Local and Global Texture Fields

Mesut Ceylan
University of Zürich
ceylanm@kof.ethz.ch

Berk Kaya
ETH Zürich
bekaya@student.ethz.ch

Erik Sandström
ETH Zürich
esandstroem@student.ethz.ch

Abstract

The problem of 3D texture reconstruction is studied using Texture Fields, a continuous 3D function, parameterized by a neural network. Taking inspiration from the single image 3D reconstruction literature, we extend Texture Fields and present a way to extract additional local appearance features from the 2D input image. Compared to our work, only the extraction of global appearance features was studied previously. Specifically in our work, each 3D point is projected onto the 2D input image and local features are extracted from the feature maps. Local feature extraction is crucial because it allows the capture of fine appearance details, which is not possible to reconstruct with only global appearance features. Our results show that local appearance features are an important part in achieving high quality 3D texture reconstructions.

1. Introduction

Recent progress in deep learning has enabled accurate 3D object reconstructions from single-view images [25, 6, 26, 21, 27, 23]. Although deep learning models can estimate the geometry of the object, they are not successful in adding texture to these reconstructions. In this work, our aim is to improve upon previous texture reconstruction work.

The usage of implicit fields to model 3D objects has gained popularity due to its ability to represent complex surfaces [11, 17, 2, 13] at arbitrary resolution. Recently, Wang *et al*. proposed the Deep Implicit Surface Network (DISN) [28] and obtained state-of-the-art performance in single-view 3D reconstruction. Moreover, Texture Fields [15] has successfully combined the texture of an RGB image with a shape, allowing the recovery of textured 3D surfaces. Texture Fields, however, assumes a viewpoint invariant global feature representation of each RGB input image and thus local and high frequency details are partly missing. Local and global feature extraction using a viewpoint variant feature representation is proposed by Wang *et al*. [28] in order to reconstruct both global and local geometries better, using an implicit field that regresses the signed distance function (SDF) to the surface.

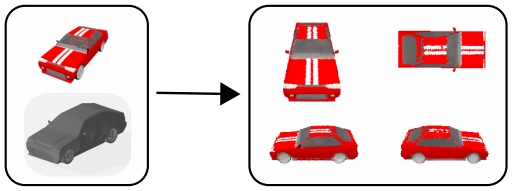

Figure 1: **Local and Global Texture Fields:** Using a 3D model and a corresponding condition image, our method learns a mapping to represent the texture of the object and renders novel views.

Inspired by DISN, we aim to combine local and global appearance features with the texture model proposed by Oechsle *et al*. [15] to achieve more accurate texture reconstructions. Our idea is based on the intuition that the features from the visible regions of the input image should be correlated with their corresponding location on the 3D shape. Within this scope, the contributions of our work include 1) an image encoder that extracts local and global appearance features. We believe that we are the first at extracting local appearance features for texture reconstruction. 2) An occlusion module that makes sure that the local features correspond to the queried 3D points. In comparison, DISN does not use an occlusion module. 3) Qualitatively and quantitatively improved results on the original Texture Fields model.

The outline of the report is as follows. Section 2 presents an overview of the related work, specifically single-view shape and texture reconstruction. Section 3 covers the methodology, while introducing a detailed review of Texture Fields [15]. Section 4 describes the experimental setup and results. The results are discussed and concluded in section 5 including the work distribution.

2. Related Work

2.1. Single-View 3D Reconstruction

As inferring the 3D structure from a single view is a highly ill-posed problem, the scientific community did not experience significant progress in this task until the advent

of deep neural networks. Traditionally, the problem has been approached via shape-from-shading [5, 9, 30], which aims to compute the depth from a single image, and therefore it can only reason about the visible parts of a surface.

With the introduction of deep learning, researchers have tried to find a compelling 3D representation to work with. Traditional representations such as voxels, meshes, and point clouds are problematic when working with neural networks. For instance, the memory footprint of voxel-based methods cubically increase with the resolution, thus limiting the practical usage. While it is possible to reduce the memory footprint by using e.g. octrees [19] or voxel hashing [14], these approaches are still limited with relatively small 256^3 voxel grids. Point clouds are missing the connectivity structure between the vertices and are therefore in need of further post-processing to produce a dense 3D model. Moreover, mesh representations are typically based on deforming a template mesh and thus cannot handle arbitrary topologies.

To circumvent these problems, continuous implicit surface representations were introduced [12, 18, 29]. Mescheder *et al*. [12] proposed to learn an implicit representation by predicting the probability of each 3D point in the field being occupied or not, i.e., being inside or outside of a 3D model. Park *et al*. [18] introduced DeepSDF for shape completion using an auto-decoder architecture. Their network, however, requires additional optimization during inference which reduces the applicability and efficiency of the approach. Finally, Xu *et al*. [29] introduced DISN which not only predicts the sign (i.e. being inside or outside) of sampled points, but also the distance in the form of a signed distance function (SDF), which is continuous. More importantly, they introduce local and global feature extraction from the input image, which helps to reconstruct both global structure and local details.

2.2. Textured 3D Reconstruction

While great progress has been made when it comes to the task of single-view 3D reconstruction, the reconstructions are purely geometric and usually do not provide texturing due to lack of attention given to the task of texturing. However, the first work that employs a continuous implicit representation for texture is PIFu. PIFu [20] takes as input an RGB image and reconstructs both the 3D model and adds texture to the model in a single framework. This is achieved by using an implicit occupancy field which predicts whether a 3D point is inside or outside of the mesh. The inputs are local image features from the image obtained by projecting the 3D point into the image and the depth of the 3D point in the camera coordinate frame. Texture is predicted per 3D point by sampling 3D points on the reconstructed mesh in a similar fashion by using local image features as well as the depth of the 3D point from the camera coordinate system. Texture Fields [15] is concurrent work to PIFu and assumes, instead, a setting where the 3D shape is known and the appearance is encoded by an input image. Since Texture Fields relies on global appearance features, high frequency details are not captured and only diffuse material properties can be captured. In concurrent work [16] to ours, Oechsle *et al*. extends Texture Fields to accommodate for the need to render more realistic images. This is achieved by adding light features as input to the texture field network. Local features are, however, not extracted from the input image.

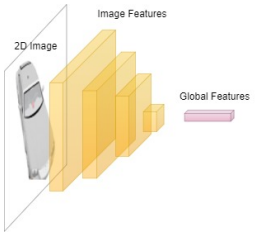

Figure 2: The image encoder of Texture Fields [15].

3. Method

This section is outlined as follows: We start by reviewing the work of Oechsle *et al*. [15] in section 3.1, which forms the basis of our method. Then, in section 3.2, we describe in detail how our framework is designed.

3.1. Texture Fields

Texture Fields [15] describes a way of adding texture to 3D shapes. Here, we specifically aim to describe the conditional setting of Texture Fields where the objective is to add texture to the 3D shape, when texture has only been observed from one view in the form of a 2D image. The 2D image is passed through an image encoder, Fig.2, (based on ResNet-18 [7]) which extracts a fixed length 512-dimensional viewpoint-invariant global appearance vector denoted $z \in Z$. The complete model of Oechsle *et al*. can be visualized by replacing the encircled red box in Fig.3 with the architecture in Fig.2. Apart from the appearance vector $z \in Z$, the 3D mesh is additionally sampled to form a point cloud which is processed with a Point Net based encoder, adopting the architecture found in Occupancy Networks [11]. This forms another 512-dimensional shape encoding $s \in S$. Along with the appearance and shape encodings z an s, a 3D point $p \in \mathbb{R}^3$ from the 3D shape is finally extracted. This triplet forms the input to the Texture Field t which predicts the RGB color at the 3D point (1), regressing this value by minimizing the L1 loss between the predicted color $\hat{c}_i$ and the ground truth color c_i.

$$t : \mathbb{R}^3 \times S \times Z \rightarrow \mathbb{R}^3 \tag{1}$$

It would be possible to directly regress the RGB color for each 3D point on the mesh, however, mesh-based appearance representations require additional UV-mappings

for representing texture. Thus, Texture Fields performs supervision in the 2D image space. To this end, a random pose $T_r = [R_r\, t_r]$ from camera to world space is requested and a depth map is rendered from this pose. Next, all foreground pixels u_i (pixels belonging to the object) are sampled and unprojected using (2) into 3D space forming the collection of 3D points p_i.

$$p_i = D_r(u_i) R_r K^{-1} [u_i\ 1]^T + t_r \tag{2}$$

The texture field regresses the RGB color in image space based on the rendered color image from the requested pose and the loss is formed using all foreground pixels N_b from the requested pose over a mini batch B of requested poses as shown in (3).

$$\mathcal{L} = \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{N_b} ||t(p_{bi}, s_b, z_b) - c_{bi}||_1 \tag{3}$$

For further details, we refer the reader to the paper [15].

3.2. Framework

The framework that we chose to design basically follows the same implementation as Texture Fields [15] apart from how the image features $z \in Z$ are constructed. This section, therefore, aims to give a detailed presentation of how the image features are extracted.

3.2.1 Image Encoder

The image encoder is illustrated in Fig.4 and is essentially a ResNet-18 [7] network, but with the modification that the final fully connected layer of ResNet-18 is replaced to give an output of size 256 instead of 512. The final fully connected layer is trained from scratch, contrary to the other layers, which are pre-trained on the ImageNet dataset [4]. For local feature extraction, the feature maps are extracted from the convolutional layers in order $\{1, 5, 9, 13\}$. This follows a similar feature extraction strategy to [28], but instead of VGG16, we used ResNet-18. In general, shallow layers are more localized but they contain less complex information compared to deeper layers. On the other hand, deeper layers are less localized, since the receptive field grows with the depth. There is, therefore, a trade-off to be made when it comes to what local features to extract. Only extracting features at shallow layers means that the information will be well localized, but may not be utilized by the texture fields network since the features only carry low-level information (e.g. edges) which may be less useful when putting texture on a complex object. A combination of well localized shallow features and less localized deeper features could therefore be a good compromise. In order to find what local features to extract at each layer, the layers are first upsampled to the input image size using bilinear interpolation. At the scale of the input image, the features at the spatial location v_i (see Fig.3) are extracted along the full channel depth. Then, the features are concatenated and fed to a fully connected layer that outputs a 256-dimensional local feature vector. Finally, the local and global features are fed to the occlusion module for further processing.

3.2.2 Occlusion Module

The task of the occlusion module is to ensure that the 3D point p_i, extracted from the depth map D_r, is also visible from the input image. If p_i is not visible from the pose T_c, it is not sensible to add local features to $z \in Z$, since the features at v_i correspond to a different 3D point than p_i. Apart from the global and local features, the input to the occlusion module are the 3D point p_i, the pose of the conditioned input image T_c as well as the depth map of the input image D_c along with the pixel v_i, which is the result after projecting p_i to the image plane of the input image. The occlusion test is very simple; if the depth of the 3D point p_i from the pose T_c is greater than the depth at pixel v_i in D_c, then p_i is occluded. In practice, we allow for slight deviation λ from the depth at v_i. If the depth of p_i from the pose T_c deviates more than $\pm\lambda$ from the recorded depth at v_i, p_i is assumed to be occluded and we multiply the local features with zero before stacking the local and global features together into a 512-dimensional appearance vector z. If p_i is not occluded, the local and global features are stacked and fed to the Texture Field.

3.2.3 Our Texture Field Module

The original implementation assumes identical appearance features for all 3D points that are sampled. In our case, the appearance features depend on the point being sampled. Thus, instead of feeding the entire appearance feature vector $z \in Z$ to a fully connected layer, we feed the global appearance vector (dimension 256) to a fully connected layer, but choose to process the local features (dimension 256) via a 1x1 convolution, bringing the dimension down to 128. Then, we add these 128-dimensional local features to the point features and finally, just as the original implementation, the global appearance features and shape features are added.

4. Experiments and Results

This section describes the experimental setup and results in detail. First, the dataset and implementation details are explained. Then, the description of the experimental setup and evaluation metrics are introduced. The section ends with the quantitative and qualitative evaluations of the baseline and proposed models.

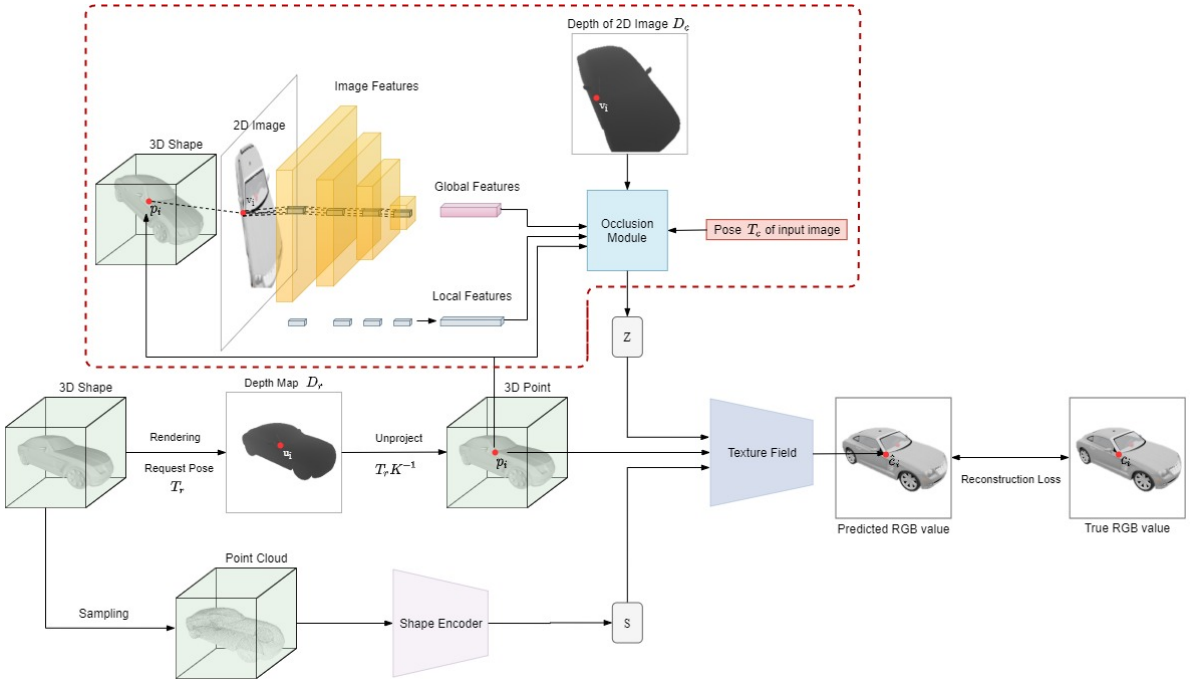

Figure 3: **Model Overview:** The shape encoder, the texture field and the image encoder (Fig.4) are trainable modules of our system and they are parameterized by neural networks. One forward pass works as follows: A 3D shape in the form of a mesh is selected for texturing and a pose T_r, from which we aim to render the mesh from, is requested. A depth map is rendered from the requested pose T_r and for each pixel u_i on the surface of the object, the pixel is unprojected into 3D to find the corresponding 3D point p_i. The 3D point p_i is used directly as input to the texture field. Using the camera parameters of the input image, it is also projected into v_i of the 2D input image plane. The local features are identified on the feature map layers of the image encoder and finally, the extracted features are concatenated to get the local features at the 3D point p_i. The local features are then post-processed (not shown in the image) using a fully connected layer to bring down the size to 256. The global and local feature vectors are then fed to the occlusion module which produces the image feature Z which is used as input to the texture field. The occlusion module also takes as input the 3D point p_i, the depth map of the input image as well as the camera pose T_i of the input image. The final input to the texture field module is the shape encoding. This is constructed by first sampling points on the 3D shape, forming a point cloud and then processing it via the shape encoder, forming a fixed length input vector S. Given the inputs p_i, Z and S, the texture fields module predicts the color $\hat{c}_i$ at the point p_i.

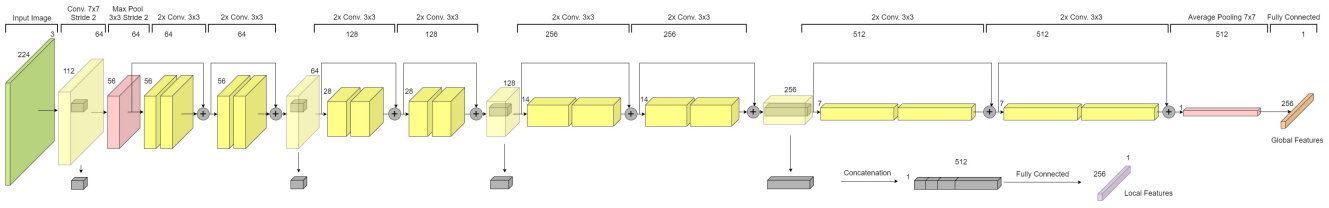

Figure 4: **Image Encoder:** The image encoder is essentially a ResNet-18 [7] network, but with the following modification: The final fully connected layer of ResNet-18 is replaced to give an output of size 256 instead of 512. Local feature extraction works as follows: First, the feature maps from which we want to extract features are spatially bilinearly upsampled to the size of the input image. At the scale of the input image, we extract the features at the spatial location v_i (see Fig.3). Then, the features are concatenated and fed to a fully connected layer which outputs a 256-dimensional local feature vector.

Figure 5: Requested fixed poses from a ground truth model. From left to right: top, front, side right, back, side left.

4.1. Dataset

We used the *"car"* category of the ShapeNet [1] dataset to train and test our models. Baseline Texture Fields uses 10 RGB images and depth maps for each object to train the model. They also use the image renderings provided by Choy *et al.* [3] for the input images. Our conditional setting, however, additionally requires depth information for the occlusion module. For this reason, we created our own renderings and utilized them to train the our model.

We created 24 different views of each object, assuming a fixed camera distance of 1.5 m between the object center and the camera. Then, we sampled azimuth and elevation angles from a Gaussian distribution to determine the camera location. For the azimuth angle, we divide the 360° into 24 bins and use the bin centers to set the mean of the distribution. For the elevation, we set the mean angle to 25°.

4.2. Implementation Details

We used 224×224 images in training. We trained our model using Adam optimizer [10] with learning rate 10^{-4}. We performed 190.000 iterations with batch size of 8. We also set $\lambda = 0.02$ in occlusion module to allow small deviations after projection. We refer the reader to [15] for details of the network architectures and implementation details.

4.3. Experimental Setup

To test whether the local features assists during texturing 3D shapes, we devise the following testing strategy. Simply, if the requested pose of the output image is "similar" to the pose of the input image, a performance gain should be evident, compared to the case when the poses differ a lot. To this end, we fix the output pose into one of five poses: $\{top, front, back, side\,right, side\,left\}$ as seen in Fig.5. By keeping the requested pose fixed during evaluation but changing the pose of the input image, the texture reconstruction can be evaluated. We limit the input pose to vary in the azimuth angle. Azimuth angle 0 is aligned with all frontal poses.

4.4. Evaluation Metrics

To evaluate the models, we adopt the evaluation metrics of Texture Fields [15]. The Frechet Inception Distance (FID) [8] is used to measure how well is the appearance distribution of the predicted images compared to the ground truth images. The structure similarity image metric (SSIM) [24] is used to estimate the distance between the predicted images and the ground truth images on a per-instance basis. The authors of [15] found that the SSIM metric mainly captures local properties of images, and thus additionally introduce more global similarity measure called the $Feature\text{-}l_1\text{-}metric$. This metric is computed, similar to FID, by embedding the predicted image and the ground truth image into a feature space with an Inception network. Using the two feature vectors, the $Feature\text{-}l_1\text{-}metric$ is computed as the mean absolute distance between the two vectors.

For completeness, we additionally compute the per pixel L1 error between the predicted and ground truth images which Texture Fields does not report.

	0°	90°	180°	270°
FID	113.06	107.15	111.38	107.23
SSIM	0.915	0.918	0.914	0.917
FeatL1	0.247	0.245	0.246	0.245
L1	0.025	0.024	0.027	0.025

Table 1: Evaluation scores when requesting the frontal pose and changing the azimuth angle for the baseline Texture Fields model.

	0°	90°	180°	270°
FID	41.21	105.66	105.94	112.19
SSIM	0.955	0.926	0.910	0.925
FeatL1	0.182	0.241	0.254	0.245
L1	0.011	0.017	0.024	0.017

Table 2: Evaluation scores when requesting the frontal pose and changing the azimuth angle for our framework.

4.5. Results of Original Texture Fields

The baseline model was trained for 78,250 iterations with batch size 18 over a period of 48 hours. Fig.6 shows a graph of the evaluation metrics (averaged over the entire test set) when the frontal pose is requested and the azimuth angle of the input image is changed. Tab.1 demonstrates the numerical results of four such azimuth angles. Fig.7 shows typical outputs for one such object.

4.6. Results of Local and Global Texture Fields

Here we show the results for proposed texture fields that is obtained by utilizing our method. Fundamentally, effect of including local features is evident in our experiments. We observe that local features helped us to obtain fine detailed textures which is not obtainable with compared model. Please refer to appendix for detailed results. Specifically, when we compare front pose of two models, we observe that our model has better results both qualitatively and quantitatively. As expected, the best evaluation results are obtained when respected pose matches with azimuth angle. Similar outcomes are proven for other requested poses in supplementary document. Overall, we are able to generate

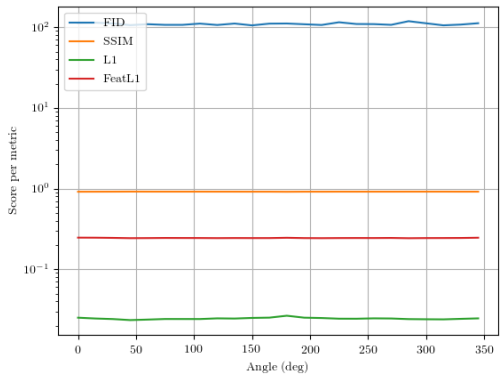

Figure 6: Results per evaluation metric and angle (split into 24 bins) for the baseline Texture Fields model for the front view of the car class.

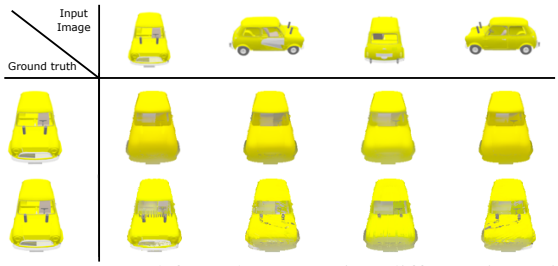

Figure 7: Textured frontal poses using different input images for the baseline Texture Fields model (top) and our model (bottom).

better results compared to original Texture Fields when requested pose and input pose is closer, whereas our results are as good as those of baseline when there is huge difference between poses.

5. Discussion

It is clear that the baseline Texture Fields model does not utilize local information for texture inference. This is evident from the fact that the graphs in Fig.6 are virtually constant with respect to the viewing angle. This is additionally backed up by the numerical results in Tab.1 where the evaluation scores are very similar across four distinct azimuth angles. Qualitatively, this can also be seen in Fig.7, where the output is more or less constant with respect to the input image.

When results of our model analyzed, it is clear from Fig.7 and Fig.8 that local appearance information is effectively utilized wherever possible and elsewhere, performance is on par with the original Texture Fields model. One can observe that given requested pose of the image, our model generates better detailed front pose image compared to baseline Texture Fields model. We observe the same outcome for other compared image objects, please refer to supplementary document. Numerically, we also observe aligned results with qualitative results.

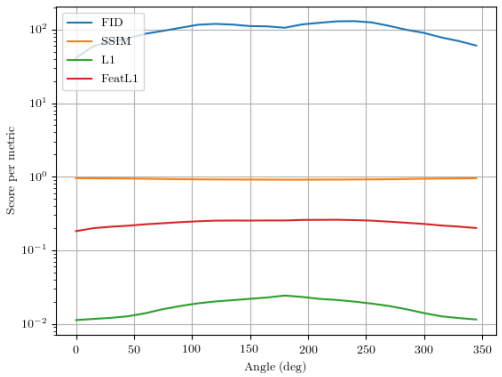

Figure 8: Results per evaluation metric and angle (split into 24 bins) for the our Texture Fields model for the front view of the car class.

Limitations: Our model assumes that the input image comes from the same identical shape as the input shape. In the original Texture Fields formulation, any image can be used as appearance conditioning. We argue, however, that our formulation is natural in the context of single image 3D reconstruction with texture, where one takes a single image and tries to reconstruct both the shape and texture from the single observation. Further, our model requires known poses as well as known depth of the input images during both training and testing. In a realistic setting, the pose needs to be estimated which introduce further complexity to the problem, but that is outside of the scope of this project.

Another limitation is that the model uses a so called object-centered coordinate system, which aligns objects of the same semantic category to a common orientation. Aligning objects this way makes it particularly easy to find spatial regularities. As discussed by [22], it encourages learning based approaches to perform recognition rather than reconstruction, which may not be the optimal approach of doing either shape or texture reconstruction.

Conclusion: In this project, we have presented a framework for learning texture representation. Our key idea is to additionally extract local features and use them to render new images. Our results have shown that local appearance information helps in obtaining more detailed texture reconstructions.

Work Distribution: Erik gathered the test results and visualizations from the baseline Texture Fields model as well as writing the main parts of the report which also includes making the architecture diagrams. Berk created the new dataset for training and implemented our proposed framework. Mesut arranged the dataset, created the test results and corresponding visualizations of our proposed model, proof-read the report and contributed to the supplementary material.

References

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[3] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] J.-D. Durou, M. Falcone, and M. Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109:22–43, 01 2008.

[6] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

[9] B. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 10 2004.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[12] L. M. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018.

[13] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019.

[14] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 2013.

[15] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4531–4540, 2019.

[16] M. Oechsle, M. Niemeyer, L. M. Mescheder, T. Strauss, and A. Geiger. Learning implicit surface light fields. *ArXiv*, abs/2003.12406, 2020.

[17] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *arXiv preprint arXiv:1901.05103*, 2019.

[18] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019.

[19] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. *CoRR*, abs/1611.05009, 2016.

[20] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *arXiv preprint arXiv:1905.05172*, 2019.

[21] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[22] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.

[23] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[24] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004.

[25] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pages 540–550, 2017.

[26] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[27] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–662, 2018.

[28] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 490–500, 2019.

[29] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. *CoRR*, abs/1905.10711, 2019.

[30] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:690–706, 08 1999.