Reconstructing Learned Geometry and Texture on Human Bodies

Anton Baumann
Technical University Munich
anton.m.baumann@tum.de

Poyraz Kivanc Karacam
Technical University Munich
poyraz.karacam@tum.de

Umesh Rajesh Ramchandani
Technical University Munich
umesh.ramchandani@tum.de

Yigit Aras Tunali
Technical University Munich
yigitaras.tunali@tum.de

Abstract

Most of the prior work of inferring 3D texture focuses on texture atlases or colored voxels which have their shortcomings. The texture atlas approach requires UV-mappings hence introduces discontinuities, while the voxel approach is quite memory inefficient and has limited resolution. Other approaches such as predicting RGB colors at XYZ coordinates given 2D images also exist but in Texture-IF-Net [6] it is shown that high-quality 3D reconstructions for texture and geometry can be achieved from partial 3D scans. The Texture-IF-Net and the IF-Net [5], which are used to reconstruct geometry, create a pipeline to completely reconstruct 3D geometry and texture from partial scans. In this project, we investigate both IF-Nets in detail and adjust them for real-world scenarios. We propose several data augmentations on the SHARP dataset and training on top of the pretrained IF-Net networks. With the augmented dataset and training we aim to generalize the Single View Reconstruction from a front view that IF-Nets can accomplish to any viewpoint with real data scanned by a Kinect camera.

1. Introduction

Complete 3D scans can be accomplished using expensive and high-tech cameras, which require a studio setup. In the last few years commodity, handheld scanners became more and more widespread. One of the successful dense-reconstruction works with the hand-held RGB-D scanners was demonstrated in Kinect Fusion [11]. Since it is unrealistic to expect the same performance from the handheld scanners as with studio set-ups, the resulting scans can be noisy or partially missing.

The approach in Texture-IF-Net [6] involves a 2 stage pipeline. The first one being the geometry reconstructing IF-Net. The IF-Net accepts various inputs, making it quite flexible. It can work with low and high-level resolution voxel grids, sparse or dense point clouds, either complete or incomplete. Since the main goal of this project is to reconstruct humans from a Single View partial scan, the focus of our approach will be the Point Cloud version of the network. From the input, a renderable continuous and complete surface is reconstructed.

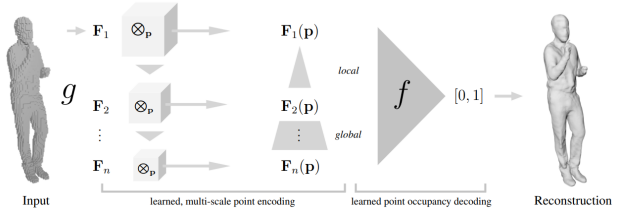

Figure 1. Complete pipeline of the IF-Net.

As can be seen from figure 1, the input first is used to create a multi-scale deep feature grids of shape $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_k \in \mathscr{F}_k^{K \times K \times K}$ with $k = \frac{N}{2^{k-1}}$ and N being the input dimension. In previous works the 3D shape would have been encoded into a latent vector $z \in \mathcal{Z} \subset \mathbb{R}^m$ and then used to learn a neural function to classify if a point in $\mathbb{R}^3$ is inside the surface or outside.

$$f(\mathbf{z}, \mathbf{p}) : \mathcal{Z} \times \mathbb{R} \rightarrow [0, 1] \tag{1}$$

Instead of the latent vector approach, in this work the shape is encoded into a *multi-scale deep feature grids*. The 3D convolutions are applied similar to their 2D counterpart. The convolutions are applied, then followed by down scaling the input, creating growing receptive fields and channels but shrinking the resolution. The earlier feature grids such as $\boldsymbol{F}_1$ capture shape detail and the later ones like $\boldsymbol{F}_k$ capture global structure.

$$\{\mathbf{p} + a \cdot \mathbf{e}_i \cdot d \in \mathbb{R}^3 | a \in \{1, 0, -1\}, i \in \{1, 2, 3\}\} \tag{2}$$

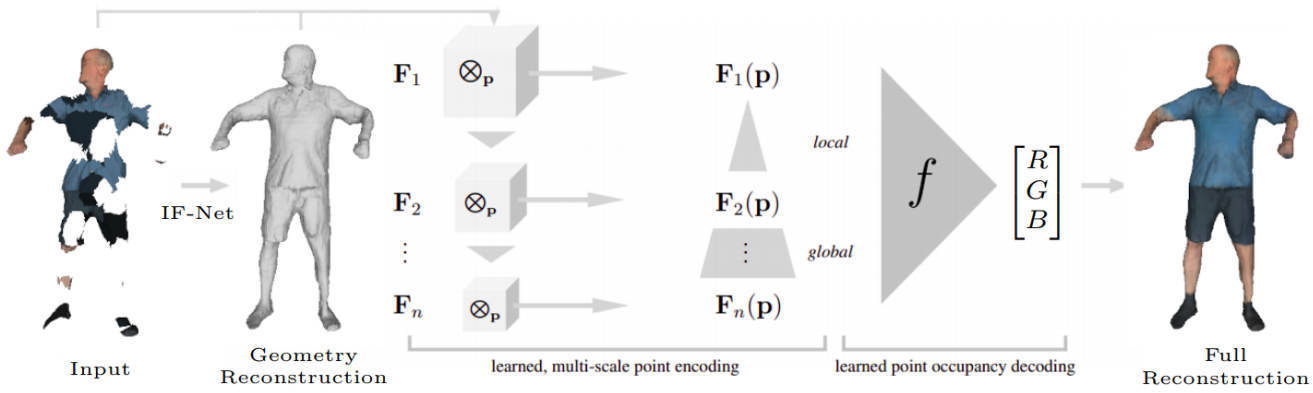

Figure 2. Complete pipeline of the Texture-IF-Net.

For reconstructing the geometry, the point coordinates aren't classified directly but extracted from the learned deep feature grids at those locations. Since the encoding has a 3D structure that is aligned with the input, this is easily accomplished compared to other methods. With feature grids being discrete the 3D continuous points are queried using trilinear interpolation. To also encode the neighborhood information at all the grids, a distance $d \in \mathbb{R}$ of any point that will be extracted is also included with formula (1) (where $\mathbf{e}_i \in \mathbb{R}^3$ is the i'th Cartesian unit vector).

The second part of the pipeline consists of the IF-net with texture that uses the reconstructed geometry to predict colors on it. The model accepts as input the partial textured scan and the complete geometry, in total 4 channels, as can be seen in 2. The geometry input is reconstructed with the original IF-Net. This network works in a similar way where again multi-scale learned deep feature grids are created with 3D convolutions. The point, just as in the IF-Net, is again extracted the same way continuously and then fed to a point-wise decoder, which is parametrized by a fully connected neural network with ReLU activations to regress the RGB color at that exact point.

For this project, we are interested in the SVR (Single-View Reconstruction) version of the network. The performance of the networks was tested, for the case of SVR, with front-facing synthetic data. We observed that the network which was pre-trained with this kind of data was not able to generalize well for scans from other views. To circumvent this shortcoming we propose data augmentations to the original dataset which include random rotations and projections to create more Single-View data that is seen from different angles. We then use this extended dataset and train on top of a pre-trained SVR model. In the following sections, we summarize related work, our methods for the data augmentation, experiments, and results.

2. Related Work

Over the years there have been many methods to reconstruct 3D objects and humans. Most notable of these methods, for humans, are voxel, implicit, parametric and mesh based representations.

2.1. Parametric & Mesh Based

These kinds of approaches try and learn the parameters of a parametric model or learn to fit the human on a mesh-based model. One of the limiting factors of these approaches is the quality of the parametric or the mesh model. Most of the research done with these models uses the SMPL [9] model or templates [8]. Some examples of such work include clothed 3D reconstruction using 2D images from a single RGB camera or video sequences [2, 4, 3] and predict deformations from the base model. Another shortcoming of the model-based CNN approaches is that the results being over-smoothed.

2.2. Voxel Based

Voxel-based approaches [13, 15, 7] tend to produce more details than the mesh-based ones, because the predictions align with the input data. Most of these approaches process image pixels. One of the shortcomings of these approaches is the possibility of having missing parts of the body. To circumvent this some approaches [13] do a post-process of fitting the SMPL [9] model to the reconstruction. Voxel-based approaches also have limited resolution bounded by their grid size.

2.3. Implicit Functions

Truncated Signed Distance Functions have been used to reconstruct static [11] and dynamic scenes with non-rigid tracking [10] before. Using similar approaches and combining it with the SMPL model produces robust and accurate tracking [14]. There are works that take as an input an

image and produces humans in clothing using an implicit network [12]. The reconstruction is done by predicting the occupancy of 3D points is done point-wise with regards of the spatial location and the 2D image features.

3. Methods

3.1. Data Processing

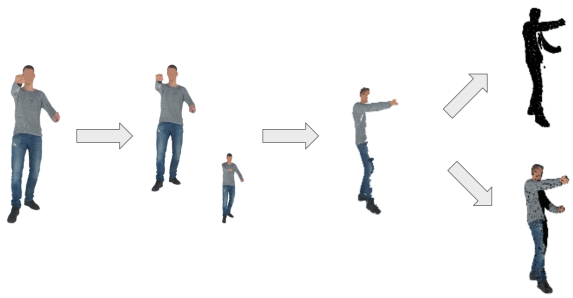

Figure 3. Data Processing Pipeline. Given an initial mesh, it is first aligned and normalized. Then a random rotation is applied to simulate a random single view and a partial mesh is obtained. Finally, points are sampled from partial and complete meshes to prepare the final input for the model.

In this project, we used the same dataset used in [6]. Initially, the dataset consists of 3D meshes of people acting out various poses with varying clothing, and the meshes have full texture and geometry. Some of the people have regular clothing, and some of them have minimal fitness clothing. We process the initial data to create a new dataset that would serve our end goal. This process comprises the following steps:

- Alignment and normalization
- Random rotation
- Point cloud voxelization

In the end, the processed data consists of partial meshes of people that are observed from a single view. The complete data processing pipeline is demonstrated in Figure 3.

3.1.1 Alignment and normalization

To align a given mesh, we first compute its centroid and apply the following translation:

$$\delta = (x_c, y_c, z_c), t = (-x_c, 0, -z_c) \tag{3}$$

where δ is the mesh centroid and t is the applied translation. Then, we normalize the height of the mesh and apply a second translation:

$$t' = (0, -0.5, 0) \tag{4}$$

and obtain a mesh that is centered around the origin point and has a height that falls inside the interval $[-0.5, 0.5]$.

3.1.2 Random rotation

To each aligned & normalized mesh we apply 4 different random rotations around the vertical axis to create 4 partial meshes observed from different views.
After a rotation is applied, the view is rendered as a 2D image, and then back-projected into the 3D world with the known depth. Thus, we obtain a partial point cloud of a surface from a single view. The inverse of the initial random rotation is applied to the obtained point cloud for spatial consistency. We further query these points on the complete mesh to remove any unmatched points from the unobserved space along with their corresponding faces. As a result, we obtain a partial mesh observed from a single view.

3.1.3 Point cloud voxelization

Point cloud voxelization is done similarly as in [5] and [6]. For geometry reconstruction, we create a grid with 256 resolution and $[-0.5, 0.5]$ bounding box. Then we sample $10,000$ points on the partial surface, and query them in the created grid and find the occupancies.
For texture reconstruction, we again create a grid with the same properties. Then we densely sample $100,000$ points on the partial surface and find their occupancies. In addition, we also densely sample and voxelize $100,000$ on the full but untextured surface.
Finally, all the voxelized information is encoded as described in [5] and [6] to prepare the final input for the respective models.

3.2. Real World Data

The Microsoft Kinect Azure camera is a consumer device that can give various outputs in RGB-D format. But the networks take voxel representation as input. So we had to design a pipeline to convert Kinect's RGB-D output to fulfill our needs i.e. in our case, occupancy grids. We decided to do this task by first converting the RGB-D output to a point cloud representation using Microsoft's Kinect SDK. Later we observed that this input has a lot of extra noise (i.e. Surrounding Objects) in it. We then decided to clean this input by masking the depth channel of the Kinect's output with the human segment mask obtained by Microsoft Kinect's SDK. As you can see in figure 4 we get the masked depth. And then generating a point cloud from this masked input gives us the result with the only human subject. Later we preprocessed this point cloud by taking a bounding box of

the input point cloud and rotating it in an upright position by applying the inverse of its rotation to the point cloud. Later this point cloud was converted to a voxel representation.

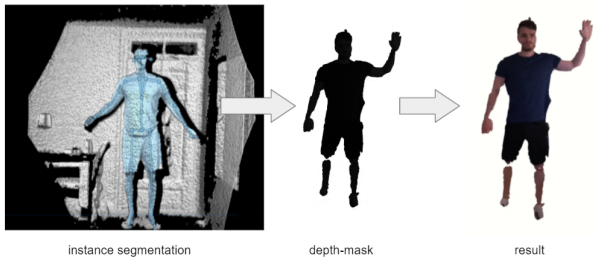

Figure 4. Human Segmentation of Pointcloud Input.

4. Results & Experiments

Figure 5. Texture reconstruction from partial pointcloud input given the ground truth geometry. Left-most image is the ground truth, middle and right-most images are model reconstructions from a back and a front view respectively. The partial mesh is observed from a front view.

We have tested the integrity of the models in two different scenarios. First, we looked at the texture reconstruction only and used the ground truth geometry without the texture in the input. Afterwards, we applied full inference on the partial point cloud data by initially reconstructing the geometry and then using this reconstruction in the input to further infer the full texture.

We observed that the performance of the texture reconstruction model depends on the observed view. In cases where the view is captured mostly from the front, the model performs well as can be seen in figures 5 and 6. However, when the view is captured mostly from the back, it fails to do a realistic reconstruction.

We argue the reason thereof is, most of the time the front view is more detailed and the back view is relatively more uniform in terms of color distribution. A person might have

Figure 6. Texture reconstruction from partial Point Cloud input given the ground truth geometry. Top row images are views from the ground truth, and the bottom row views are from the model reconstruction.

Figure 7. Texture reconstruction from partial Point Cloud using complete pipeline. The view was created with an angle from behind.

a t-shirt and jacket with the front open, and the back view of such a person would only contain the backside of the jacket and the t-shirt wouldn't be visible. In such a case (Figure 7) if the person is observed from the back, it is hard for the model to create finer details and it rather assumes the reconstructed color distribution should have less variance.
The most challenging cases in geometry reconstruction are the "side-ways" views. When a part of the person, such as an arm or a leg, is almost completely occluded because of the view (Figure 8), the reconstruction is not able to regenerate these missing parts. But, when the limbs are captured properly, as in a mostly back or a front view, we observed that the model performs well.

In some cases, ground truth meshes might contain color deformations. The learned model has a smoothing effect on the regions which prevents it to be influenced from such deformations and actually corrects them (Figure 9). On the other hand, the said smoothing effect also prevents the model the retain fine texture details.

Besides reconstructing the synthetic dataset, we also

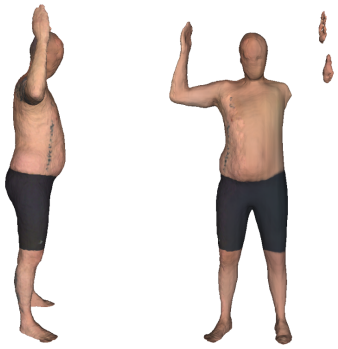

Figure 8. A fully reconstructed mesh. The observed view on the left, and a front view of the same mesh on the right. A side-ways case where the geometry reconstruction doesn't perform well.

Figure 9. A snapshot from a ground truth mesh. Some ground truth meshes have "corrupted" color regions. However, texture reconstruction is not affected from such cases.

Figure 10. Couple of examples reconstructed with the full pipeline.

tested our geometry and texture models on real-world data from the Kinect Azure camera. Since the data coming from the Kinect camera is not from the same distribution as our training data (real vs synthetic) and has inherent noise, we expected that the reconstructions wouldn't be pristine. The results can be seen in the figure 11, while it is not perfect it performs rather well given the problem is a gap-filling between simulation and the real world.

Figure 11. The final reconstruction on real world data from point cloud captures. Left is the original input, middle and right is the reconstruction.

5. Evaluation

For observing quantitative metrics, we have used the original evaluation metrics from the CVPR Sharp 2021 Challenge [1]. In table 12 we have compiled the evaluation results for 20 predicted samples from our test set. For Surface Face Areas, the metrics for Reconstructed Mesh and the Ground Truth should be closer to each other. Since Reconstructed Mesh being smaller than Ground Truth indicates that some part of the mesh is missing or has a lesser area than the ground truth. And a larger metric indicates exaggeration of some form in the reconstruction.

Shape Accuracy and Texture accuracy are a measure of similarity between the Reconstructed Mesh and the Ground Truth. The value of the reconstructed mesh shows the similarity of a sampled point on the Reconstructed mesh with a triangle on the Ground Truth Mesh. And Ground Truth shows the similarity of a point in the Ground Truth mesh to the reconstructed mesh. Texture Accuracy performs a similar function to shape Accuracy, it takes a Euclidean distance of the color of the point in the source mesh and a similar point in the destination mesh. This metric is subjected to minimization.

Surface hit rates indicates similarity between the source and the destination mesh. For this metric, a value of 1 is the perfect score. Our method gets 0.93 when measured from Reconstructed mesh to Ground Truth, which shows that 93% of points in the source mesh have a similar point in the destination mesh.

	Reconstructed Mesh	GT Mesh
Surface face areas	0.6783707222	0.744
Shape Accuracy	3.25e-03	7.94e-02
Texture Accuracy	3.08e-03	8.16e-02
Surface Hit Rates	9.38e-01	8.68e-01

Figure 12. Metrics for the reconstruction

6. Conclusion

Based on the experiments and observed results, we found that the model's performance is overall satisfactory. Although most of the reconstructions done by the model achieve good results, there are some edge cases like losing body parts that are mostly occluded in the view. Since the main focus of this project was to generalize the model to work with different view angles and with real-world data scanned by a Kinect camera, the final step we took was to conduct such tests. We have seen that we were able to reconstruct humans in an acceptable manner.

There are some improvements one can use to get better results, especially for the real-world data. First one is using an SMPL or SMPL-X model as a prior in a post-processing step and non-rigidly deform it to fill out the missing body parts. To improve on the texture reconstruction, performing Segmentation or addition of a positional encoding layer might help in getting better results for the texture predictions. Another step would be to train both models even more epochs. Due to hardware and time limitations the amount of epochs trained for both models were not as high as the original paper [6] and we trained over the pretrained models to have a good prior. We are confident that more training, and especially training from scratch instead of using pretrained models as the base weights for the model, would also increase the performance considerably.

References

[1] Sharp2021 metrics. `https://gitlab.uni.lu/cvi2/cvpr2021-sharp-workshop/-/blob/master/doc/evaluation.md`, 2021.

[2] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1175–1186, Jun 2019.

[3] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *IEEE Conference on Computer Vision and Pattern Recognition*. CVPR Spotlight Paper.

[4] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *International Conference on 3D Vision*, pages 98–109, Sep 2018.

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[6] Julian Chibane and Gerard Pons-Moll. Implicit feature networks for texture completion from partial 3d data. In *European Conference on Computer Vision (ECCV) Workshops*. Springer, August 2020.

[7] Andrew Gilbert, Marco Volino, John Collomosse, and Adrian Hilton. Volumetric performance capture from minimal camera viewpoints, 2018.

[8] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video, 2019.

[9] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.

[10] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136. IEEE Computer Society, 2011.

[12] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization, 2019.

[13] Gl Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. Bodynet: Volumetric inference of 3d human body shapes, 2018.

[14] Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor, 2018.

[15] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. Deephuman: 3d human reconstruction from a single image, 2019.