Approximating Lighting with a Conditional Generative Adversarial Network

Zach Dixon, Max Urbany
Brown University
18th December 2017

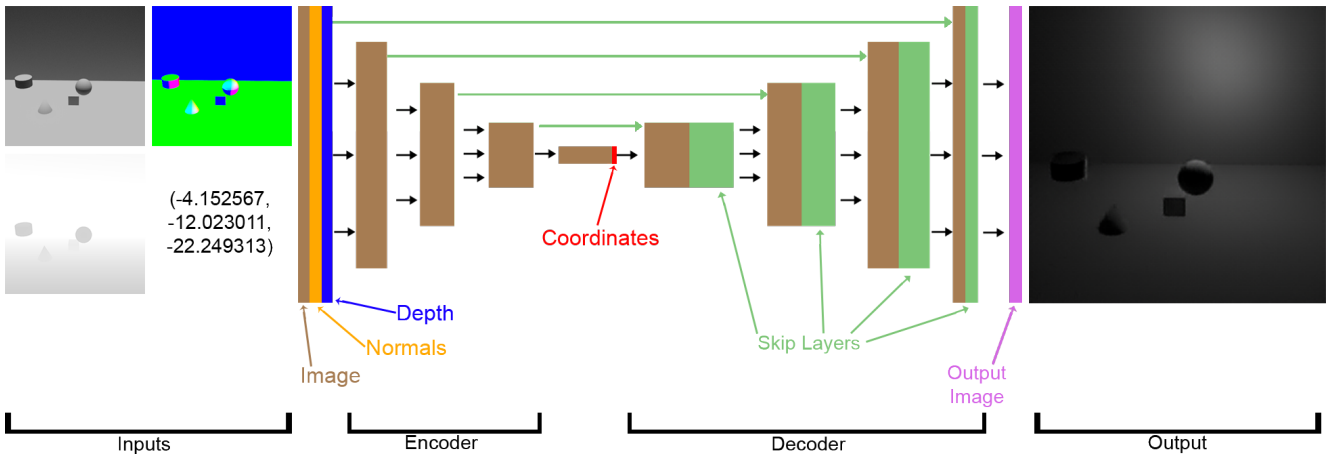

Figure 1: Our network inputs, structure, and output.

Abstract

We present a method for approximating rendering of 3D scenes at a fixed evaluation cost using a conditional Generative Adversarial Network (cGAN). The system incorporates the outputs of multiple simple lighting passes and information about lights in the scene as inputs to a generator network which can produce render approximations. Results on several tests show promising results, and the approach allows for broader generalization to more complicated lighting contexts.

1. Introduction

Rendering is a computationally intense process where any change to the lighting of a 3D scene requires a complete rerender. The time cost of rendering increases with the complexity of the lighting to the point that it can be prohibitively expensive to render intermediate changes at low resolutions. The goal of this project is to generate a representative equivalent of a rendered image in fixed time using a cGAN conditioned on lighting information. This would be a valuable tool in production environments where many intermediate renders are necessary during development, such as 3D animation, visual effects, or game development.

2. Related Work

Isola et al. [1] created pix2pix, a conditional generative adversarial network (cGAN) based image translator. pix2pix translates an image to a corresponding image in a different domain. However, pix2pix is conditioned only on an input image, and does not allow for the inclusion of other relevant information. Our method uses a modified version of pix2pix as a foundation, adding in the ability to include other relevant information.

Thomas et al. [4] created a cGAN-based method for approximating the lighting of a single object within a scene, also based on the pix2pix model. However, their method relies solely on additional image buffers, and was trained in limited environments (a fixed camera, or a small, low-poly world). Additionally, their approach is intended to be retrained for each scene, whereas ours is intended to be more generalized (though our testing was restricted).

Deep learning has been used to approximate rendering techniques in other ways, as well. Nolbach et al. [2] used a CNN with a similar structure to our generator network to approximate different shading techniques (ambient occlusion, depth of field). However, our network approximates lighting more generally, and is trained adversarially, theoretically providing more convincing results.

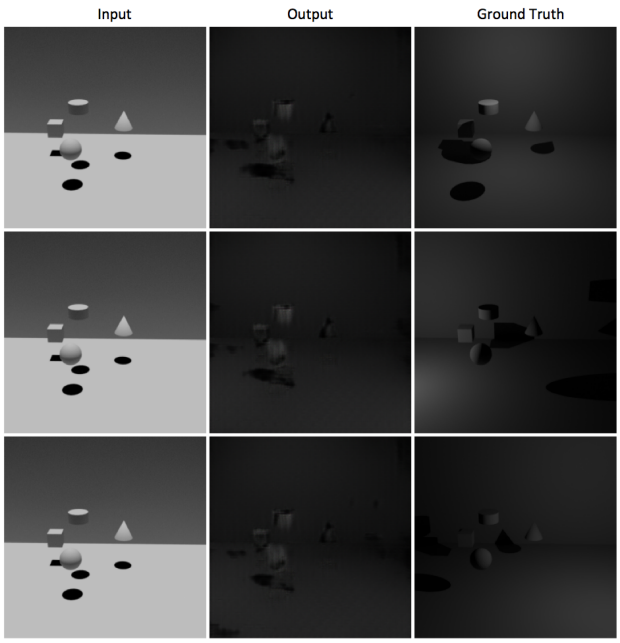

Figure 2: No light location (after 200 epochs)

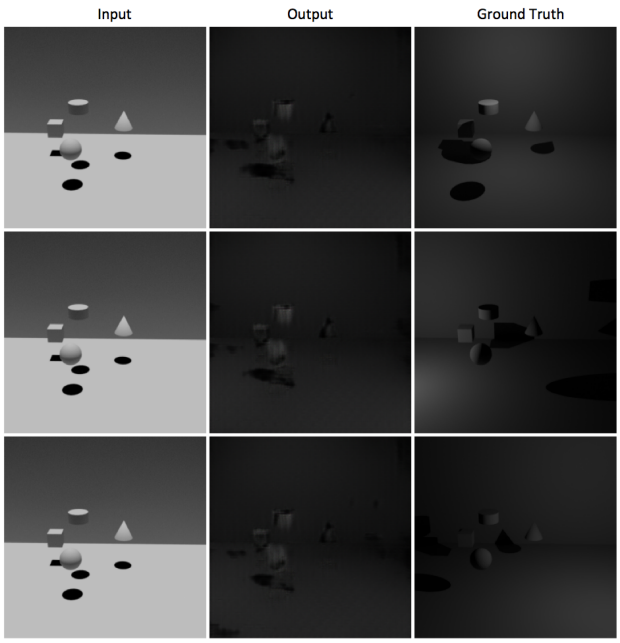

Figure 3: Light location included (after 200 epochs)

3. Method

Our approach uses a conditional GAN, consisting of a generator network and a discriminator network. The generator and discriminator are trained concurrently, with the generator generating results, and the discriminator attempting to determine if given inputs are generated or real.

3.1. Generator

Our generator consists of a U-Net [3] with eight encoder layers, a bottleneck layer, eight decoder layers, and skip connections between the encoder and decoder layers. See Figure 1 for a simplified version of our generator architecture. It uses batch normalization between each encoder and decoder layer. The multiple image input layers are combined to form a 9 layer input (3 layers from the image, normals, and depth renders), and the coordinate data is appended directly to the middle bottleneck layer. The bottlneck layer is one dimensional, so arbitrary, low dimensional data can be appended directly to it.

3.2. Discriminator and Loss

The discriminator and loss function are unchanged from [1], a patchGAN consisting of 5 encoder layers with batch normalization, which outputs a confidence value of whether the input is real or generated.

3.3. Training

Our network was implemented in TensorFlow, based on a port of [1] available here. We used a dropout value of 0.5, a batch size of 1 (as was recommended by [1]), and a learning rate of 0.0002. Code is available at https://github.com/zdixon/cgan-renderer.

4. Data Generation

Using the 3D modeling software Maya, we generated a basic scene containing four primitive shapes (sphere, cube, cylinder, and cone) along with a ground and back wall plane. Each primitive was randomly placed in a bounding box to ensure its visibility. With each scene arrangement a single point light was randomly placed in the bounding box. The prepared scene was then rendered using Arnold 5, a physically based renderer, into several distinct passes: full lighting, normal, z-depth, and constant lighting. During generation, coordinate information of the light in the scene is also recorded into a text file. Additionally, the compositing software Nuke was used to normalize the z-depth passes as the Arnold renderer generates them non-normalized and Photoshop was used to batch convert the output images to a format supported by TensorFlow. Data generation code is available at https://github.com/zdixon/cgan-renderer.

5. Results

Initially we did not include the normal and z-depth passes or the light coordinates when training our network. These results are shown in Figure 2. The network was trained on 100 image sets for 200 epochs. This first set of results demonstrates the network's inability to learn shadow locations from the input data provided. Additionally, due to the lack of light

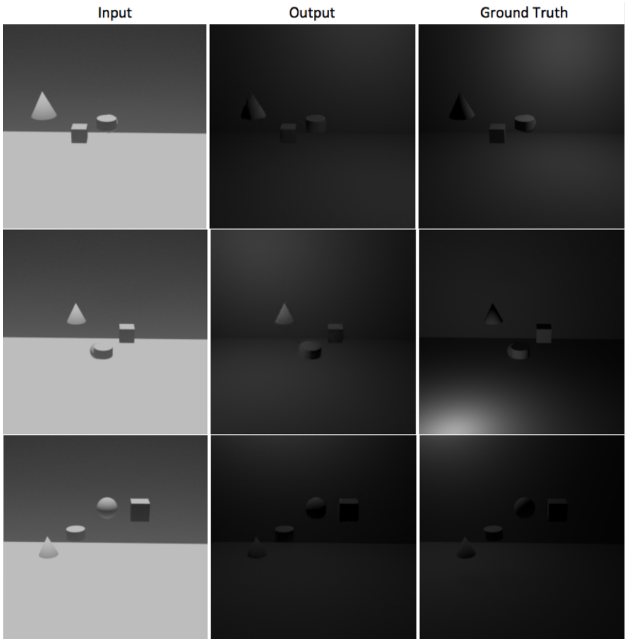

Figure 4: All passes included (after 500 epochs)

coordinate data, the network approximates an average lighting.

In Figure 3 normal and z-depth passes are still not included, but the light coordinates are. Using the same data set as in our initial test we trained the network for 200 epochs. Objects remain blurred and distorted because of the small size of the data set and the low number of epochs. Shadows are still not correctly placed, but variation in the color of the ground and wall planes show the impact of including the light-coordinates.

As a simplification of our problem we rendered a new data set of 2490 scene and lighting variations without shadows to test the validity of the network configuration. The network was trained with all render passes and lighting coordinates for 500 epochs. The results in Figure 4 demonstrate a significant in both the quality of the image as well as lighting variations across different areas of the surface of the primitives. Full sample results for these runs and for the following experiments are available at https://github.com/zdixon/cgan-renderer.

6. Experiments

After establishing results with a basic greyscale scene we conducted further experiments.

6.1. Colored Objects

In Figure 5 are the results of a network trained on images containing primitives of various colors. Trained on 1000 image sets for 100 epochs, the results show varied color over primitive surfaces even capturing darker areas of light occlusion. The quality of the evaluated images is lower, however, as the size of the training set and length of the training were decreased.

6.2. Position Alteration

As different training scenario more in line with practical application, we trained the network to approximate a lighting change from a fully lit scene, rather than a constantly lit scene. We replaced our baseline of the scene lit under a vertical directional light to fully lit render of the scene under a single point light. Additionally we modified the lighting information to represent the vector change between the baseline and target image point lights' positions. An increase in quality of the evaluated images can be attribute in part to decreasing the total pixel color change required compared to previous tests. The results also appear to better capture circular highlights from point lights very close to objects. This test appeared to produce the most compelling results of all our tests.

7. Discussion

Our system is able to reasonably approximate lighting changes in a simple scene. With sufficient data for the network to train on and expanded input data, this system could be incredibly useful in a production environment where re-rendering a scene under complex lighting can take several minutes if not hours. That being said, our system is currently restricted to a single scene due to the effectively infinite size of all scene and lighting configurations. This approach is similar to that of [4].

Our approach is relatively effective for input generalization as the image inputs themselves are available in all commercial renderers. Additionally, we have the advantage of being able to encode arbitrary lighting information because of the way we append information to the bottleneck layer of the network. One particular disadvantage is the low resolution of the image, as in its current iteration the evaluated images are not sufficient to replace fully rendered images as well. The system also is unable to effectively reproduce shadows, or bright specular highlights from a light in close proximity to a surface.

8. Conclusion and Future Work

In this paper, we presented a cGAN-based system which produces reasonable approximations of lighting calculations at a fixed evaluation cost. We implemented this using a modification of the approach in [1], with additional image input layers, and an additional one dimensional input for lighting information (coordinates). We produced convincing results for greyscale arrangements of objects, for colored objects,

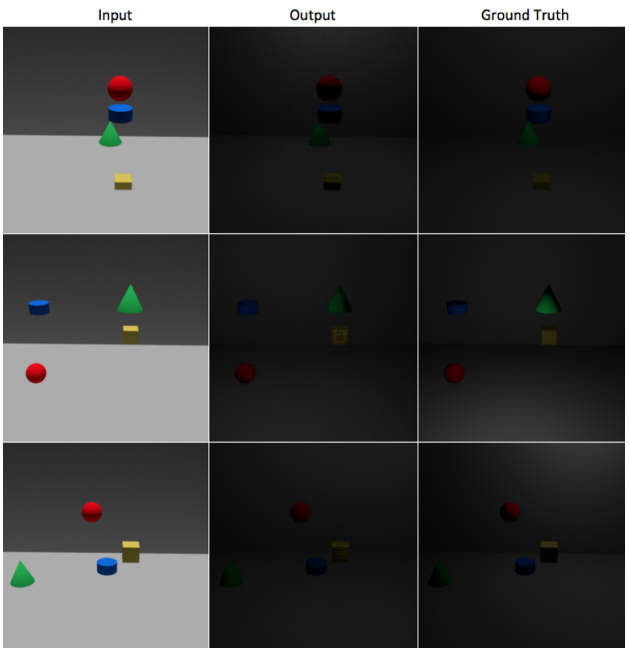

Figure 5: Colored objects (after 100 epochs)

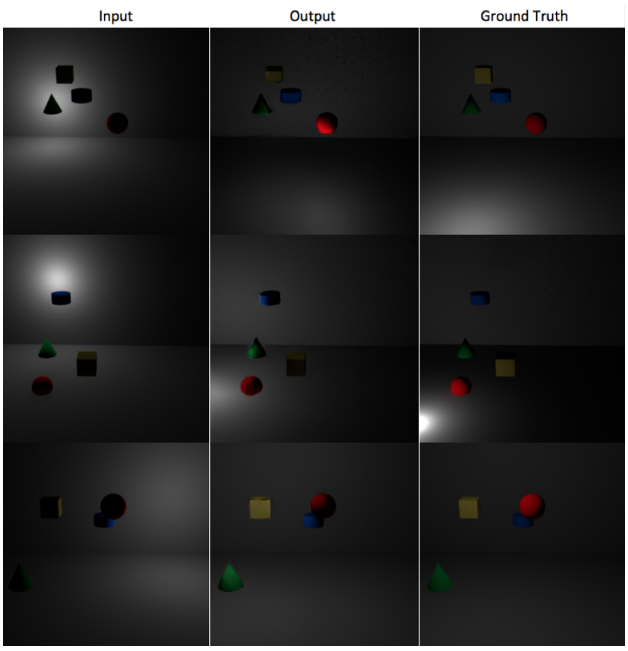

Figure 6: Light displacement instead of position (after 200 epochs)

and for an alternate encoding design utilizing light translations instead of positions.

Further enhancements of this system include a larger number of inputs for the network to train on such as breaking down a fully lit render into specular, diffuse, and ambient passes. Also including a shadow pass could possibly teach the network the relationship between shadows and objects. Augmenting the light information could also increase the generalizability of the system by incorporating more light parameters such as color and exposure, and even including multiple types of lights like area lights or spot lights.

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. 1, 2, 3

[2] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel. Deep shading: Convolutional neural networks for screen space shading. *Computer Graphics Forum*, 36(4):65–78, 2017. 1

[3] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, pages 234–241. Springer International Publishing, Cham, 2015. 2

[4] M. M. Thomas and A. G. Forbes. Deep illumination: Approximating dynamic global illumination with generative adversarial network. *CoRR*, abs/1710.09834, 2017. 1, 3

Appendix

Team contributions

Max Urbany Python-based system for data-set generation using Maya's scripting language and Arnold renderer API as well as rendering of the dataset.

Zach Dixon Adaptation and modification of Pix2Pix architecture to support multi-layer and coordinate-based inputs. Network training and evaluation on a GPU.