Physics-based Feature Extraction and Image Manipulation via Autoencoders

Winnie Lin
Stanford University
CS231N Final Project
winnielin@stanford.edu

Abstract

We experiment with the extraction of physics-based features by utilizing synthesized data as ground truth, and further utilize these extracted features to perform image space manipulations. Inspired by recent literature, we model our network as an adversarial autoencoder, and train our encoder to extract physical properties of the image scene.

1. Introduction

Motivated by recent work on interesting applications of deep learning to image synthesis, we explore a hybrid technique between completely data-based methods and physics-based generative models, by training a joint encoder-decoder network that partly performs extraction of geometry, material, and lighting models on the encoder end, and attempts the 2D rendering problem on the decoder end. Additionally, an existing issue with style transfer is that results are often aesthetically pleasing yet not very realistic. Given prior results on disentangling style from images via autoencoders [6] we think that it may be possible to utilize autoencoders to enforce physical constraints on image generation via semi-supervised methods, resulting in realism preservation in style transfer.

2. Related Work

Here is a brief survey of existing works that are of interest to us.
In computer graphics, the task of inverse rendering is one that has been mainly focused on augmenting renderers with gradient descent algorithms to perform direct reconstruction of geometry, textures, and lighting, such as the differentiable renderer architecture as described in OpenDR [4], and to perform more precise and constrained material parameter matching, such as the fabric appearance matching system as outlined in Khungurn 2015 [2]. These methods attempt to fit image data to parameters and geometry, and the final image outputs are still wholly synthesized from scratch using existing generative models and shaders in 3D.

Existing work has also been done on 2D image space manipulation for image synthesis. In particular, on the specific topic of image relighting, Ng 2003 [8] shows that we caan approximate a pixel-by-pixel light transport matrix that encodes information about how light sources at different positions and of different intensity can affect pixel radiance. Recently, Ren 2015 [10] extended this method and showed that with neural networks, we can drastically reduce the number of image inputs needed to effectively approximate the light transport matrix.

While image generation within the deep learning community seem to have been originally motivated by the need for deeper understanding of network features [5], there have been many interesting applications to non-photorealistic style transfer recently, and papers such as Upchurch 2016 [11] demonstrate the potential of performing image space modificarions via feature vector manipulation to generate realistic looking images. In addition, there have been a few existing works on inverse graphics networks, most notably Kulkarni 2015 [3] which uses a variational autoencoder network with explicit constraints on a small subset of features to extract, or disentangle, certain physical properties from the image such as shape orientation and lighting. We were very intrigued by this particular paper, and base our experiments off methods described in this paper and Eigen et. al.'s paper on joint prediction of depth and normals via a common convolutional architecture [1].

3. Methods

We use a similar framework as [3], where we train a encoder on some number of extrinsic features such as depth, surface normals, texture, and lighting, as well as some variational amount of hidden intrinsic parameters, and train a decoder to act as a 2D image space renderer, which attempts to output the original image given the feature vector as generated by the encoder.

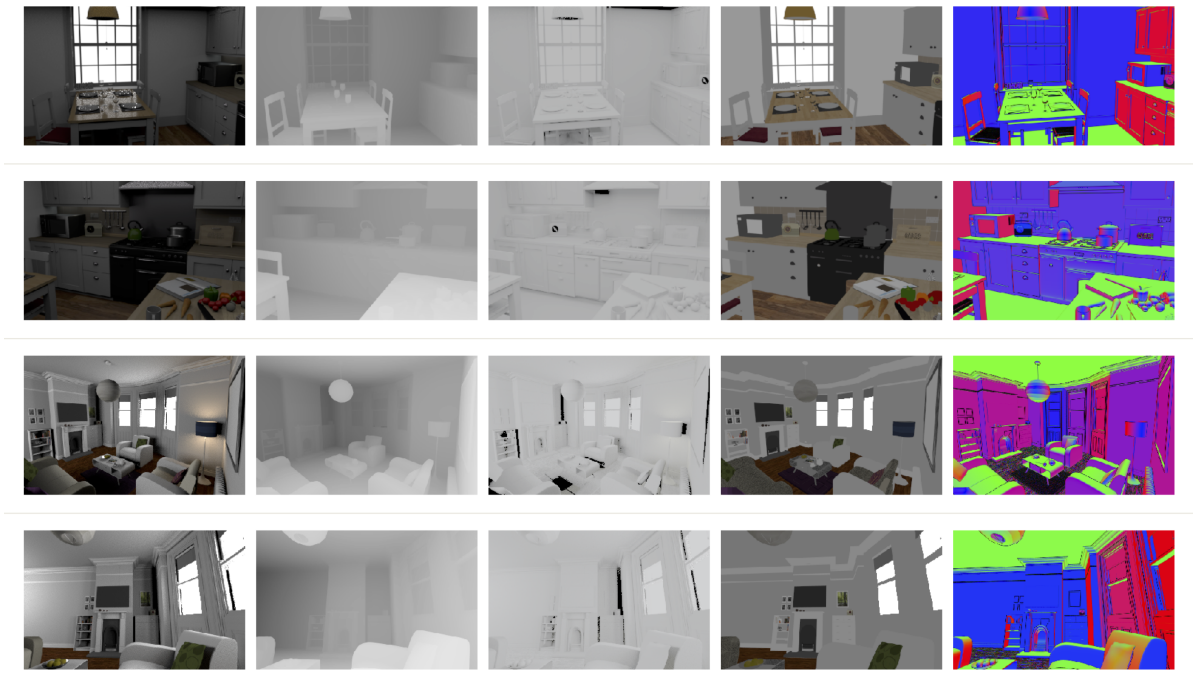

Figure 1. From left to right: original image, depth map, log-normalized light intensity map, approximated diffuse material map, and normal map.

3.1. Dataset

The main challenge of this project is the data collection; in order to effectively train the intrinsic variables, we need a large amount of ground truth that is not easily measurable in the real world. Few existing datasets go beyond RGBD, so we spent a considerable amount of time collecting and generating our data. After some struggle with existing datasets ([7] looked promising but ultimately did not yield usable extractions of material and lighting properties,) we attempted to generate our own data via the pbrt renderer. We synthesize our own data by heavily augmenting pbrt3 [9], a state of the art research-oriented renderer, to output information including material approximations, surface normals, depth, lighting, etc. A visualization of two scenes from different angles are shown in Figure 1.

3.2. Adversarial Autoencoders

The main difference of our method and [3], besides the nature of our input, is that we use an adversarial network instead of a KL loss term in the variational autoencoder to enforce a prior distribution on our encoder output. We utilize [6]'s architecture where in our training stage our encoders simultaneously are used as generators, with the encoded samples passed into a discriminator that treats samples from our enforcable prior as ground truth data. The architecture can be split into three general components; the encoders, the decoder, and the discriminators.

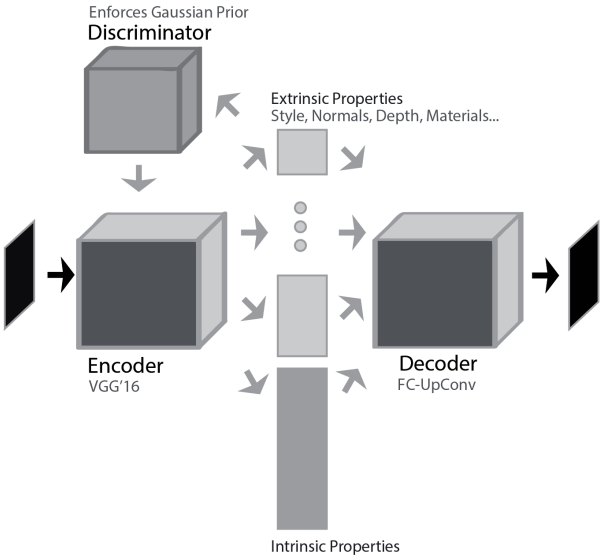

Figure 2. Diagram of architecture

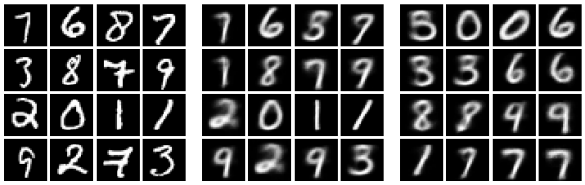

Figure 3. From left to right, the original digits, the reconstructed digits, and the digits produced by feeding grid values $(i, j) \in [-2, 2] \times [-2, 2]$ to the decoder

3.3. Details

We train separate encoders via semi-supervised methods. Initially, for the intrinsic feature vector, we append the ground truth features we obtained to the encoder output to pass to the decoder. For each extrinsic feature, we remove the corresponding ground truth and train an feature encoder as its replacement, still appending the other ground truth features and the trained intrinsic vector.
For each feature encoder, we utilize an adversarial network discriminator during training time to enforce a Gaussian prior on the encoder outputs.
For our decoder, we swap out and retrain our first fully connected layer each time we replace the ground truth features with the feature encodings. Our encoder outputs explicitly correspond to geometric (depth and normal).

4. Results

4.1. Experiments

We weren't able to simulate enough data in time to train a full scale network before the deadline, but multiple toy examples we ran lead us to believe this is a promising method to continue exploring.

4.2. Toy Example 1: Handwritten Digits

First we tried our architecture on the MNIST dataset with an unsupervised method– no extra information was appended onto the encoder input before passing into the decoder. We enforced a 2D gaussian prior with standard deviation of 1 on the encoder output, and after 1000 epochs with 3 fully connected layers on each end, our adversarial autoencoder was able to reconstruct and generate digits with a feature vector of just length 2. We show some of our results in Figure 3

4.3. Toy Example 2: Depth map appendage

Our second toy example was trained on the 20 rendered scenes we had. The images were of 400 by 250 resolution, and we passed in our encoder output as well as a downsampled 200 by 125 resolution depth map into the decoder. We used an intrinsic feature vector of size 256 for our encoder input, and instead of the pretrained VGG network (more suitable for larger datasets) we used a three stacks of conv-leakyReLU-maxpool followed by a fully connected layer as our encoder architecture, and a fully connected layer followed by a conv layer then 4 stacks of upconvolutional-leakyReLU-batchnorm layers as our decoder. As we can see in the results shown in Figure 4, with just the coarse depth maps we were able to overfit and generate fairly nice reconstructions for our original input.

Figure 4. Top to bottom: first 2 rows are original image and reconstruction after 50 epochs, bottom 2 rows are original image and reconstruction after 150 epochs

An observation to make is that we lose quite a bit of color in our reconstruction - this leads us to believe that a more complete architecture that disentangles the material properties from the image might perform better!

5. Conclusion

5.1. Future Work

As evident from this project writeup, we spent quite a bit of time on data synthesis and we only have preliminary results. However, our small examples all yield reasonably good results and we are optimistic in its scalability once we collect enough data to train more complex architecture. We intend on continuing work on this project, and some experiments we would like to conduct include trying to answer the following questions

1. How does the number of intrinsics in the encoder output affect accuracy?

2. What types of images are hard to work with? Can we capture complex phenomena such as reflection and subsurface scattering?

3. Can we effective perform image relighting, recoloring, and eventual photorealistic style transfer via the manipulation of the decoder's input?

5.2. Acknowledgements

We would like to thank Albert Haque for his feedback on the project, Leo Keselman for his advice at office hours, as well as the entire teaching staff for the well-designed and fascinating curriculum. This was a great experience!

References

[1] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[2] P. Khungurn, D. Schroeder, S. Zhao, K. Bala, and S. Marschner. Matching real fabrics with micro-appearance models. *ACM Transactions on Graphics (TOG)*, 35(1):1, 2015.

[3] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. *CoRR*, abs/1503.03167, 2015.

[4] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[5] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[6] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[7] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet RGB-D: 5m photorealistic images of synthetic indoor trajectories with ground truth. *CoRR*, abs/1612.05079, 2016.

[8] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 376–381. ACM, 2003.

[9] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[10] P. Ren, Y. Dong, S. Lin, X. Tong, and B. Guo. Image based relighting using neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):111, 2015.

[11] P. Upchurch, J. Gardner, K. Bala, R. Pless, N. Snavely, and K. Weinberger. Deep feature interpolation for image content changes. *arXiv preprint arXiv:1611.05507*, 2016.