(\mathbf{i},\mathbf{h}_\mathrm{r})\, G(\mathbf{i},\mathbf{o},\mathbf{h}_\mathrm{r})\, D(\mathbf{h}_\mathrm{r})}{4\,|\mathbf{i}\cdot\mathbf{n}|\,|\mathbf{o}\cdot\mathbf{n}|}$$

Commonly Used Functions:

GGX for D,G, and Shlick for F (dielectric) or Lazanyi (metals). Lambertian for diffuse scattering.

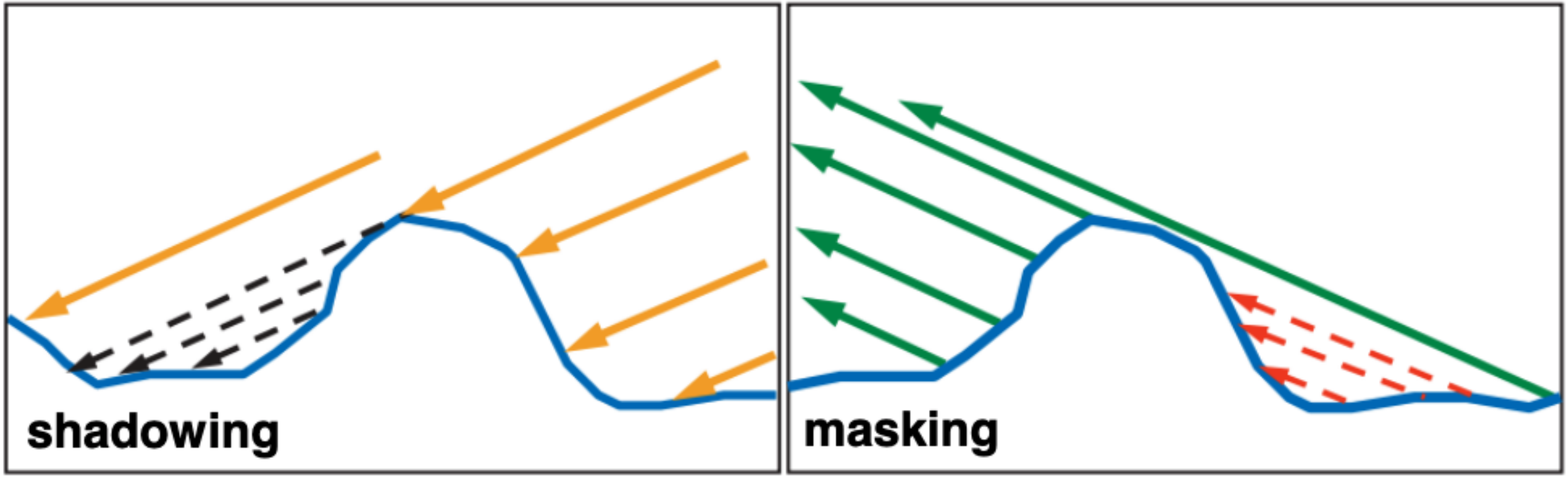
B. Walter, S. R. Marschner, H. Li, K. E. Torrance. *Microfacet Models for Refraction through Rough Surfaces*. Eurographics, 2007.

Graphics 101: Physically Based Materials

Conservation of energy $$L_o \leq L_i$$

$$L_{\mathrm{o}}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \,=\, L_e(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \,+\, \int_\Omega f_r(\mathbf{x},\, \omega_{\mathrm{i}},\, \omega_{\mathrm{o}},\, \lambda,\, t)\, L_{\mathrm{i}}(\mathbf{x},\, \omega_{\mathrm{i}},\, \lambda,\, t)\, (\omega_{\mathrm{i}} \,\cdot\, \mathbf{n})\, \mathrm{d}\,\omega_{\mathrm{i}}$$

Graphics 101: Microfacet models



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

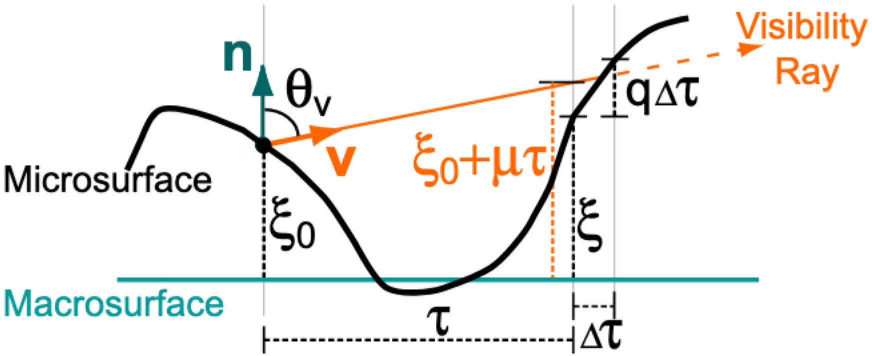
From: Physically-Based Shading Models in Film and Game Production. Siggraph Courses 2010

Graphics 101: GGX

$$k_{\text{spec}} = \frac{DFG}{4(V \cdot N)(N \cdot L)}$$

$$D(\mathbf{m}) = \frac{\alpha_g^2 \, \chi^+(\mathbf{m} \cdot \mathbf{n})}{\pi \cos^4 \theta_m \, (\alpha_g^2 + \tan^2 \theta_m)^2}$$

$$G_1(\mathbf{v}, \mathbf{m}) = \chi^+\left(\frac{\mathbf{v} \cdot \mathbf{m}}{\mathbf{v} \cdot \mathbf{n}}\right) \frac{2}{1 + \sqrt{1 + \alpha_g^2 \tan^2 \theta_v}}$$

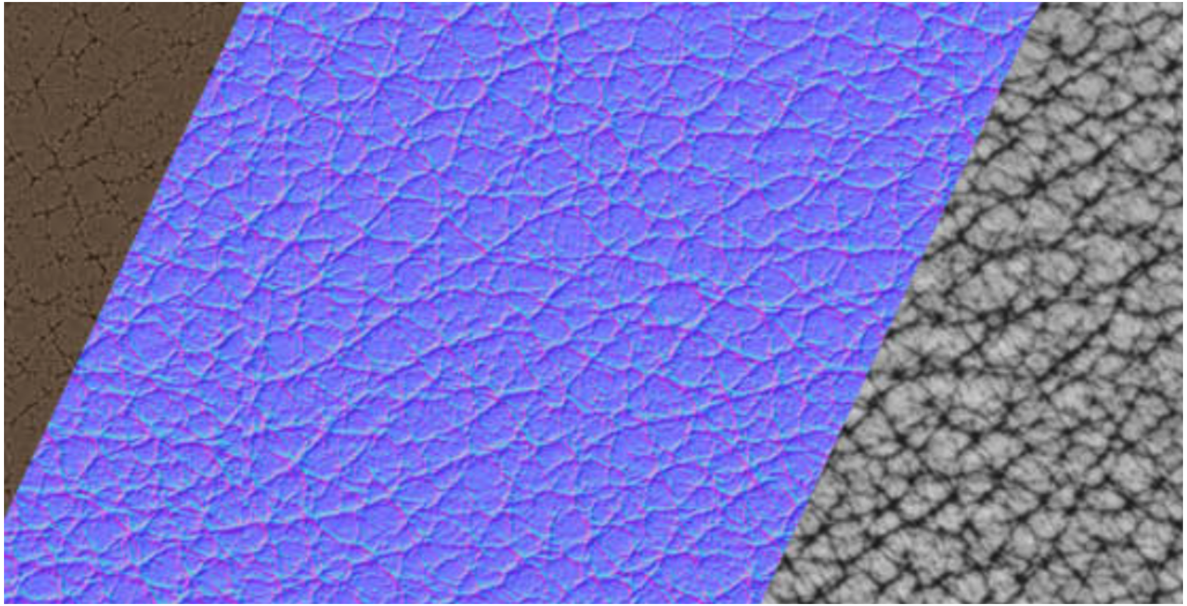


B. Walter, S. R. Marschner, H. Li, K. E. Torrance. *Microfacet Models for Refraction through Rough Surfaces*. Eurographics, 2007.

Graphics 101: Physically Based Materials

Physically Meaningful BRDF parameterization

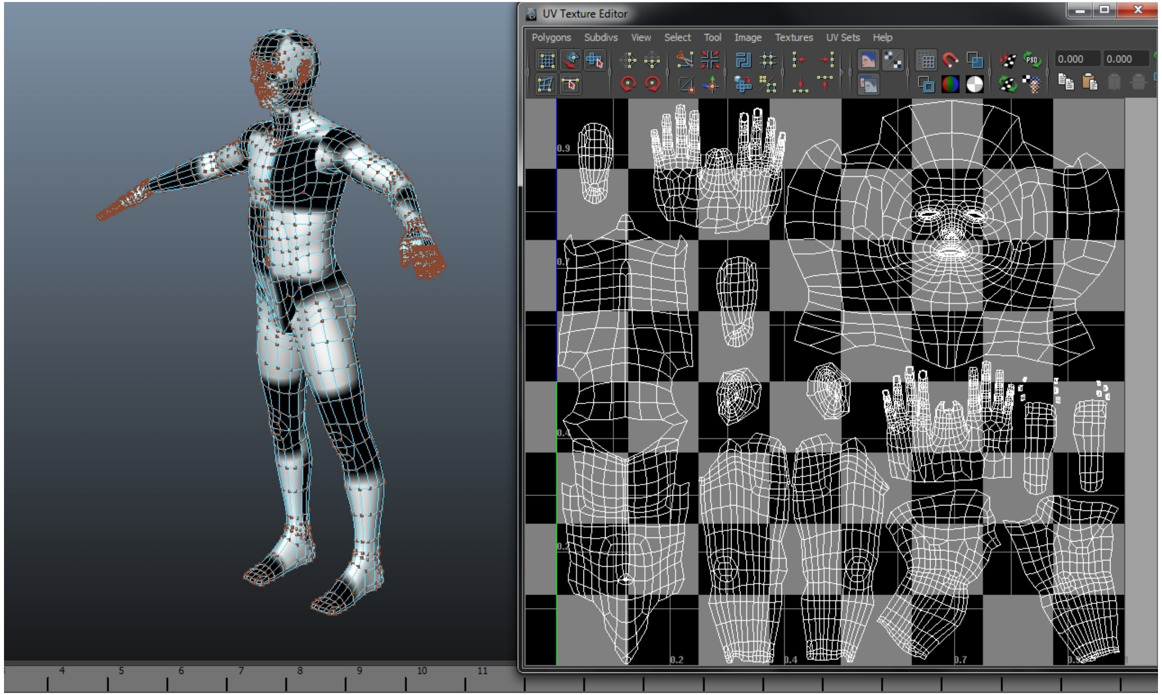
- Roughness
- Metalness
- Albedo
- Normal



www.substance3d.com

Graphics 101: Meshes

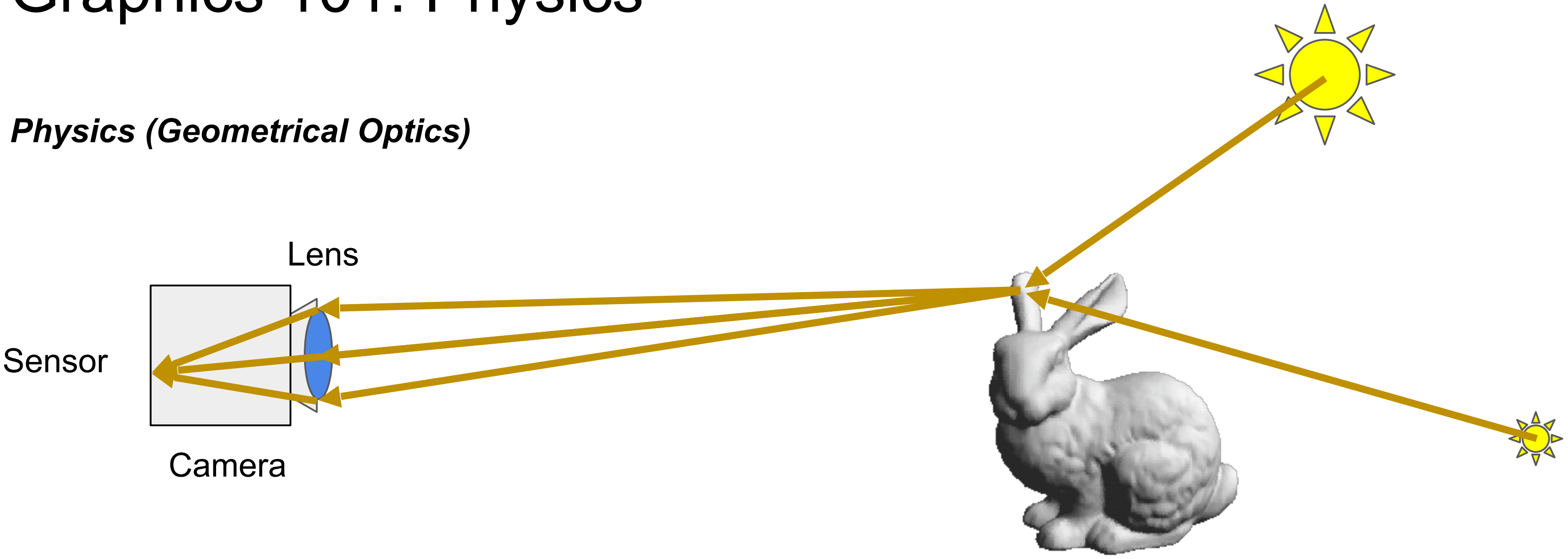
- **Geometry**
  - vertex coordinates + edges = faces (mesh)
- **Textures**
  - 2D texture mapped to position inside faces (UV coordinates)



https://fromgray.files.wordpress.com/2012/01/uvmap6.jpg

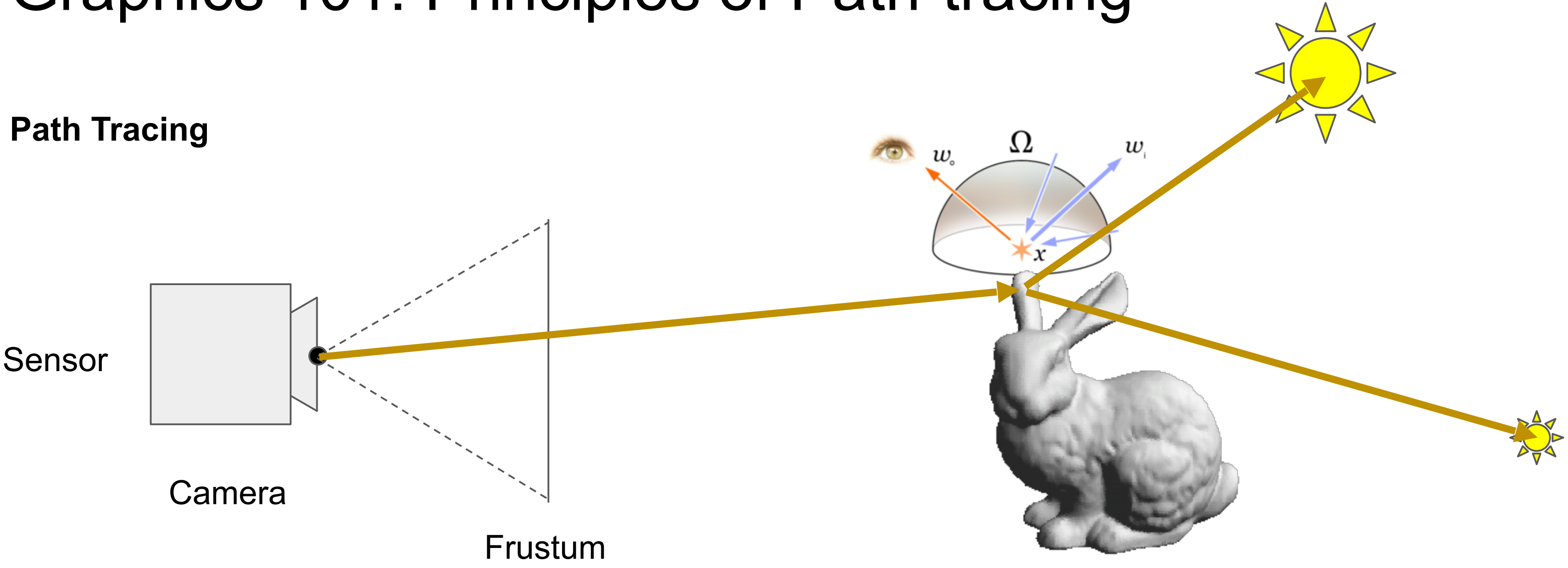
Graphics 101: Physics

Physics (Geometrical Optics)



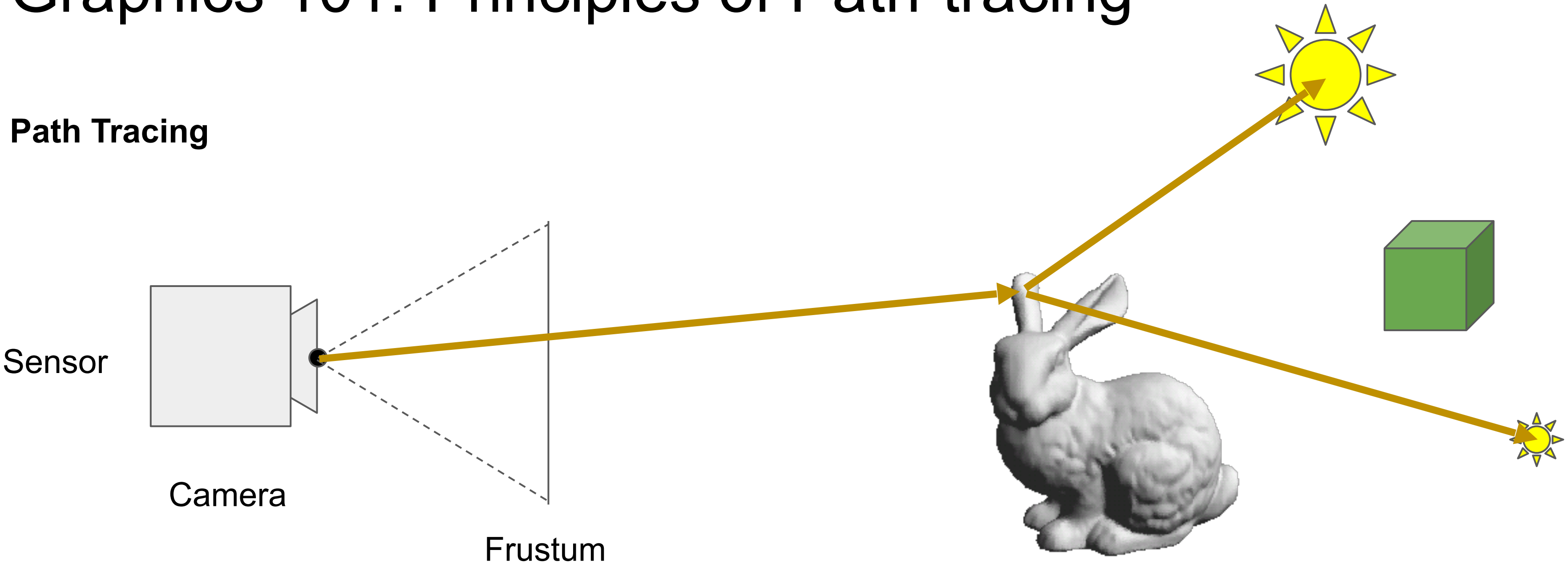
Graphics 101: Principles of Path-tracing

Path Tracing



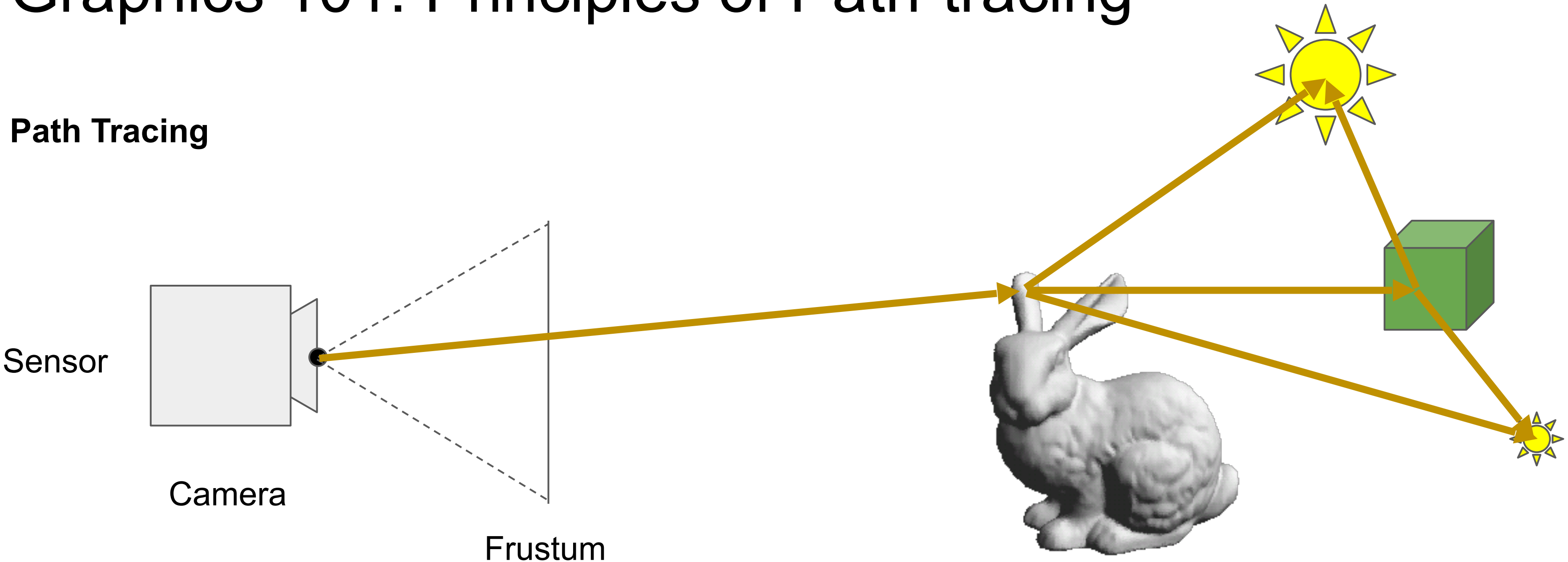
Graphics 101: Principles of Path-tracing

Path Tracing



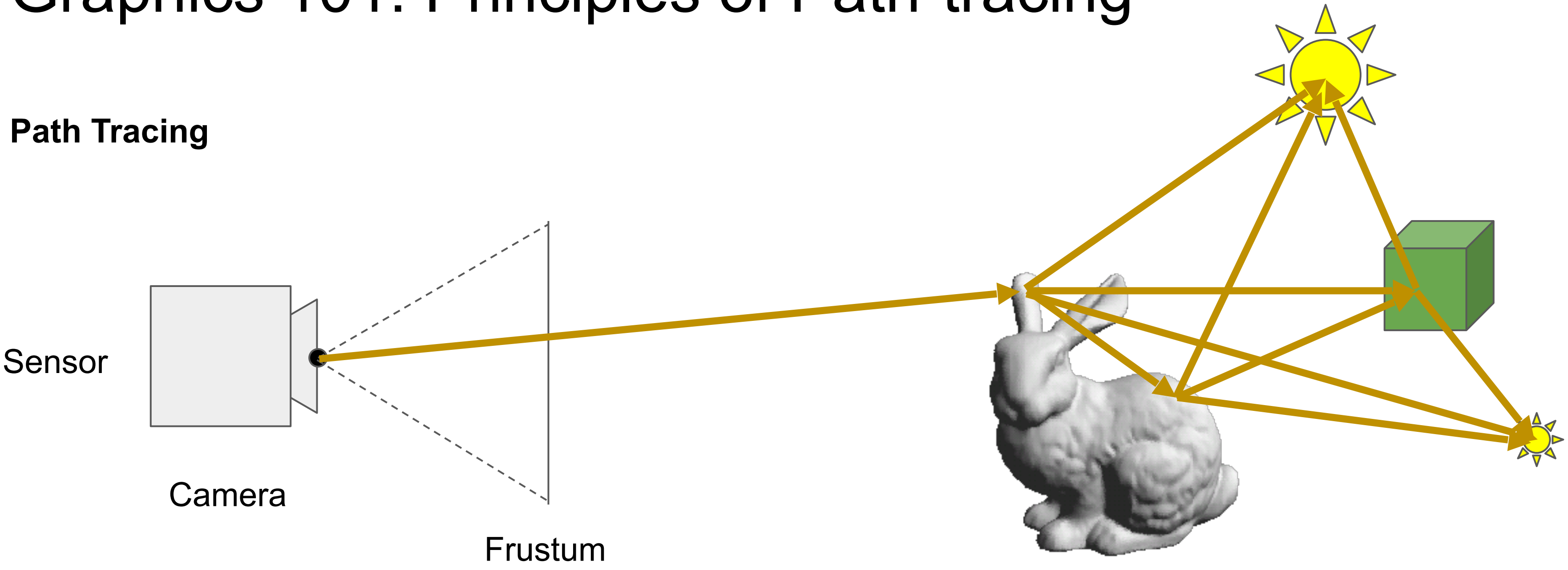
Graphics 101: Principles of Path-tracing

Path Tracing

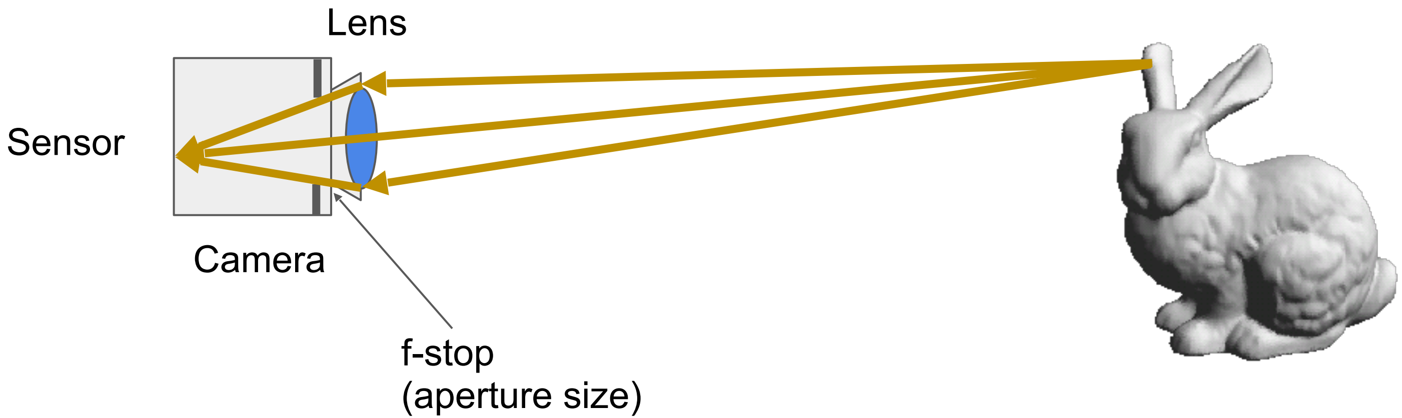


Graphics 101: Principles of Path-tracing

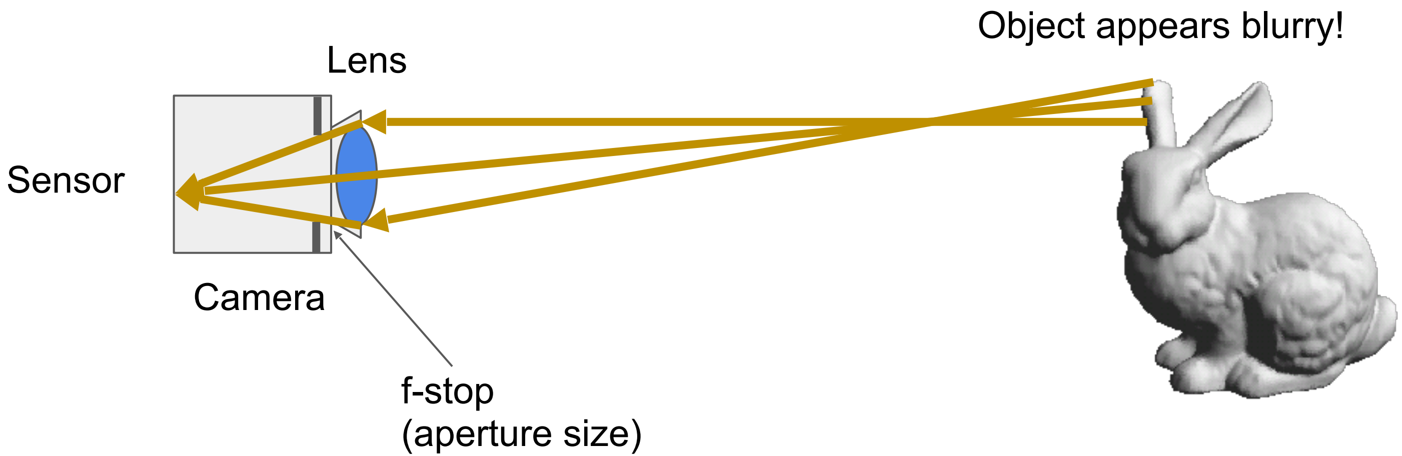
Path Tracing



Graphics 101: Camera Modeling - Focus and Depth of Field



Graphics 101: Camera Modeling - Focus and Depth of Field

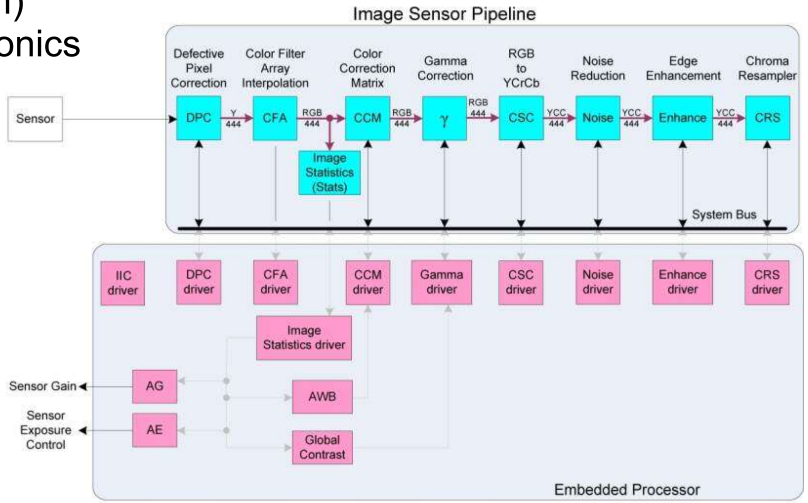


Graphics 101: Camera Modeling - Noise

Shot Noise: Poisson ~ $\mathcal{N}(\lambda, \lambda)$, where $\boldsymbol{\lambda}$ is the incident photon count
Dark Current: Thermal processes in sensor (poisson)
Read Noise: Structured noise due to read-out electronics

Low level processing can change noise model!

- Quantization
- Demosaicking
- Gamma Correction



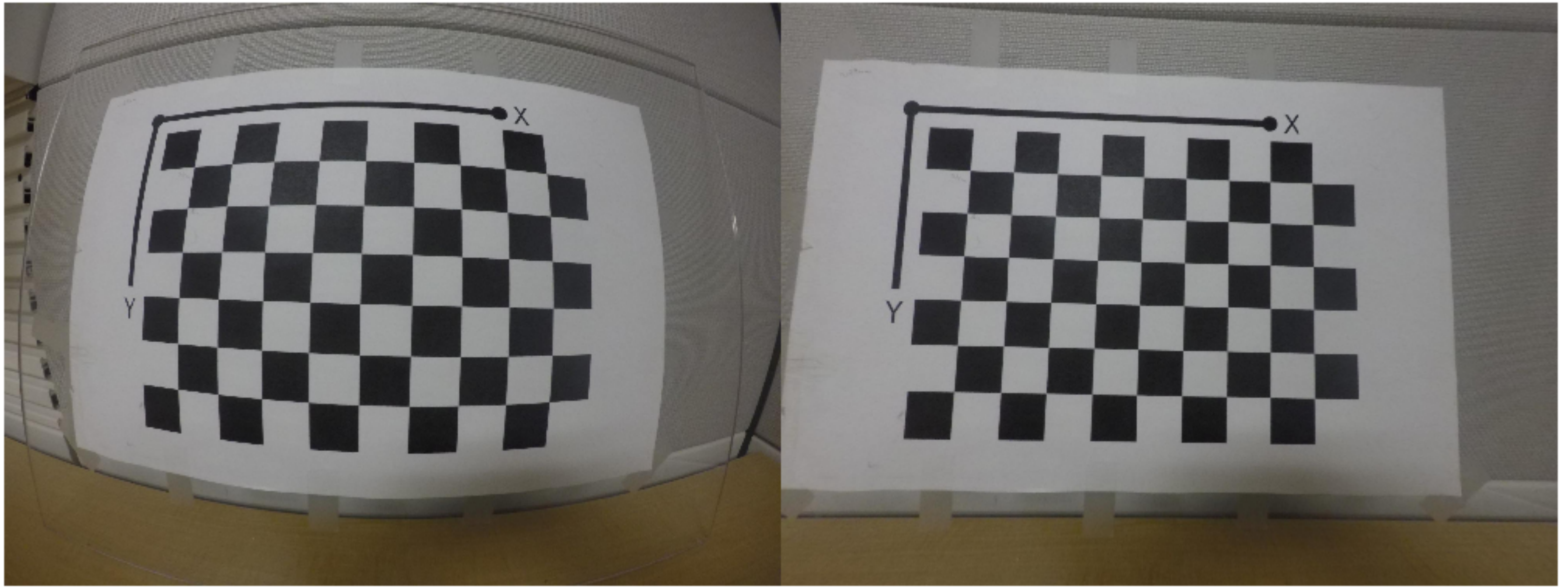
embedded.com / Xilinx

Foi et al. Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data. Trans Image Processing. 2008.

Graphics 101: Camera Modeling

Fisheye: Barrel Distortion, Chromatic Aberration, Vignetting

Original Image (left) vs. Corrected Image (right)



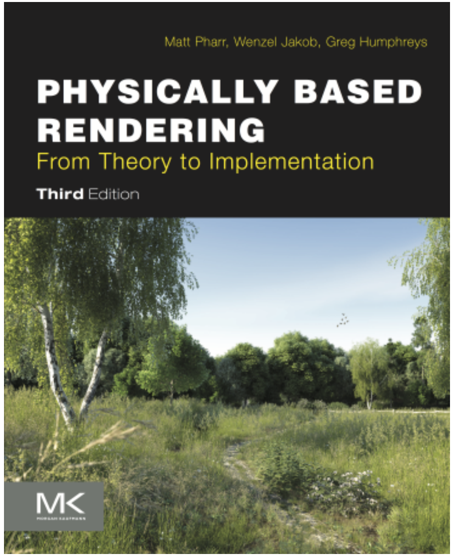
mathworks.com

Graphics 101: More References

Siggraph Rendering Courses:

see: http://renderwonk.com/publications/

PBRT: https://www.pbrt.org/



CS Graphics Courses:

Stanford CS348b: http://graphics.stanford.edu/courses/cs348b/

MIT OCW: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/

Generating Synthetic Data in Practice

Path Tracing / Physically Based

Mitsuba / PBRT: https://www.mitsuba-renderer.org/

Blender Cycles: https://www.blender.org/

Raster/Realtime

Unreal Engine: https://www.unrealengine.com/en-US/

Unity: https://unity.com/

Licenses typically free for non-commercial use

Assets: Geometry

1. “Canonical Meshes”:

- Stanford 3D Scanning Repository
  - https://graphics.stanford.edu/data/3Dscanrep/
- Cornell Box [1]
- Utah Teapot [2]

2. Create Meshes using 3D Modeling software (e.g. Blender)

3. Use repository of models (e.g. ShapeNet [3])


Stanford Bunny


Cornell Box


Utah Teapot

[1] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. *ACM SIGGRAPH 1984*
[2] Torrence, Ann. "Martin Newell's original teapot". *ACM SIGGRAPH 2006*
[3] https://shapenet.org/

Capturing Assets

Photogrammetry

- Textures
- 3D Geometry

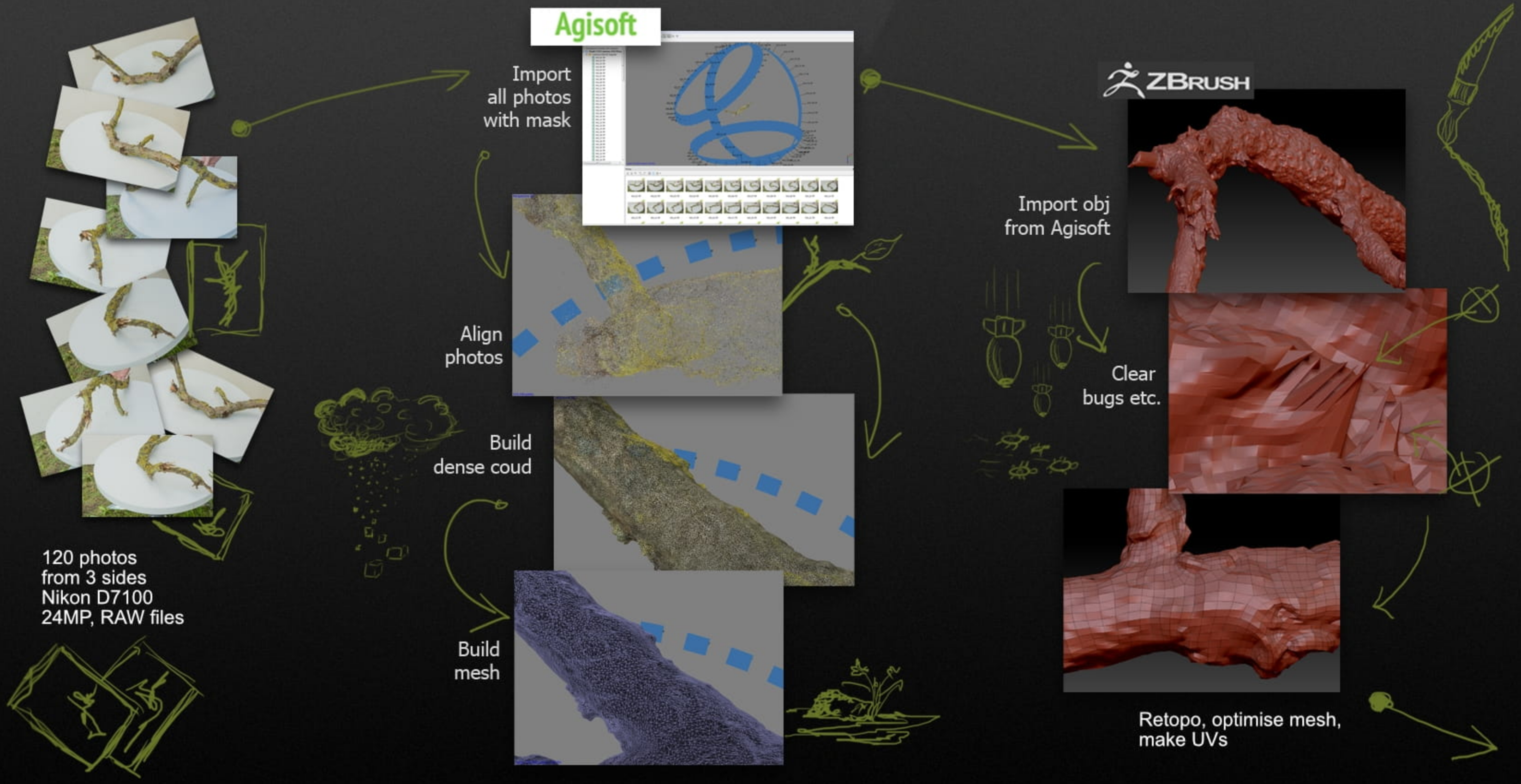
Object Specific Capture

- Digital Humans (SURREAL Dataset)
- Furniture
- ShapeNet

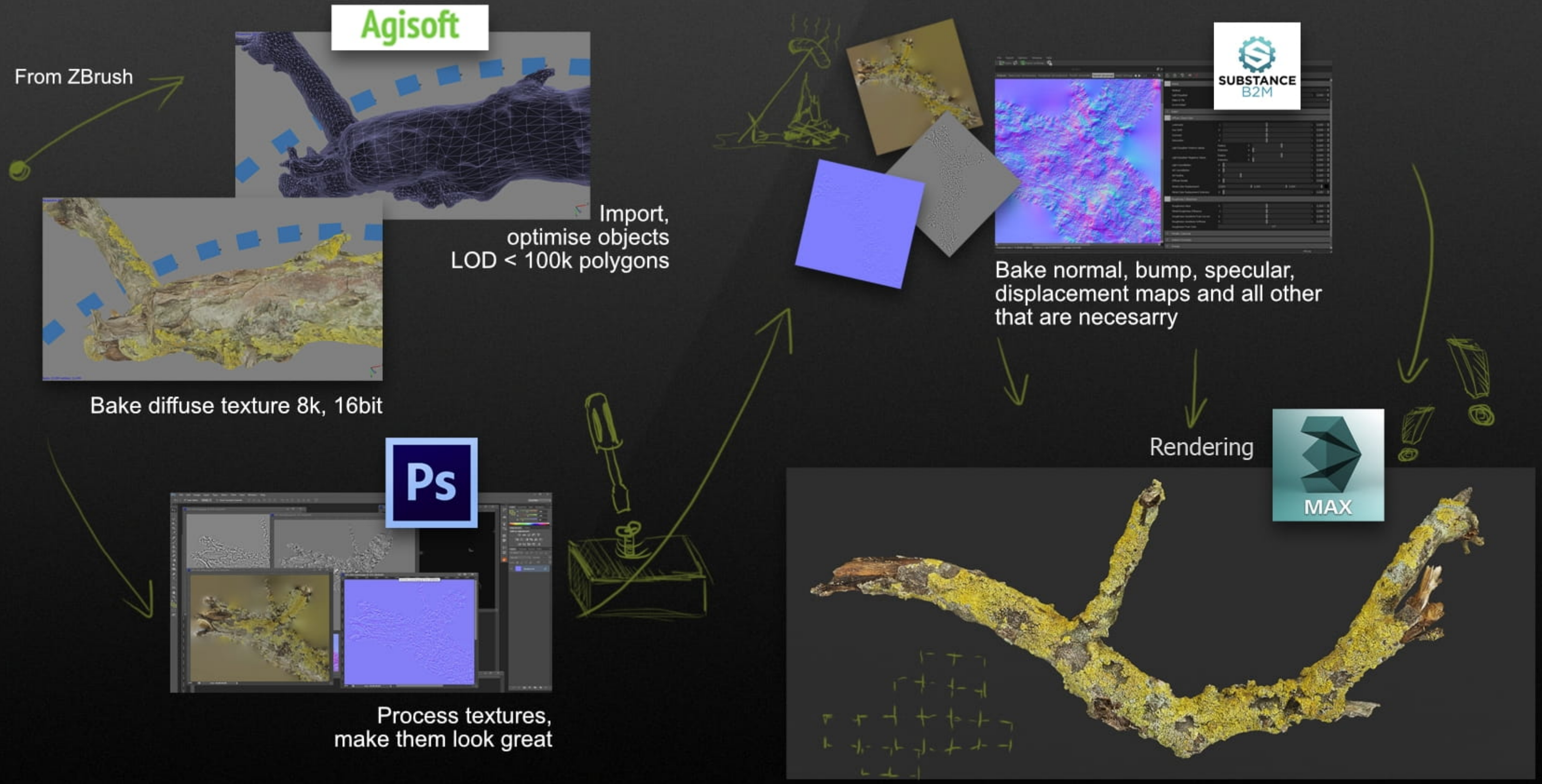
SURREAL Dataset

G. Varol et al., *Learning from Synthetic Humans*. CVPR, 2017.

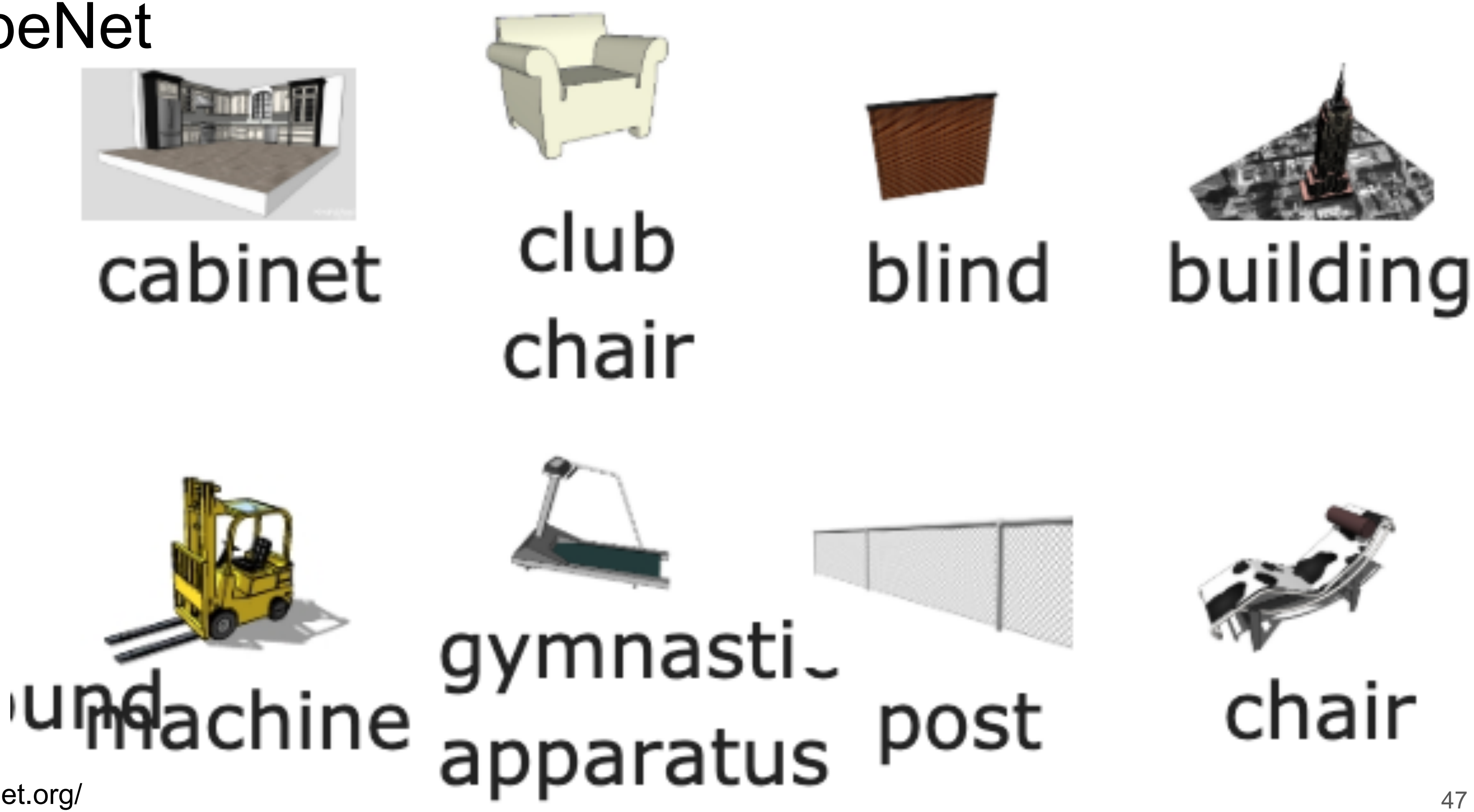
Photogrammetry



Photogrammetry



ShapeNet



https://shapenet.org/

Machine Learning and Synthetic Data

- “Real” data is almost always preferred
  - Rendering is economical in easy to parameterize scenes
  - Usually need to write a script to produce samples generatively:
    - Textures
    - 3D Geometry
    - Camera Viewpoints / Intrinsics
- Domain Transfer has been shown to work in some cases
  - GAN to translate:
    - Synthetic to Real
    - Real to Synthetic
- Difficult to learn and model noise
  - How does the model access entropy?

GPUs are useful beyond training Deep Networks!

Understanding compute architecture can be helpful: Improvements to GPU memory motivated by graphics workloads (primarily real-time)

CPUs have traditionally been used for offline rendering: Recently, improvements to hardware and implementations are making use of GPUs advantageous:

- GPUs can be run in parallel on same machine
- Larger bucket sizes than CPU (GPU memory/cache vs. CPU cache)

Typical path-tracing rendering times for typical scenes: ~5 mins to hours, can reduce to seconds for noise in renders or limiting number of bounces.

Practical Considerations

Gamma correction: know your “scene space” from your “display space”

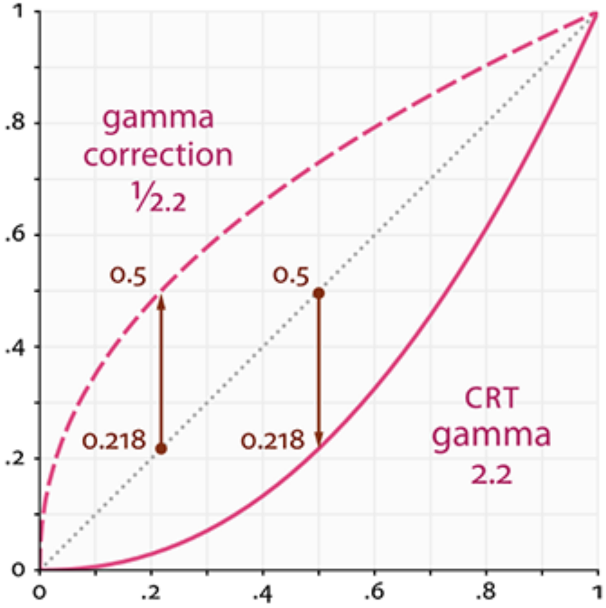
“The Importance of Being Linear”

https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch24.html

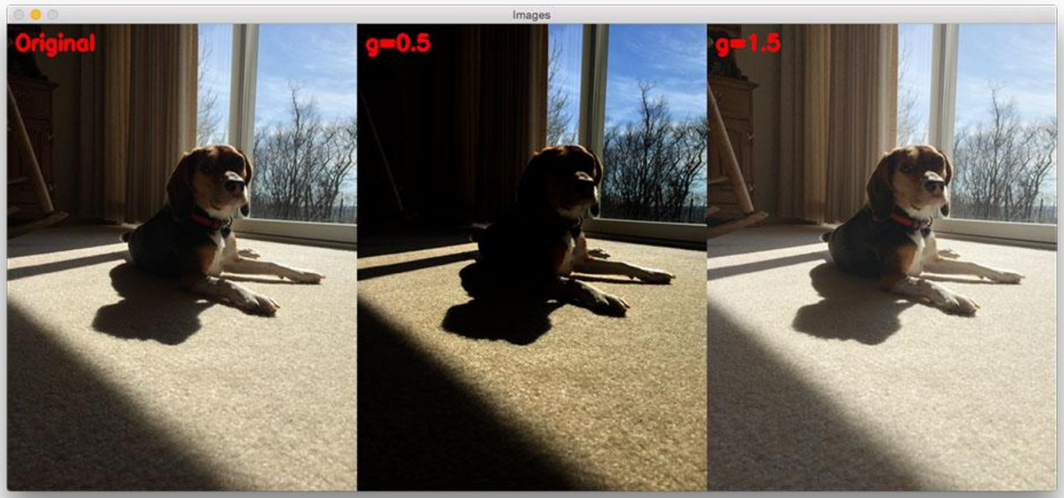
Renders are fundamentally different: RGB cameras require debayering to get to a color image, which can introduce tiny artifacts in real camera images.

Rendering with denoising: Many path tracers offer built-in denoising to speed up render time. Noise statistics of renderers using monte-carlo sampling are different than image sensors

Gamma Correction



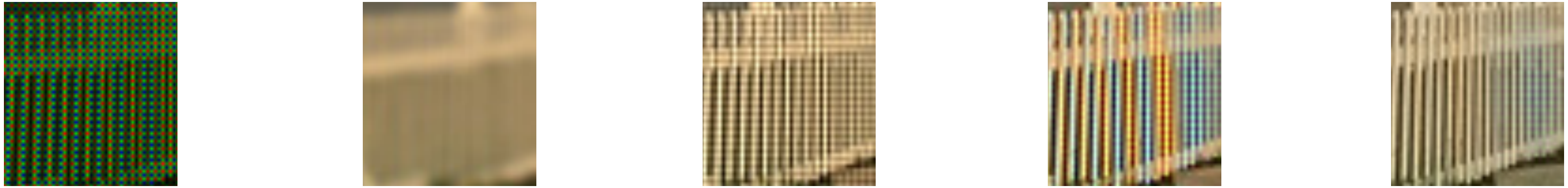
learnopengl.com



www.pyimagesearch.com

Debayering Artifacts

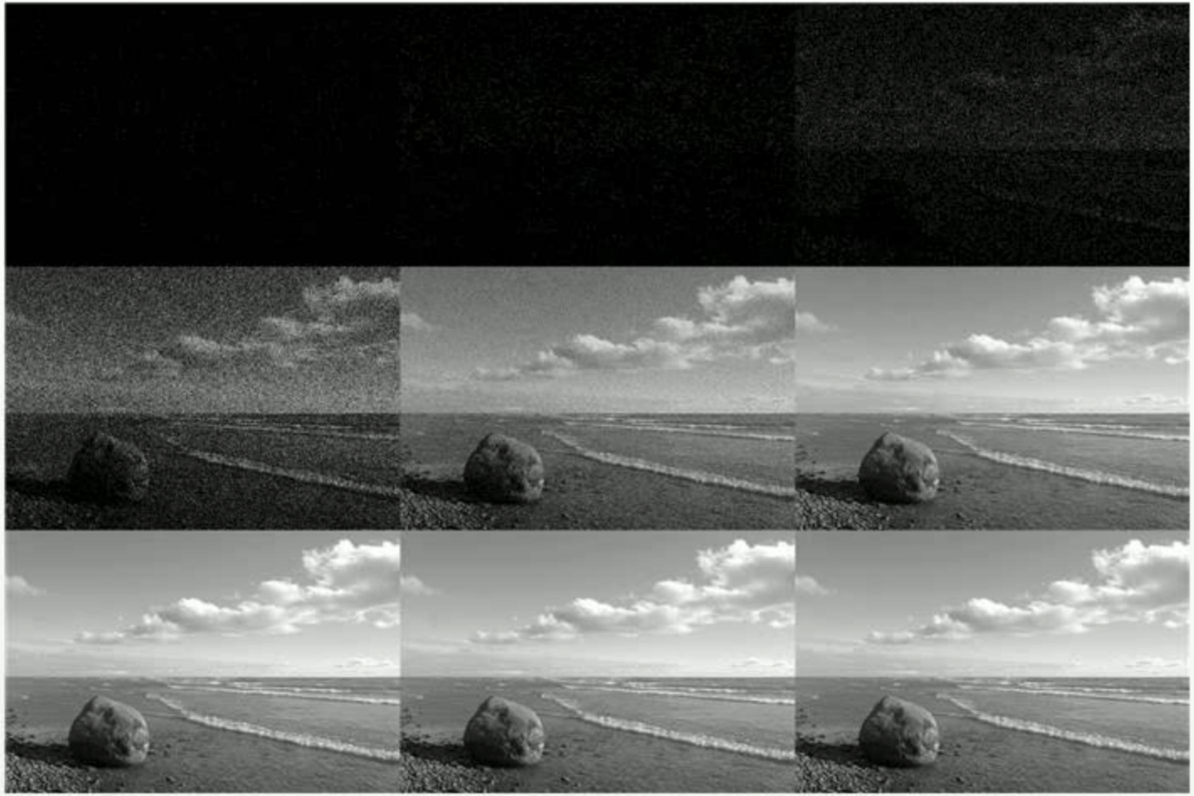
Raw | Blurring | Grid | False Color | Water Color



Sabine Süsstrunk

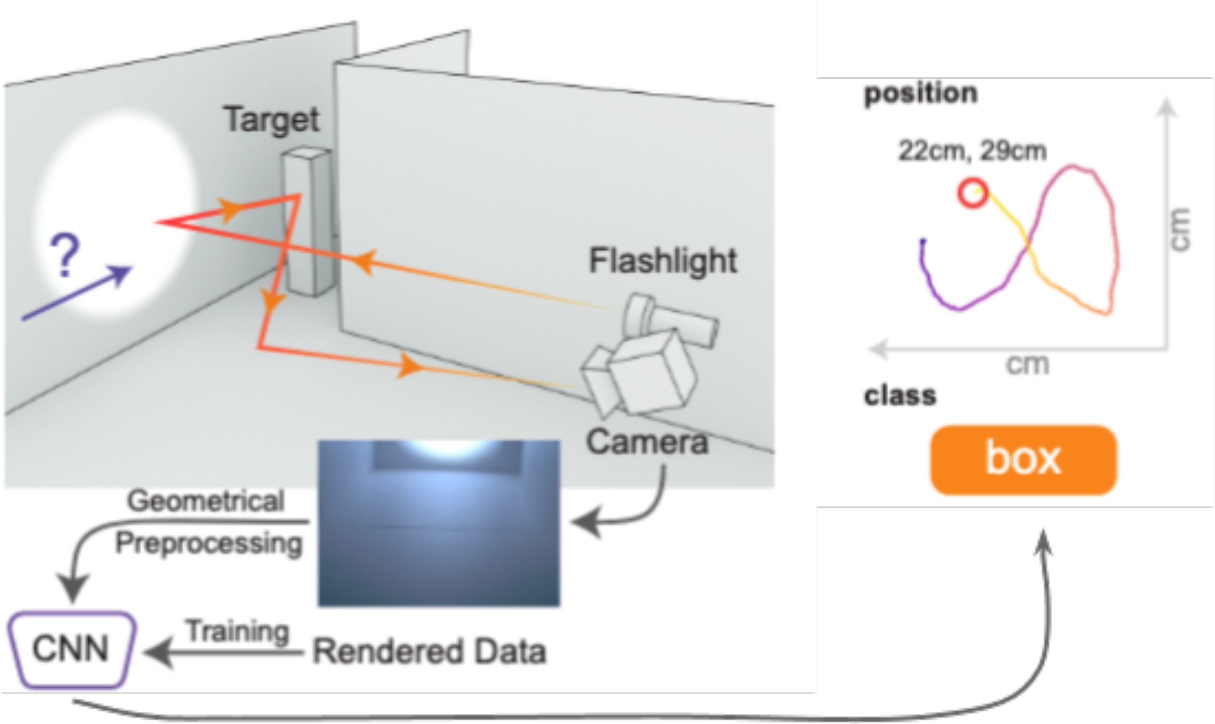
https://ivrlwww.epfl.ch/research/past_topics/demosaicing.html

Noise Models



https://en.wikipedia.org/wiki/File:Photon-noise.jpg

Case Study: Machine Learning and Synthetic Data



Tancik, Satat, Raskar. *Flash Photography for Data-Driven Hidden Scene Recovery*. Arxiv, 2018.

Case Study: Creating Diverse Simulated Data

1. Generative parameterized model: p(x, A)

1. Sample and render: p(x, y)

1. Learn: p(x | y)

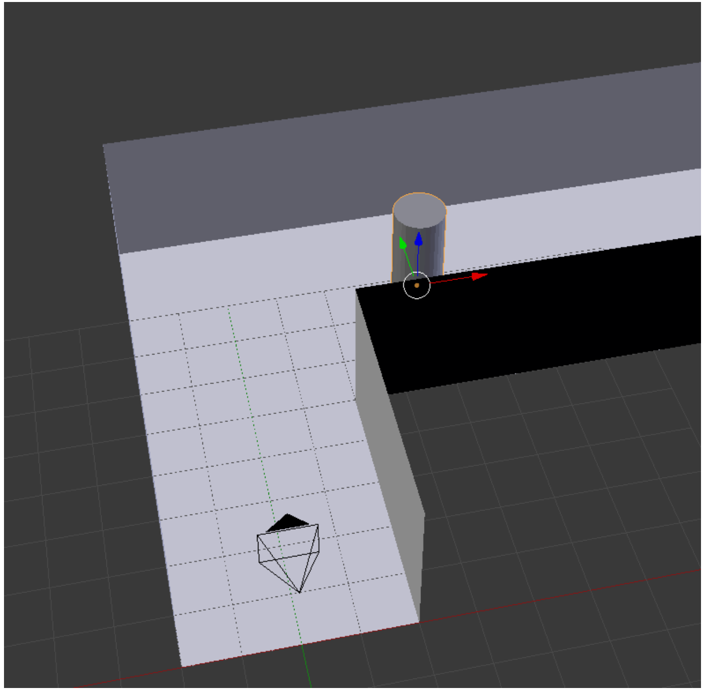
Blender Sampling Script

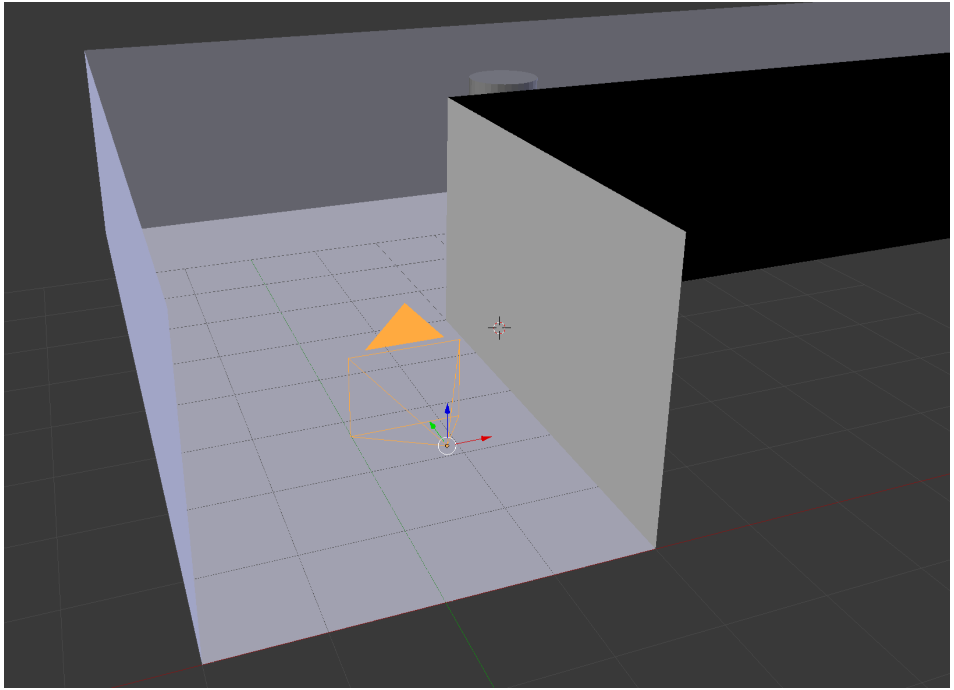
```
Geometry:
     Front Corner: x,y
     Back Corner: x,y
     Ceiling and Floor: z, z
     Extrinsic Camera Parameters: R,x,y,z
     Num Random Clutter:
          Clutter Pose: R, x,y,z

Materials:
          Blender Principled BSDF
                    Roughness: [0.2, 1]
                    Albedo:
                              Uniform: r,g,b
                              Random:
                                        Delaunay
                                        Noise


Num Targets:
     Target Postion: x,y
     Target Height: z
     Target Emmission: [0,1]
```

Random Seed:
{“b23336c-24ab”}

Tancik, Satat, Raskar. *Flash Photography for Data-Driven Hidden Scene Recovery*. Arxiv, 2018.

Case Study: Creating Diverse Simulated Data

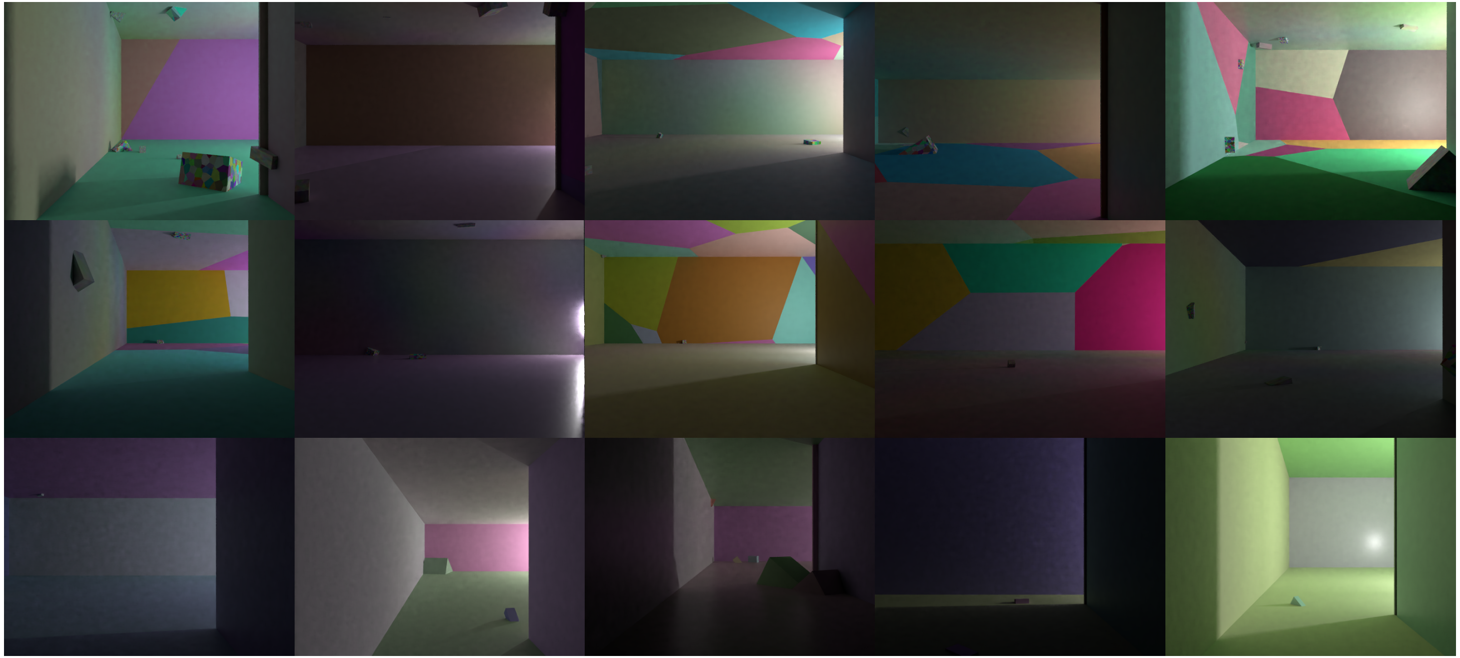




Parameterized Geometry

Tancik, Satat, Raskar. *Flash Photography for Data-Driven Hidden Scene Recovery*. Arxiv, 2018.

Case Study: Creating Diverse Simulated Data

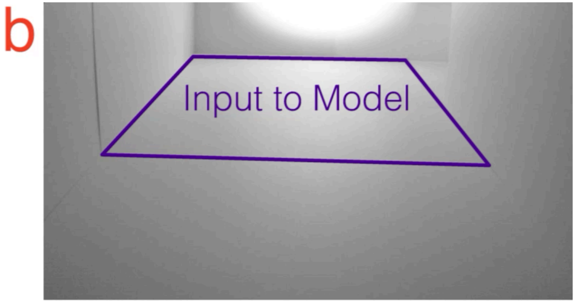


Synthetic Samples

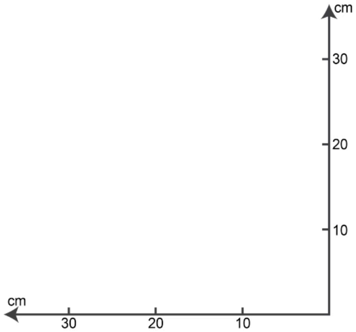
Tancik, Satat, Raskar. *Flash Photography for Data-Driven Hidden Scene Recovery*. Arxiv, 2018.

Case Study: Trained Model

Ground Truth

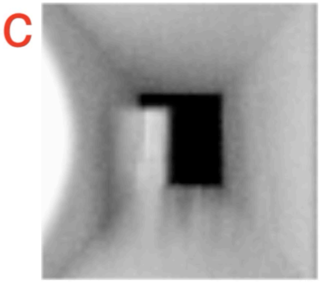


Measurement

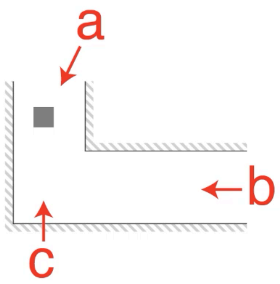


Predicted Location

Predicted Identity



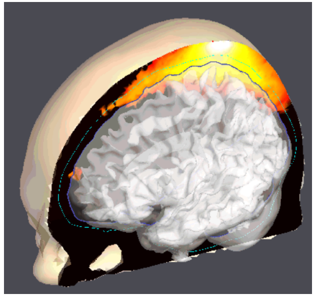
Reconstruction



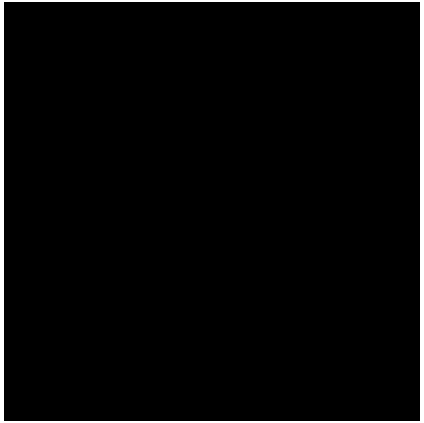
Tancik, Satat, Raskar. *Flash Photography for Data-Driven Hidden Scene Recovery*. Arxiv, 2018.

Time of Flight Rendering: Monte Carlo

MCX
http://mcx.space/

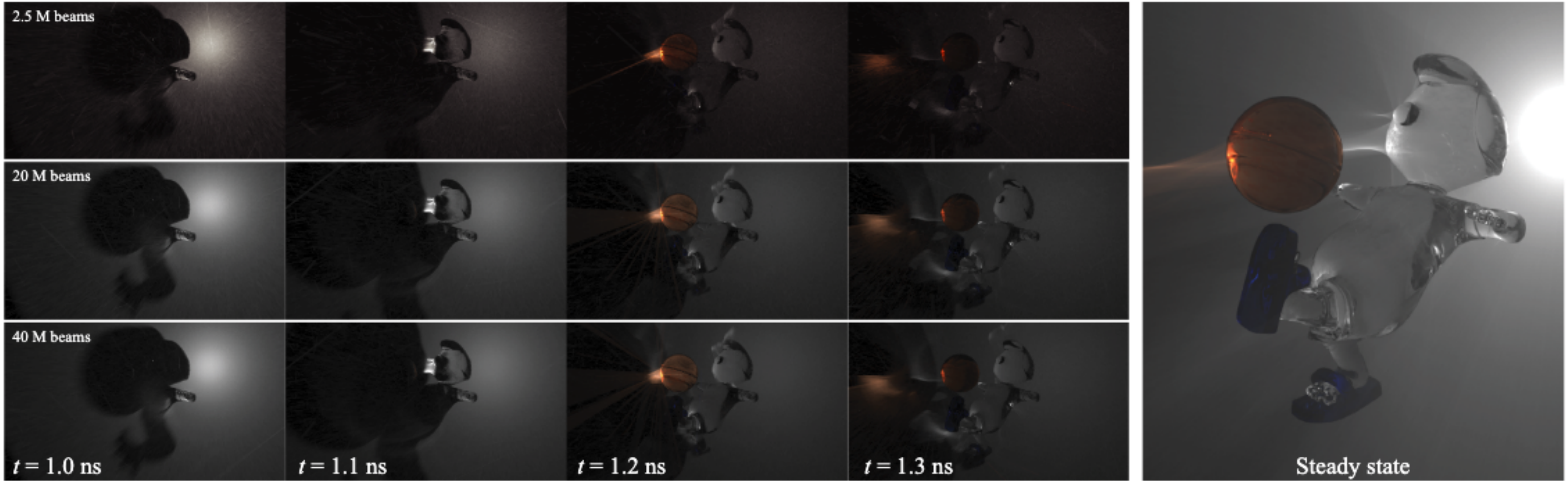


Mitsuba ToF
https://github.com/cmu-ci-lab/MitsubaToFRenderer



Camera Culture Monte Carlo Renderer
https://github.com/mitmedialab/MonteCarloRender

Time of Flight Rendering: Transient Rendering



Adrian Jarabo, Julio Marco, Adolfo Munoz, Raul Buisan, Wojciech Jarosz, Diego Gutierrez, *A Framework for Transient Rendering*. TOG (Siggraph Asia), 2014.
Julio Marco, Ibón Guillén, Wojciech Jarosz, Diego Gutierrez, Adrian Jarabo, *Progressive Transient Photon Beams.* Computer Graphics Forum, 2019.

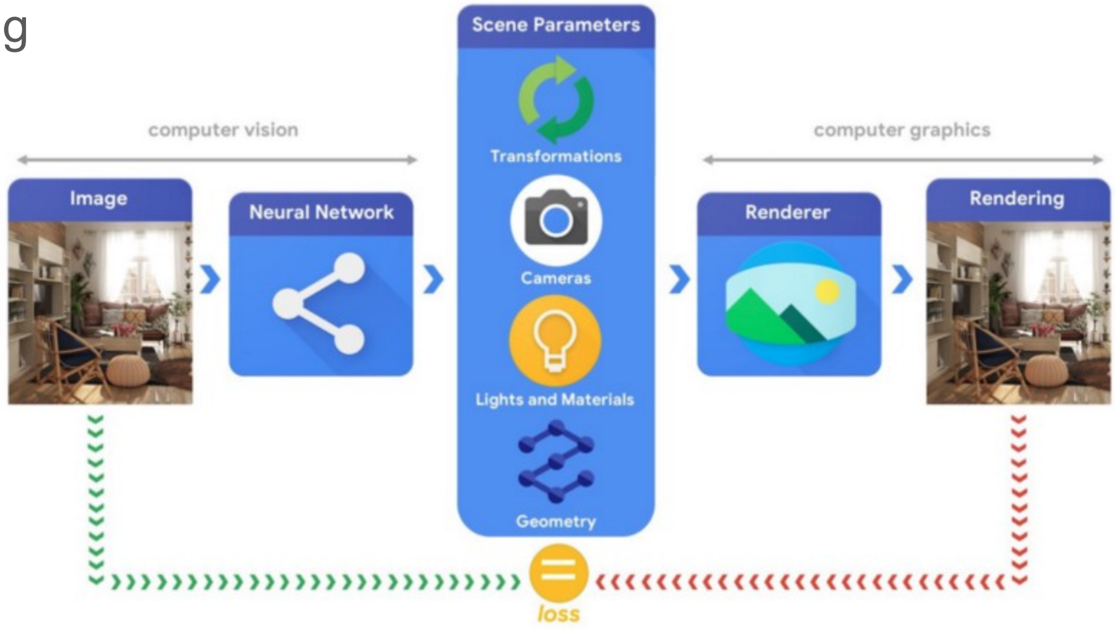
Rendering and Inverse Problems

Analysis by Synthesis: “Rendering Engine in our Head”

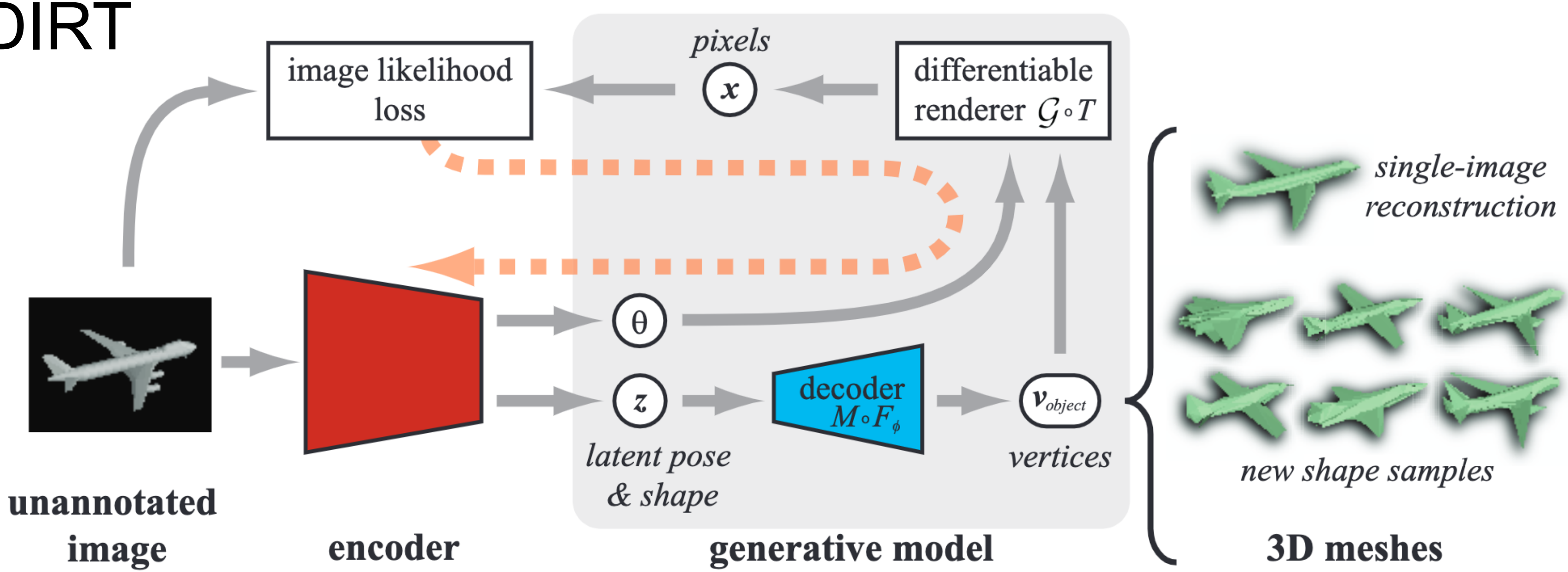
J. Wu, J. B. Tenenbaum, and P. Kohli. *Neural Scene De-rendering,* CVPR 2017

Differentiable Rendering

- DIRT
- Redner
- Inverse Transport Networks
- Tensorflow Graphics



DIRT

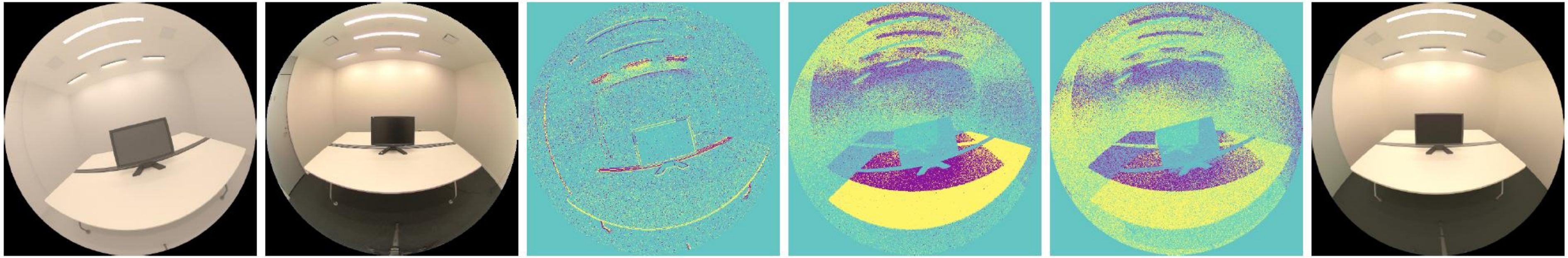


P. Henderson and V. Ferrari. *Learning to Generate and Reconstruct 3D Meshes with only 2D Supervision*, BMVC 2018.

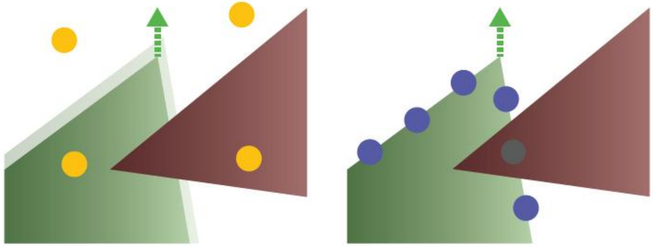
See Also:

- Loper and Black, ECCV 2014
- Kato et al., CVPR 2018
- Genova et al., CVPR 2018
- Palazzi et al., ECCV Workshops 2018

Redner



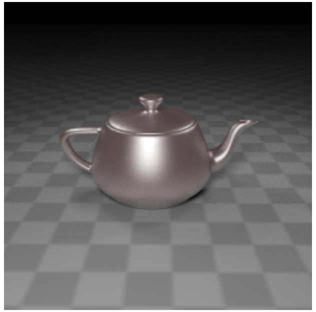
(a) initial guess (b) real photograph (c) camera gradient (per-pixel contribution) (d) table albedo gradient (per-pixel contribution) (e) light gradient (per-pixel contribution) (f) our fitted result



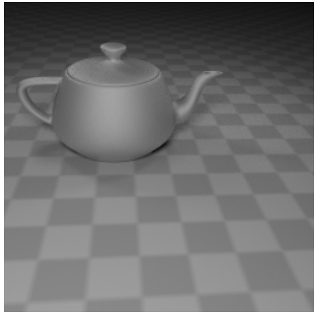
(a) area sampling (b) edge sampling

Tzu-Mao Li, Miika Aittala, Fredo Durand, Jaakko Lehtinen. *Differentiable Monte Carlo Ray Tracing through Edge Sampling,* TOG (Siggraph Asia) 2018.

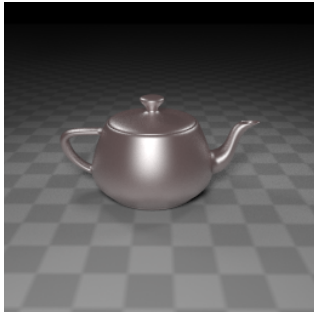
Redner



target



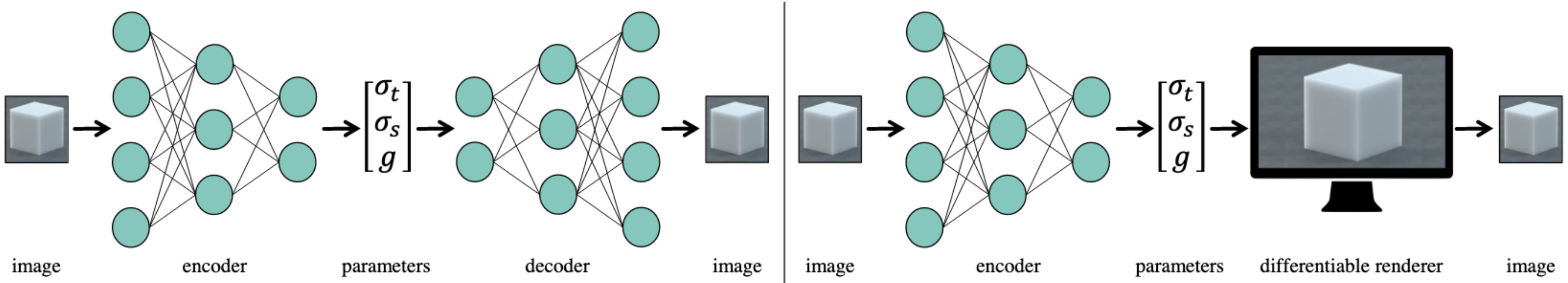
init



final

Tzu-Mao Li, Miika Aittala, Fredo Durand, Jaakko Lehtinen. *Differentiable Monte Carlo Ray Tracing through Edge Sampling,* TOG (Siggraph Asia) 2018.

Inverse Transport Networks

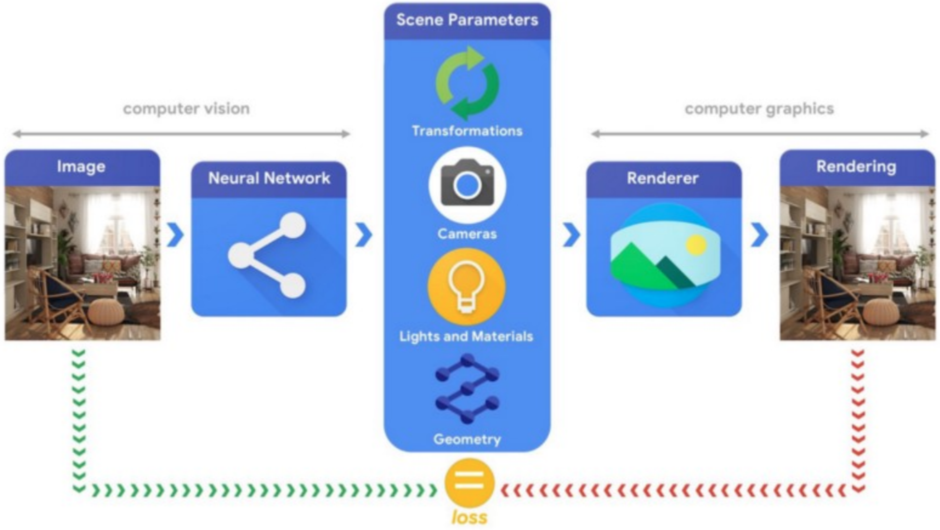


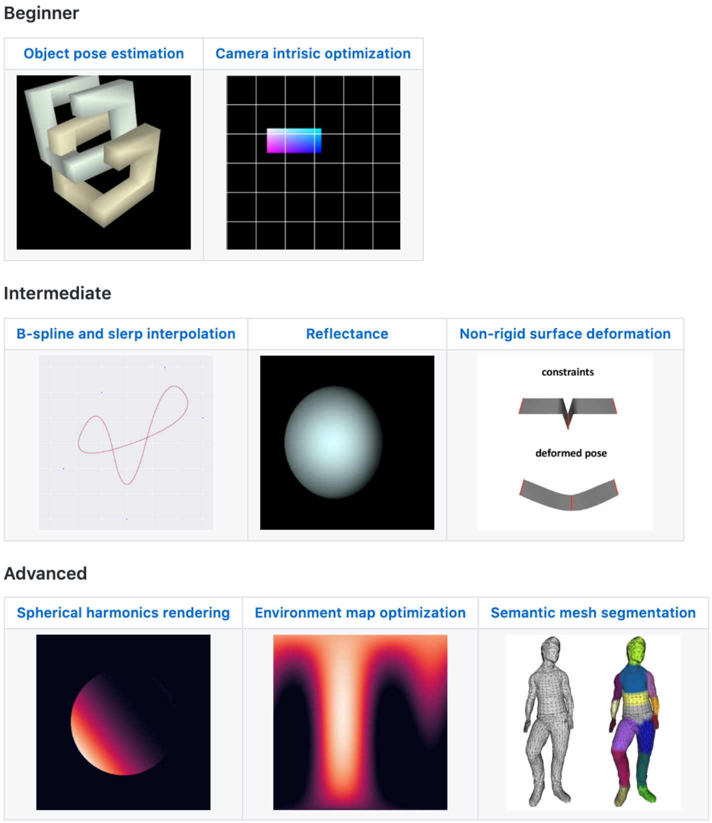
(a) traditional auto-encoder architecture

(b) proposed encoder-renderer architecture

C. Che, F. Luan, S. Zhao, K. Bala and I. Gkioulekas. *Inverse Transport Networks*. ArXiv 2018.

Tensorflow Graphics





https://github.com/tensorflow/graphics

Julien Valentin and Cem Keskin and Pavel Pidlypenskyi and Ameesh Makadia and Avneesh Sud and Sofien Bouaziz. *Tensorflow Graphics,* 2019.

Differentiable Graphics

What makes this possible?

Automatic Differentiation Frameworks!

autodiff

automatic differentiation in C++ couldn't be simpler

Forward Mode

```
dual x, y, z;
dual u = f(x, y, z);
double dudx = derivative(f, wrt(x), x, y, z);
double dudy = derivative(f, wrt(y), x, y, z);
double dudz = derivative(f, wrt(z), x, y, z);
```

Reverse Mode

```
var x, y, z;
var u = f(x, y, z);
Derivatives dud = derivatives(u);
double dudx = dud(x);
double dudy = dud(y);
double dudz = dud(z);
```

https://autodiff.github.io/

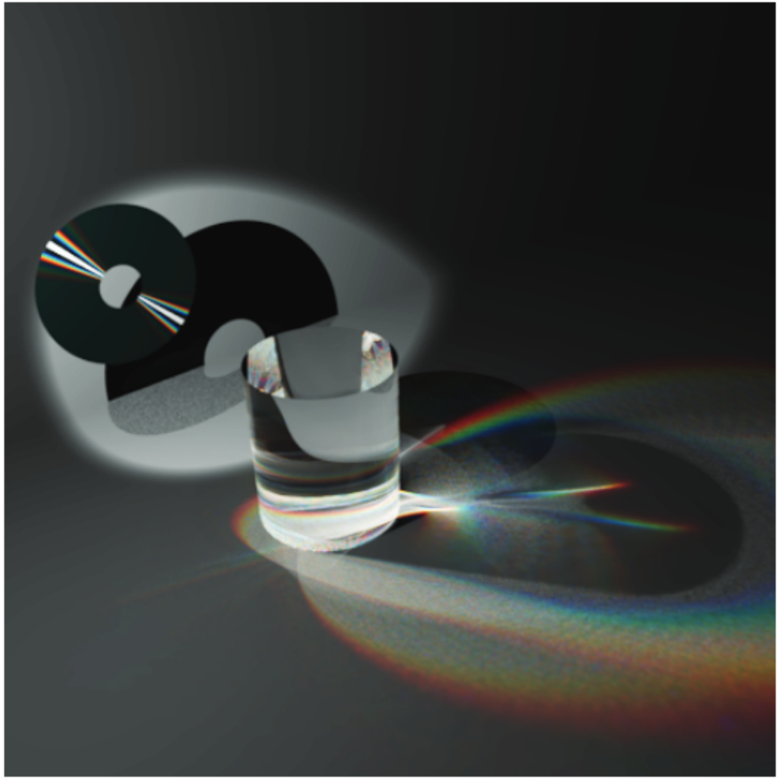
What else could we model?

Wave Optics: Coherence (e.g. diffraction, microscopy)

Light Field: Render many shifted pinhole cameras across camera aperture

Fluorescence: Materials that absorb light and emit in a longer wavelength

Non-linear Effects: Two-photon Microscopy



T. Cuypers, T. Haber, P. Bekaert, Se Baek Oh, R. Raskar, *Reflectance model for diffraction*. ACM Trans. Graphics (TOG), 2012.
T. Cuypers, R. Horstmeyer, Se Baek Oh, P. Bekaert, R. Raskar, *Validity of Wigner Distribution Function for ray-based imaging*. ICCP, 2011.
Se Baek Oh and R. Raskarl, *Rendering Wave Effects with Augmented Light Field*. Eurographics, 2010.

Summary

- Graphics is useful for creating training and test data
  - Particularly when real data is expensive to collect
  - Relevant problem for domain adaptation and transfer learning
- Physically Based Rendering (and photorealism) is is achievable with easily accessible tools
  - And increasing availability of datasets
- Computer Vision + Graphics is an exciting frontier!
  - Differentiable Rendering promises to close the loop