Project Report

3D Reconstruction

Implementation of an Unsupervised 3D Reconstruction Algorithm from Raw Single image [1]

Abouzar Moradian

Problem Summary:

The modified goal of this project is to implement an unsupervised learning-based algorithm for 3D reconstruction from raw single image based on the method proposed by Wu et al. 2020 [1]. Most of the traditional 3D reconstruction algorithms heavily rely on supervisions such as manually reconstructed 3D models, keypoints, segmentations, depth maps, etc. However, since these supervisions are not always available mainly because making these types of datasets to cover a vast variety of objects are very expensive and time consuming, it has been always a desired goal to find a way to reconstruct a model without any supervision. On the other hand, 3D reconstruction from multiple images even without supervision is another limitation of 3D reconstruction because most of the time there is only one image or view available to the training model. Furthermore, since a computer vision algorithm is more likely to be successful and more functional when it imitates the way human perceives the world, and human makes a 3D model of scene by only having one single view perception, that perfectly makes sense to seriously pursue this novel approach.

This algorithm overcomes two main obstacles in 3D reconstruction:

- Need for 3D or 2D supervision which most of the time are not available
- Need for more that one image from different viewing angles for each object which most of the time are not available either

Challenges to overcoming these obstacles:

- Without 3D supervision, the output of the model should be comparable to input to be able to calculate the loss and since the input is an image, the output must be an image as well. Therefore, we will have to compare the rendered image of the reconstructed 3D image with the original image. Now here is the challenge: “Rendering consists of projecting the vertices of a mesh onto the screen coordinates system and generating an image through regular grid sampling”. The first part is differentiable, but the second one which is called rasterization is not because it involves a discrete operation which is not differentiable and consequently prevents backpropagation. Therefore, we need an alternative differentiable renderer to make backpropagation possible. In this project I used an “approximate gradient for rendering” proposed by Kato et al. 2018 [2] which enables incorporation of rendering into neural networks. Kato et al provided a useful library called “Neural Renderer” which flows gradients into texture, lighting camera as well as object shape.

- Secondly, training based on a single image is ill posed problem because it does not provide enough constraint to make an effective loss function which is necessary for any machine learning algorithm. Therefore, we need to use the flipped version of the original image as the second view assuming that the shape is symmetric in some direction. However, since almost no actual image is symmetric, we should account for the asymmetries in the input images if we want to keep our symmetry assumption. Wu et al. [1] found two ways to account for this asymmetry. Some part of asymmetry in an image comes from the illumination direction, therefore by disentangling shading from the texture we can eliminate this source of asymmetry. The second source of asymmetry comes from the shape itself, for this part of asymmetry they introduced a confidence map trained in the model which gives the probability of each pixel having a counterpart in the input image.

I chose this topic because I am personally interested in 3D reconstruction as the most important area (in my opinion though) in computer vision and AI in general. In addition, I found this method very interesting because it draws maximum information from minimum input. Nowadays we could train a model in every topic by creating a huge training dataset, but what is more practical and efficient is designing a model which learns with minimum supervision and input.

Related works:

3D reconstruction from multiple 2D images is a traditional research area in computer vision. However, during last few years, there has been an increasing interest on learning-based 3D reconstruction from single image. The works on 3D reconstruction from single image can be divided into 3 categories: reconstruction with 3D supervision, 2D supervision, and without supervision. Here, I will first briefly mention the studies on 3D reconstruction from multiple images, then discuss these three categories of single-image 3D reconstruction respectively.

Multi-view stereo algorithms are of different types. The first type performed by [3, 4, 5 6, 7] first computes a cost function on 3D volume, then surface will be extracted from this volume. The second type of algorithm minimizes the cost function by evolving a surface iteratively. [8, 9, 10, 11] The third method called image-space method works on a computation of a set depth maps. The 3D shape is reconstructed by enforcing consistency among depth maps as the only constraint. [12, 13, 14] The last method is performed by first extracting and catching a set of feature points, and then fitting a surface to those reconstructed features. [15]. Seitz et al. [16] has done a through overview of the different methods on multi-view 3D reconstruction.

For single-view 3D reconstruction, most of the learning-based methods rely on 3D supervision. Wu et al. 2016 [17], and Zou et al. 2017 [18] are two examples of this kind of methods which have utilized from manually 3D constructed datasets such as ShapeNet [18] and ModelNet [19] the main problem with this method is that 3D supervision datasets are very expensive to obtain.

The second type supervised single-view 3D reconstruction method, requires keypoints annotations on 2D training images instead of 3D models. Vicente et al. 2014 [20]; Kar et al. 2015 [21]; Kanazawa et al. 2018 [22] are some examples of weakly supervised method. Although this method is more feasible compared to those ones which require a 3D supervision, they are still not the ideal method because of requiring some kinds of supervision which might not be available in most of the cases.

The newest and most interesting category of learning-based single-image 3D reconstruction is totally unsupervised method which model only access unannotated raw 2D images as input without any ground-truth pose, keypoints, or 3D shapes with only a single view image for each shape. There are a few works which have recently focused on this challenging type of 3D reconstruction. One example of is Henderson et al. 2020 [23] who exploited shaded cues to reconstruct 3D shapes. Another example is Szabo el al. 2019 [24] which used adversarial training method to reconstruct 3D mesh without supervision. The algorithm which I chose to implement is proposed by Wu et al [1] and another variant proposed by Ho et al, 2021 [25] which outperformed the other mentioned unsupervised single-view 3D reconstruction. I will explain the algorithm in some details in the following sections.

Description of work:

What I have actually accomplished in this course project is implementing the algorithm proposed by Wu et al [1], including training a deep neural network model on CelebA [26] dataset, and testing the results. At the beginning, as implied in the first update, my goal was to first implement the existing algorithm and train it exactly based on what has been mentioned in the Wu et al [1] paper, then try different types of loss functions based on some combination of geometric properties of faces as well as some features in the input images and the reconstructed one. In addition, I was planning to train a renderer function separately and jointly with the first training part to come up with my own renderer function which takes four elements, d, a, v, I as input and outputs an image in any given viewpoint and illumination. This Renderer model was supposed to be like a black box without requiring any geometric computation. The other idea that I was pursuing was training a model to first classify input image into some categories of objects, then feed the image into appropriate pathway based on its category to be reconstructed. However, because of the extreme complexity of the algorithm, the only thing that I managed to do is training the model and successfully find the desired result. Therefore, the main reason which prevented me to meet all my initial goals was the complexity of the project which was beyond a course project. Maybe I should have picked a simpler topic to be able to find enough time to allocate to my own novel ideas. However, I am happy with what I accomplished because the project was a complicated one both in terms of theory and implementation.

Now I will concisely explain the main ideas and concepts behind this 3D reconstructions algorithm.

The algorithm has been designed to train model which learns a function such that maps an input image to five elements i.e., canonical depth map, canonical albedo, illumination, viewpoint, and confidence map. Then from depth map, albedo, illumination, and viewpoint we can reconstruct the input image.

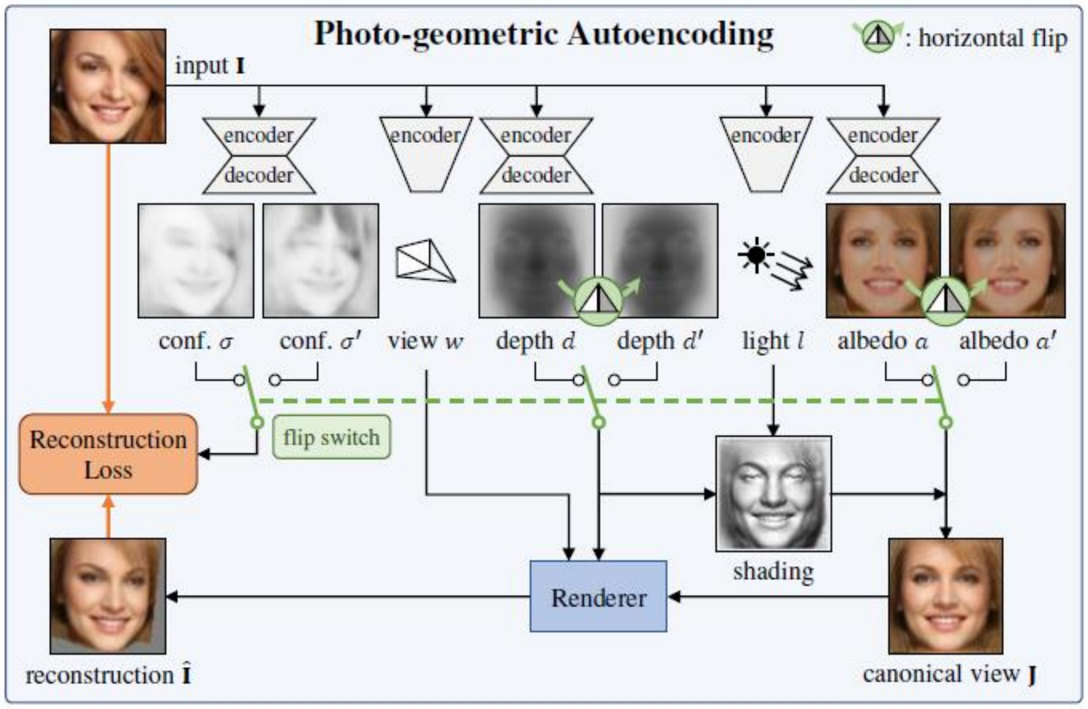
Assuming that the object is symmetric, we can easily obtain a second view of the object by mirroring the input image. Then we can achieve 3D reconstruction using stereo reconstruction. As mentioned in the previous section, some asymmetry in the input image comes from shading therefore we extract illumination as an additional information which account for this apparent asymmetry. For that part of asymmetry, which is rooted in the shape itself, we incorporate a confidence map into the loss function which will be learned along with other functions such as albedo, depth map, etc. to account for those inherently asymmetric part of the shape. Therefore, for those pixels which has lower confidence map value, the discrepancy between the original and reconstructed image are less penalized.

After the model decomposes an image into d, a, l, and v, we reconstructs the input image in two steps. First function L takes d, a, and l as its arguments and outputs shaded canonical image. Then function R takes shaded canonical image along with canonical depth map and the viewpoint to reconstruct the image in the original viewing angle. In fact, function R takes the shaded canonical image and transform it to the given viewpoint v. the figure below shows the procedure.

L (d, a, l), v = 0 **Shaded canonical image**

$\hat{I}$ = R (L, d, v) = R (L(d, a, I), d, v) **Reconstructed image**

d = canonical depth, a = canonical albedo, l = lighting direction, v = viewpoint



loss function

For the loss function, the first thing which is obvious is that the model is supposed to minimize the difference between the original image and the reconstructed image which we can show it as follows:

$$\mathbf{I} = \hat{\mathbf{I}}$$

In addition, assuming that the object is symmetric in canonical view, canonical depth and canonical albedo of the original image and flipped image must be the same. i.e., d = d’, a = a’ this gives us another constraint in our optimization problem.

$$\widehat{\mathbf{I}'} = \mathbf{R}\ (\mathbf{L}\ (\mathbf{d},\text{’}\ \mathbf{a}\text{’}, \mathbf{I}), \mathbf{d}, \mathbf{v})$$

Therefore, we will have two constraints according to which we should form our loss function:

$$\mathbf{I} = \hat{\mathbf{I}}$$

$$\widehat{\mathbf{I}'} = \hat{\mathbf{I}}$$

Here is the loss function which incorporates these to constraints as proposed by Wu et al [1]

$$\mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}, \sigma) = -\frac{1}{|\Omega|} \sum_{uv\in\Omega} \ln \frac{1}{\sqrt{2}\sigma_{uv}} \exp -\frac{\sqrt{2}\ell_{1,uv}}{\sigma_{uv}}$$

Where $\ell_{1,uv} = |\hat{I}_{uv} - I_{uv}|$ is the distance between the intensity of two corresponding pixels in the original image and the reconstructed image, σ is confidence map which is the probability of a pixel having a counterpart in the image. We should penalize the discrepancy of two pixels with higher confidence more. That’s why we incorporate it into our loss function to take into account the asymmetric parts of the object.

$$\mathcal{E}(\Phi; \mathbf{I}) = \mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}, \sigma) + \lambda_{\mathrm{f}}\mathcal{L}(\hat{\mathbf{I}}', \mathbf{I}, \sigma')$$

where $\lambda f = 0.5$ is a weighing factor

We also need to add another loss function to take care of the blurry result from the previous loss function. Wu et al [1] used image encoder VGG16 to control for the loss of features in the reconstructed image by comparing the features in the original and reconstructed images. They called it perceptual loss.

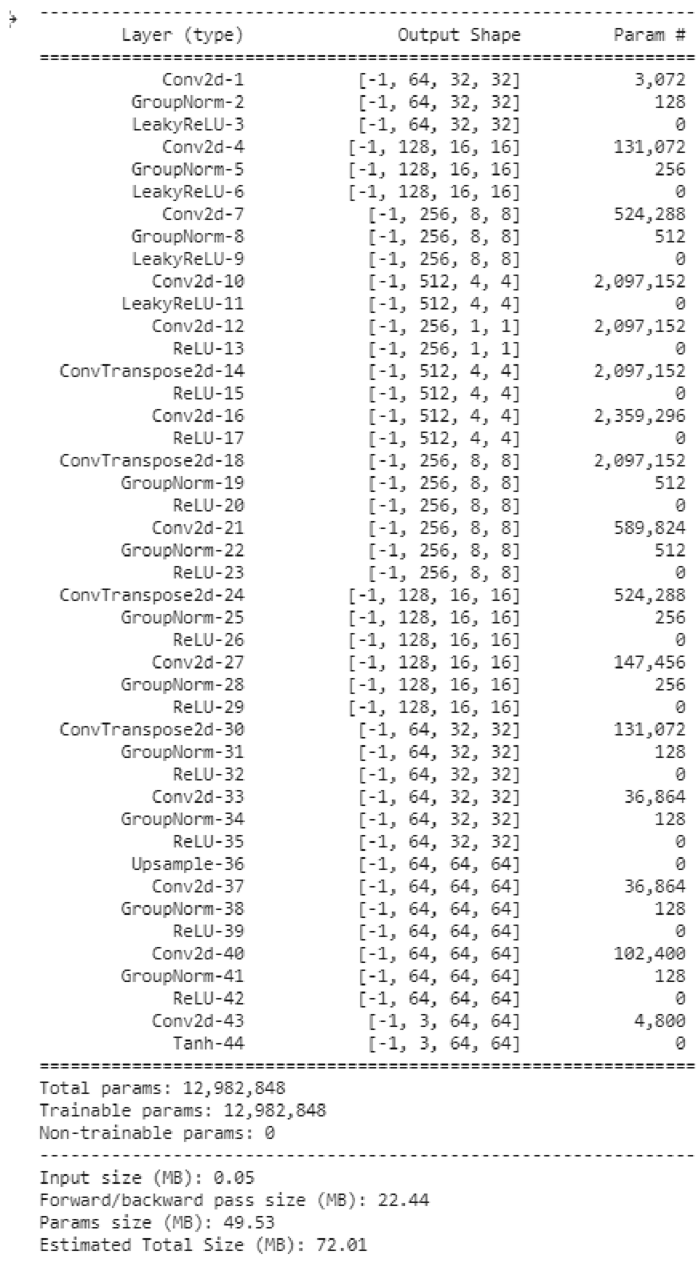
Implementation

As mentioned earlier the goal is to train a model to input an image and output 5 elements, i.e., depth map, albedo, illumination, viewpoint, and confidence map. Each of these elements is a separate function. Therefore, there are 5 separate neural networks which are being trained together. For depth map and albedo map, I implemented two encoder-decoder neural networks using CNN algorithm, while for illumination and viewpoint which are regression, I implemented encoder networks. The neural networks were implemented in PyTorch. I trained the model only on dataset called CelebA consists of over 200K images of real human faces.

I trained the model using Adam optimization on batches of 64 images. The model inputs images of size 64×64 and outputs albedo and depth map of the same size. I preprocessed the images including using pretrained face detector to locate the face in the raw image, and then cropped it into the desired size. For the visualization, I upscaled images using interpolation to have image of size 256×256 with better resolution.

After the images go through all of the convolutional layers, the output was undertaken some geometric processing before feeding into differentiable rendering functions called “Neural Renderer”. Then the image is reconstructed and the loss function is calculated and the parameters get updated through backpropagation.

The implementation of the algorithms was complicated with so many deep neural networks, libraries, pretrained models, geometric manipulation of those four decomposed elements. Implementing a neural network in PyTorch is way more complicated than Tensorflow because you have to take care everything by yourself. However, this in return gives us the flexibility to build your own neural networks and directly access of the output of whatever layer you want. Therefore, PyTorch was the best option for this project.

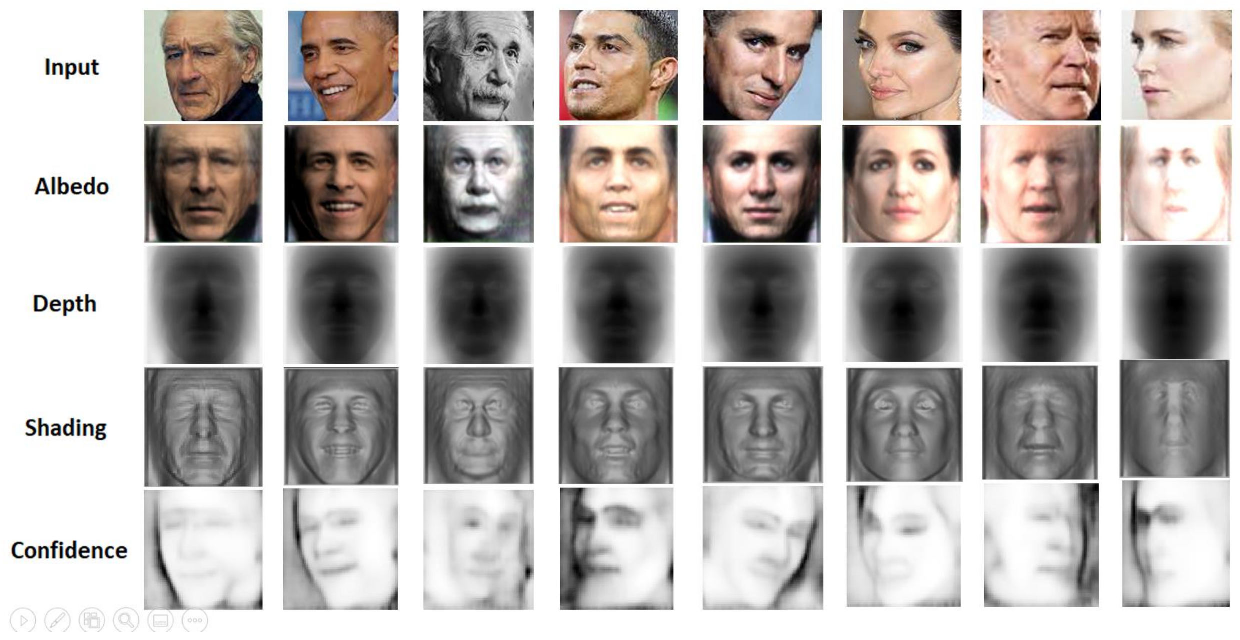


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3,072
         GroupNorm-2           [-1, 64, 32, 32]             128
         LeakyReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4          [-1, 128, 16, 16]         131,072
         GroupNorm-5          [-1, 128, 16, 16]             256
         LeakyReLU-6          [-1, 128, 16, 16]               0
            Conv2d-7            [-1, 256, 8, 8]         524,288
         GroupNorm-8            [-1, 256, 8, 8]             512
         LeakyReLU-9            [-1, 256, 8, 8]               0
           Conv2d-10            [-1, 512, 4, 4]       2,097,152
        LeakyReLU-11            [-1, 512, 4, 4]               0
           Conv2d-12            [-1, 256, 1, 1]       2,097,152
             ReLU-13            [-1, 256, 1, 1]               0
  ConvTranspose2d-14            [-1, 512, 4, 4]       2,097,152
             ReLU-15            [-1, 512, 4, 4]               0
           Conv2d-16            [-1, 512, 4, 4]       2,359,296
             ReLU-17            [-1, 512, 4, 4]               0
  ConvTranspose2d-18            [-1, 256, 8, 8]       2,097,152
        GroupNorm-19            [-1, 256, 8, 8]             512
             ReLU-20            [-1, 256, 8, 8]               0
           Conv2d-21            [-1, 256, 8, 8]         589,824
        GroupNorm-22            [-1, 256, 8, 8]             512
             ReLU-23            [-1, 256, 8, 8]               0
  ConvTranspose2d-24          [-1, 128, 16, 16]         524,288
        GroupNorm-25          [-1, 128, 16, 16]             256
             ReLU-26          [-1, 128, 16, 16]               0
           Conv2d-27          [-1, 128, 16, 16]         147,456
        GroupNorm-28          [-1, 128, 16, 16]             256
             ReLU-29          [-1, 128, 16, 16]               0
  ConvTranspose2d-30           [-1, 64, 32, 32]         131,072
        GroupNorm-31           [-1, 64, 32, 32]             128
             ReLU-32           [-1, 64, 32, 32]               0
           Conv2d-33           [-1, 64, 32, 32]          36,864
        GroupNorm-34           [-1, 64, 32, 32]             128
             ReLU-35           [-1, 64, 32, 32]               0
         Upsample-36           [-1, 64, 64, 64]               0
           Conv2d-37           [-1, 64, 64, 64]          36,864
        GroupNorm-38           [-1, 64, 64, 64]             128
             ReLU-39           [-1, 64, 64, 64]               0
           Conv2d-40           [-1, 64, 64, 64]         102,400
        GroupNorm-41           [-1, 64, 64, 64]             128
             ReLU-42           [-1, 64, 64, 64]               0
           Conv2d-43            [-1, 3, 64, 64]           4,800
             Tanh-44            [-1, 3, 64, 64]               0
================================================================
Total params: 12,982,848
Trainable params: 12,982,848
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 22.44
Params size (MB): 49.53
Estimated Total Size (MB): 72.01
```

NetA: Encoder-Decoder for Albedo Map

Results

As I mentioned in the earlier section, I managed to train this neural network model on CelebA dataset which consists of more than 200K wild images of human faces from different viewing angles. I tested the model on some images from google to see the model performance. The trained model takes a 64×64 image as input and gives 5 separate outputs. Depth map of size 64×64×1, albedo map of size 64×64×3, confidence map of size 64×64×1, illumination, and viewpoint. Then these outputs are fed into function L (d, a, l) as defined in the previous section to output shaded canonical image, and finally function R(L, d, v) transform the shaded canonical image into the reconstructed image of the same viewing angle as original input image. Below are some of the results.



Analysis of work

New results: I did not achieve any new result because what I accomplished was an implementation of an existing algorithm proposed by a recent paper. However, I can list my accomplishments as follows:

- A through literature review on 3D reconstruction specially shape completion using neural networks. It was a good opportunity for me to learn about the state of the art in 3D reconstruction because I plan to work on this field in the future. I literally started from scratch with zero knowledge about Geometric modeling in general and 3D reconstruction particularly. overall, I am happy with what I learned from literature review.
- Understanding the implemented algorithm with going through most of the details such as the neural networks architecture, the geometric parts, and some interesting loss functions. I realized that loss functions in 3D reconstruction is one of the most important parts of the algorithm which go beyond some general well know loss functions which are typically used in deep learning project. In fact, for each project we need to come up with the appropriate loss function to get the best desired result.
- I managed to train a large and complicated deep neural networks model in PyTorch for the first time which is an accomplishment for me. I had trained several neural networks in Keras and Tensorflow before but training a model with PyTorch in this scale was a challenging job which I managed to make in the last moments.

Meeting goals:

As I mentioned breifly in my proposal and updates, my original goal was to first implement the algorithm, and them improve it in different ways.

- One way was testing some new loss functions to compare the results with the existing algorithm. The idea was that we can compare the input and the output images in many different ways. One way is to simply compare the intensity of the corresponding pixels as proposed by the algorithm I implemented. But we can also compare it in terms of their lower level as well as some higher-level features in both images. We could also compare a batch of filtered version of two images and define the loss function in terms of the average of the difference between the corresponding pixels in each pair of filtered images.
- The other idea which I was planning to test was to train a classifier to classify the image inputs into different categories and then train our main neural networks. That way the model would have more specified ways of reconstructing the objects based on the categories the belong to.
- However, as I mentioned in the previous section, the implementation of the original algorithm took time much more that I had estimated. Part of the reason had to do with the face that I was new in pytorch and I had to learn many new things. In addition, because there were many libraries being used, it was hard to handle all of them together. The most challenging part was incorporating "Neural Renderer" library. When I tried to feed the output of my neural network to some of the functions of that library, I kept getting error about the incompatibility of the input data. I ended up going through the details of many of the functions in the library to trace the source of the error. Some parts of the package were written in C++ and Cuda and it was really hard and time consuming to figure out where the error comes from. Since all of the training process was based on the incorporation of "Neural Renderer" as a differentiable rendering function which enables backpropagation, there was no way to replace it with any other library. As a result, I just managed to train the model in the last moments did not have any extra time to experiment my own new ideas and compare the results with the results of the existing algorithms.

- **Future work:** I will definitely keep working on this project because it is completely related to what I am up to. As I mentioned, there are may venues I was planning to improve the algorithm. Since I ran out of time and did not manage to test my idea for this course project, I will continue working on this project after finishing this course. In addition, I am thinking of training a renderer model which inputs canonical depth map, canonical albedo, illumination, and viewpoint, and output the image in the given viewpoint and illumination.

References

[1] Wu, S., Rupprecht, C., & Vedaldi, A. (2020). Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1-10).

[2] Kato, H., Ushiku, Y., & Harada, T. (2018). Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3907-3916).

[3] S. Roy and I. Cox. A maximum-flow formulation of the Ncamera stereo correspondence problem. In ICCV, pp. 492– 499, 1998.

[4] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. IJCV, 35(2):151–173, 1999.

[5] A. Treuille, A. Hertzmann, and S. Seitz. Example-based stereo with general BRDFs. In ECCV, vol. II, pp. 457–469, 2004.

[6] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In CVPR, pp. 391–398, 2005.

[7] Y. Furukawa and J. Ponce. High-fidelity image-based modeling. Technical Report 2006-02, UIUC, 2006.

[8] T. Fromherz and M. Bichsel. Shape from multiple cues: Integrating local brightness information. In ICYCS, 1995.

[9] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3-D objects from multiple calibrated camera views. In ICASSP 99, pp. 3509–3512, 1999.

[10] K. Kutulakos and S. Seitz. A theory of shape by space carving. IJCV, 38(3):199–218, 2000.

[11] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In CVPR, vol. 2, pp. 49–54, 1999.

[12] R. Szeliski. A multi-view approach to motion and stereo. In CVPR, vol. 1, pp. 157–163, 1999.

[13] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In ECCV, vol. III, pp. 82–96, 2002.

[14] O. Faugeras, E. Bras-Mehlman, and J.-D. Boissonnat. Representing stereo data with the Delaunay triangulation. Artificial Intelligence, 44(1–2):41–87, 1990.

[15] C. J. Taylor. Surface reconstruction from feature based stereo. In ICCV, pp. 184–190, 2003.

[16] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006, June). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (Vol. 1, pp. 519-528). IEEE.

[17] Wu, J., Zhang, C., Xue, T., Freeman, B., & Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), Advances in neural information processing systems 29 (pp. 82–90). Curran Associates, Inc.

[18] Zou, C., Yumer, E., Yang, J., Ceylan, D., & Hoiem, D. (2017). 3D-PRNN: Generating shape primitives with recurrent neural networks. In Proceedings of the international conference on computer vision.

[18] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shape modeling. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[19] Vishwanath, K. V., Gupta, D., Vahdat, A., & Yocum, K. (2009, September). Modelnet: Towards a datacenter emulation environment. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing* (pp. 81-82). IEEE.

[20] Vicente, S., Carreira, J., Agapito, L., & Batista, J. (2014). Reconstructing PASCAL VOC. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[21] Kar, A., Tulsiani, S., Carreira, J., & Malik, J. (2015). Category-specific object reconstruction from a single image. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[22] Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J. (2018). Learning category-specific mesh reconstruction from image collections. In Proceedings of the European conference on computer vision.

[23] Henderson, P., & Ferrari, V. (2019). Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, 1-20.

[24] Szabó, A., Meishvili, G., & Favaro, P. (2019). Unsupervised generative 3d shape learning from natural images. *arXiv preprint arXiv:1910.00287*.

[25] Ho, L. N., Tran, A. T., Phung, Q., & Hoai, M. (2021). Toward Realistic Single-View 3D Object Reconstruction with Unsupervised Learning from Multiple Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 12600-12610).

[26] Liu, Z., Luo, P., Wang, X., & Tang, X. (2018). Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, *15*(2018), 11.