GAN Based Texture Rendering

Joon Jung (joonjung@stanford.edu)

Stanford University

Motivation

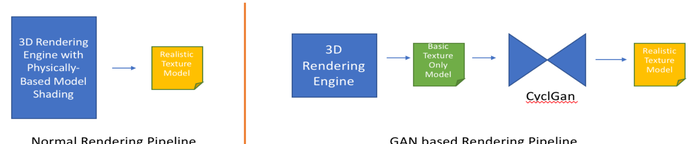
Rendering a physically realistic 3D model requires intensive computations, performing high resolution texture sampling and shading repeatedly. Among the intensive computations, the memory bandwidth, busy looping to read and writ the textures, cost the most significant amount of resources.

Recently, various researches on extracting latent features from images and reconstructing the original from it have indeed progressed significantly. Now we can even create non-existing images based on the learned latent space features. [1]

What if we can extract the low dimensional latent features of a high resolution 3D model, such as the light reflection distribution [3], and run through a GAN based generator to reconstruct the final 3D image? If so we might be able to completely skip the repetitive shading operations and avoid time consuming memory operations. We even might be able to just feed in the raw polygons and paint the textures without the 3D pipeline's help. Just maybe someday soon.

Left: an input image. Right: a target output image.

Model



CyclGan

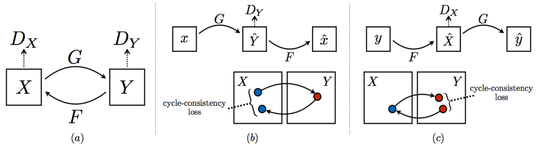


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

- A CycleGan model is composed of two generators (G, F) and two discriminators (DX , DY).
- Each generator and discriminator pair employs a min-max adversarial strategy to learn a good representation of the latent space.
- Given a pair of two domain images (X, Y) as shown in the figure left, G learns to fake X to Y repeatedly, while the paired discriminator DY constantly trying to distinguish between a real image (Y) and a faked image(G(X)) generated by the generator .

CycleGan Model(excerpt from [1])

Losses

Adversarial Loss:

$$L_{GAN_G}(G, D_Y, X, Y) = \mathbb{E}_{x \sim p_{data}(x)}[(D_Y(G(x)) - 1)^2]$$

$$L_{GAN_D}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{data}(x)}[(D_Y(G(x)))^2]$$

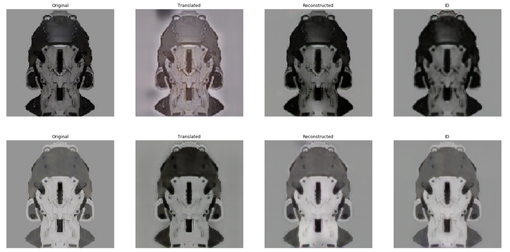
Cycle Consistence Loss:

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

Identity Loss:

$$L_{identy}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(x) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(y) - y||_1]$$

Result Example



- Train with 1000 images from different camera distances and rotations with 200 epochs.
- The row 1 shows the main result of the generator G while the row 2 shows for the generator F.
- The column "Original" shows the original images and the column "Translated" shows the generated images.
- The column "Reconstructed" shows the "Translated" images cycled through the opposite generator.
- The column "ID" shows the result in which the original fed to the generator to show if it changes the original image.

Losses

Discriminator Loss: 0.000191. Generator Loss: 2.440339

Physically Based Rendering(PBR)

Reflectance Equation using Bidirectional Reflective Distribution Function (BRDF)

$$L_o(p, \omega_o) = \int_\Omega f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$

Incoming Radiance: $L_i(p, \omega_i)$ **Incoming & Outgoing Solid Angle:** ω_i, ω_o **Surface Normal:** n **Hemisphere of Reflection:** Ω

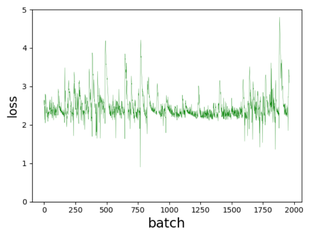
BRDF Scale Factor: $f_r(p, \omega_i, \omega_o)) = k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}$

D: MicroFacet Normal Distribution Function

F: Fresnel Equation

G: Geometry Equation

Error Analysis & Next Steps



On the above, it shows the loss trend of the generator near the last epoch. It is hard to tell from this if it converges but the loss got improved at the end of the training. It is possible that running longer might improve the loss further.

- **Mean Squared Error**: For the particular example shown above, the MSE for the pixel differences between the PBR rendered image and the Generator rendered image is **0.05527**. As a comparison, the MSE between the PBR and the reconstructed images is **0.00933,** almost 50 fold of differences.
- **Human Eye Inspection:** Just by inspecting the generated image of F (row 2, 'Translated' on the figure above), we can see the light reflection is completely missing. This can be confirmed in the G generated image as well. In the generated image of G, you can see that it left out the light reflection intact for the generated image. This is not supposed to be so since the image represents non PBR, light reflection-less image.
- **Next Steps:** For the remaining of the project, we will try to train the deep model to recognize the light reflection. We will separately train the generators by different light distributions (BRDF Scale Factor Function) and try to produce a better target output.

References

[1] Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017. (* indicates equal contributions)

[2] Generative Deep Learning by David Foster. Publisher: O'Reilly Media, Inc. Release Date: June 2019 ISBN: 9781492041931 https://github.com/davidADSP/GDL_code

[3] Details on PBR: https://learnopengl.com/PBR/Theory

[4] Rendering Physically-Based ModelIO Materials August 5, 2018 by Warren Moore

Presentation URL

- Google Drive Link: https://drive.google.com/file/d/1nsnVWznsaqx4W_uiMyRRTEVfjHACqg27/view?usp=sharing