Differentiable Graphics for ML

Udit Jain

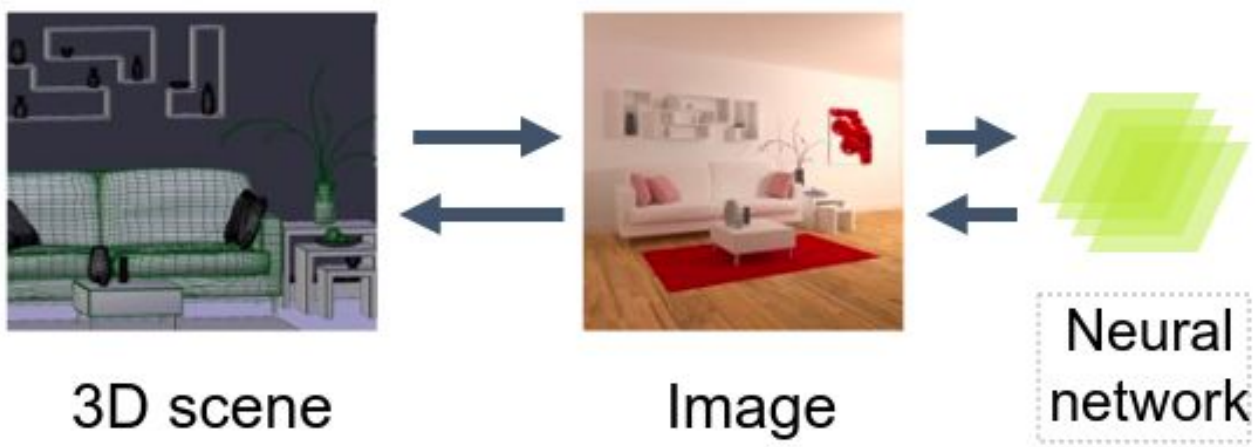
Project Description

Differentiable rendering:

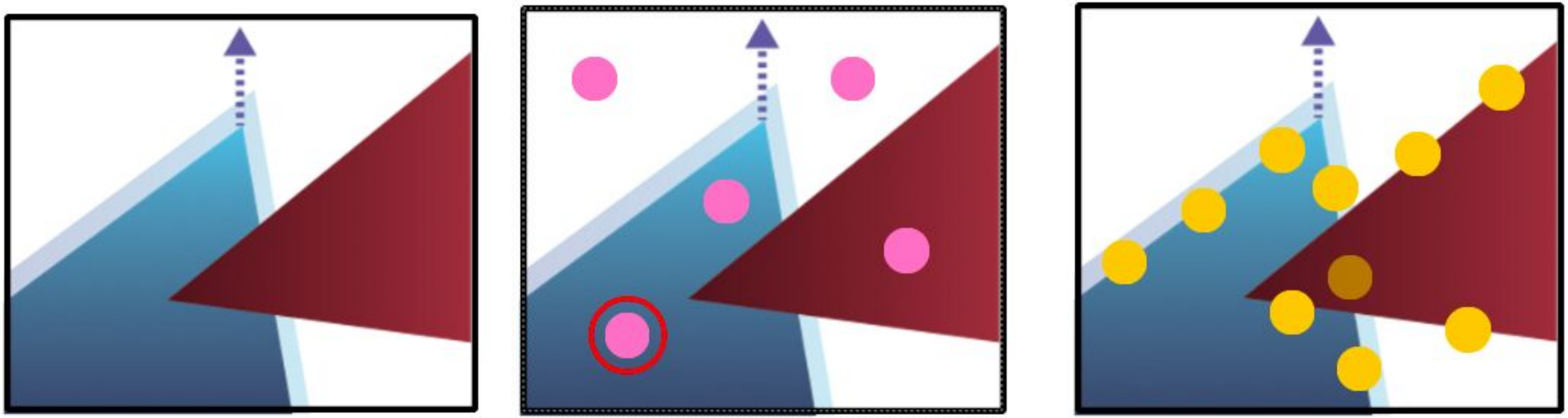
1. Motivation
2. Challenges
3. Algorithm - Differentiable MC Ray tracing through Edge sampling
4. Results and Applications
5. My Experiments

Motivation

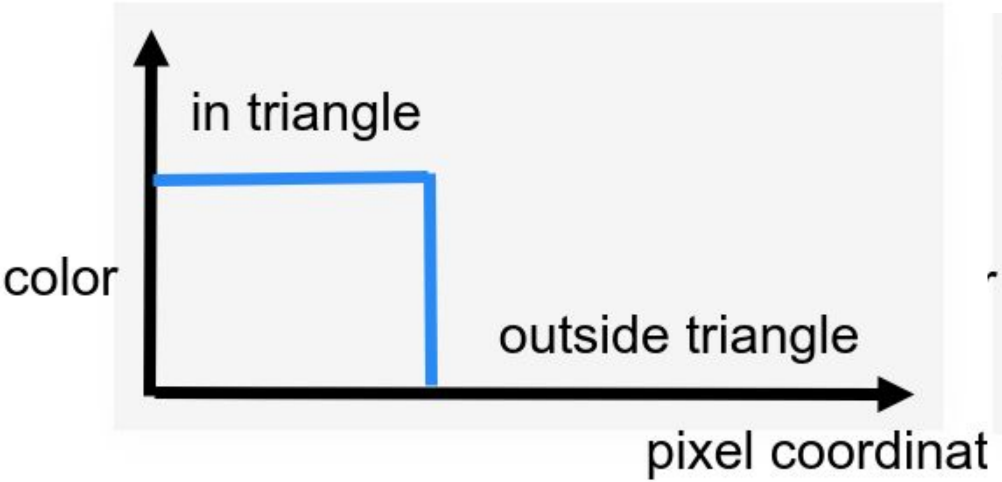
1. Derivative computation central to Graphics, Vision and ML.
2. Major role in Deep learning via back-propagation.

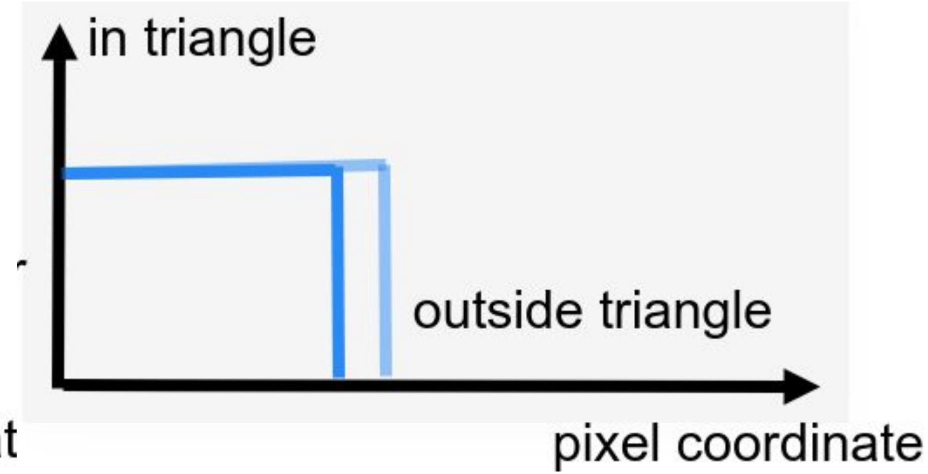


Differentiable rendering

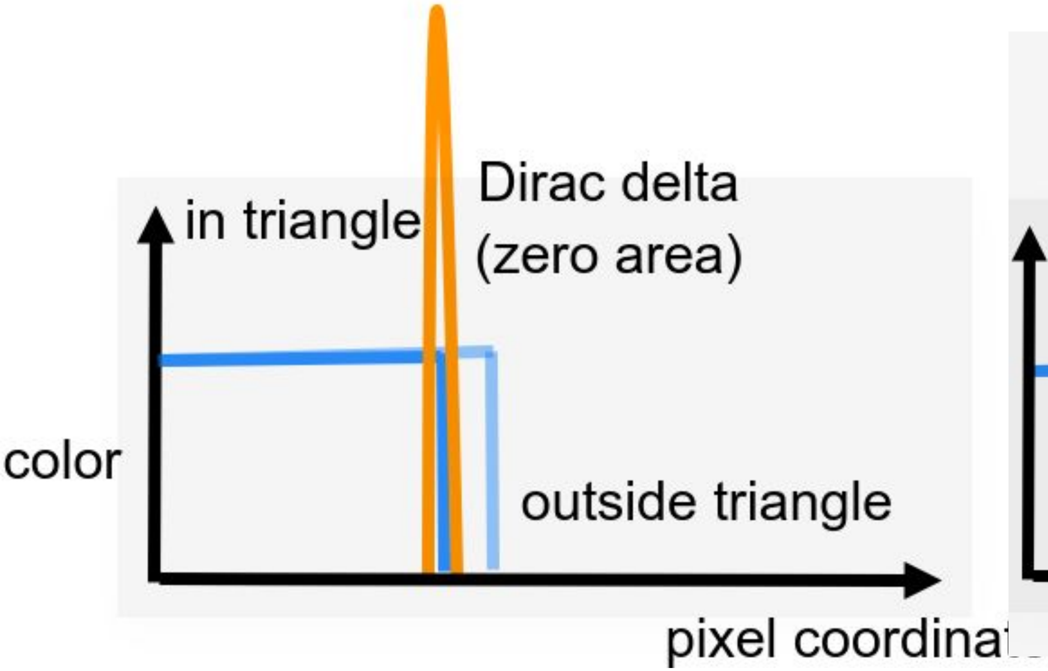


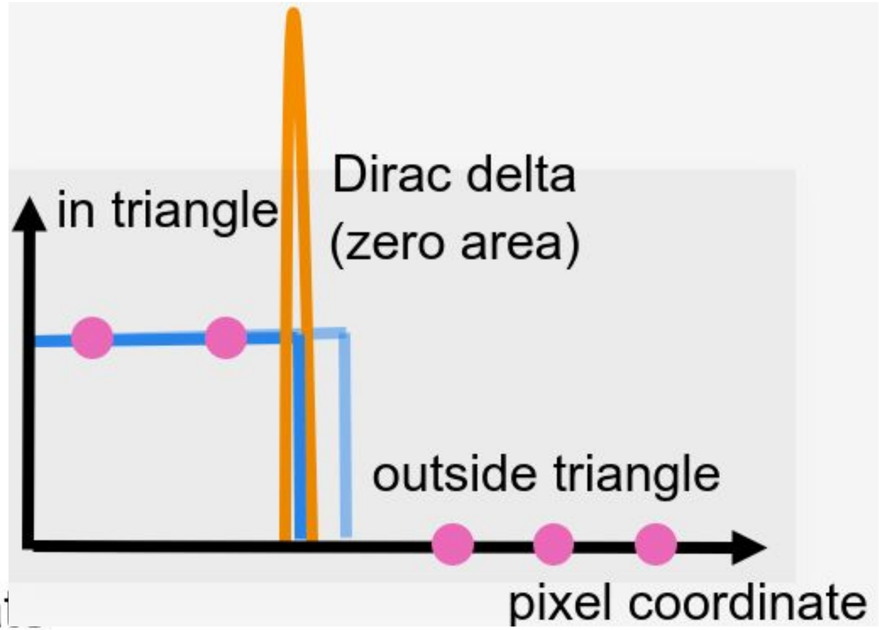
Mathematical Formulation and Challenges





Mathematical Formulation and Challenges Cont.

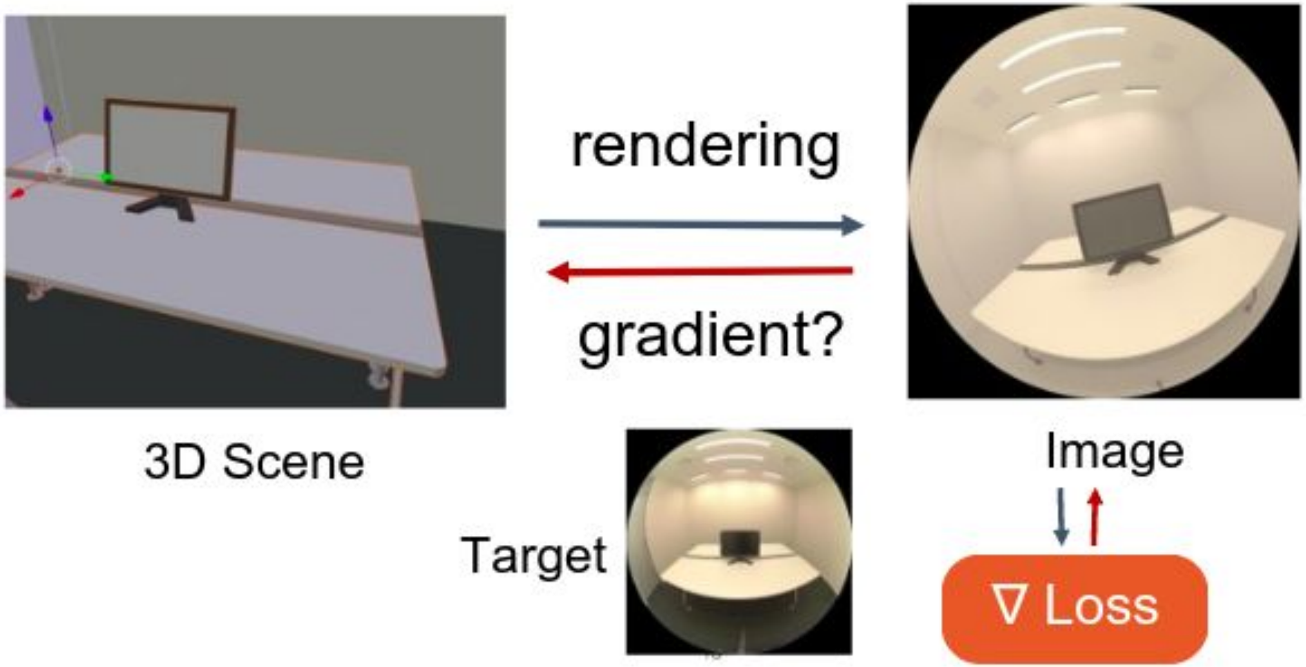




Differentiable MC Ray Tracing through Edge Sampling

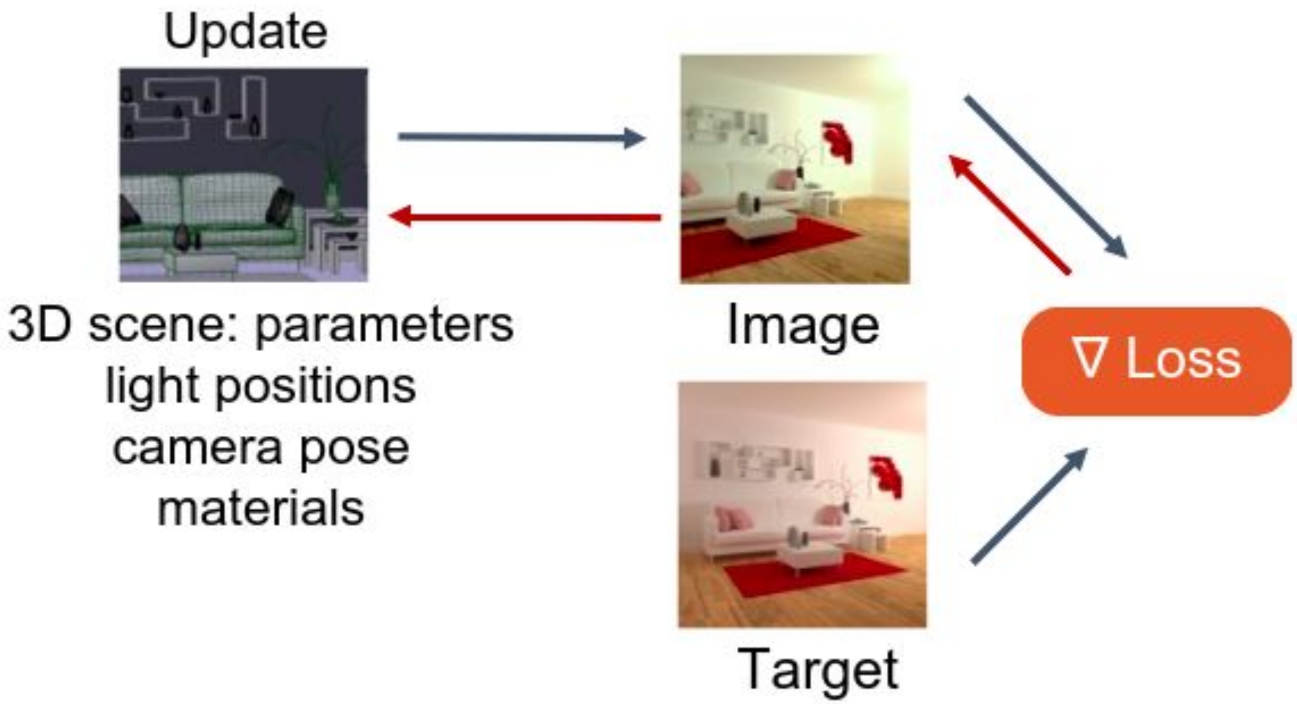
1. Novel Edge Sampling algorithm. (With proof of correctness)
2. Increasing efficiency, hierarchical sampling
3. Application on image and computing gradients.
4. Adversarial applications

Model



Applications

·



Limitations

- Modern models efficient on GPU and this is not GPU compatible.
- Effects such as motion blur can be supported.

Tasks Accomplished

1. Understanding
   i. The Algorithm from Paper
   ii. The codebase and and the Tutorials.
2. Usefulness: Pyrender incredibly useful in Deep learning.
3. Extensions:
   i. Made a model to estimate camera's position parameters given a target image
   ii. More involved: Made a model to estimate light intensity values given a target image.
   iii. Directly useful for our project : Vary Learning function.

Pyrender Tutorials Key Points

Tutorial	Description	Usefulness
Materials and Textures	Material manipulation, specular vs diffused materials etc. Changing roughness, texture etc. Experimenting with light sources, environment map light source.	Intro to Render API, path tracing algorithm working.
Pose Estimation	Estimating object's translation and rotation parameters by calculating gradients and learning.	A learning task, an application of getting the gradient.
Camera Models	Different camera behaviour and how it matters in rendering.	Introducing parameters of the camera and their variation.
Path Tracing	Experimenting with global illumination, increasing intensity, playing with shadow, and increasing samples for realism.	A powerful use of Render to generate realistic objects.

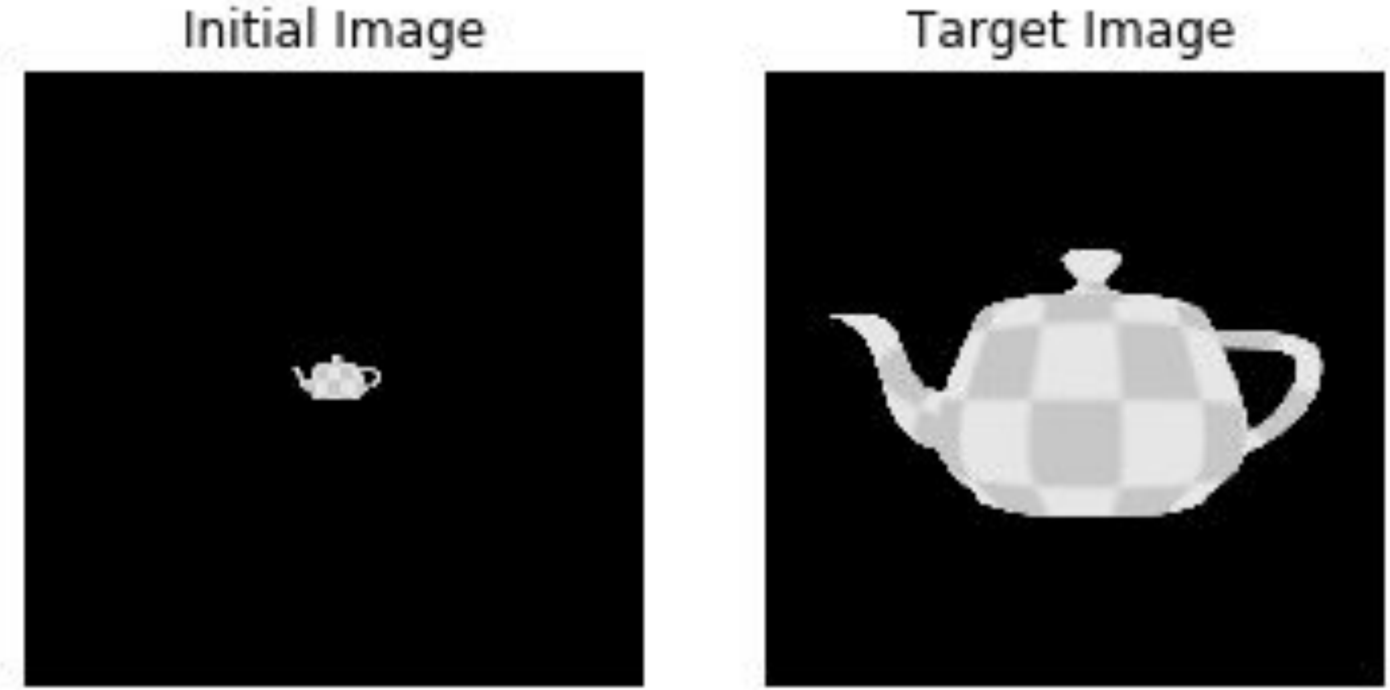
Prototype 1: Estimating camera position

Task - Start with target image and an initial camera position. To estimate the cam pos with derivatives.

Challenges:

- Effect is not a regular function and large search space.
- Translation and rotation are one of many camera parameters .
- Reaching Global maxima is dependent on initial values.

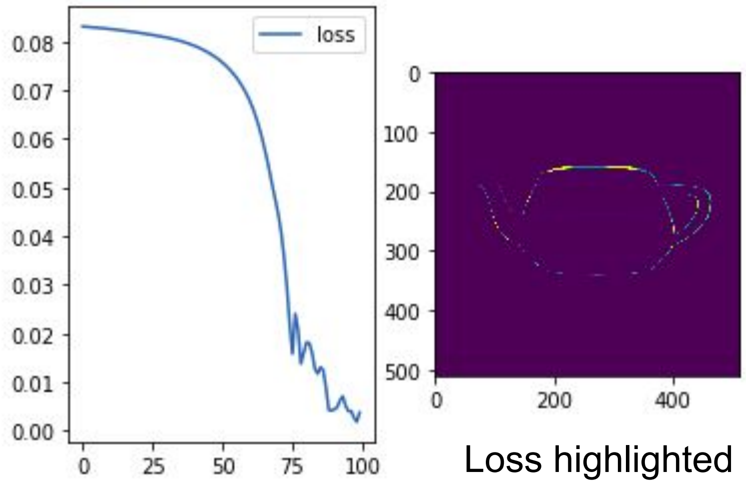
Goal and initial camera parameters



Model

1. Input: target image and input parameter (camera’s position).
2. Displace the camera and calculate the L2 loss. Calculate the gradients wrt to the camera position.
3. Adjust the camera’s parameters by -alpha(lr)*grad.
4. Repeat until loss < threshold.

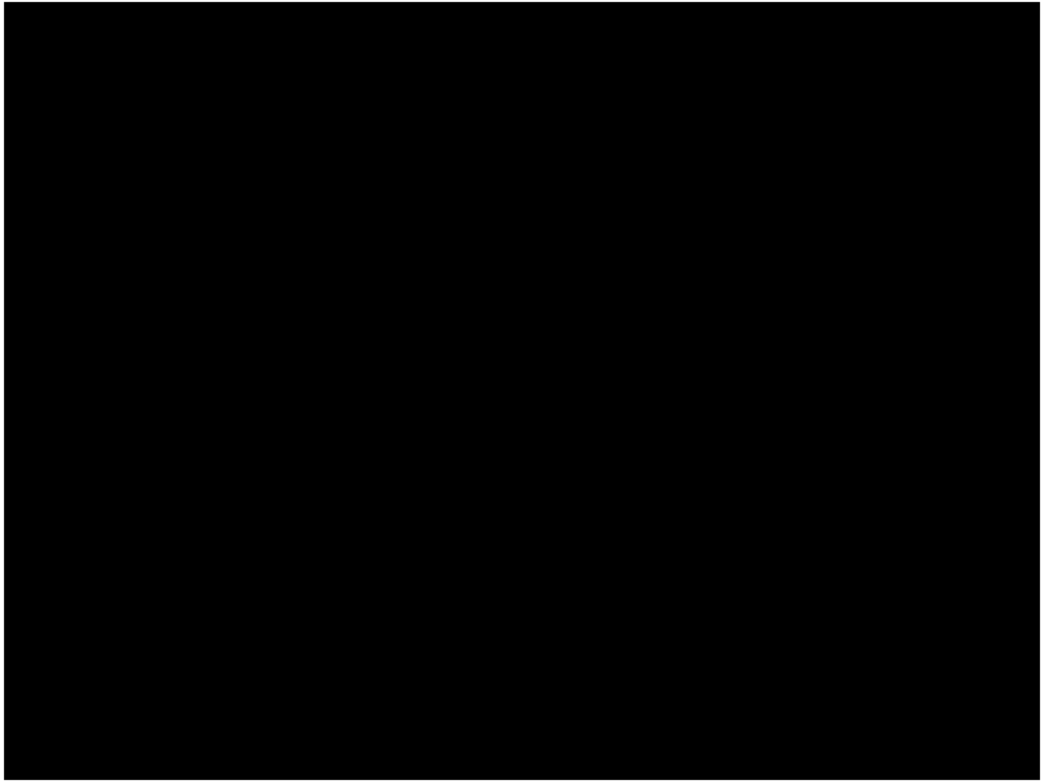
Loss Values



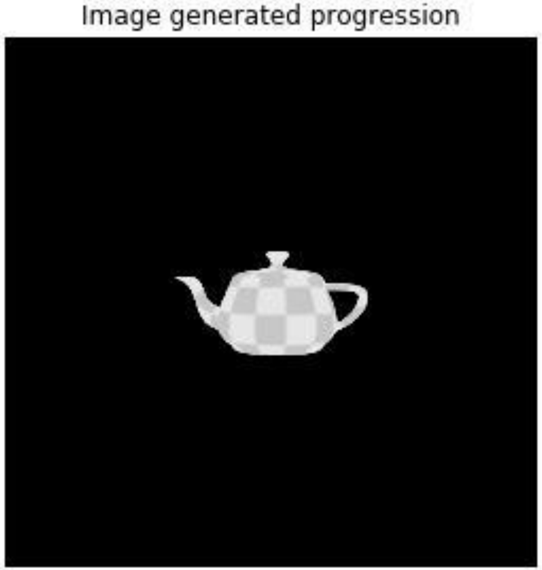
Iterations

Loss highlighted (HeatMap)

Pixel wise loss with iteration



Generated Image Progression



Prototype 2 - Estimating light intensity

- The standard classification models may give wrong result when classifying images with different light intensities.
- Given: input and a target image, vary the image parameters(intensity) suitably to get a ‘standard’ image of an object.

1 — 2 — 3 — 4

Input Image

Rotated,, translated dim image of the object

Transformation Model

To transform scene parameters

Output Image

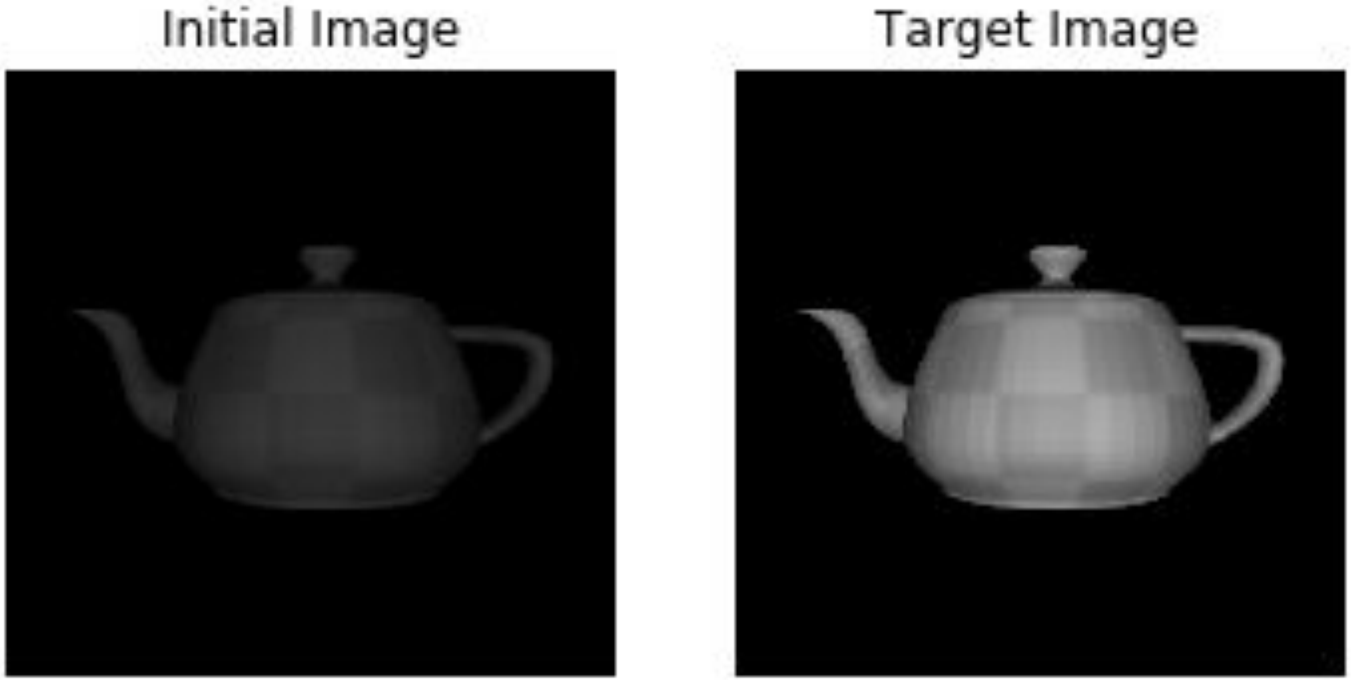
Similar to a ‘collective image’ from the dataset .

Classifier

Classification with increase confidence

Light Intensity Variation

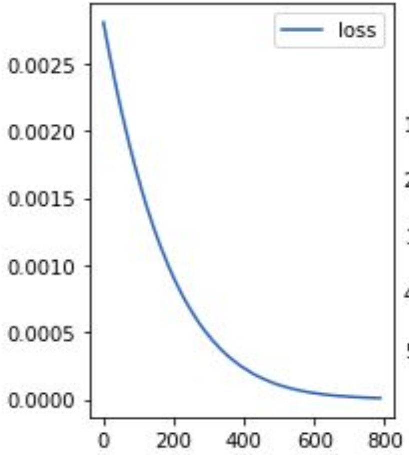
Goal and initial light intensities



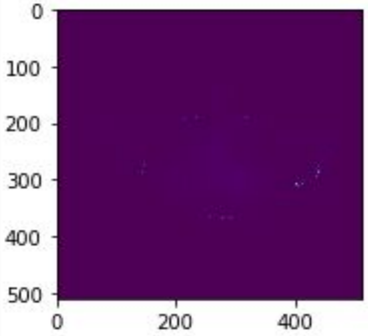
Steps and Challenges

- Light intensity values vary erratically with light source.
- Type of light source and material also plays a role in exposure.
- Change the loss function accordingly

Loss Values



Iterations



Loss highlighted (HeatMap)

Pixel wise loss with iteration

Generated Image progression

Target Image

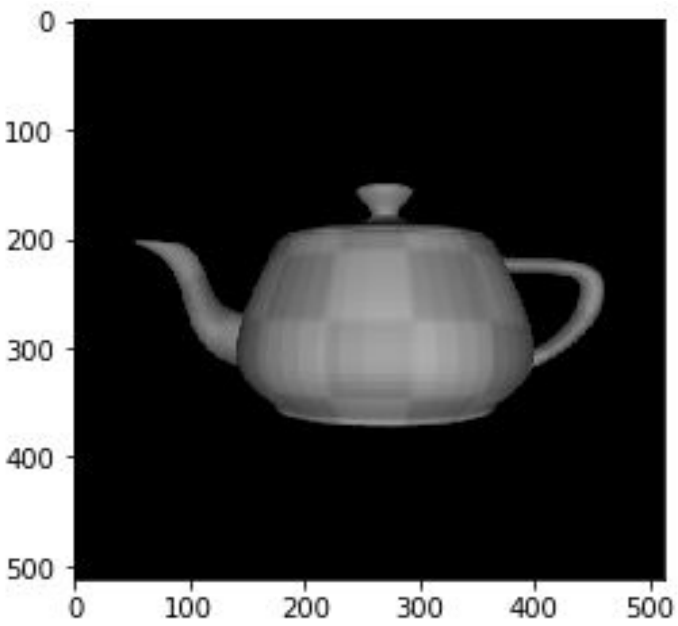
Image generated progression

Results

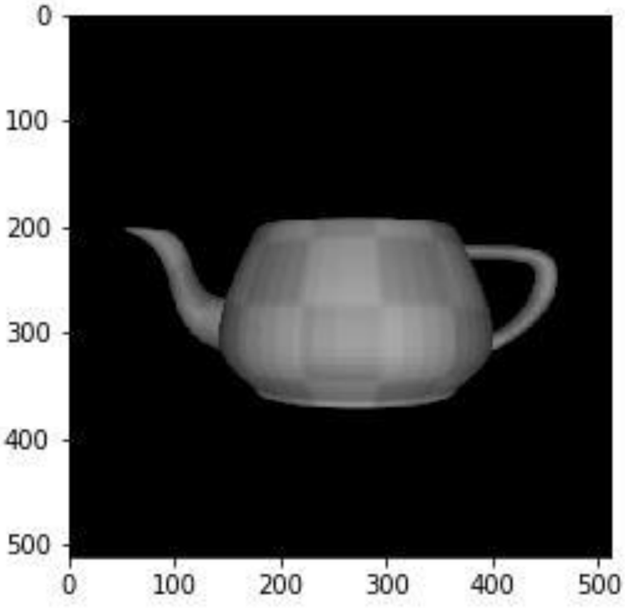
1. Target image:Generated with light intensity 50,000.
2. Initial image: Light intensity 5,000.
3. Model improves estimate and reaches and it’s final estimate: 48,506 (in 800 iterations).

Sampling(5-8 sec animation with stoppage at last)

Sampled 1 out of 30 images to observe the changes in intensity.

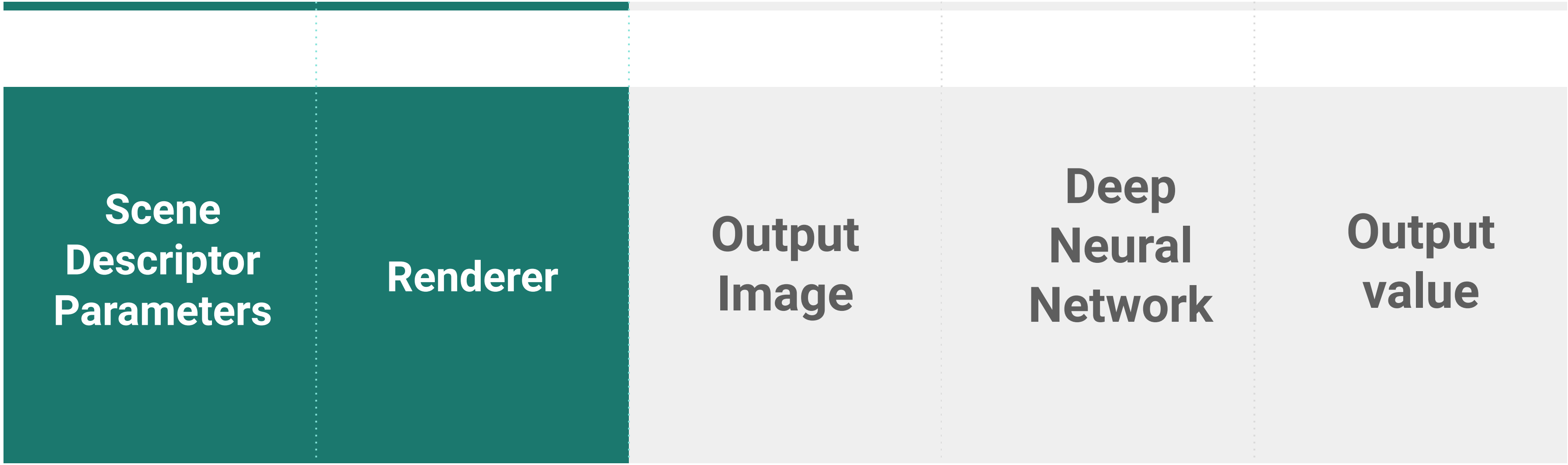


Target Image



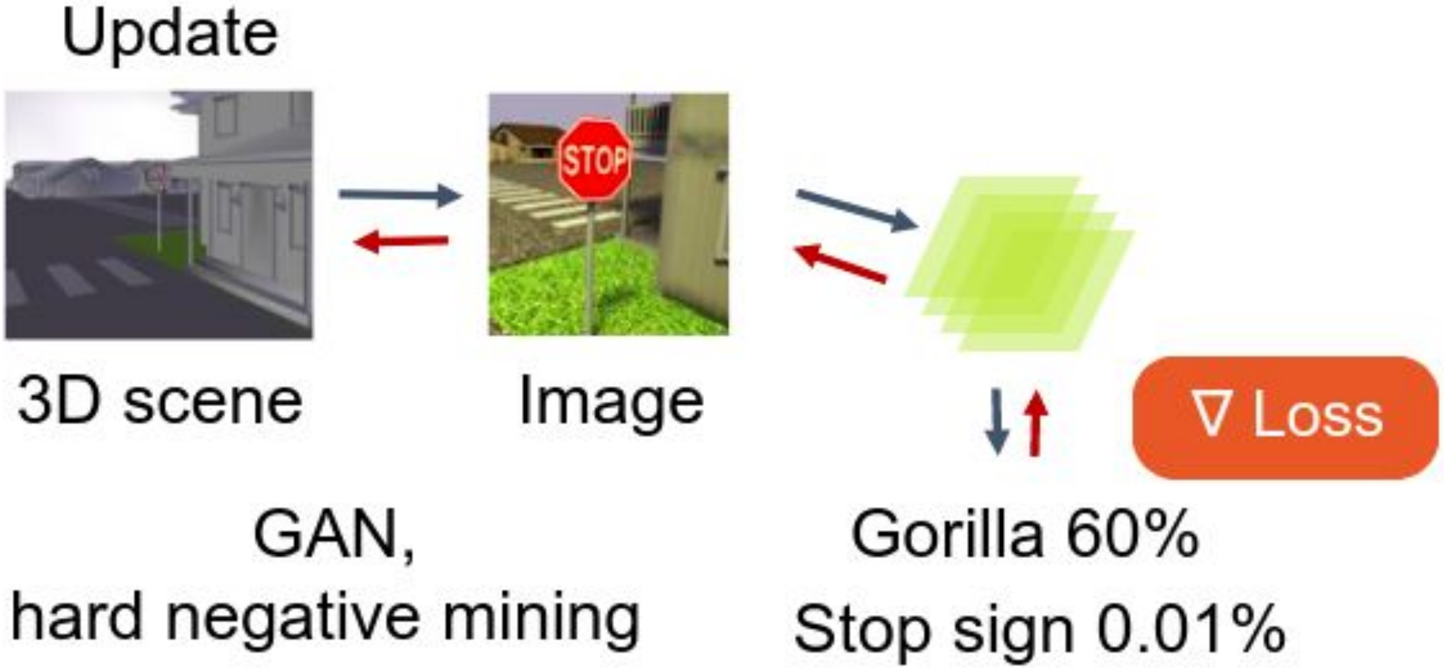
Final Model

Forward Pass ->



<- Backward Pass (Gradients)

Next steps


Update
STOP
3D scene
Image
∇ Loss
GAN,
hard negative mining
Gorilla 60%
Stop sign 0.01%

Conclusion and Future work

- A general differentiable path tracer which handles geometric discontinuities
- Useful for inverse rendering, deep learning.
- PyTorch compatible.

1. Can experiment with other loss functions.
2. Experiment with different learning frameworks.
3. GPU compatibility as large deep learning models on GPU.

Thank You

Questions ?

References

https://github.com/BachiLi/redner