3D Model Reconstruction from Single View Image

Parika Goel
parika.goel@tum.de

Angela Dai
angela.dai@tum.de

Matthias Nießner
niessner@tum.de

Abstract

With the recent advances in convolutional neural networks (CNNs) and the availability of large scale 3D datasets like Shapenet, there has been increased interest in learning based approaches for 3D shape reconstruction. Unlike images, there is no standard representation for 3D shapes. Until now, deep neural networks have used variety of representations for 3D shapes like voxel grids, point clouds, deformable meshes or patches and graph based structures. Voxel grids have been a popular choice to represent 3D shapes because of their underlying regular grid structure. Two most popular voxel grid representations used have been ***Occupancy Grids*** *and* ***Distance Fields****. We have used both representations to reconstruct 3D shape of an object from a single image and provided a comparison of the accuracy of generated 3D models. We have performed the evaluation on the 3D models from Shapenet dataset.*

1. Introduction

The task of inferring and understanding the 3D geometry and structure of objects from 2D images has drawn huge attention in the computer vision research community. Being able to construct the 3D model of the object has its utility in many applications such as animations, simulations, visual renderings and robotics.

Most of the SFM or SLAM based methods [15, 11] for 3D object reconstruction require images from multiple viewpoints. These techniques work by matching features between pair of images. Then the corresponding 3D points are calculated using triangulation process. Feature matching process requires pair of images to be captured from slightly different viewing angles. It has been demonstrated [21] that due to self-occlusion and local appearance changes, finding feature correspondences is extremely difficult if viewpoints are separated with large baseline.

3D reconstruction methods like Shape from Silhouette (SFS) [19] overcome this difficulty of large baseline. SFS method uses the silhouette images of the object to reconstruct 3D shape. The geometric entity produced by SFS is called Visual Hull. But the generation of Visual Hull requires images from multiple viewpoints of the same object taken by well calibrated cameras. Due to which, this approach cannot be used when it is desirable to recover 3D structure from a single view or only a handful of views. Also, it requires the cameras to be well calibrated which is not the case in many applications.

Inspired by how humans could predict the shape of an object, a different approach of using prior knowledge about the geometry and appearance of objects has gained popularity. With the advances in deep learning and the availability of large scale 3D datasets like Shapenet [5], there have been increased interest in these learning based methods for 3D reconstruction. The advantage of using priors is that since the reconstruction method does not rely on feature correspondences, it can work with fewer images and arbitrary viewpoints.

Until now, deep neural networks for 3D shape synthesis have employed a variety of representations to represent 3D shapes like voxel grids [8, 14, 30, 33, 6], point clouds [23, 24] and mesh representation [16, 25, 32]. Of these, point clouds and mesh representations have been popular because of vectorized representation of 3D data and lesser memory requirements. But these representations are very irregular and thus not in line with the convolutional neural networks (CNNs). CNNs require highly regular data format in order to perform weight sharing or kernel optimizations. Voxel Grids provide such a regular data format. Voxels in 3D space are extensions of pixels in 2D space.

Because of their simplicity and their underlying regular grid structure, voxel grids have been a popular choice to represent 3D shapes. Most of the approaches using voxels to represent 3D shapes either use Occupancy Grids or Distance Fields. We have used both the representations for the task of predicting the 3D shape of an object given its image from single view. Then we have provided an evaluation of which representation performs better for the problem at hand. We have used the encoder-decoder architecture for our network. We have experimented with two different network architectures and provided a comparison of their performance for the task. In Section 2, we have provided a summary of related work on object reconstruction using different output representations. Section 3 describes our data

generation pipeline. In Section 4, we have explained our network architectures and the loss functions we have used for two different representations. Results and evaluation of our work is shown in Section 5, and we draw conclusion from this in Section 6.

2. Related Work

Learning based 3D-shape generative approaches have used variety of representations to represent 3D shapes which can be broadly categorized into voxel based, point cloud based and mesh based representations.

Many of the 3D generative approaches represent 3D shapes in the form of binary occupancy grids. Occupancy Grids is a form of voxel representation in which each cell in the voxel grid contains a binary value: 0 for voxels inside and outside the object and 1 for voxels around the surface. Choy et al. [8] proposed to use the ability of deep neural networks to learn a mapping from 2D images to their underlying 3D shapes. They build upon the idea of leveraging the power of LSTM (Long Short Term Memory) to retain previous observations and incrementally build the output reconstruction as more observations become available. Their model takes one or more images of an object as input and outputs reconstruction in the form of occupancy grid. Wu et al. [34] proposed to represent 3D shape as probability distribution of binary variables on a 3D voxel grid. They used Convolutional Deep Belief Networks to learn these probability distributions.

Zhang et al. [33] used volumetric convolutional neural networks and generative adversarial nets to learn 3D shapes from probabilistic space. Girdhar et al. [12] proposed a TL-embedding network that reconstructs the 3D shape of an object in the form of occupancy grid. Their network contains two parts : an autoencoder that aims to reconstruct the voxel grid and a ConvNet that learns to predict an intermediate embedding. Tulsiani et al. [31] learns to predict the 3D shape of an object in the form of occupancy probabilities in a discretized 3D voxel grid given an image from a single view. To do so, they proposed to use differentiable ray consistency which allows to make incremental updates to the predicted 3D model by computing gradients from each new image. Soltani et al. [29] advocated the use of multiview depth images or their corresponding silhouettes for generative modelling of 3D shapes. Their approach uses deep generative networks to model and sample from space of 2D images, and then they use a rendering function to synthesize 3D shapes from sampled 2D images.

Other approaches [9, 26] have advocated the use of implicit field representations like Truncated Signed Distance Field (TSDF) to represent 3D shapes. TSDF is a form of voxel representation where the value in each cell represents its truncated distance to the nearest surface point. Chen et al. [6] used implicit field representation and introduced an implicit field decoder, which they called IM-NET for 3D shape generation. Implicit fields assign a value to a point in 3D space such that the surface can be extracted using an iso-surface. They trained their model to assign this value by a binary classifier. The assigned value indicates whether the point is outside the shape or not.

A major limitation with voxel based approaches is that due to cubic growth of volume with higher resolution, these approaches usually predict coarse resolution voxel grids. To improve upon this, Tatarchenko et al. [30] proposed using an octree representation instead of regular voxel grids. On similar lines, Häne et al. [14] proposed a hierarchical surface prediction approach. Their idea is that since we are interested in the shape of the object, voxels around the surface can be predicted with higher resolution. The voxels inside and outside the object can be predicted with coarse resolution.

Qi et al. [23, 24] pioneered the use of point clouds to represent 3D shapes for discriminative tasks. Fan at al. [10] represented 3D shape in the form of point cloud for the task of shape reconstruction. They proposed a novel point-set generator architecture and loss function to be able to predict directly unordered point sets to represent 3D shape. They introduced two new distance metrics : Chamfer Distance and Earth Mover's distance as loss functions for point sets. But their approach requires images with cleaner backgrounds and fixed viewpoints. To be able to use images with random backgrounds, Xia et al. [35] proposed to use prior knowledge about 3D shape in the network. Given an image, their approach first retrieves a nearest shape model from a database of 3D models. The image and retrieved 3D shape are then fed into the network to generate a 3D point cloud. Choi et al. [7] proposed an approach to first extract shape information from the input image. They then embed point-specific and global shape information into the point cloud. Then they used the embedded point cloud features to deform a randomly generated point cloud into the final representation. Lin et al. [20] used images from multiple viewpoints to predict 3D structures in the form of dense point clouds. These approaches require post processing steps [3, 4, 17, 18] to convert resultant point clouds into surface meshes.

Recently there has been an increased interest in representing 3D shapes with deformable meshes or patches [13, 28, 37] that naturally infers a surface representation of the shape. Kanazawa et al. [16] present a framework that learns to predict the 3D shape, camera and texture of an object given its image from a novel view. They train their predictor on an annotated image collection of the object category. They represent the 3D shape as a deformable 3D mesh model of an object category. It is parameterized by a learned mean shape and per-instance predicted deformation. Ranjan et al. [25] have used mesh representation to

generate 3D faces. They have introduced a Convolutional Mesh Autoencoder that learns a non-linear representation of a face. Wang et al. [32] learns to predict the 3D shape in the form of triangle mesh from single view RGB image using graph based convolutional neural networks. They progressively deform an ellipsoid in a coarse-to-fine manner to produce the correct geometry. They use the perceptual features extracted from the input image for the deformation process.

3. Data Generation

Our model is trained on table dataset from ShapeNetCore [5]. ShapeNetCore is a large dataset containing 51,300 unique 3D models from 55 common object categories. For our work, we have used the subset containing tables. The encoder-decoder model requires several types of data for training. For ground truth, we generate voxelizations of 3D models. The voxelization process is explained in Section 3.1. For input to the encoder, we generate a visual cone from the 2D image rendered from 3D CAD model in its canonical orientation (0°). Section 3.2 explains the process to generate this visual cone which serves as the input 3D model fed into the encoder.

3.1. Voxelization

For ground truth data, we have used two voxel representations : Occupancy Grid and Truncated Distance Field. To generate the unsigned distance fields of Shapenet CAD models, we have used the level-set generation toolkit by Batty [1]. To compute truncated distance field, we have used truncation distance of 3.0. The occupancy grids are computed from the distance field grids. A voxel is set to be occupied if its distance value is less than 1.0. We have used the resolution of 32 x 32 x 32 for voxel grids. Figure 1 shows an example of a 3D CAD model and the generated voxelized occupancy grid.

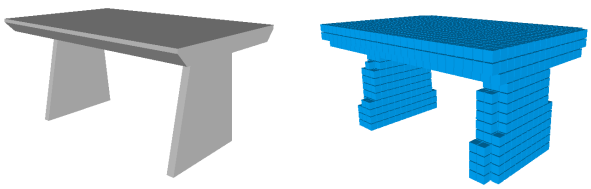

Figure 1: (Left) 3D CAD model from Shapenet dataset. (Right) Occupancy Grid

3.2. Encoder Input

Inspired by the concept of Visual Hull [2], we used a similar approach to generate a 3D model from a 2D image

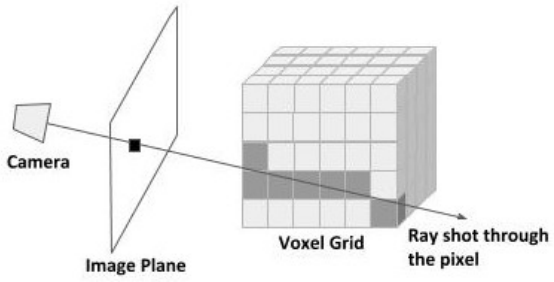

Figure 2: Illustration of approach used to generate visual cone from rendered images. The generated cones are the 3D models that are fed as input into the encoder

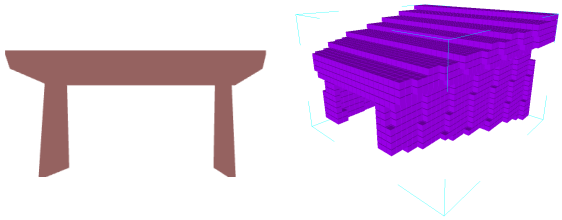

Figure 3: (Left) Image rendered using pyrender [22] library. (Right) Input to encoder generated using approach explained in Figure 2

which we feed as the input into our encoder. Visual Hull is a 3D representation of an object generated by Shape-From-Silhouette [19] technique. The basic principle behind Visual Hull is to create a 3D representation of an object from its silhouettes from different viewpoints. Each of the silhouettes when back projected into 3D volume using camera parameters form a cone, called visual cone. The intersection of all these cones gives an approximate of object's shape. Since our approach uses image of the object from only one viewpoint, we take the corresponding visual cone as the input 3D model. Our intuition behind using visual cone from a single viewpoint is that the actual 3D object lies inside this volume, so it can serve as a good initialization of the 3D model for our network. Below we have described the technical implementation of how we generate the visual cone.

The idea is that we have a camera that is looking at the rendered image. Imagine our voxel grid is located behind this image. What we see on the image is the projection of the voxel grid. Imagine we send out rays originating from the camera through each pixel on the image. Each pixel can be accessed by one ray. If we continue following the ray behind the image, it will pass through some voxels in the voxel grid. We set all these voxels that the ray passes through to occupied if the corresponding pixel in the image belongs

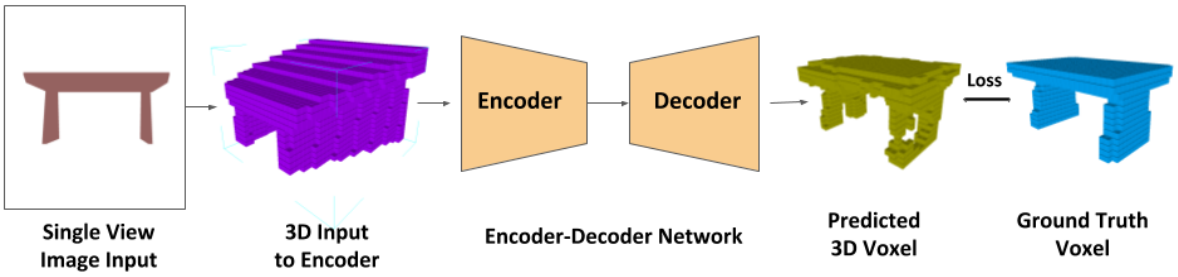

Figure 4: System Pipeline

to the object. The cells are set to not occupied if the pixel belongs to the background. Since we are only interested in the occupancy information, we do not use the color information from the images. Figure 2 shows the diagrammatic illustration of the explained method.

We have used the pyrender [22] python library to render images of Shapenet CAD models. The image rendered from 3D model in its canonical orientation (0°) is used to generate the input 3D model. Figure 3 shows an example of rendered image and the corresponding visual cone.

4. Method Overview

Figure 4 shows an overview of our method. We use the image rendered from canonical orientation (0°) of 3D model as input. Then we generate input 3D model from this image using approach explained in Section 3.2. This 3D model is fed as input into our encoder-decoder network. The input voxel grid is processed by the encoder to a feature representation. This is then fed to the decoder, which returns the 3D model predictions. These predictions are then compared to our ground truth 3D models and loss function is calculated accordingly.

We have experimented with two representations for ground truth data : Occupancy Grid and Truncated Distance Field. Further, we have compared their performance which is shown in Section 5. The representation of 3D input to the encoder remains same in both the variants which is occupancy representation.

4.1. Network Architecture

In this section, we discuss the architecture of our encoder-decoder network which we call Net3D shown in Figure 5. The 3D convolutional encoder has 6 convolution layers (8, 16, 32, 64, 128, 256 channels with kernel size of 3x3; first convolution layer has stride 2, padding 1; other convolution layers have stride 1). The 3D convolutional decoder has 6 transposed convolution layers (128, 64, 32, 16, 8, 1 channels with kernel size of 3x3; all layers have a stride of 1 except last layer which has a stride of 2, padding 1 and

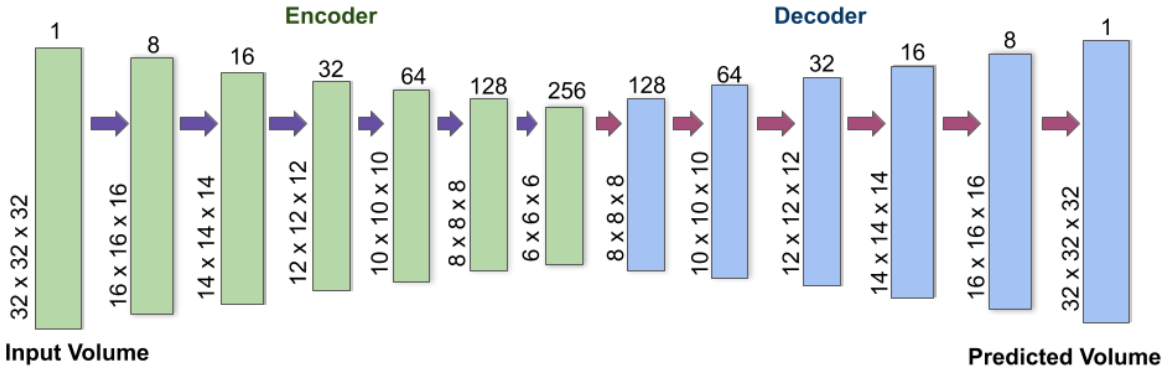

Figure 5: Net3D Architecture: Each box represents a feature map. The number of channels is written on top of the box. The width x height x depth dimensions are written at the lower-left edge of each box. The input volume is encoded using a set of 6 convolution layers followed by instance norm and relu (denoted by purple arrows). The decoder consists of a set of 6 transpose-convolution layers followed by instance norm and relu (denoted by red arrows) except the last layer.

output padding 1). All convolution layers are followed by instance normalization and relu except the last layer which returns the predicted 3D model.

4.2. Loss Function

When using Occupancy Grid representation, we train our model to be able to classify each voxel in the grid as being occupied or not. We apply Sigmoid activation on the output of the network which are interpreted as the probability of a voxel being occupied or not. We use binary cross entropy between the probability predictions and the ground truth occupancy grid as the loss function.

When using Truncated Distance Field (TDF) representation, the model is trained to predict the distance value at each voxel. We formulate the loss for the generated 3D geometry on the predicted TDF values using an $\ell 1$ loss with the target truncated distance field values.

5. Evaluation

We have a dataset of 8433 table models, with train/val/test split of 5400/1340/1693 samples. Throughout the evaluation, we show the visual results as occupancy grid representation. We first present the metrics we have used to compare the quality of our reconstructions. Then, we show the results of the following experiments :

- Comparison of Occupancy Grid and Distance Field representations
- Comparison of two different architectures for our Encoder-Decoder Network

5.1. Metrics

Intersection over Union (IoU)

$$IoU = \frac{p \cap t}{p \cup t} = \frac{\sum_{0 \leq i < n} (p_i \geq 0.5).t_i}{\sum_{0 \leq i < n} [(p_i \geq 0.5) + t_i]} \quad (1)$$

where n is the length of the vectors. p_i is the occupancy probability of predicted voxel i which can lie between 0 and 1. t_i is the value of target voxel i which can be either 0 or 1. IoU is a metric that allows us to evaluate how similar our predicted bounding box is to the ground truth bounding box. The idea is to compare the area where the boxes intersect to their total combined area. It is a good metric for geometry since it only considers the surface voxels in the prediction and ground truth, and not the empty area. In case of Occupancy Grid representation, we first apply the Sigmoid activation on the network prediction to get the occupancy probabilities of each voxel. We consider the voxels with a probability ≥ 0.5 as occupied. In case of TDF representation, we first convert the predicted and target TDF grids into occupancy grids. We consider the voxels within a truncation distance of 1.0 to be occupied. Then we compute the IoU metric over it.

L1 Error Metric

$$L_1 = \frac{1}{n} \sum_{0 \leq i < n} |p_i - t_i| \quad (2)$$

L1 error is the mean of the absolute difference between the prediction vector and the target vector. We used the L1 error metric between the predicted and target truncated distance field values to measure the completion quality. In case of occupancy representation, we first transform the predicted and target occupancy grid into a distance transform and then compute the L1 error between them.

5.2. Comparison: Occupancy Grid and Distance Field

In this section, we discuss about the accuracy of the predictions made by our network Net3D. For training, we have used the Adam optimizer with a learning rate of 0.001 and weight decay of 1e-5. The learning rate is decayed by half after every 10 epochs. We train the network for 50 epochs with the batch size of 8.

	IoU ↑	L1 ↓
occ	0.592	0.118
tdf	0.583	0.114
tdf(log)	**0.597**	**0.111**

Table 1: Quantitative shape prediction results on different Ground Truth representations

Table 1 shows the quantitative evaluation of the shape predictions by Net3D. In case of TDF representation, we have also experimented with using log transform as a continuous weighting scheme. While training, we apply log transform on the predicted and target tdf values before calculating loss. The reason behind using this is that log transform will give more weightage to the voxels near the surface. tdf(log) in the last row shows the prediction accuracies when this weighting scheme is used. We observed that occupancy performs a little better than distance field in terms of geometric accuracy of predicted shape. With log transform predictions made by distance field representation have almost similar IoU as occupancy. We also observed that occupancy representation converges a lot faster than distance field. It took our model 26 epochs to converge on occupancy and 43 epochs to converge on distance field with log weighting.

Figure 6 shows some example predictions made by our model. We observed that though the predictions made by tdf representation looks better, the number of false positives are higher than other two versions. By false positives, we mean the voxels which are predicted as occupied by model but are unoccupied in ground truth. This can be observed if we focus on the leg of the table in the first row and the

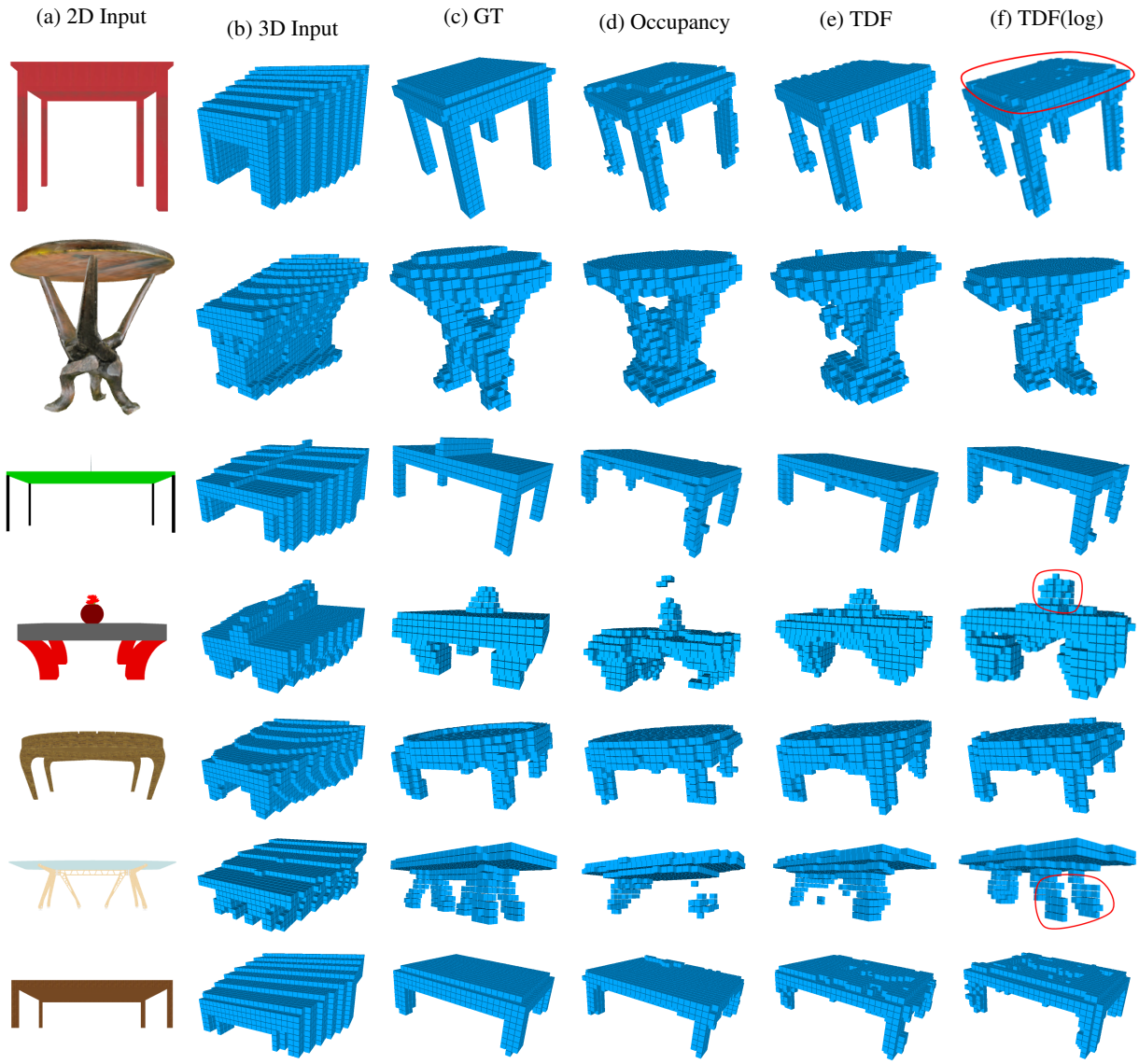

Figure 6: Example shape predictions for different ground truth representations by our network Net3D. Columns from left to right: (a) Input 2D Image (b) Volume Cone generated from input image by approach explained in Section 3.2 which is fed as input to the encoder (c) Ground Truth 3D Volume (d) Network predictions when GT is in the form of Occupancy Grid (e) Network predictions when GT is in the form of Truncated Distance Field (TDF) (f) Network predictions when GT is in the form of TDF and log weighting is used.

fourth row in the figure. This explains the lower IoU value for tdf representation.

Occupancy representation performs better than tdf. If we again observe the legs of the table in first row, occupancy version is closer to ground truth than tdf since the false positives are lesser. tdf with log weighting performs even better than occupancy. Again observing the first row in the figure, we can see the corner part of the top layer of the table is missing in occupancy prediction. In the fourth row of the figure, we can see that tdf(log) is able to predict the flower pot on the table better than occupancy. In some of the cases, occupancy gives better geometric predictions than other two version as can be seen in seventh row of the figure.

Overall, we can say that occupancy representation performs better than other two versions since it gives comparable performance in terms of geometric accuracy of the predictions and converges much faster than other two. The fact that our input is in the form of occupancy grid could also have contributed to this behaviour.

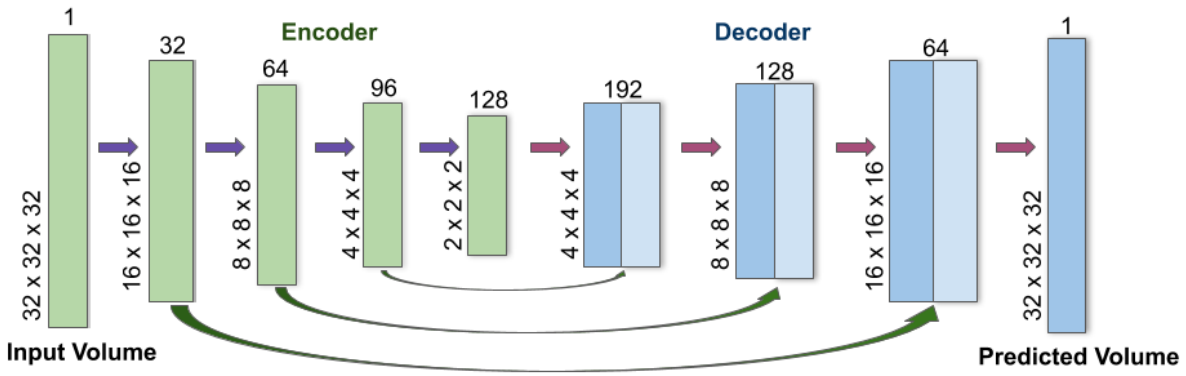

Figure 7: UNet3D Architecture: Each box represents a feature map. The number of channels is written on top of the box. The width x height x depth dimensions are written at the lower-left edge of each box. The input volume is encoded using a set of 4 convolution layers followed by instance norm and relu (denoted by purple arrows), each set reducing the spatial dimensions by a factor of 2. The decoder consists of a set of 4 transpose-convolution layers followed by instance norm and relu (denoted by red arrows) except the last layer. The green arrows at the bottom represents the concatenation operation which combines feature maps of same spatial resolution.

5.3. Comparison: Net3D and UNet3D

We experimented with a different architecture for our encoder-decoder network which we have called as UNet3D and compared its performance with the previous architecture Net3D. Figure 7 shows the architecture of UNet3D. The 3D convolutional encoder has 4 convolution layers (32, 64, 96, 128 channels with kernel size of 3x3, stride 2 and padding 1). The 3D convolutional decoder has 4 transposed convolution layers (96, 64, 32, 1 channels with kernel size of 3x3, stride 2, padding 1 and output padding 1). All convolution layers are followed by instance norm and relu except the last layer which returns the predicted 3D model. Similar to U-Net [27], we use skip connection between encoder and decoder part of our network. Feature maps of same spatial resolution are taken from encoder, concatenated with output of transposed convolution layer in decoder and fed as input into next decoder layer.

	occ		tdf(log)	
	Net3D	UNet3D	Net3D	UNet3D
Mean IoU ↑	0.592	**0.596**	**0.597**	0.584
L1 Error ↓	0.118	**0.118**	0.111	**0.110**

Table 2: Quantitative evaluation of the predictions made by Net3D and UNet3D networks

We have used the occupancy grid and distance field representation of the ground truth to compare both the architectures. In case of distance field, we have applied log transform at the time of training before calculating loss. For training the UNet3D network, we have used the Adam optimizer with a learning rate of 0.001 and weight decay of 1e-5. The learning rate is decayed by 0.1 after every 10 epochs. We trained the network for 50 epochs with a batch size of 8.

Table 2 shows the quantitative comparison of the shape predictions made by both the networks. Our Net3D network has approximately 2.3 million trainable parameters and UNet3D network has approximately 1.3 million trainable parameters. We observed that UNet3D with less trainable parameters gives comparable performance to Net3D. Also, it took our UNet3D model 12 epochs to converge which is faster compared to Net3D model which took 26 epochs to converge.

Figure 8 shows some example predictions made by Net3D and UNet3D. We observed that in some cases UNet3D gave better predictions than Net3D. This can be observed in the first, second and fourth row examples (highlighted by red circle).

6. Conclusion

We have presented the comparison of Occupancy Grid and Distance Field representation for the task of 3D shape generation. Our experiments show that distance field representation along with log weighting gives better performance in terms of geometric accuracy. But model trained on occupancy representation takes less time to converge. We have also presented the comparison between two architectures for our Encoder-Decoder network : Net3D and UNet3D. Our UNet3D model uses skip connections on similar lines of [27]. Our experiments show that UNet3D with less trainable parameters gives comparable performance to Net3D. Also, UNet3D model takes fewer epochs to converge.

As future work, we would like to improve upon the accu-

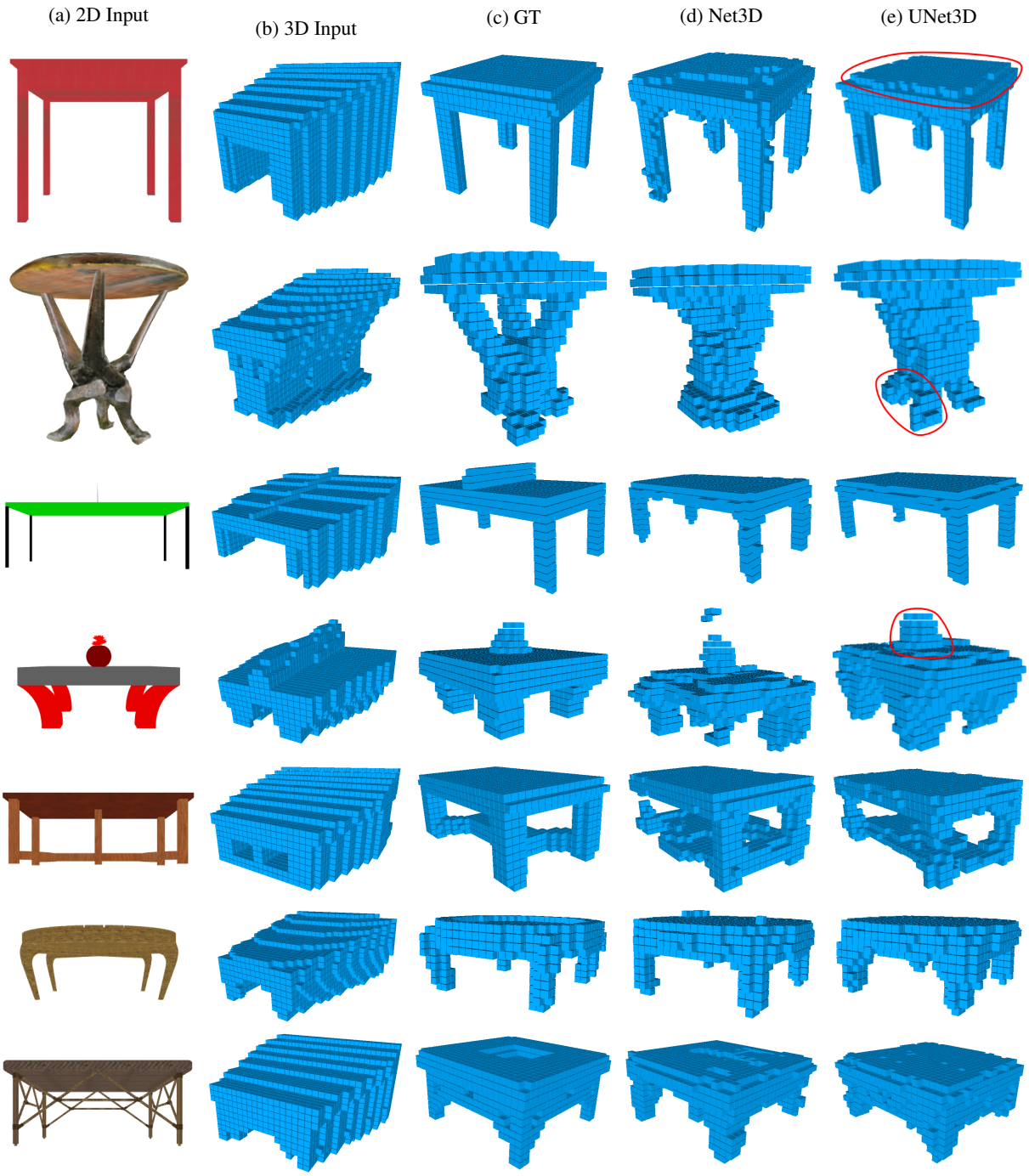

Figure 8: Example shape predictions by Net3D and UNet3D on Occupancy (row 1-4) and DF (row 5-7) representations. Columns from left to right: (a) Input 2D Image (b) Volume Cone generated from input image by approach explained in Section 3.2 which is fed as input to the encoder (c) Ground Truth 3D Volume (d) Network predictions by Net3D (e) Network predictions by UNet3D

racy of our 3D predictions. One idea would be to project the predicted 3D model to multiple viewpoints and then compare the generated silhouettes to ground truth silhouettes from same viewpoints. This can be on similar lines as projection layer in [36]. We would also like to train on multiple object categories. Lastly, it would be interesting to explore

generating 3D models directly from the images without the visual cone generation step in between and improve upon that.

References

[1] Christopher Batty. SDFGen. `https://github.com/christopherbatty/SDFGen`.

[2] Bruce G. Baumgart. Geometric modeling for computer vision. 1974.

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349359, Oct. 1999.

[4] F. Calakli and Gabriel Taubin. Ssd: Smooth signed distance surface reconstruction. *Computer Graphics Forum*, 30:1993 – 2002, 11 2011.

[5] 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.

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *CoRR*, abs/1812.02822, 2018.

[7] S. Choi, A. Nguyen, J. Kim, S. Ahn, and S. Lee. Point cloud deformation for single image 3d reconstruction. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2379–2383, 2019.

[8] Christopher B. 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.

[9] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *CoRR*, abs/1612.00101, 2016.

[10] 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.

[11] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.*, 43(1):5581, Jan. 2015.

[12] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. *CoRR*, abs/1603.08637, 2016.

[13] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR*, abs/1802.05384, 2018.

[14] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *CoRR*, abs/1704.00710, 2017.

[15] Peters Gabriele Hming, Klaus. The structure-from-motion reconstruction pipeline a survey with focus on short image sequences. *Kybernetika*, 46(5):926–937, 2010.

[16] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. *CoRR*, abs/1803.07549, 2018.

[17] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP 06, page 6170, Goslar, DEU, 2006. Eurographics Association.

[18] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), July 2013.

[19] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[20] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *CoRR*, abs/1706.07036, 2017.

[21] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91110, Nov. 2004.

[22] Matthew Matl. Pyrender. `https://github.com/mmatl/pyrender`.

[23] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[25] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3d faces using convolutional mesh autoencoders. *CoRR*, abs/1807.10267, 2018.

[26] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. *CoRR*, abs/1704.01047, 2017.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[28] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. *CoRR*, abs/1703.04079, 2017.

[29] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D. Kulkarni, and Joshua B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2511–2519, 2017.

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

[31] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. *CoRR*, abs/1704.06254, 2017.

[32] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. *CoRR*, abs/1804.01654, 2018.

[33] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *CoRR*, abs/1610.07584, 2016.

[34] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.

[35] Yan Xia, Yang Zhang, Dingfu Zhou, Xinyu Huang, Cheng Wang, and Ruigang Yang. Realpoint3d: Point cloud generation from a single image with complex background. *CoRR*, abs/1809.02743, 2018.

[36] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *CoRR*, abs/1612.00814, 2016.

[37] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *CoRR*, abs/1712.07262, 2017.