Transparent Object Reconstruction

in GGX Microfacet Model

Ren Liu

2020.5

Theorem

- 1. GGX Microfacet BSDF Model

  Walter B, Marschner S R, Li H, et al. Microfacet Models for Refraction through Rough Surfaces[J]. Rendering techniques, 2007, 2007: 18th.

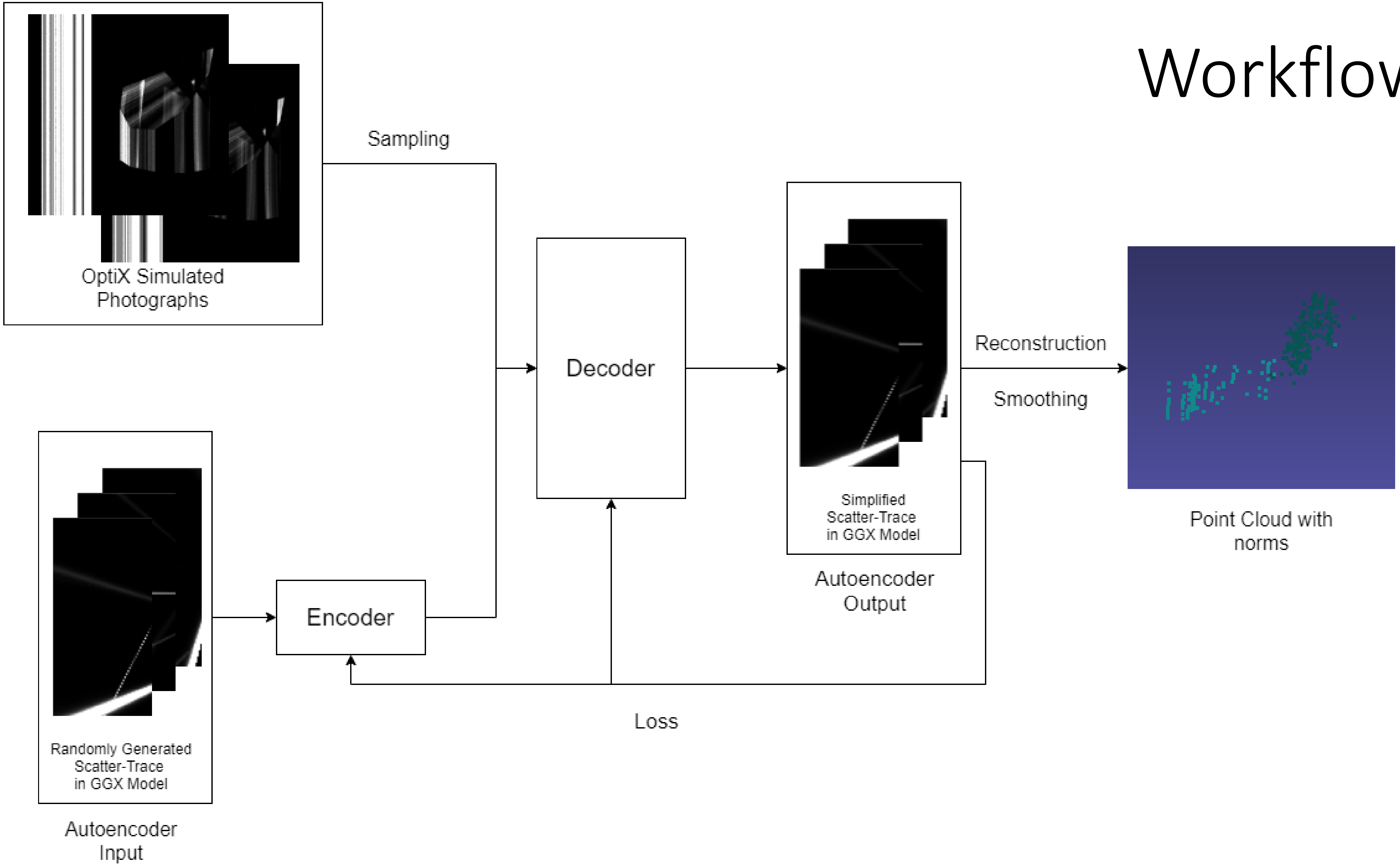
- 2. Scatter-Trace Method for Transparent Object Surface Reconstruction

  Morris N J W, Kutulakos K N. Reconstructing the surface of inhomogeneous transparent scenes by scatter-trace photography[C]//2007 IEEE 11th International Conference on Computer Vision. IEEE, 2007: 1-8.

- 3. L-DAE alike autoencoder – a physically acquired encoder and a decoder with dense layers.

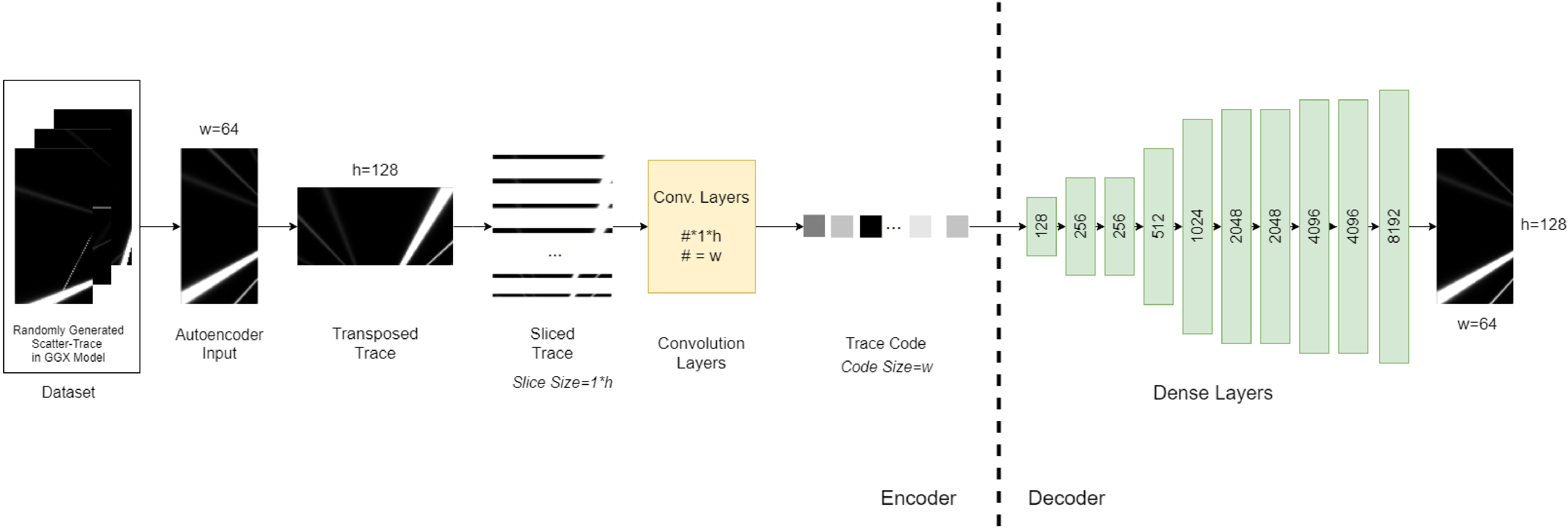
  Kang K, Chen Z, Wang J, et al. Efficient reflectance capture using an autoencoder[J]. ACM Trans. Graph., 2018, 37(4): 127:1-127:10.

Workflow

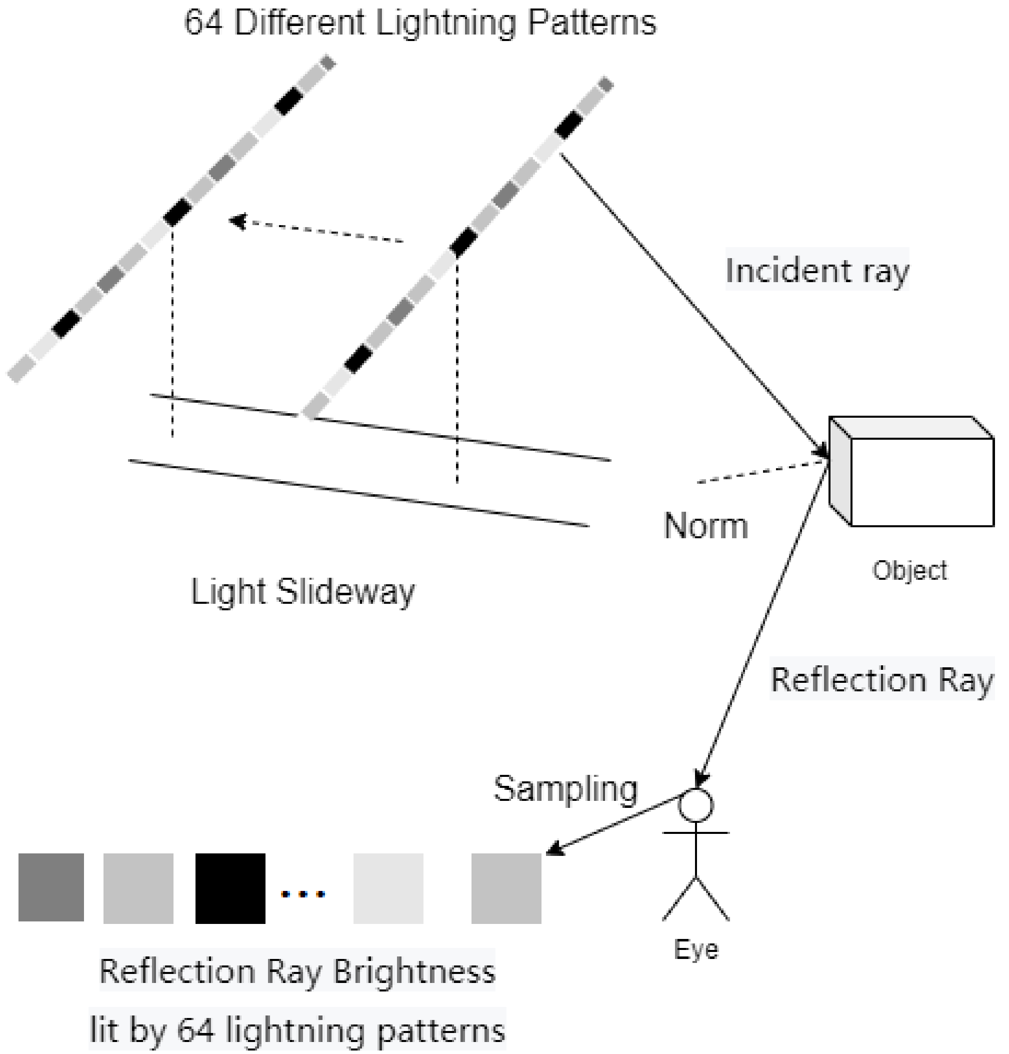


OptiX Simulated
Photographs
Sampling
Decoder
Simplified
Scatter-Trace
in GGX Model
Autoencoder
Output
Reconstruction
Smoothing
Point Cloud with
norms
Randomly Generated
Scatter-Trace
in GGX Model
Autoencoder
Input
Encoder
Loss

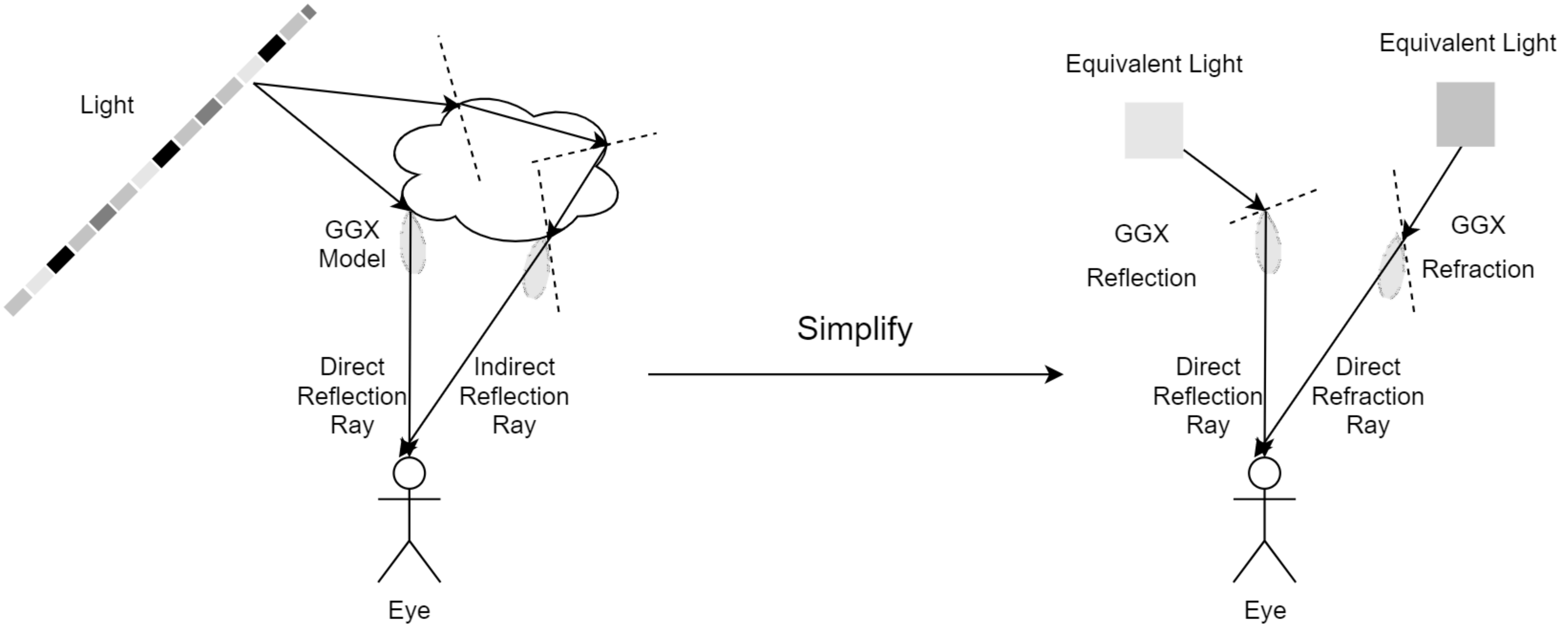
Autoencoder - Structure



Autoencoder – Physical Part



Autoencoder - Dataset

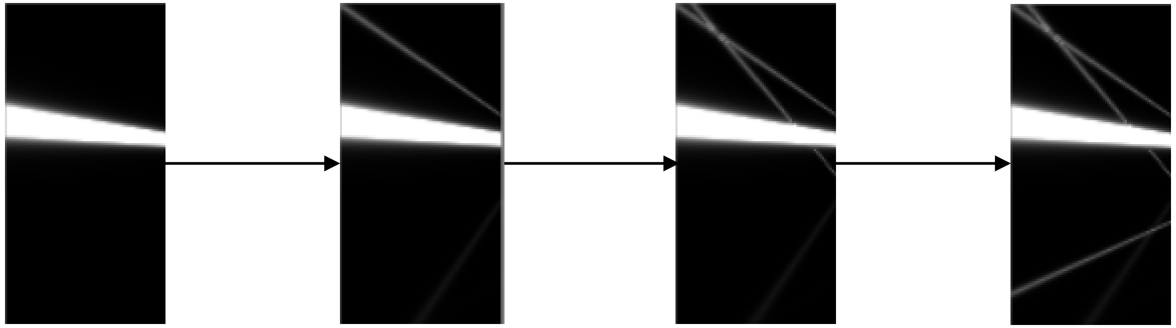


Autoencoder – Dataset - Assumptions

- For every single spatial point on the object's surface, its scatter-trace can be simplified as combination of several equivalent ***direct*** reflection & refraction traces.
- For not so complex objects, the number of remarkable traces on the scatter-trace for one point is limited.
- According to GGX Model, every GGX reflection/refraction trace can be determined by the following 11-dim parameters:
  - Eyesight vector, light vector, major norm, light intensity, GGX distribution (Ag)

Autoencoder – Dataset - Generator

- To randomly generate a scatter trace record, 5-6 virtual reflection/refraction micro-surfaces should be firstly randomly generated in a limited area. Then add their contributions to the scatter-trace record **separately**.



A randomly generated scatter-trace

Autoencoder – Training

- To avoid overfitting, the training set is dynamically generated while training.

- To maintain as much feature of the scatter-trace as possible, a 2-stage training process with different loss functions is designed.

  - Stage 1: Loss = $\sqrt{\frac{\sum[(output_{ij}-label_{ij})(3*label_{ij}+1)]^2}{w*h}}$: to emphasize bright feature
  - Stage 2: Loss = $\sqrt{\frac{\sum[(output_{ij}-label_{ij})(2*label_{ij}+2)]^2}{w*h}}$: to reduce interference

Autoencoder – Training

Trained 64 lightning patterns

Speed

170-190 s / epoch

Loss

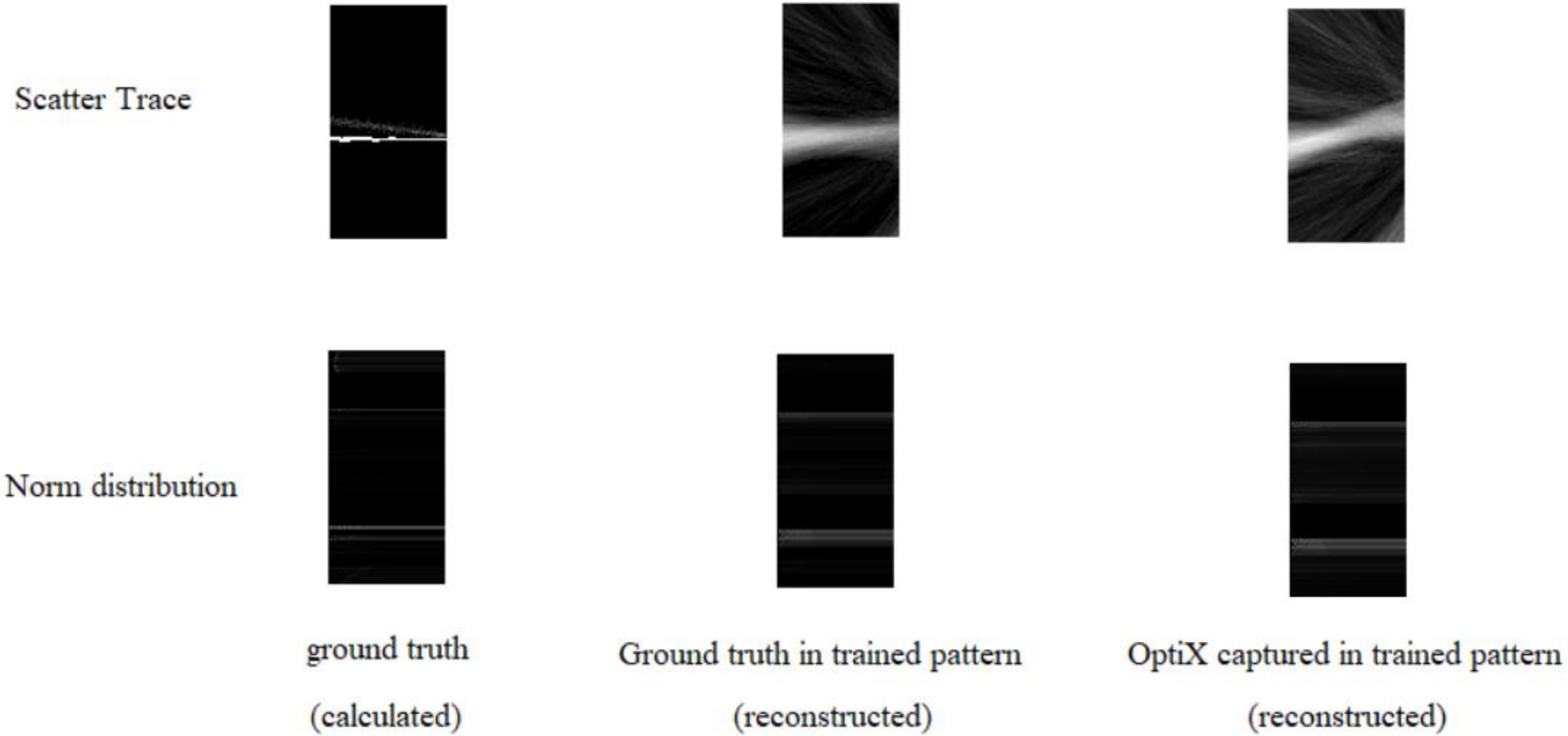
Stage 1: 0.7543 – 0.0866 (in 200 epochs)
Stage 2: 0.1230 – 0.0845 (in 100 epochs)

Platform & Device

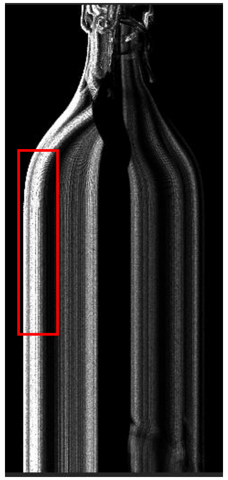
PyTorch 1.5.0
Cuda 10.0
Nvidia GeForce GTX 1070
Windows 10
(also Google Colab)

Reconstruction

- Simulated by Nvidia OptiX 6.0.0



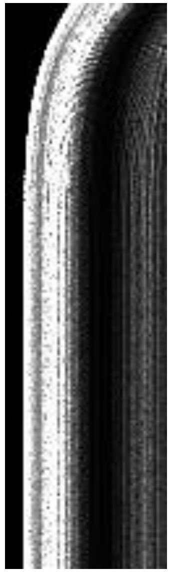
Bottle (no GGX)



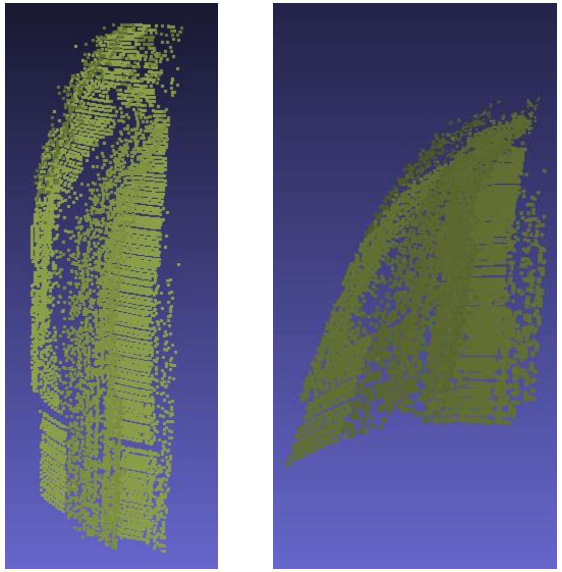
Valid Percentage: 73.8%

Average Absolute Coordinate Error: 0.055

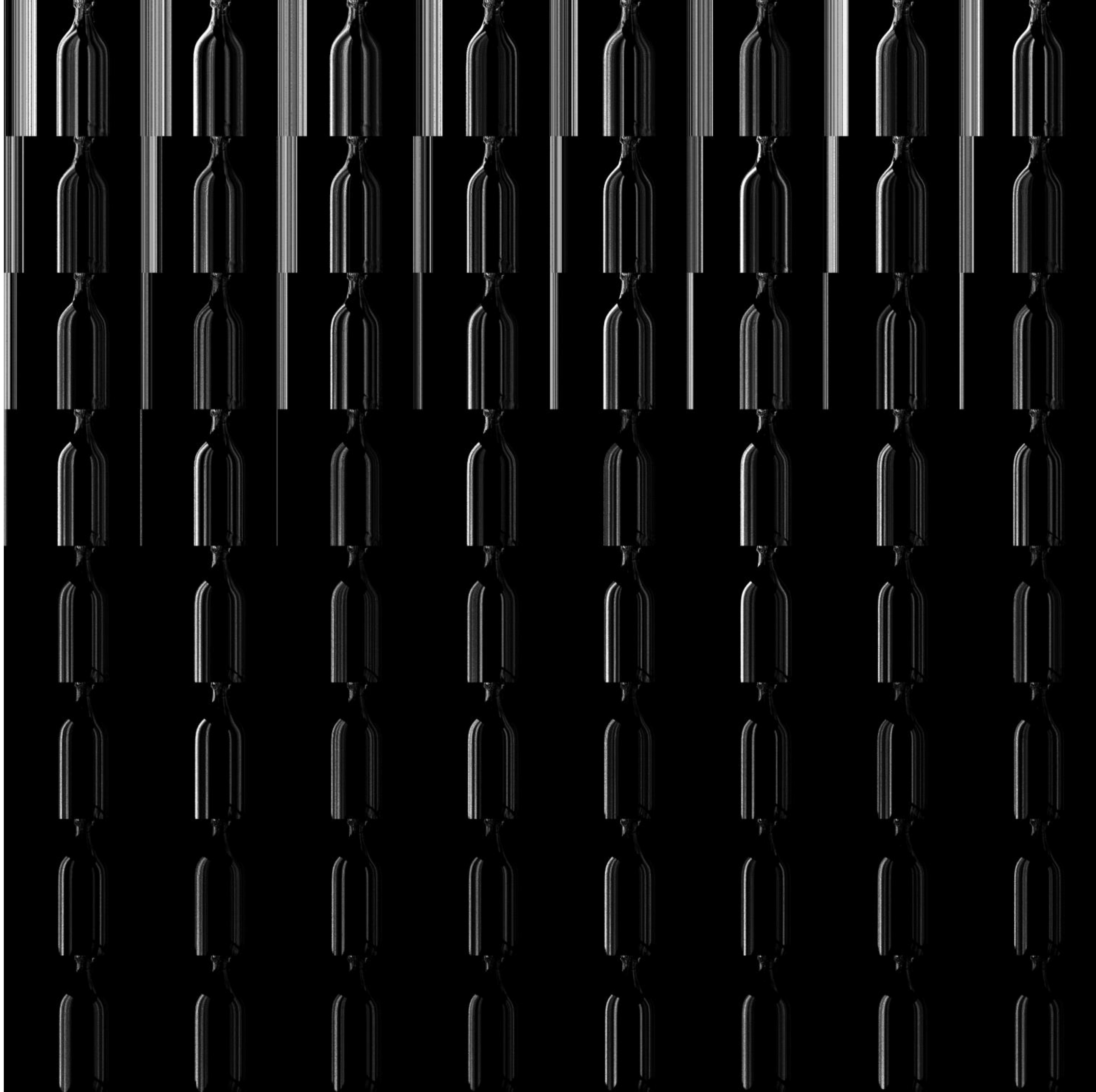
Average Absolute Normal Error: 7.58 (deg)



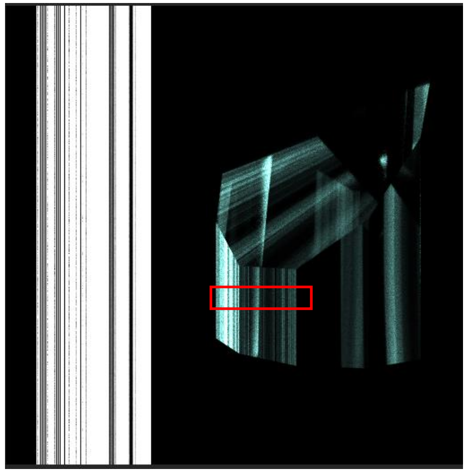
Sample Area



Reconstruction Results

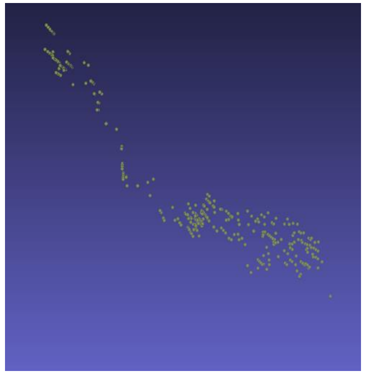


Prism (Ag =0.0023)

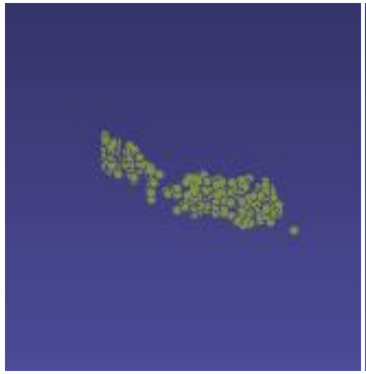


Sample Area

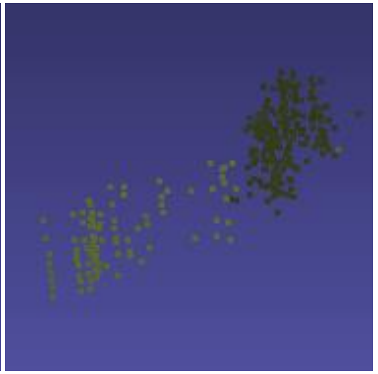
Reconstruction



Top view

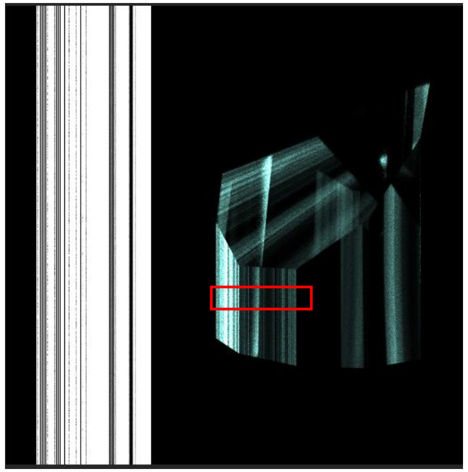


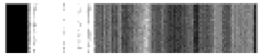
Front view



Normal view

Prism (Different GGX parameters)

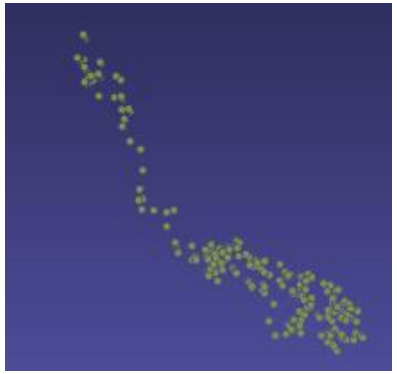




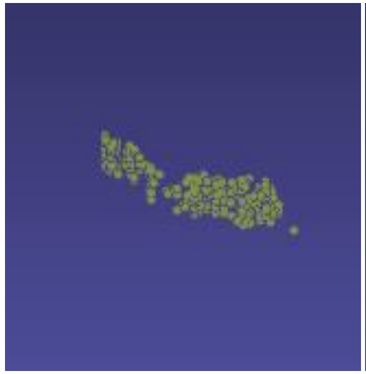
Sample Area

Ground Truth index: 1.5

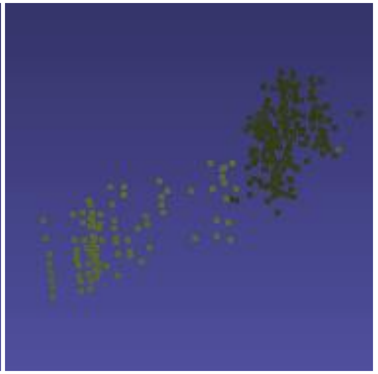
Reconstruction (Ag =0.0023)



Top view

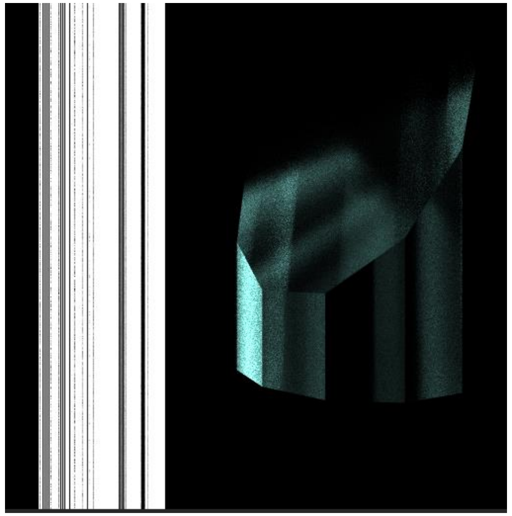


Front view

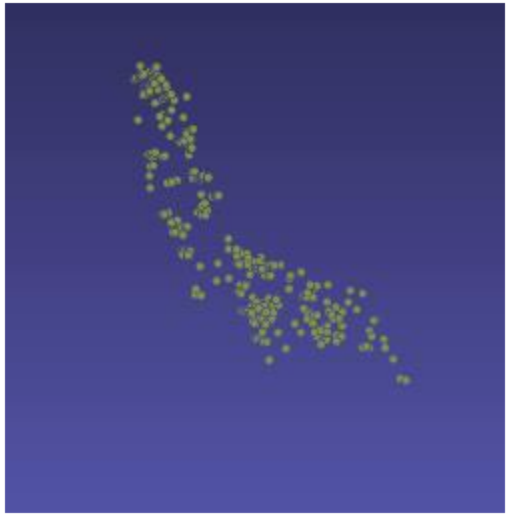


Normal view

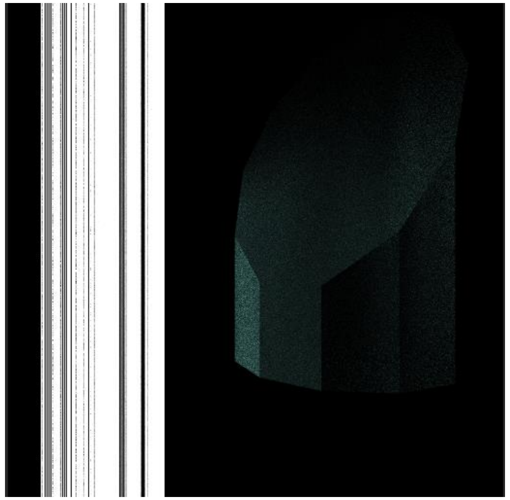
Reconstructed index: 1.51



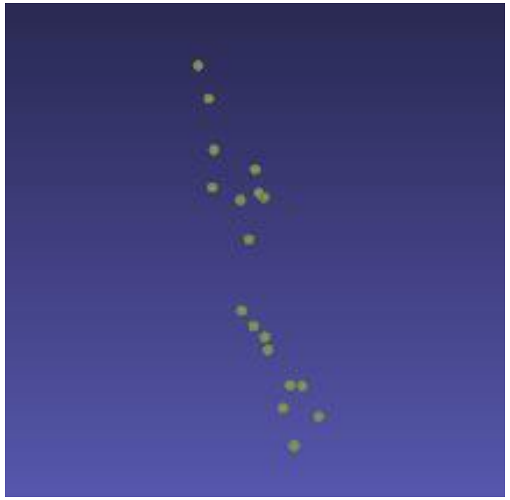
Ag =0.023



Reconstructed index: 1.53



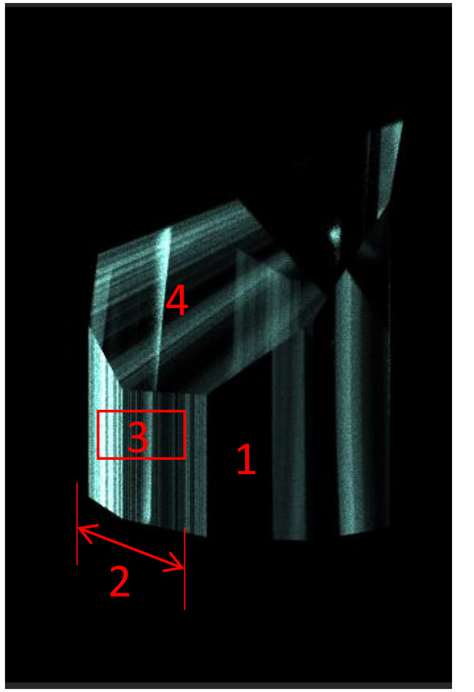
Ag =0.23



(not good for large Ag)

Reconstructed index: 1.61

Error Analysis



- 1. Cannot reconstruct <10° or > 80° normal.
- 2. Limited Object Scale (cannot be too delicate because of autoencoder's scatter-trace reconstruction is not very precise).
- 3. Limited Target Area Size (did not use tags to calibrate height of the pixel).
- 4. Not good when the normal is out of x-z plane.