Study of Unified Volume Based 3D Reconstruction Techniques for Mobile Deployment

Alexa Rockwell
Stanford University
arkwl@stanford.edu

Abstract

This project experiments with several options for 3D object reconstruction models for deployment. The aim here is to scan an object, and export it as a .obj file, which is compatible with third party modeling applications. Third party systems like PolyCam send captured 2D images for processing to a server for 3D descriptor analysis and correspondence matching, but my project will look for optimizations for performant reconstruction on systems with limited memory. Specifically, this paper will consider two learning solutions: one with a voxel grid and another with a point cloud implementation. After training both models, I'll evaluate their accuracy and memory footprint. The findings will show that neither solutions are suitable for real world examples, and existing commercial mobile applications may use SFM solutions.

1. Introduction

The goal of this project was to train and deploy a 3D reconstruction model that only uses N 2D views as it's parameter, and outputs a mesh model. This would allow novices and experts in 3D graphic fields to feel uninhibited by the overhead of creating new detailed objects and scenes for their projects.

In order to reconstruct models that are compatible with applications like Blender. Many techniques make use of LiDAR, to generate point clouds without the need of 2D images. However, we choose to use N 2D images of a subject to generate a 3D representation of their joined images, since it's significantly more accessible than a LiDAR camera. There are options such as point clouds or voxel occupancy grids. This project will take a look at two solutions for object reconstruction, compare them, and propose optimizations to make the models suitable for deployment.

2. Problem Statement

2.1. Object Reconstruction

My project will compare and contrast 3D-R2N2 & Point Cloud Generation for Dense 3D Object Reconstruction and propose alternatives and optimizations. The dataset I will use is ShapeNet, and conveniently, both papers use it to train their models.

Major milestones in the field of 3D object reconstruction include 3D-R2N2[1], Point Cloud reconstruction [2], NeRF[3] and NSVF[4], among many other model architectures. While NeRF and NSVF are state of the art, they mainly specialize in view synthesis and are not suitable for multi-category scenarios. Strictly geometric solutions do exist, but they are also victim to noise and exponential processing time. This report will go into greater detail about the available solutions for mesh reconstruction.

The expected result will vary in form, including 3D voxel grid and point clouds. Similarly, the error metrics are different between the different representations, more detail later.

3. Related Work

3.1. 3D-R2N2

This model's architecture comprises of learning reconstruction through the use of a recurrent 3D ConvNet. Using a generic encoder and decoder structure, this paper predicts a voxel occupancy grid. Its novelty lies in the, "3D-Convolutional LSTM which allows attention mechanism to focus on visible parts in 3D", or the surface of the object. In more detail, the LSTM is responsible for the locality of the voxel unit. Another benefit to using this model is that it creates decent reconstructions using a variable number of views. It also argues it succeeds in cases where SFM and SLAM fail, especially in the cases where views may be separated by a large baseline.

The primary metric the paper uses is a voxel Intersection-over-Union (IoU) between a 3D voxel reconstruction and its ground truth voxelized model. It's loss function is a "sum of voxel-wise cross entropy" [1]:

$$L(\mathcal{X}, y) = \sum_{i,j,k} y_{(i,j,k)} \log(p_{(i,j,k)}) + (1 - y_{(i,j,k)}) \log(1 - p_{(i,j,k)})$$

This means the goal of this model is to minimize loss between the predicted and ground truth occupancy mesh. Lastly, the prediction voxel grid is of size (32, 32, 32), but the exported obj file ranges from 20 to 200 kilobytes of data. Surprisingly, this takes up less on disk memory when comparing it to the predictions of 3D Point Cloud Generation. [2]

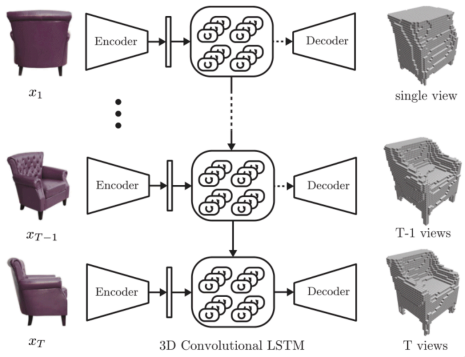

Figure 1: Overview of 3D-R2N2

3.2. Point Cloud Generation

The other paper uses, "2D convolutional operations to predict points clouds that shape the surface of the 3D objects". [2] The authors additionally report this works very well on single category training, and not multiple categories. Intuitively, the paper's structure generator "predicts the 3D structure of the object at N different viewpoints". The viewpoints can be then transformed into 3D coordinates using basic camera transformations to generate their 3D points. While we benefit from the final output taking up considerably less memory than a voxel grid the 2D convolutions still contribute to the model's memory footprint.

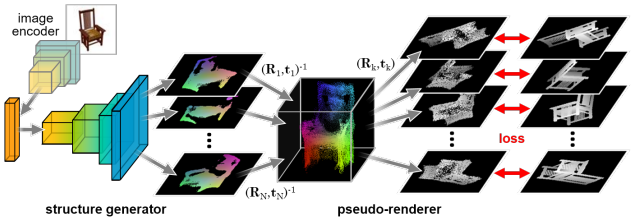

Figure 2: Point Cloud Generation Network Architecture

Lastly, this paper uses "average point-wise 3D Euclidean distance" between the predicted point cloud and the ground truth as it's error metric. It's critical to evaluate the metric bidirectionally as the "the former measures 3D shape similarity and the latter measures surface coverage." [7]. The loss function used is also a cross entropy loss for the K number of viewpoints:

$$\mathcal{L}_{\text{mask}} = \sum_{k=1}^{K} -\mathbf{M}_k \log \hat{\mathbf{M}}_k - (1 - \mathbf{M}_k) \log \left(1 - \hat{\mathbf{M}}_k\right)$$

3.3. NeRF

As a brief reminder, NeRF proposes a network whose input is a, "single continuous 5D coordinate (spatial location (x,y,z) and viewing direction (θ,ϕ)), and output is the volume density and view-dependent emitted radiance at that spatial location" [3]. It uses classic volume rendering techniques and camera rays to predict the color and intensity of the result. Though the main problem it solves is view synthesis, researchers have found other applications for it such as model reconstruction, pose estimation, and obstacle avoidance algorithms. While NeRF trains it's function approximator based on the input views, handles occlusion very well, and it has a significantly lower number of trainable parameters, it is not a suitable candidate for deployment. To handle any scenario of reconstruction, the model would have to retrain its weights in a limited environment.

3.4. NSVF

Neural Sparse Voxel Fields combine the idea behind voxel learning with camera ray projection. Namely, "learn the underlying voxel structures with a differentiable ray-marching operation". Since it uses a sparse voxel octree, the speed of rendering is considerably faster since it skips octets without any information. The authors report 10 times faster rendering speed due to this optimization [4]. This paper does mention multi-scene learning yet didn't show any results. With more time, I'd like to explore this method of object reconstruction.

3.5. SFM

Structure from Motion is also another great candidate for object reconstruction. It produces a 3D reconstruction by overlapping key feature points between N views. While it is a scene and object agnostic method, it does not perform well with occlusion and a high level of detail. When reconstructing small objects, its performance is reasonable.

4. Technical Approach

4.1. Image preprocessing

The typical image resolution of a photo taken from standard mobile cameras is (3024, 4032, 3). However, 3D-R2N2's input is of size (127, 127, 3). In addition to resizing the input, we also can compare how the model performs with natural backgrounds or a matte backdrop. Image segmentation is commonly used [5] as it eliminates noise from the input. This step also contributes to lower test errors when the network is trained with ShapeNet. The dataset has images with blank backgrounds, and the problem's constraint is that we don't have any depth knowledge. While occluded data isn't present in the dataset, it's still a common problem encountered when

validating if a model is suitable for a real-world environment.

4.2. Implementation

My approach is to take both implementations of 3D-R2N2, and Point Cloud Generation, train both models using multiple categories in ShapeNet, and evaluate if they're suitable for 3D object reconstruction in a limited setting. My contributions include training, data preprocessing, adding summary reporting, and altering base repositories to report additional data metrics. However, the authors or 3D Point Cloud Generation did not add their additional preprocessing step to the repository, so I will only use their single category subset for training.

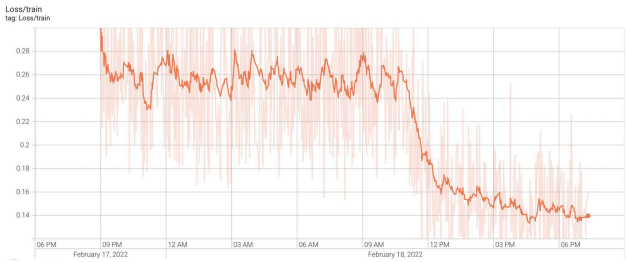

Figure 3: Training Loss per Iteration for 3D-R2N2 (200k Iterations)

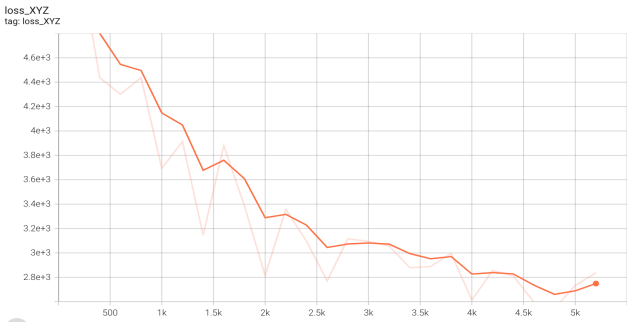

Figure 4: Training Loss per Iteration for 3D Point Cloud Generation (500k Iterations)

5. Experiments

5.1. 3D-R2N2

After training 3D-R2N2, I observed the following results in voxel grid form (Figure 4 & 5). Figure 4 shows the prediction of a sample from the training set on the left and the prediction of a sample from the test set on the right. To see if the models are suitable for real-world objects, I captured the voxel predictions for a real-world figurine (Figure 5). When qualitatively comparing the real-world input and prediction of the pretrained...

Figure 4: Input and Voxel Grid Predictions using Pre-Trained 3D-R2N2

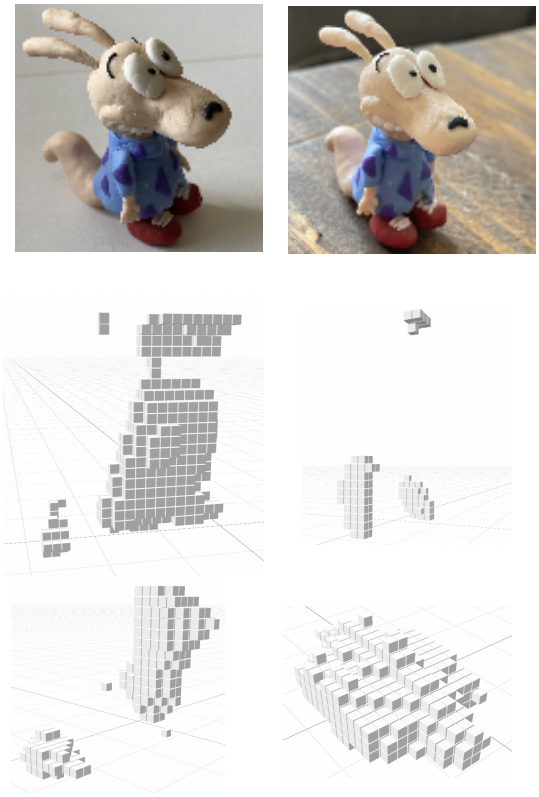

Figure 5: Input (Top Row), Pre-Trained Predictions (Middle Row) and Multi-Category Custom Trained Voxel Prediction

… network, the model somewhat succeeded with detecting

figurine's chin, neck, and body. When we try the same object with noisy background, the prediction is significantly less accurate. Lastly, the custom trained model's performance was significantly lower than the model loaded with the pretrained weights.

3D-R2N2 Pre-trained Single Category Error Metrics:

	AP	IoU(40%)	IoU(60%)	IoU(80%)
Training	0.892	0.6341	0.6114	0.5327
Test	0.818	0.5877	0.5192	0.4045

3D-R2N2 Trained Multi Category Error Metrics:

	AP	IoU(40%)	IoU(60%)	IoU(80%)
Training	0.546	0.4992	0.4673	0.3981
Test	0.432	0.5877	0.5192	0.4045

These results also show that average precision (AP) can obfuscate the true performance of a volumetric representation, and IoU is a lot more reliable. Since AP factors in false positives and not false negatives, it increases when objects are more compact and have smoother features, since they will be easier to predict. However, as soon as the focus object increases in detail, we will see the AP stay about the same, but due to the increase of false negatives, IoU will decrease.

5.2. Point Cloud Generation for Dense 3D Object Reconstruction

Since the authors of this paper did not release their preprocessing code, I evaluated its single category performance only. It shows in the following results that single category models do not succeed in a real world setting since this neural network seems to have encoded the nature of its subject and not the behavior of its surface.

Point Cloud Generation Trained Single Category Error Metrics:

	Prediction→GT	GT→Prediction
Training	1.768	1.763
Test	0.020404	0.013139

The prediction for the trained dataset looked qualitatively decent, the model clearly shows signs of overfitting. This should be solved by introducing multiple categories. If that doesn't work, other solutions to consider are reducing the number of layers, adding regularizations, and using Dropout layers.

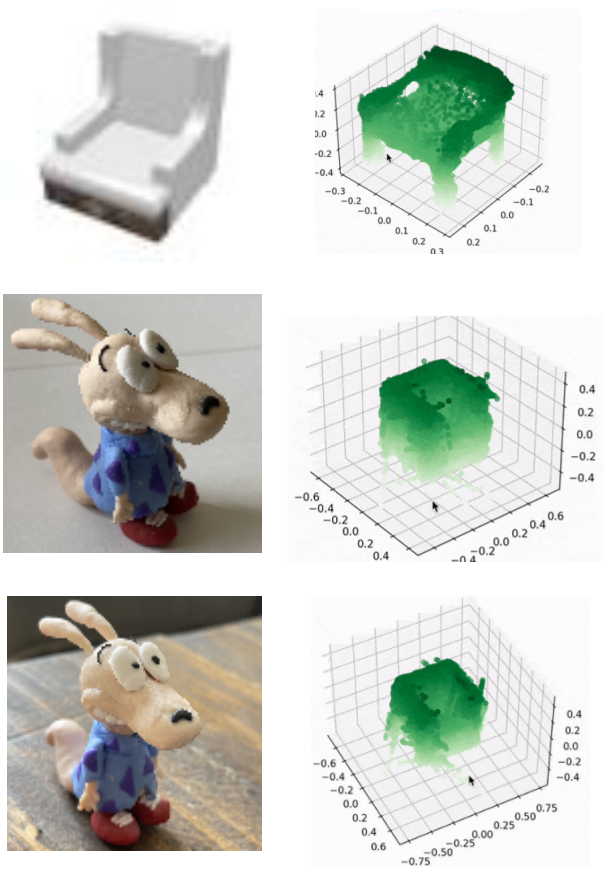

Figure 5: Test input (first column) and output (second column) for Point Cloud Generation

5.3. Memory Efficiency

Another metrics we're going to compare are the runtime memory usage and the size of the trained weights. Considering an environment with limited memory, we have to ensure inference can be performant. Lin argues, "3D ConvNets are extremely wasteful in both computation and memory in trying to predict much useless data with high-complexity 3D convolutions, severely limiting the granularity of the 3D volumetric shapes that can be modeled even on high-end GPU-nodes commonly used in deep learning research."[2] While he makes this claim, the number of parameters surpasses the number of parameters in 3D-R2N2.

	Number of Trainable Parameters
3D-R2N2	4,894,722
Point Cloud Generation	108,389,792

While the output of the point cloud generation is not as large as the voxel, the convolutional layers still increase the size of the network.

A naïve approach to making 3D-R2N2, or any neural network more robust to finer details of a model would be to increase the input size from (127, 127, 3) to (256, 256, 3). That increase in input size would expand the trainable parameters size, into the 14 digits at minimum. This tells us that increasing the number of layers or increasing the size of voxel grid predictions is not an optimal solution for object reconstruction.

Possible techniques to reduce the number of trainable parameters include, replacing dense convolutional layers with average pooling [8] and trying radiance field solutions [3].

5.4. Comparison between the two aforementioned techniques

All in all, the only difference between 3D-R2N2 and 3D point cloud learning was their output representation. Otherwise, their architecture was very similar, they share the same loss function, and they both belong in the category of volume-based learning.

6. Conclusion

6.1. Overview

From these findings, I surmise that volume learning implicitly encodes the nature of the categories the models where trained with. 3D Point Cloud showed that when a model is trained on a single category, the output will always be in the form similar to the training category, no matter what input. The multi category training done with 3D-R2N2 showed that it was unable to handle the detail of an object it hasn't seen before. Intuitively, it could not reason its surface shape since it wasn't similar to the images in the training set. If we want a reliable object reconstruction method, ignore these methods and either use NeRF on accelerated hardware, or use SFM if noise is acceptable.

The biggest lesson learned from this project is that object reconstruction is not meant for limited constraints. It makes sense why apps like PolyCam do send N 2D images to render farms. The number of trainable parameters and the size of the predicted output both scale exponentially. Next, within the scope of object reconstruction, errors can arise at any step. They can be sourced anywhere from data preparation, noise, overfitting vs underfitting, to not having enough views, to in adequate model architecture, to insufficient training.

6.2. The Issue of Dimensionality

Something that surprised me was that most model input sizes were (127, 127, 3). If a person with a mobile phone wanted to process N images of default size (3024, 4032, 3), they would either have to down sample the images considerably or take segmentation masks. Or both. Either solution loses detail or will result in a low-quality rendering if we were to use a volume-based approach to volume rendering. In this case, SFM may have an advantage as it can process images of any size without the need of trainable weights.

6.3. Future Work

With more time, I'd certainly try multi-category learning with the 3D Point Cloud Learning architecture. Since 3D Point Cloud Learning's implementation requires 24 viewpoints, I could also restructure it to train on fewer or variable views, or tangentially: train the model on noisy and occluded versions of ShapeNet images. Another thing I would like to try with more time would be to encode the surface texture into the predicted reconstruction.

I attempted to try PolyCam's Photo Rendering mode on the real-world figurine and was able to extrapolate some details from their rendering. While the majority of the object looks well reconstructed, the method seems to have failed in areas where the structure was occluded, near the neck and underside of the figurine.

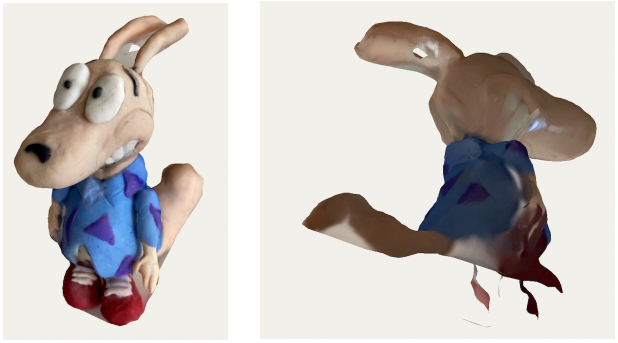

Figure 6: PolyCam's Object Reconstruction (48 views)

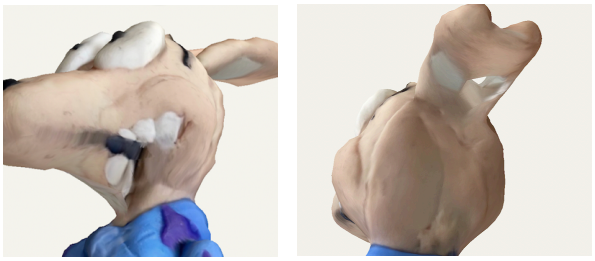

Figure 7: Flaws in PolyCam's Object Reconstruction (48 views)

My hypothesis is that this method does use SFM to combine the views into the final reconstruction. Firstly,

the high number of views indicate that the reconstruction was likely stitched together, and there was no method of in painting at occluded regions. Next, the reconstructed result blended the figurine's ears together. Likely, any feature descriptor method like SLAM or SIFT could have been confused by their similarities and assumed it was the same shape. SFM's weaknesses is that it's not robust to cases like this.

Lastly, I also wanted to port these models to CoreML or compress them into a smaller representation with a smaller number of trainable parameters. With more time, I'd like to pursue that and evaluate error before & after conversion.

6.4. Model Optimizations

PyTorch's Performance Tuning Guide recommends the following techniques for pursuing model optimization. This continues my last point of future works and techniques I would like to try with more time:

6.4.1 Enable async data loading and augmentation

Using PyTorch API, one could define the number of threads to load in the dataset for training and inference.

6.4.2 Disable gradient calculation for validation or inference

Gradients aren't needed for validation or inference. Removing them or adding flags to bypass gradient calculation will accelerate execution and reduces the amount of required memory.

6.4.3 Enable channels_last memory format for computer vision models

PyTorch 1.5 introduced support for channels_last memory format for convolutional networks. This format is meant to be used in conjunction with AMP to further accelerate convolutional neural networks with Tensor Cores.

6.4.4 Avoid unnecessary CPU-GPU synchronization

PyTorch also recommends avoiding unnecessary synchronizations, to let the CPU run ahead of the accelerator as much as possible to make sure that the accelerator work queue contains many operations. This include removing use of cuda_tensor.item() and memory copies like tensor.cuda()

6.4.5 Use PyTorch Profiler

This new profiler collects both GPU hardware and PyTorch related information, correlates them, performs automatic detection of bottlenecks in the model, and generates recommendations on how to resolve these bottlenecks. All of this information from the profiler is visualized for the user in TensorBoard.

7. Acknowledgments

Thank you teaching team for a great CS231A!

8. Link to Repository

https://github.com/arkwl/cs231A-project

References

[1] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, Silvio Savarese: "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", 2016

[2] Chen-Hsuan Lin, Chen Kong, Simon Lucey: "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction", 2017.

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng: "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", 2020

[4] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, Christian Theobalt: "Neural Sparse Voxel Fields", 2020

[5] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In European Conference on Computer Vision, pages 371–386, 2018.

[6] Alessandro Simoni, Stefano Pini, Roberto Vezzani, Rita Cucchiara. Multi-Category Mesh Reconstruction From Image Collections. 2021.

[7] C. Kong, C.-H. Lin, and S. Lucey. Using locally corresponding cad models for dense 3d reconstructions from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[8] MachineCurve. (2020, January 30). What are Max Pooling, Average Pooling, Global Max Pooling and Global Average Pooling?