DEPTH MAP RECONSTRUCTION

Alex Hanson
Department of Computer Science
University of Maryland
College Park, MD 20740

Daniel Lichy
Department of Computer Science
University of Maryland
College Park, MD 20740

1 Introduction

Reconstructing 3D geometry from a single image is a highly challenging problem, but is never-the-less important due to its many applications in computer graphics and computer vision. For example, 3D reconstruction is necessary for physically accurate 3D relighting, rendering objects from a different view point, and editing of 3D shapes.

The main challenge of 3D reconstruct from a single image is due to the problem's ill-posedness, that is many different geometry and lighting conditions can produce the same image. Thus other constraints must be imposed on a scene to generate physically plausible objects. These constraints include assumptions about topology, planarity, minimal surfaces, volumes, and learned shape priors [1]. We concentrate mainly on the shape priors; we attempt to select the shape that is the most probable for producing a given image.

In this work we conduct a number of experiments to reconstruct objects from a single image by learning priors on the models in the ShapeNet dataset. We represent 3D objects as multiview depth maps from known views. Depth maps have the advantage that they are image like data and thus can be used as input to a standard convolutional neural network (CNN) without modification. However, they have the disadvantage that they are not sampled uniformly on the surface of the object. For example, regions where the tangent plane is nearly orthogonal to the image plane the density of sample points is low, however this issue is alleviated when multiple views are given.

2 Prior Work

2.1 Classical Approaches

There are many classical approaches to 3D reconstruction from a single image. Usually these approaches propose some prior assumptions about the 3D object and then formulate reconstruction as an optimization problem. We do not provide a detailed review of this large body of literature, but we refer the interested reader to the introduction of [1].

2.2 Deep Learning Based Approaches

In recent years, many deep learning based approaches to 3D reconstruction have been proposed. There are two main challenges to applying deep networks to 3D reconstruction. The first is choosing how to represent 3D geometry in a CNN, and the second is how take a 2D image to the geometry that best explains it in the chosen representation. We briefly review some of these methods.

Wu proposed representing 3D shapes as binary voxel occupancy. This has the advantage that convolutions naturally generalize to it but the memory it requires grows with the cube of the resolution and thus does not scale well to higher resolutions [2]. Although Wu doesn't apply this directly to 3D reconstruction from 2D images this representation is applied to 3D reconstruction in [3]. [4] proposes a sparse way to represent voxel occupancy, by storing just the starting and stopping points of each voxel tube. This can also be interpreted as a generalization of a multiview depth map, since a depthmap from the front and rear of an object can be thought of as the starting and stopping point of the tube. [5] also addresses these storage concerns by developing an octree based volume decoder to more efficiently represent voxel information.

Alternatively to voxel based representations, [6] proposed a way to preform convolutions on point clouds and uses this to develop a CNN to reconstruct point clouds from images. In another approach, [7] develops a differentiable

mesh renderer and trains a network to generate deformation vectors for each vertex on a template sphere, such that the rendering of the deformed sphere is similar to the original image. [8] proposes an encoder that takes an input image of an object to a view independent representation of the object, and a decoder that takes a camera pose and returns an RGBD rendering of the shape from the input image.

3 Approach

3.1 Dataset

The ShapeNet [9] dataset is a large-scale repository of common objects represented by 3D CAD models. In this work, we use ShapeNetCore, a densely annotated subset of ShapeNet consisting of 55 common object categories and ~51,300 unique 3D models. The object representations are stored in .obj mesh files and associated texture/material .mdl files.

Due to the long-tailed distribution of these object classes, we chose to further refine the dataset and use the 10 most populous classes, listed here in descending order: table, chair, airplane, car, sofa, rifle, lamp, vessel (boat), bench, and loudspeaker. This refinement still leaves ~36,000 models with the largest category, table, represented with ~8400 models and the smallest category, loudspeaker, represented with ~1600 models.

We evaluate our 3D reconstruct of cars on a subset of the Carvana dataset [10]. Originally collected for a car segmentation challenge, the dataset consists of ~5000 car images with ground-truth masks, and ~100,000 car images without masks. We only used the masked training images for testing.

3.2 Data Generation

3.2.1 Renderer

Neural networks require a very large amount of data to train. Therefore, we wanted a renderer that could generate data very quickly. Furthermore, we needed a renderer that could generate depth maps, or could be easily modified to generate depth maps. For these reasons, and also because we wanted to learn about OpenGL and non-photorealist real time rendering, we chose to implement our renderer based on the tutorials available at `https://learnopengl.com/` and code at `https://github.com/JoeyDeVries/LearnOpenGL`.

The code available already included methods to load .obj files and the companion .mdl files. These methods only required slight modification to read diffuse and specular coefficients from the .mdl files if textures are not available, which is the case for some Shapenet objects.

Most of the rendering computations in OpenGL are done in small shader kernels. For simplicity we used a shader that only supports directional lights. We modified it to store depth values in a fourth channel usually reserved for transparency values. We chose to render images as 32 bit floating point values. This has the advantage of not having to scale the depthmaps at the cost of requiring more memory. Initially we wanted to generate our data on the fly in order to avoid storing 100s of GBs of memory on the hard drive. This would make it easier to train the network on different machines without having to transfer large datasets. Additionally, changing the rendering parameters would not require a regeneration of the data. Unfortunately, we did not get far enough on developing this feature for the renderer and ended up storing images on disk anyways. Images are stored as binary files to avoid relying on images libraries that support floating point, four channel images.

The final renderer takes in a model, a sequence of views, and the properties to specify two directional lights per view. It then renders the object from each of these views with the corresponding two directional lights.

3.3 Network Architectures

3.3.1 Simple U-net

The simplest architecture we use is a U-net with skip connections. This architecture reconstructs 3D faces in a follow up to [11] submitted to TPAMI 2019. Details of the architecture are shown in Figure 1.

3.3.2 Six Head U-net

For mulitivew depth map reconstruction, we replicated the head of the simple U-net to produce six outputs of one channel each.

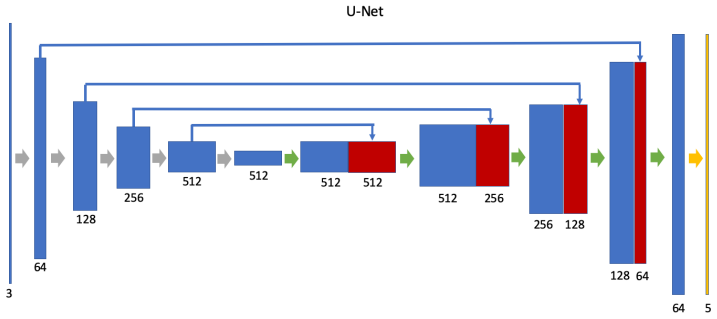

Figure 1: Simple U-net architecture. Blue and red rectangles represent data passing through the network, below them are the number of channels. Three input channels correspond to the input RGB image and the five output channels correspond to one RGB image and two depth maps. Red rectangles are simply copies of the data represented by the blue rectangle that point to them. Gray arrows represent convolutions with kernel size 3, stride 2 followed by batch normalization and leaky ReLU (slope=0.2) activation. Green arrows are convolution with stride 1, and kernel size 3, followed by bilinear upsampling, batch normalization, and ReLU activation. The gold arrow is a simple convolution layer with ReLU activation.

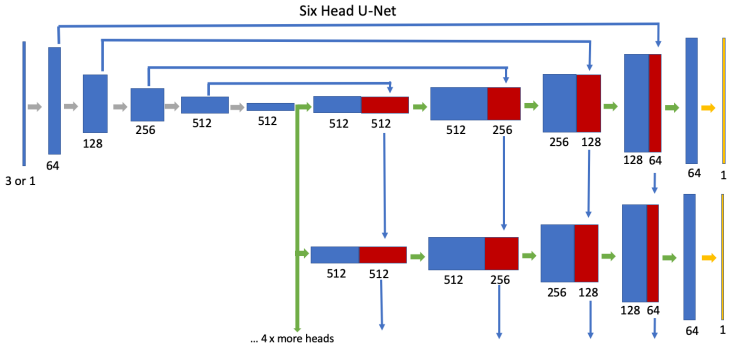

Figure 2: Six Head U-net architecture. This architecture is identical to the simple U-net architecture except that the head of the U-net is replicated so as to produce six outputs. The input to this network is either an RGB image of three channels or a depth map of one channel. The outputs are depth maps of one channel.

3.3.3 Six Head U-net with intermediate loss

We additionally experiment with applying an intermediate loss to the upsampled features of the Six Head U-net. To do so, we add a convolution layer to each upsampled feature map constructing a one channel output at each layer.

4 Experiments

4.1 Multiview Depth Map Reconstruction Experiment

4.1.1 Concept

In this experiment, we attempt to learn depth maps from multiple views to represent a 3D object given a 2D input, such as an RGB image or depth map. We hypothesized that a U-net can learn relevant shape representations to construct multiple depth map views simultaneously.

4.1.2 Data

To simplify the task, we chose fixed camera positions for each input and output. Since all ShapeNet objects are centered at the origin and oriented along the axes, we position the input camera at $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, point it at $(0, 0, 0)$, and position "up" to align with the y-axis, resulting in a diagonal view of the objects. The plane that is perpendicular to this view vector

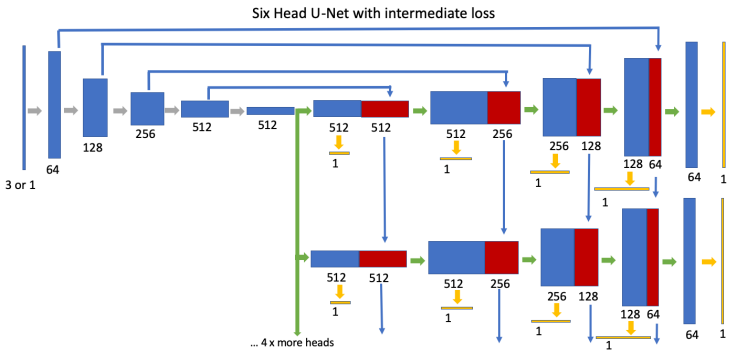

Figure 3: Six Head U-net architecture with intermediate loss. This architecture is identical to the Six Head U-net, with the addition of a simple convolution layer applied to each upsampled feature map to construct intermediate one channel outputs in addition to the final network one channel outputs.

and goes through the point $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, intersects the unit sphere in a circle of radius $\frac{1}{2}$. All points along this circle form a 30° angle between the input camera and the origin. We chose six equally spaced output camera positions along this circle: N, NW, SW, S, SE, and NE, where N is the point on the circle the camera would hit if traveling "up" in the circle plane.

In addition to fixing the views, we fix two diffuse lights. One at $(1, 0, 0)$ and the second at $(0, 0, 1)$, both facing the origin.

Given these fixed views and lights we generate input/output images by simply swapping objects in the renderer. We do this for all 10 classes of objects listed in section 3.1. The average input/outputs for the airplane class are given in Figure 4 and the average input/outputs for the lamp class are given in Figure 5. Similarly the average input/outputs of all classes is given in Figure 6.

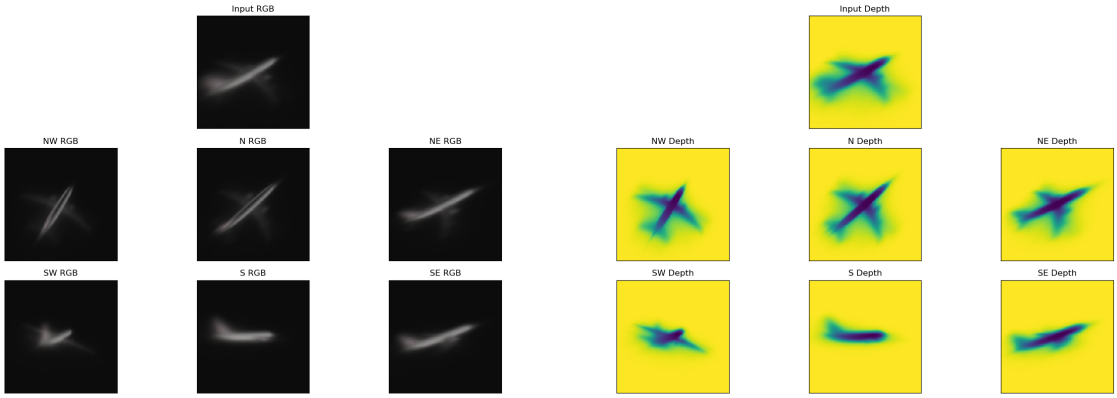

Figure 4: Average RGB image and depth map views of the airplane class

When comparing Figure 4 and Figure 5, it's clear that some classes have a distinctive average "structure" while others do not. We suspect that a network will perform better on classes with this structure.

To confirm this, we chose to train two networks on the airplane class, one whose input is RGB images and the other whose input is depth maps. The outputs of both networks are six depth maps: N, NW, SW, S, SE, NE. Similarly we chose to train another two networks on the lamp class and a further two networks on all classes.

Each dataset is randomly split into 90% training data and 10% test data. The split is done by object and not by image. As such, all images of a particular object are either all in training or all in test with no overlap.

4.1.3 Procedure

We make a slight modification to the simple U-net architecture described in section 3.3.1 by replicating the upsampling "head" such that the U-net produces six outputs instead of one. This is shown in Figure 2 and section 3.3.2.

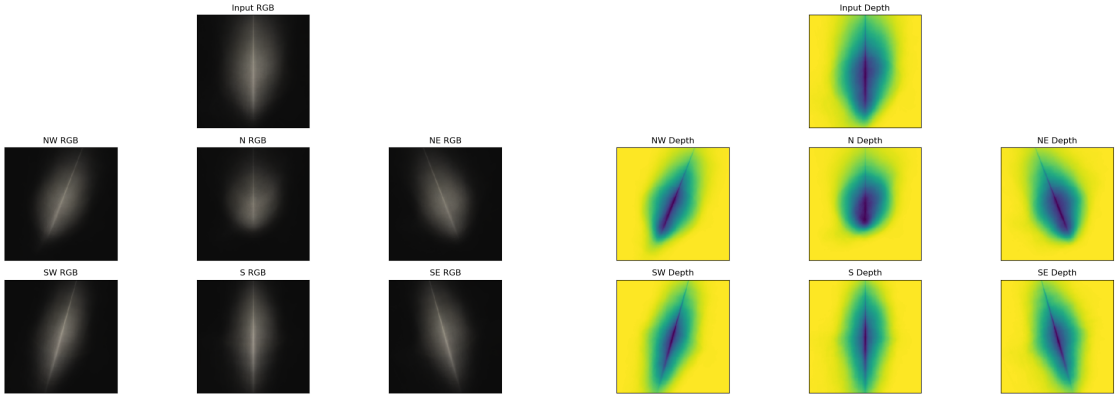

Figure 5: Average RGB image and depth map views of the lamp class

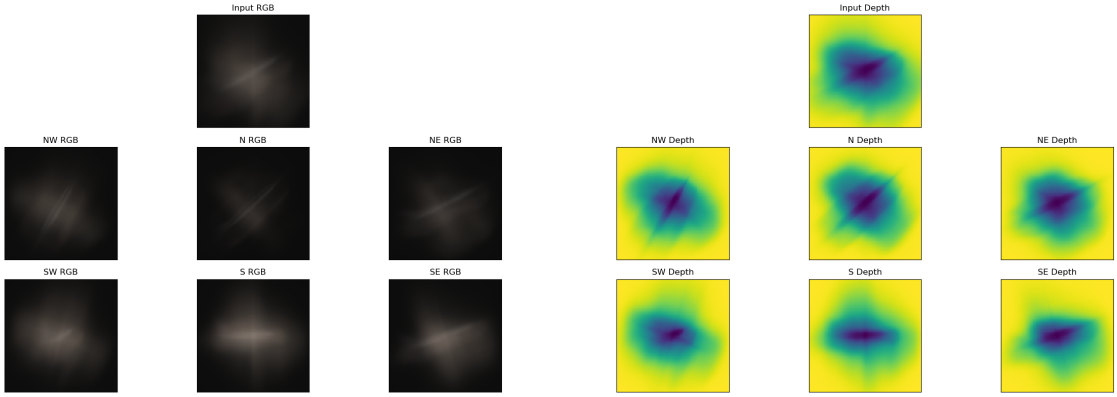

Figure 6: Average RGB image and depth map views of all classes

We train the network for 50 epochs with the following loss function:

$$\|o_1 - d_N\|_1 + \|o_2 - d_{NW}\|_1 + \|o_3 - d_{SW}\|_1 + \|o_4 - d_S\|_1 + \|o_5 - d_{SE}\|_1 + \|o_6 - d_{NE}\|_1$$

Where $\|\cdot\|_1$ denotes the $L1$ norm, o_n is the output of head n , d_X is the ground truth depth map for direction X.

We extended this experiment by attempting to guide the intermediate upsampled feature maps by applying a loss to these feature maps during training. We do this by adding a convolution layer to these feature maps to produce a one channel output per feature map. Then perform an L1 loss between each of these feature map outputs and a downsampled ground truth depth map of the respective head. We scale these losses by the ratio of the overall output size to the feature map output size and sum all losses. This slightly more complicated loss is given by:

$\sum_{i=0}^{4} \frac{1}{4^i}\|o_1[i] - d_N[i]\|_1 + \frac{1}{4^i}\|o_2[i] - d_{NW}[i]\|_1 + \frac{1}{4^i}\|o_3[i] - d_{SW}[i]\|_1 + \frac{1}{4^i}\|o_4[i] - d_S[i]\|_1 + \frac{1}{4^i}\|o_5[i] - d_{SE}[i]\|_1 + \frac{1}{4^i}\|o_6[i] - d_{NE}[i]\|_1$

Where $\|\cdot\|_1$ denotes the $L1$ norm, $o_n[i]$ is the output of head n at feature map i, $d_X[i]$ is the ground truth depth map for direction X downsampled by $\frac{1}{4^i}$. i indexes from the output of the final layer inward, so index 0 is the full sized depth map with resolution of 256×256, whereas index 4 has a resolution of only 16×16. Thes networks are also trained for 50 epochs.

4.2 Front-Back Depth Map Reconstruction Experiment

4.2.1 Concept

After some experimentation, we notice that U-nets with skipped connections seem to be good at "recoloring" images. Since a depth map from the front and back of an object captures most of its geometry we propose to learn a depthmap of an image of an object as well as a "reverse" or "back" depthmap.

If the value of a depth map at a pixel is the distance along the ray of projection passing though that pixel to where the ray first intersects the mesh, then the value of the "back" depthmap, at the same pixel, is the distance where the same ray last intersects the mesh. In the orthographic limit, this is the same as a depthmap with the camera on the opposite side of the object and mirrored about the horizontal axis. This is how we generate the back depth map in practice, though it can also be done with a z-buffering change by overwriting the criteria from less than to greater than.

4.2.2 Data

We conduct this experiment on two ShapeNet objects: car and table. For each car, 15 camera positions are sampled randomly on a sphere of radius 14. For each table 10 camera positions are sampled, also from posiitons on a sphere or radius 14. Azimuth angles were selected from a uniform random distribution. Elevation angles are sampled from a Gaussian with mean $\pi/2$ and standard deviation 0.2. The "to" direction is always the origin, where the object is located, and the "up" direction is always in a plane with the y axis.

For simplicity, lighting is always along the plus and minus x-axis in world coordinates. We use diffuse lighting with intensity 0.8 for each channel and ambient lighting at intensity 0.2 for each channel.

The dataset is randomly split into 90% training data and 10% test data. The split is done by object and not by images. As such, all images of a particular object are either all in training or all in test with no overlap.

4.2.3 Procedure

We train the simple U-net, Figure 1, to take an RGB images of an object and predicted a depth map, back depth map and back image, the RGB image from the opposite side mirrored about the vertical axis.

We train the network for 10 epochs with the following loss function:

$$\|FD, gt\_FD\|_1 + \|BD, gt\_BD\|_1 + \|BI, gt\_BI\|_1 + 0.2 \cdot \|FN, gt\_FN\|_1 + 0.2 \cdot \|BN, gt\_BN\|_1$$

Where $\|\cdot\|_1$ denotes the $L1$ norm, FD is the front depth, BD is the back depth, BI is the back image, FN is the front depth normals, and BN is the back depth normals. The prefix $gt\_$ is used to differentiate the ground true from the network output.

The front normals and back normals are calculated by applying Sobel filters to the front depth and back depth respectively to calculate $\frac{\partial depth}{\partial x}$ and $\frac{\partial depth}{\partial y}$. Concatinating these partial derivatives and one gives $(\frac{\partial depth}{\partial x}, -\frac{\partial depth}{\partial y}, 1)$. Normalizing this vector gives the surface normals. This is done for both ground truth depths and the network output depths.

We add the loss to the normals during the first epochs and it seems to help the network converge. However, we haven't done a proper ablation study on this yet. The intuition fro why this should work is because flat surfaces should have a constant normal and thus a constant color. Hence, learning normals from color should be easier than depth, since depths will vary along constant a color.

From the depth maps we recover point clouds by concatenating each pixel's coordinates with its depth value. We color each point with the corresponding color from the original image for the front depth map and by the that of the generated back image for the back depth map. After transforming these point clouds to common coordinates we concatenate them for the final reconstruct. This process is shown in figure 7.

5 Results

5.1 Multiview Depth Map Reconstruction Results

The network trained on airplanes seems to preform well at reconstructing unseen airplanes. This is demonstrated in Figure 8, Figure 9, Figure 10, and Figure 11. Adding the feature map loss seems to help the reconstruction in thinner areas of the model (e.g. the wings of the airplane when viewed from the side). This is demonstrated in Figure 12, Figure 13, Figure 14, and Figure 15.

The network trained on lamps doesn't seem to preform nearly as well well at reconstructing unseen lamps. This is demonstrated in Figure 16, Figure 17, Figure 18, and Figure 19. Adding the feature map loss does seem to reduce disconnected regions during the reconstruction of lamps. This is demonstrated in Figure 20, Figure 21, Figure 22, and Figure 23.

The network trained on all classes (with feature map loss) seemed to also perform well on the airplane class as demonstrated in Figure 24 and Figure 25. It did not perform as well on the randomly selected lamp test image, Figure

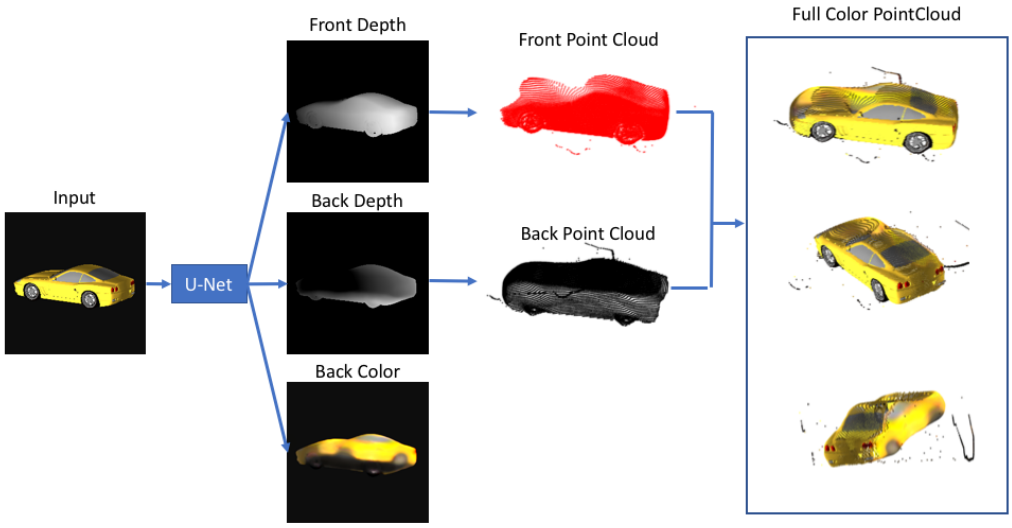

Figure 7: Front-Back depth map reconstruction process

26 and Figure 27. However, this is likely due to the deviation of this particular randomly chosen sample from the lamp class images (refer to Figure 5). This network also performed will on chair and rifle classes as seen in Figure 28, Figure 29, Figure 30, and Figure 31. We suspect this implies the network is learning to distinguish classes before attempting the reconstruction of depth maps. It would be interesting to further examine this in future experiments.

5.2 Front-Back Depth Map Reconstruction Results

We reconstruct a number of cars and tables from the test set of our data. Our method works decently although we haven't compared it to other reconstruction methods. Results for cars are shown in Figure 32 and results for tables are shown in Figure 6.

We note that the side of the object not visible in the picture is fairly blurry in the reconstruction. It would be interesting to incorporate a GAN loss to the output image to try to get a sharper result.

We also apply our method to images of real cars from the Carvana dataset [10] some results of this are displayed in Figure 34. The results are not as visually pleasing as the synthetic data. This has a few explanations. One is that our data is generated naively; we only account for diffuse lighting, but cars are metallic and thus highly specular. Additionally, our synthetic data only has lighting along the x-axis, whereas the lighting in the Carvana dataset is more varied. In the future it would be interesting to see if we can get better results by rendering with specular highlights and more varied lighting.

6 Conclusion

Reconstructing both mutiview depth maps and front-back depth maps produced reasonable looking results, but neither was photo-realistic. In particular, more work could be done to preserve the smaller more complex surfaces in the input. Future work could keep this structure from the input and attempt to locate symmetries to "copy" this structure instead of trying to reconstruct it from scratch.

References

[1] E. Töppe, C. Nieuwenhuis, and D. Cremers. Relative volume constraints for single view 3d reconstruction. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 177–184, June 2013.

[2] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, June 2015.

[3] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *CoRR*, abs/1604.00449, 2016.

[4] Stephan R. Richter and Stefan Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. *CoRR*, abs/1804.10975, 2018.

[5] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *CoRR*, abs/1703.09438, 2017.

[6] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016.

[7] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *CoRR*, abs/1901.05567, 2019.

[8] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR*, abs/1511.06702, 2015.

[9] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[10] Kaggle. Carvana image masking challenge, 2017. data retrieved from Kaggle, `https://www.kaggle.com/c/carvana-image-masking-challenge/data`.

[11] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D. Castillo, and David W. Jacobs. Sfsnet : Learning shape, reflectance and illuminance of faces in the wild. *CoRR*, abs/1712.01261, 2017.

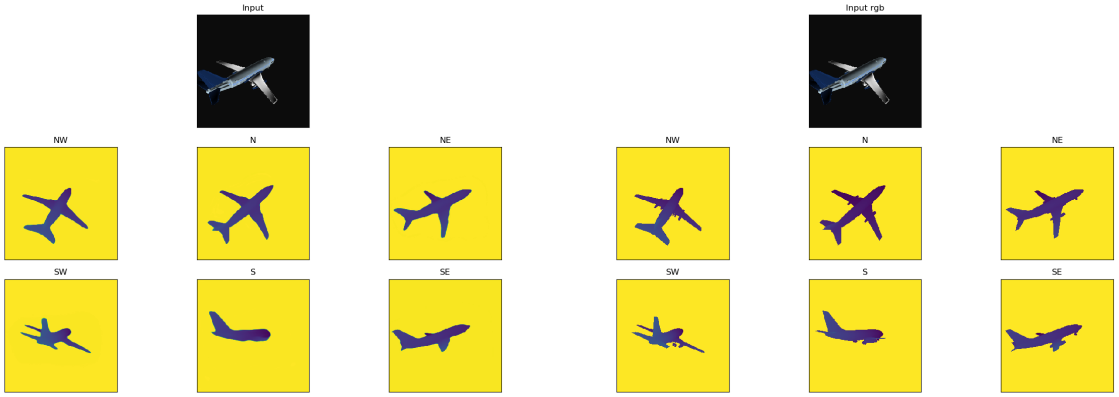

Figure 8: Example of airplane depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

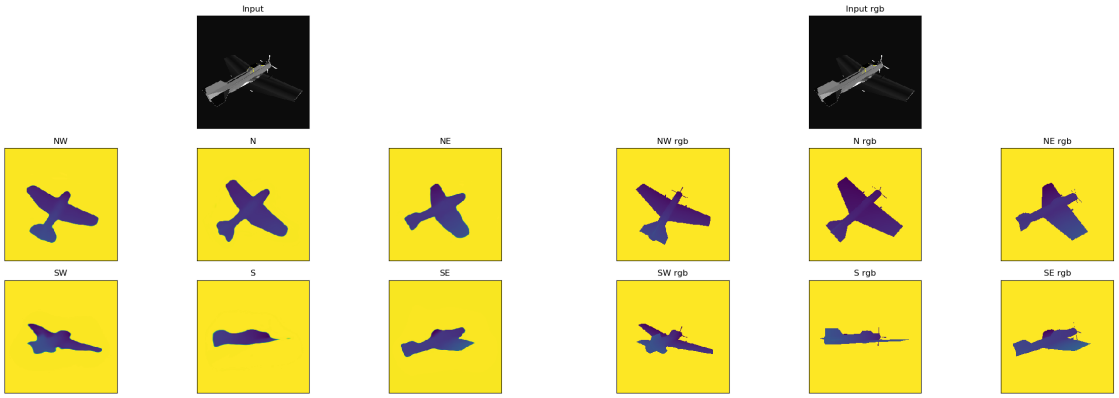

Figure 9: Another example of airplane depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

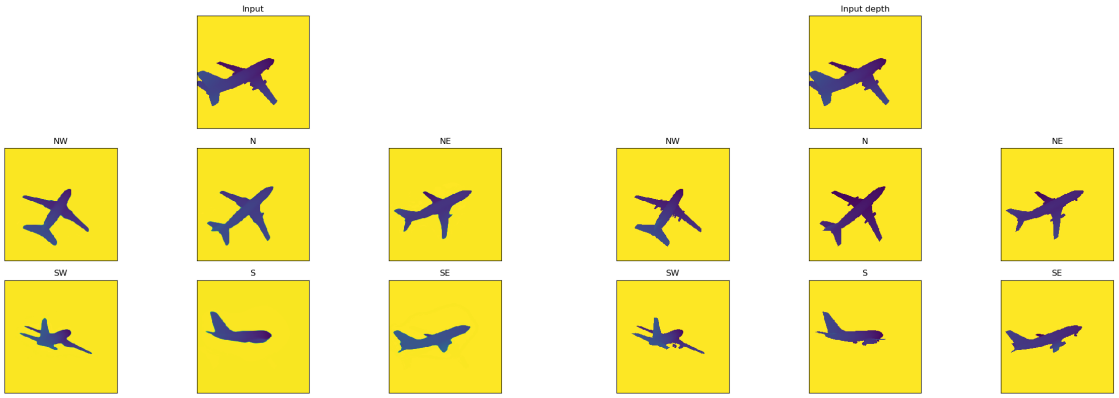

Figure 10: Example of airplane depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

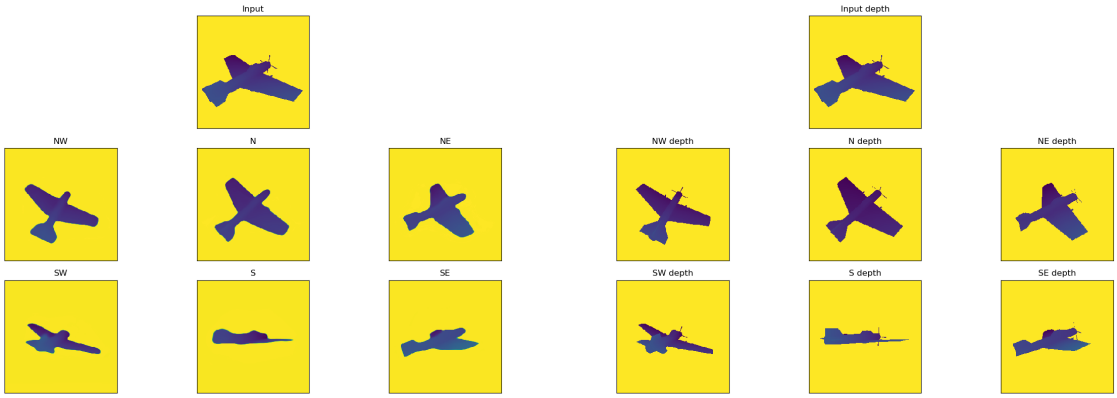

Figure 11: Another example of airplane depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

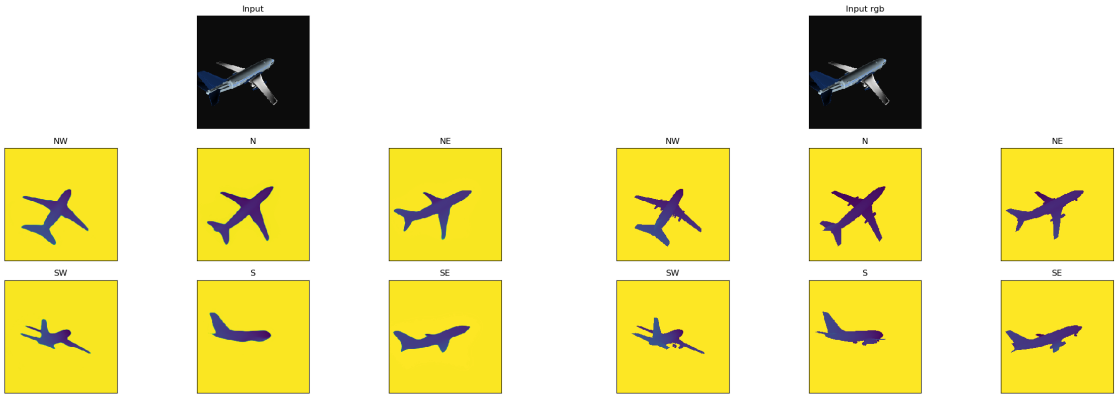

Figure 12: Example of airplane depth map predictions via input RGB image with feature map loss (left) vs. the ground truth depth maps (right)

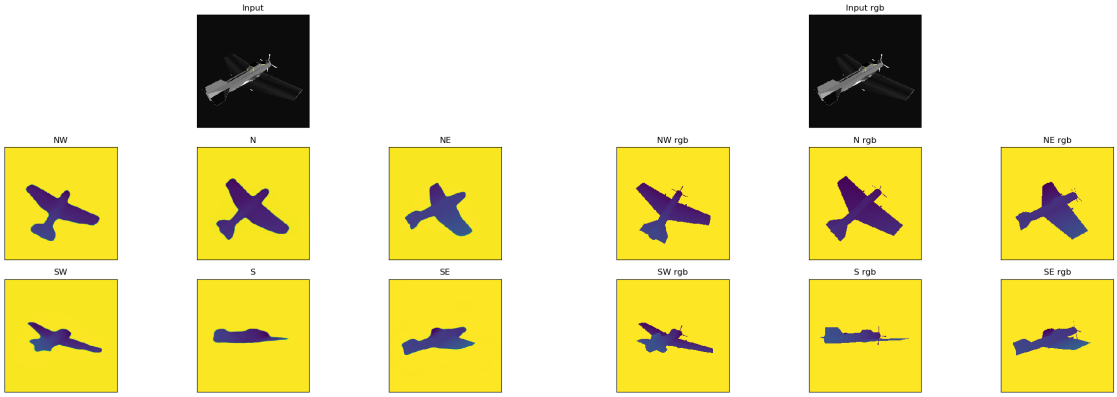

Figure 13: Another example of airplane depth map predictions via input RGB image with feature map loss (left) vs. the ground truth depth maps (right)

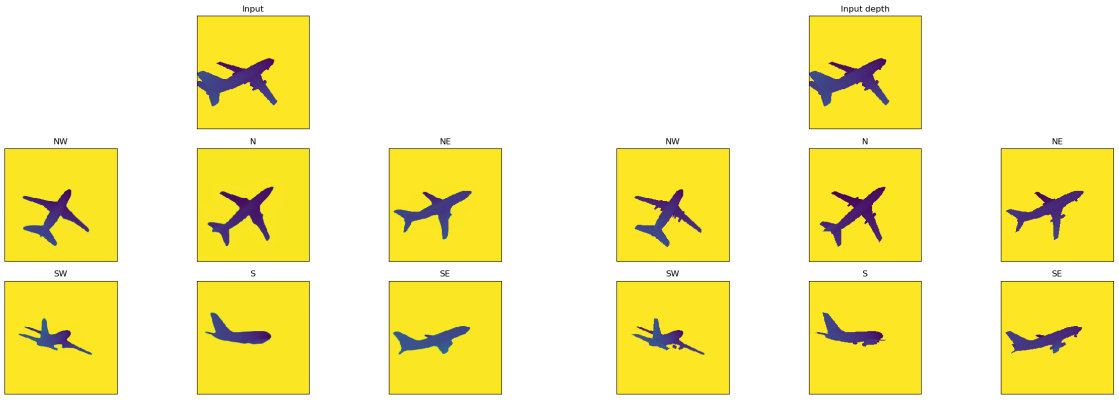

Figure 14: Example of airplane depth map predictions via input depth map with feature map loss (left) vs. the ground truth depth maps (right)

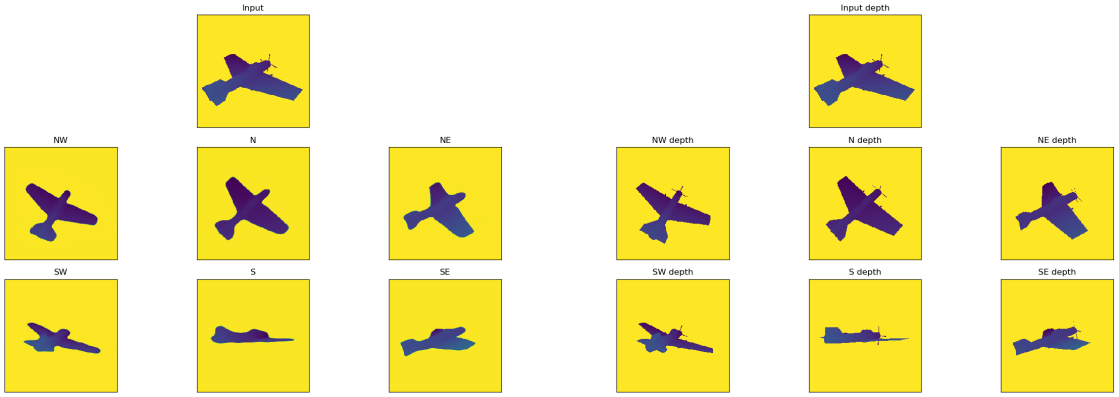

Figure 15: Another example of airplane depth map predictions via input depth map with feature map loss (left) vs. the ground truth depth maps (right)

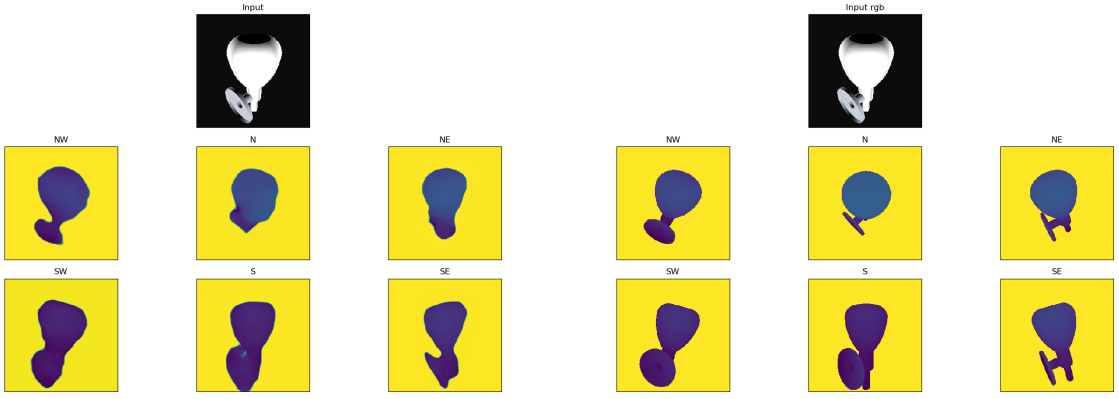

Figure 16: Example of lamp depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

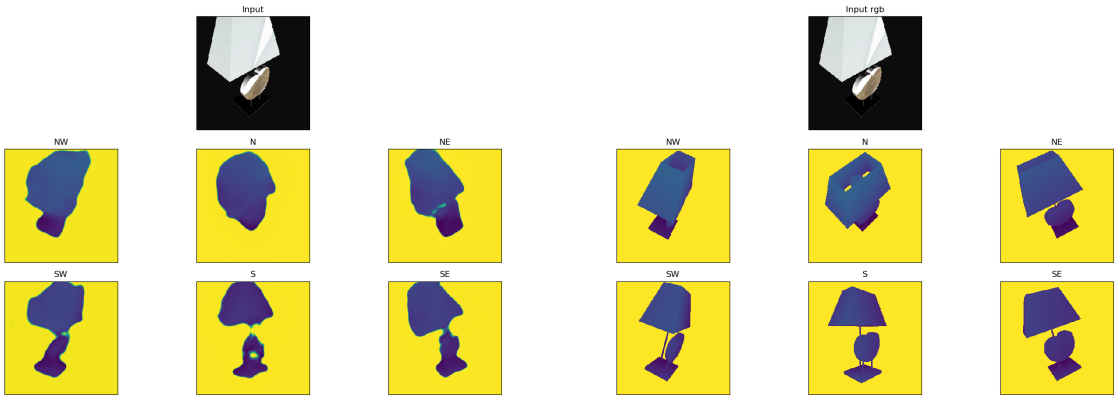

Figure 17: Another example of lamp depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

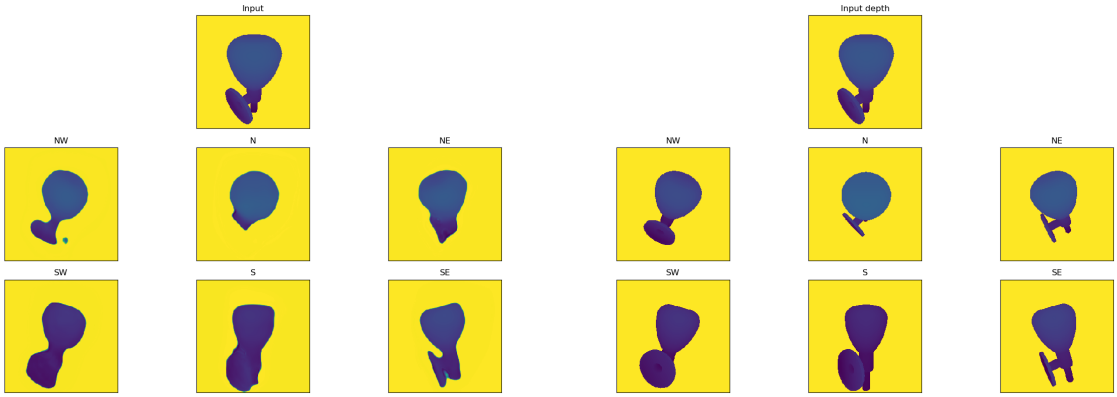

Figure 18: Example of lamp depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

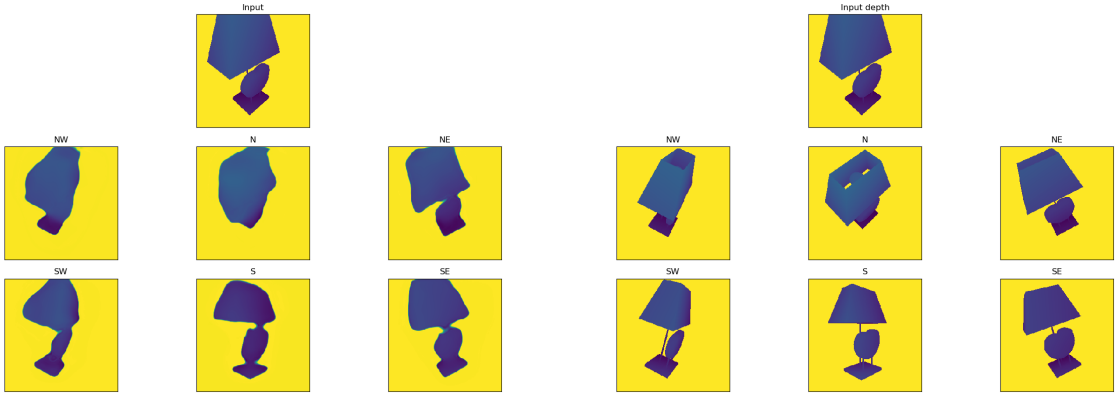

Figure 19: Another example of lamp depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

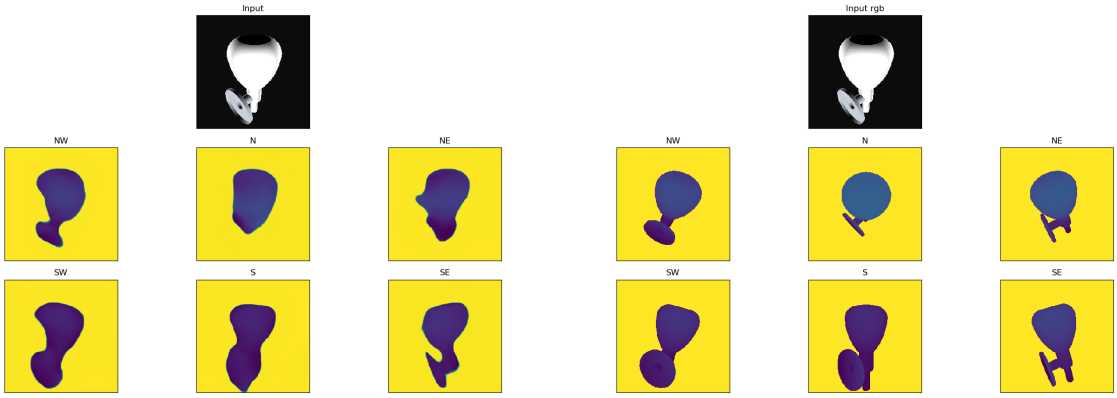

Figure 20: Example of lamp depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

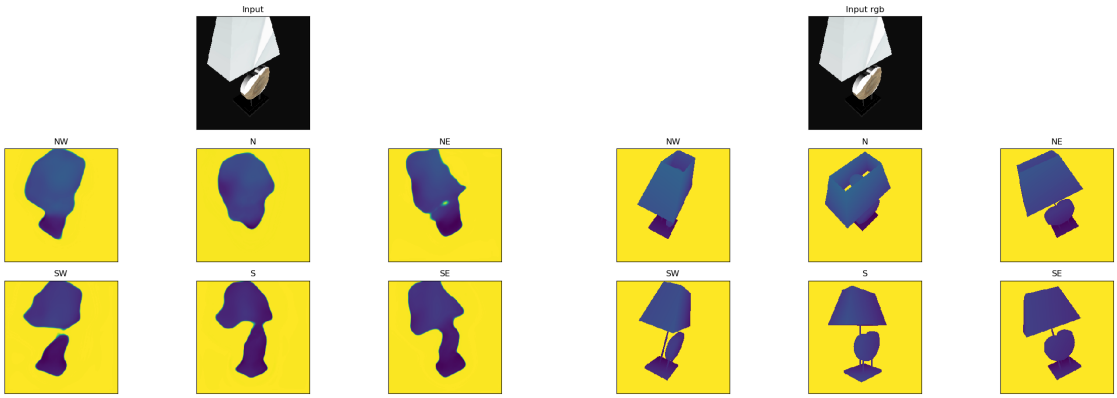

Figure 21: Another example of lamp depth map predictions via input RGB image (left) vs. the ground truth depth maps (right)

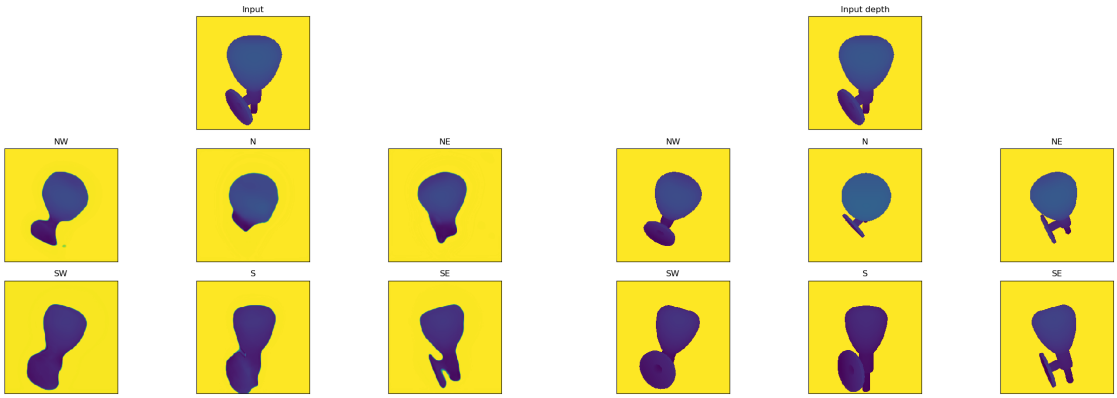

Figure 22: Example of lamp depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

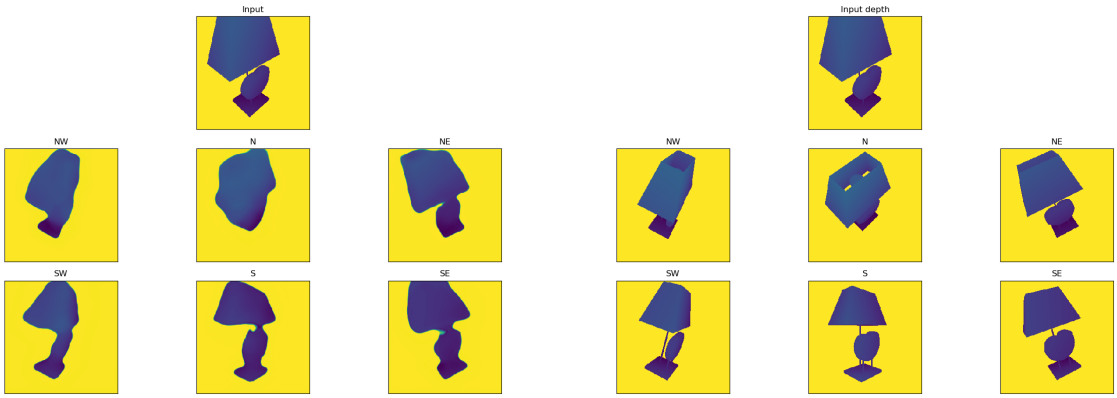

Figure 23: Another example of lamp depth map predictions via input depth map (left) vs. the ground truth depth maps (right)

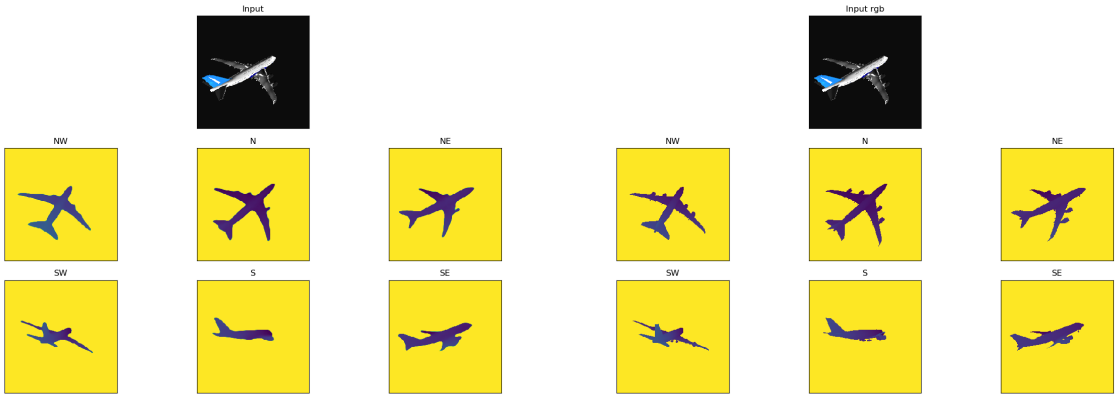

Figure 24: Airplane example of depth map predictions via input RGB image on the network trained on all classes (left) vs. the ground truth depth maps (right)

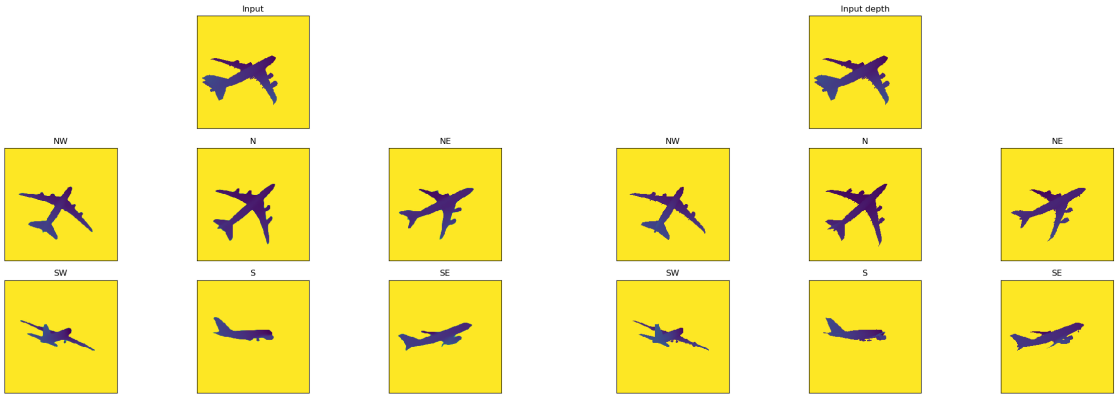

Figure 25: Airplane example of depth map predictions via input depth map on the network trained on all classes (left) vs. the ground truth depth maps (right)

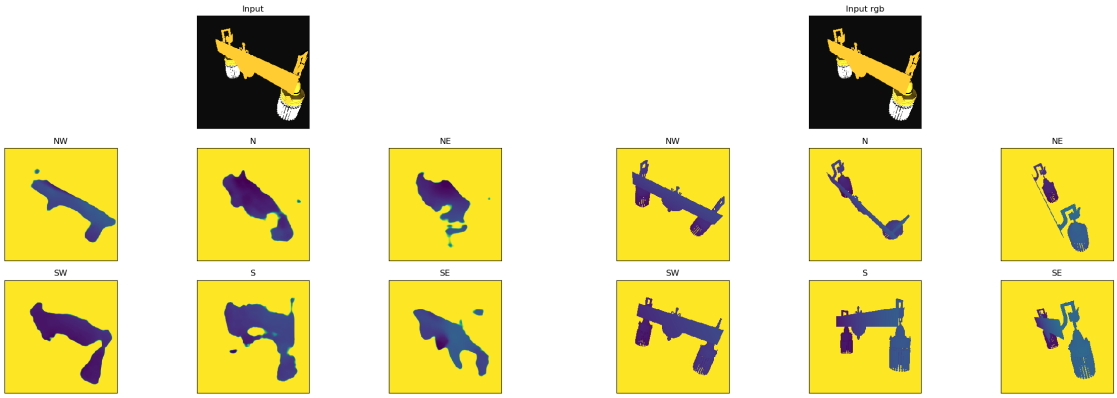

Figure 26: Lamp example of depth map predictions via input RGB image on the network trained on all classes (left) vs. the ground truth depth maps (right)

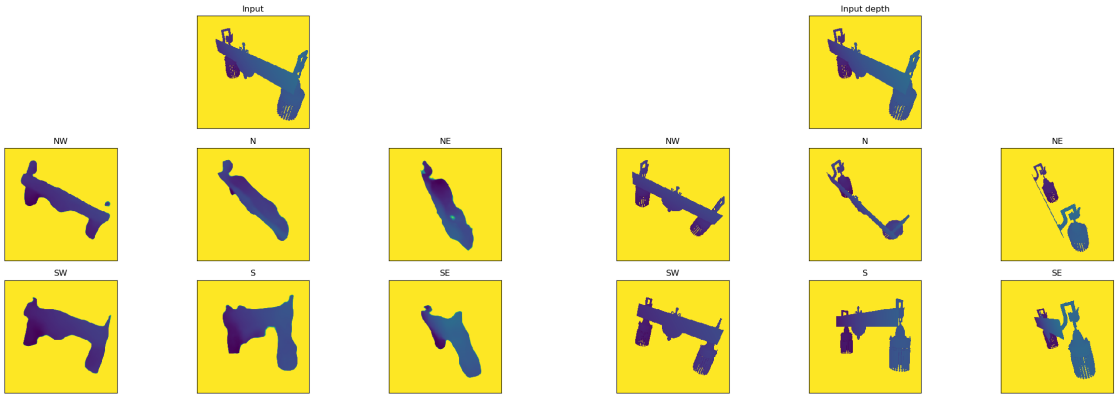

Figure 27: Lamp example of depth map predictions via input depth map on the network trained on all classes (left) vs. the ground truth depth maps (right)

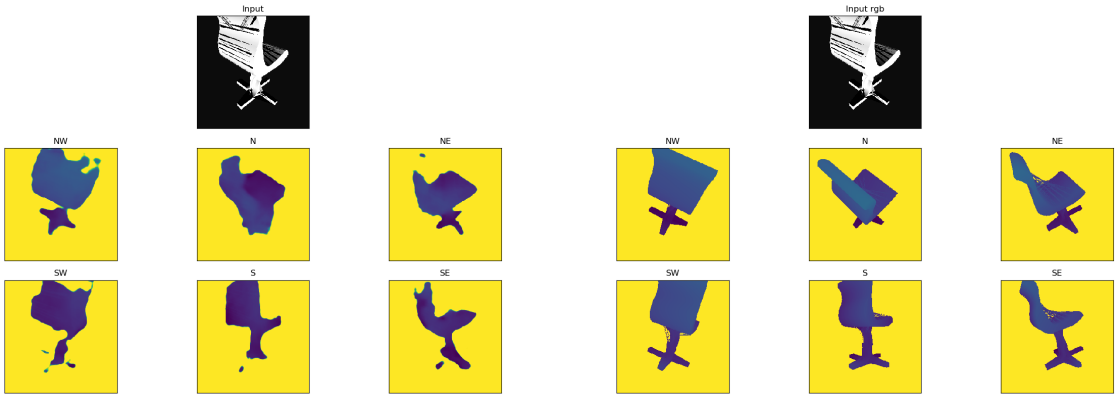

Figure 28: Chair example of depth map predictions via input RGB image on the network trained on all classes (left) vs. the ground truth depth maps (right)

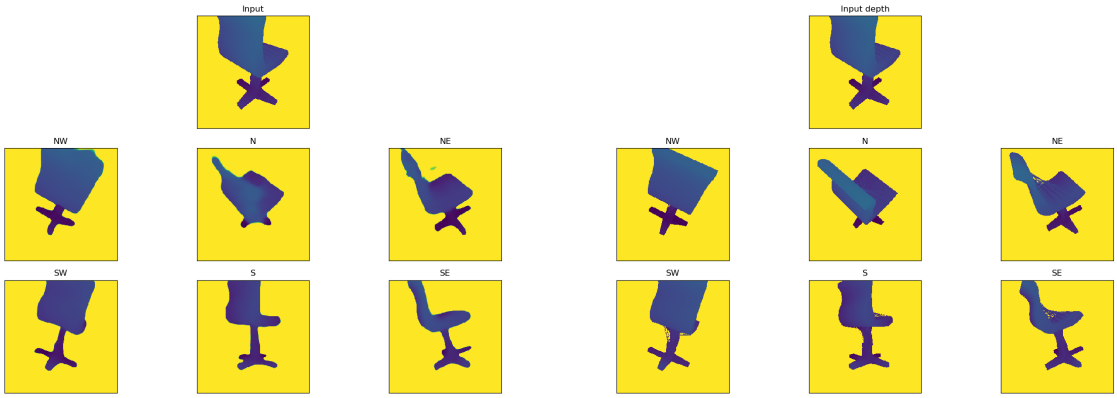

Figure 29: Chair example of depth map predictions via input depth map on the network trained on all classes (left) vs. the ground truth depth maps (right)

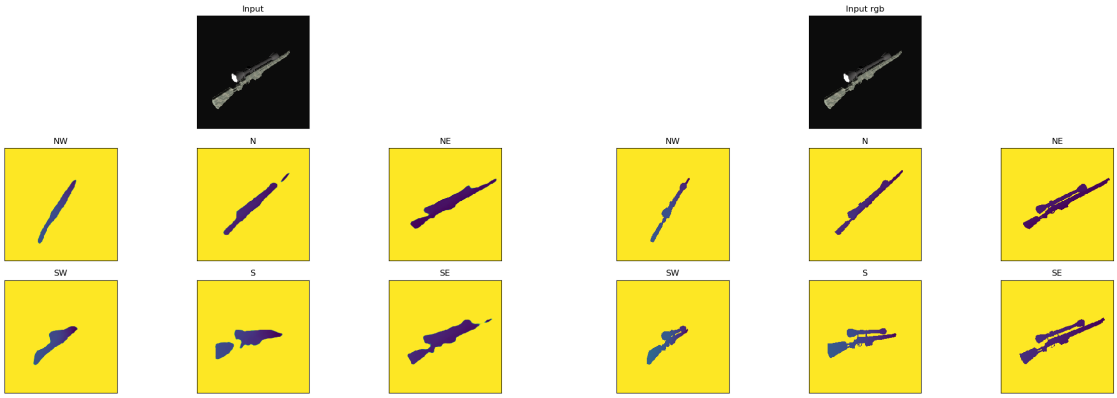

Figure 30: Rifle example of depth map predictions via input RGB image on the network trained on all classes (left) vs. the ground truth depth maps (right)

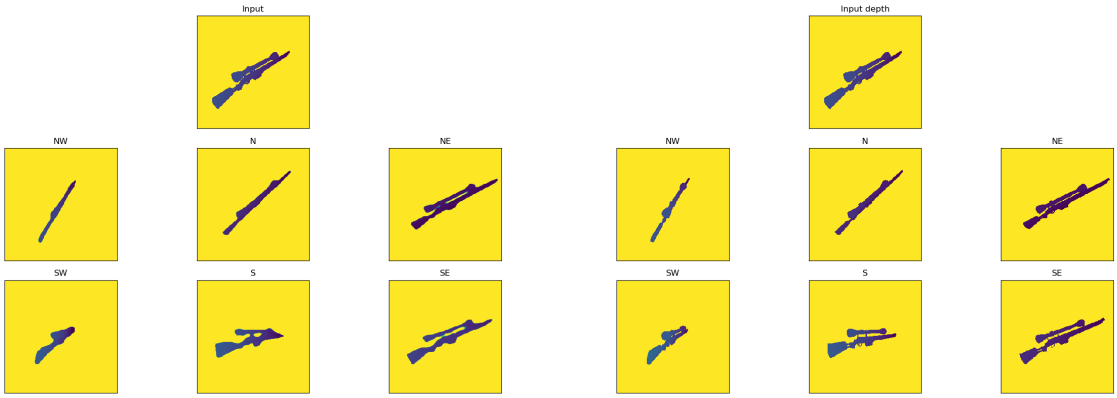

Figure 31: Rifle example of depth map predictions via input depth map on the network trained on all classes (left) vs. the ground truth depth maps (right)

Figure 32: Left column shows the input image. To the right in the same row is the ground truth point clouds constructed from the rendered depths at various angles. Below that is the reconstructed object from the same angles.

Figure 33: Same as Figure 32 but for tables

Figure 34: Carvana Results: Left column is input image, to the right in the same row is the 3D reconstruction from a number of angles.