RenderNet: A deep convolutional network for differentiable rendering from 3D shapes

UNIVERSITY OF BATH

Thu Nguyen-Phuoc
University of Bath
T.Nguyen.Phuoc@bath.ac.uk

Chuan Li
Lambda Labs
c@lambdalabs.com

Stephen Balaban
Lambda Labs
s@lambdalabs.com

Yong-Liang Yang
University of Bath
Y.Yang@cs.bath.ac.uk

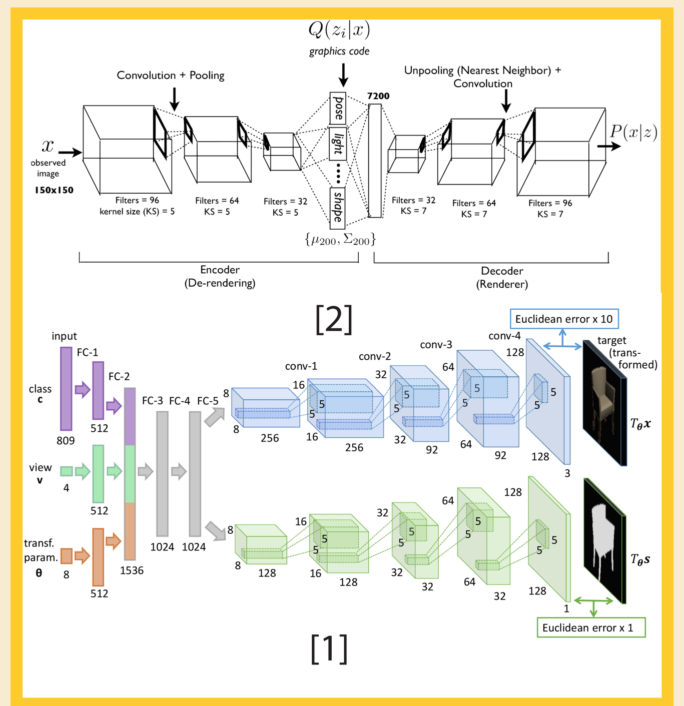
Lambda

ADVANTAGES

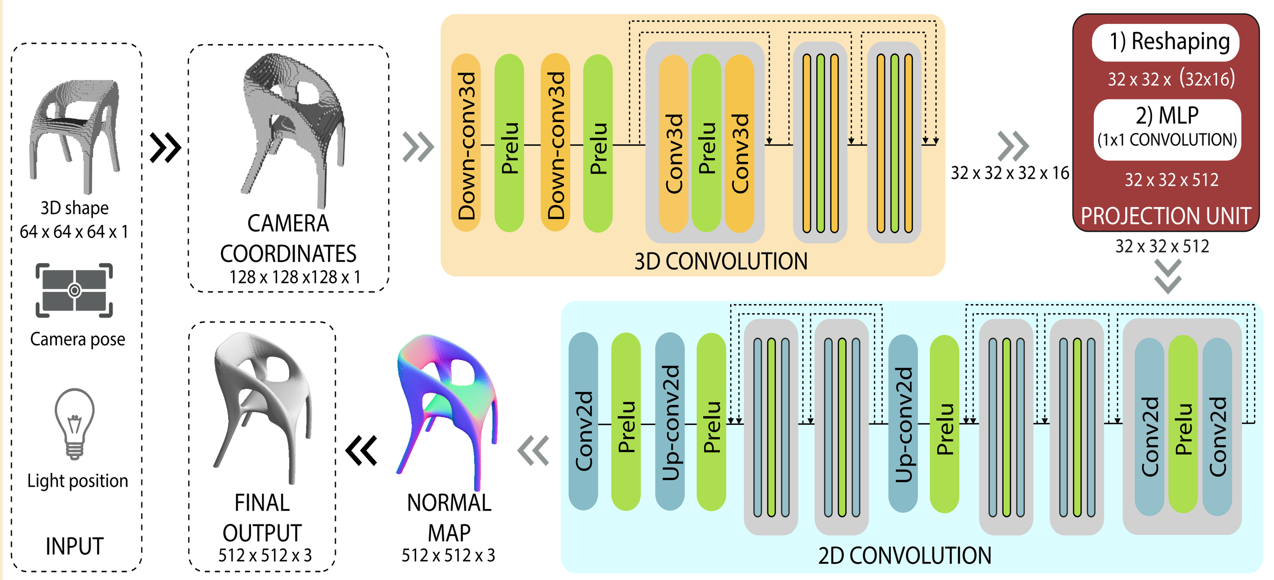
- A novel CNN architecture that enables both rendering and inverse rendering.
- Generalizes well to objects of unseen category and more complex scene geometry.
- Capable of producing textured images from textured voxel grids, where the input textures can be RGB colors or deep features computed from semantic inputs.
- Easy to integrate into other modules for applications, such as texturing or image-based reconstruction.

CURRENT APPROACHES

- Focus on losses and training regimes
- Make few assumptions about the 3D world and the image formation process
- Rotation in latent space using a CNN is hard! [1, 2]
- Do not generalise well to different object categories
- Current differentiable renderers are limited to a single fixed shader. [3, 4]



METHOD



- Rigid-body transformation (world coordinate system to camera coordinate system) followed by trilinear resampling.
- 3D convolutions morph the scene and enable perspective camera views.
- 2D convolutions compute shading color for each pixel
- Projection unit:

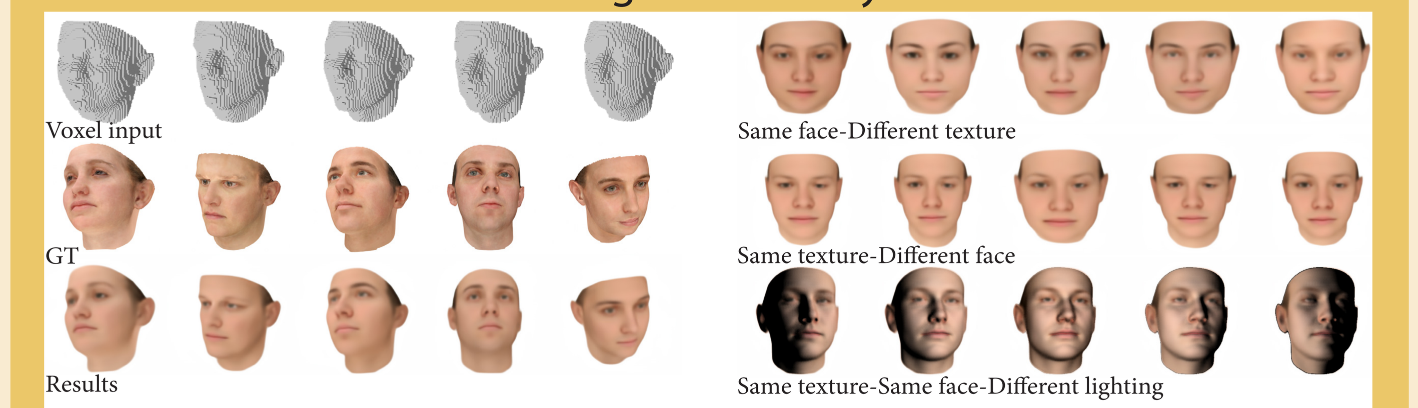
$$I_{i,j,k} = f\left(\sum_{dc} w_{k,dc} \cdot V'_{i,j,dc} + b_k\right)$$

- Pixel-wise loss function:

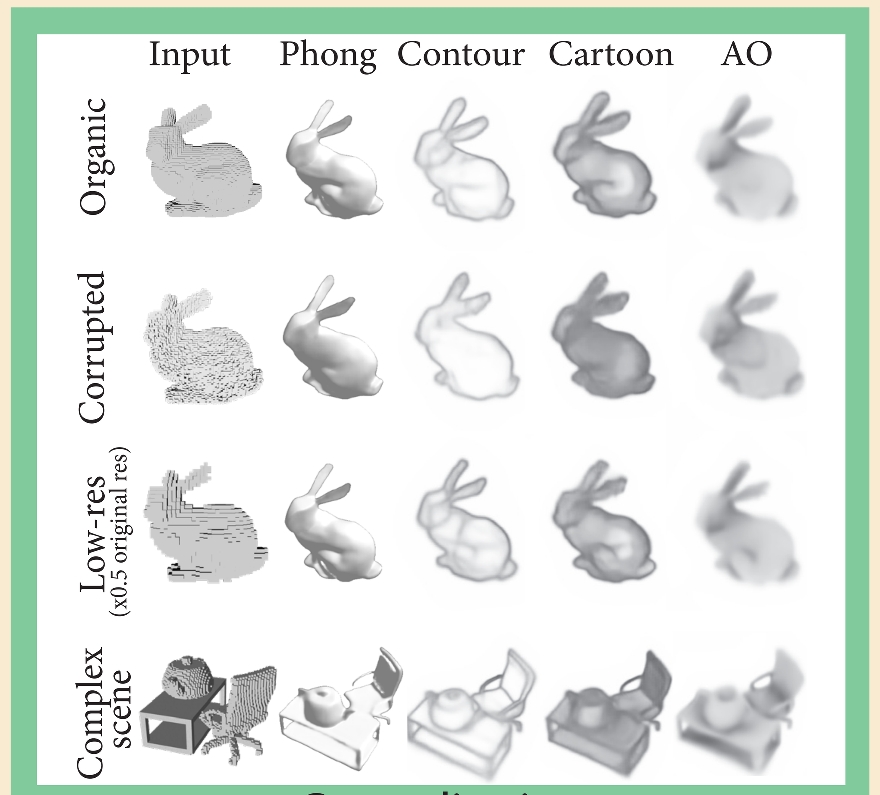
$$L_{recon} = \frac{1}{n}\sum_{i}^{n} ||y_i - y'_i||^2$$

RENDERING RESULTS

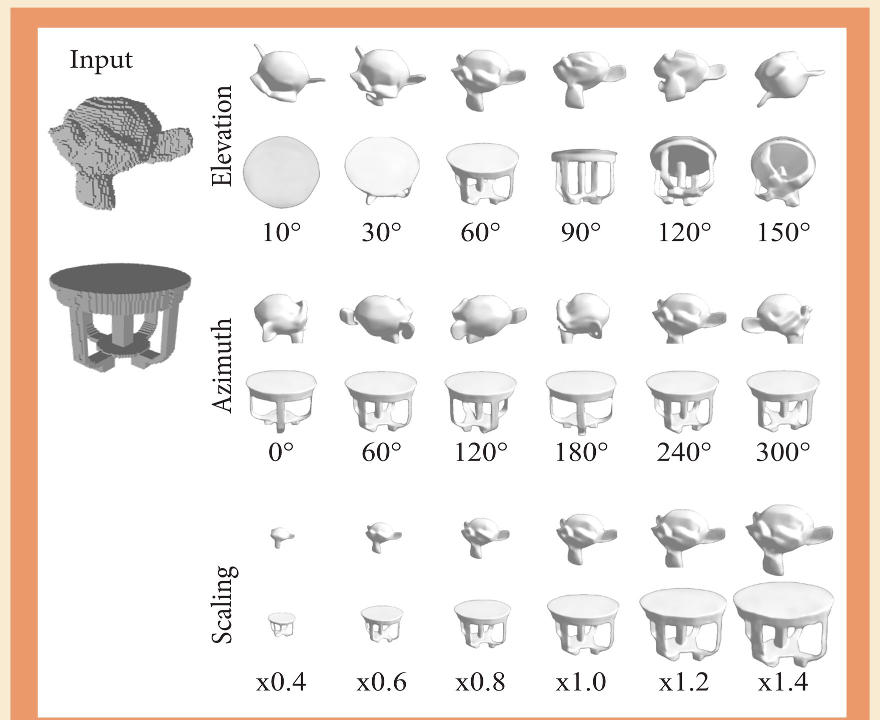
Rendering different styles



Rendering textures



Generalisation



Rigid transformation test

INVERSE RENDERING RESULTS

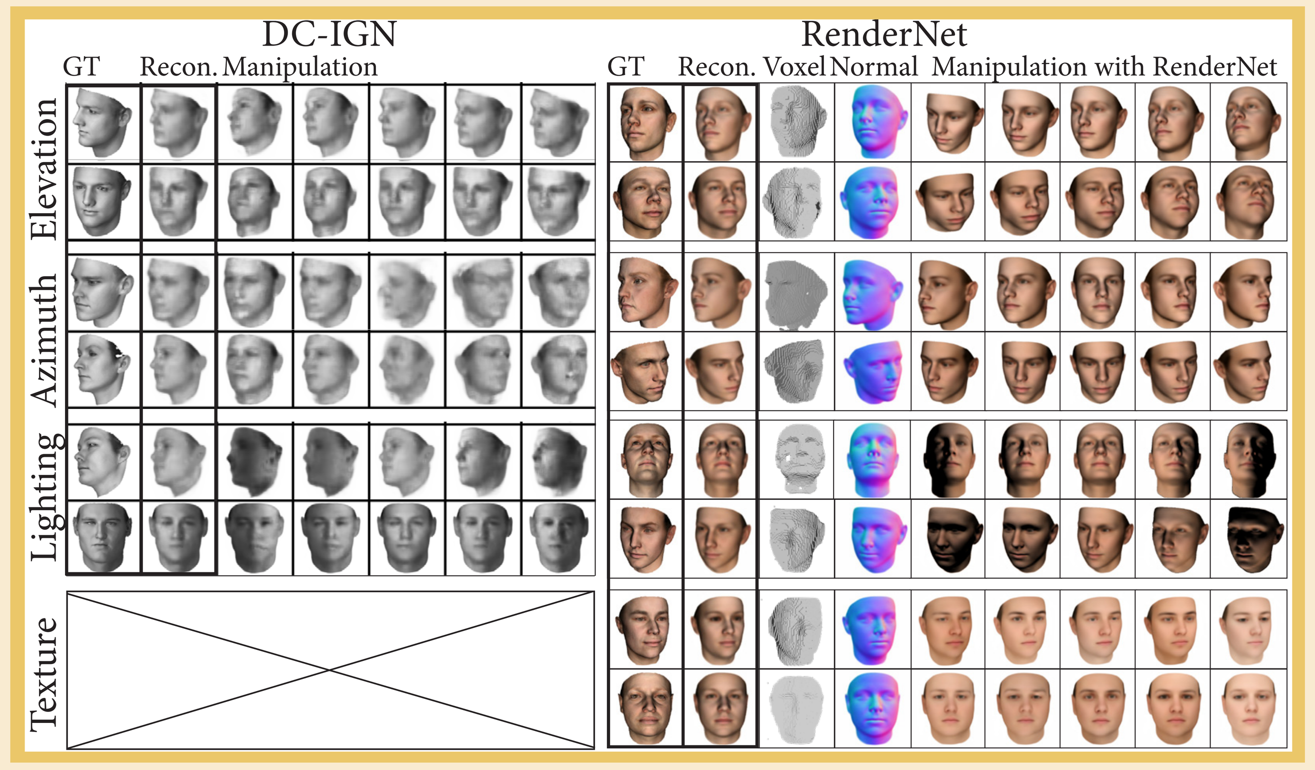
MAP estimation:

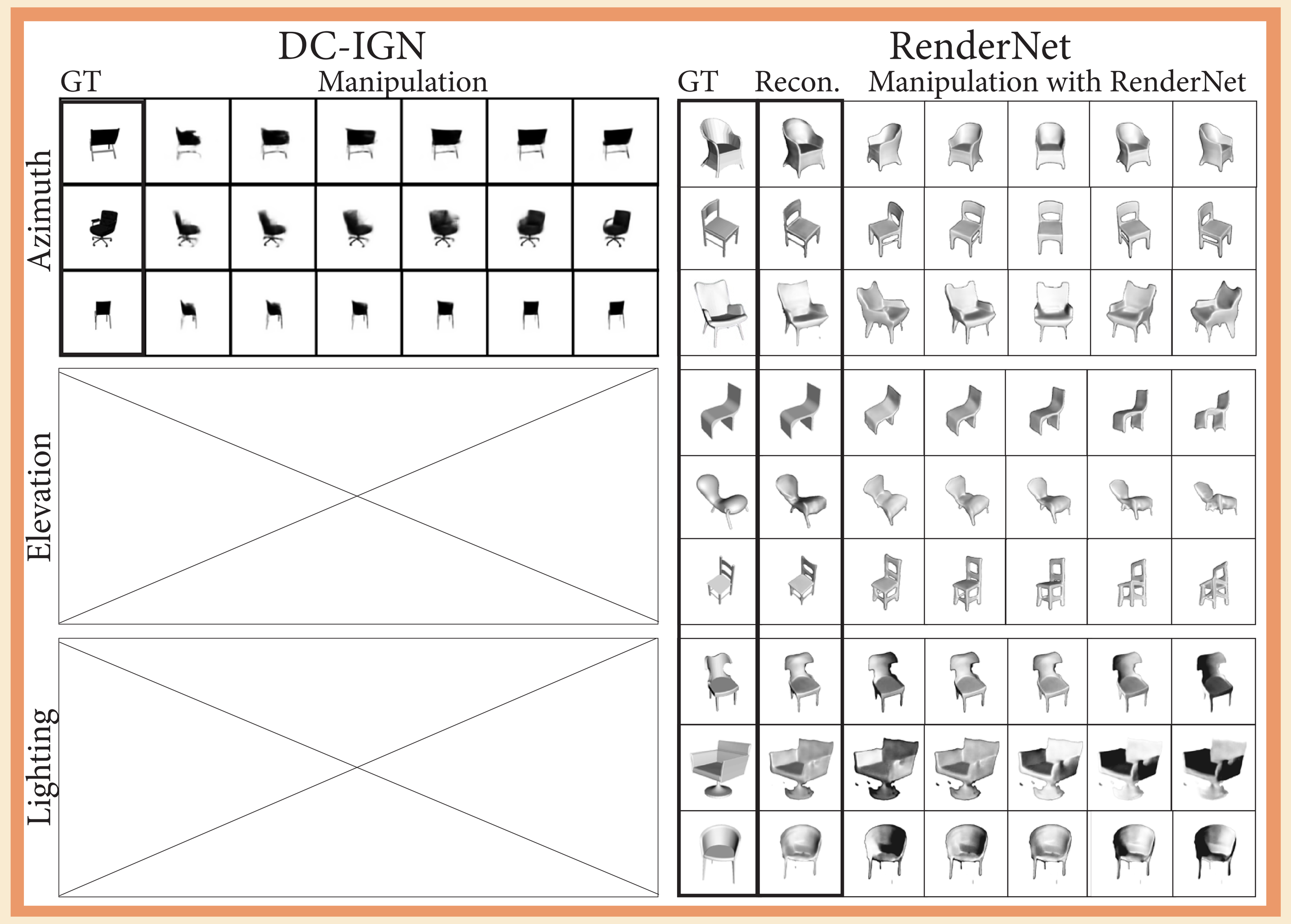
$$\underset{z', \theta, \phi', \eta}{\text{minimize}} \quad ||I - f(g(z'), \theta, h(\phi'), \eta)||^2$$

where I is the observed image and f is our pre-trained RenderNet. z' is the shape vector to reconstruct, g is the decoder of a pretrained 3D auto-encoder, θ and η are the pose and lighting parameters, and Φ is the texture vector.

$$\underset{z, \theta}{\text{minimize}} \quad \alpha||I - f(g(z'), \theta)||^2 + \beta(z - \mu)^T \Sigma^{-1}(z - \mu)$$

where μ and Σ are the mean and covariance of z' estimated from the training set respectively.





COMPARISON

PSNR score

Name	PSNR
RenderNet Phong	25.39
EC Phong	24.21
EC-Deep Phong	20.88
RenderNet Contour	19.70
RenderNet Toon	17.77
RenderNet AO	22.37
RenderNet Face	27.43

DISCUSSSION

- Using adversarial loss instead of MSE or BCE
- Using more efficient voxel representation (octree)
- Considering other 3D data types (mesh, point clouds, etc.)
- Learning multiple shaders with one network

ACKNOWLEDGEMENT

This work was supported in part by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 665992, the UK's EPSRC Centre for Doctoral Training in Digital Entertainment (CDE), EP/L016540/1, and CAMERA, the RCUK Centre for the Analysis of Motion, Entertainment Research and Applications, EP/M023281/1. We also received GPU support from Lambda Labs.

REFERENCES

1. Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. TPAMI, 39(4):692–705, 2017.
2. Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In NIPS, pages 2539–2547, 2015.
3. Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In CVPR, pages 3907–3916, 2018.
4. Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In ECCV, pages 154–169. 2014.

Code available at
github.com/thunguyenphuoc/RenderNet