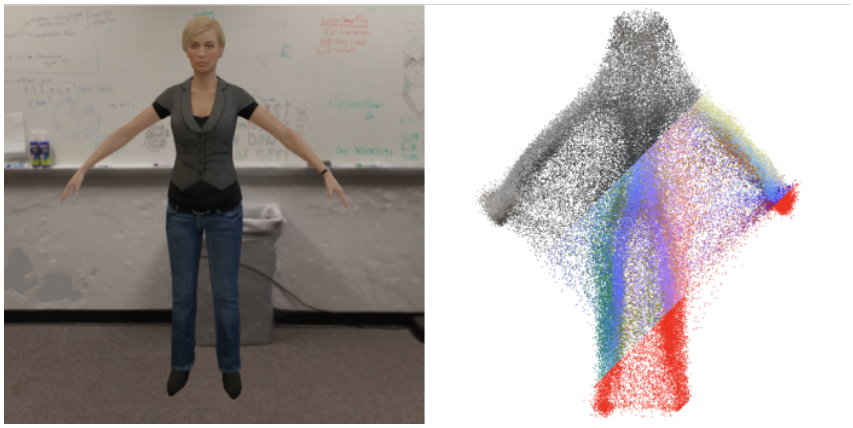
Open Source Avatars

3D Avatar Reconstruction From A Single Image

Jan-Niklas Dihlmann
Matrikelnummer 4117847
jan-niklas.dihlmann@student.uni-tuebingen.de

Abstract

Open sourcing digital avatar reconstruction seems to be a distanced goal. This is not only due to the fact, that there needs to be a lot more research to reach the goal of true canny avatars. But, is also caused by methods being impractical or expensive for being deployed as a service or technology that is made platform exclusive by big companies. In this work we first rebuild a simple, easy-to-use, state of the art method, that is not public available yet and only needs a single image to reconstruct an avatar. And second, provide tools to build up needed datasets and a viewer to easily deploy services for future models and improved versions of our work. We do so in the hope to make a step in the direction of open-sourcing avatar creation.



GitHub Repository OSA

1 Introduction

In a world where a lot of big companies bet on the next evolution of the internet, the metaverse a digitally expanded version of the universe, and that relies more and more on virtual presence communication a digital representation of ourselves is indispensable (1; 2). Creating realistic and canny virtual avatars, that also feel good when encountered, needs a lot more research, but there are methods available to create versions of scanned and animated humans (3; 4; 5). While these methods are available, they are very costly and need a lot of effort and technical know-how to be put into practice.

The goal of this work is to shrink the gap by stepping in the direction of open sourcing realistic digital avatar creation and thereby make digital avatars available for a broader audience. This is achieved

by rebuilding parts of the state of the art methods and providing tools for others to expand on this work. We decided to build upon the work of photorealistic monocular 3D reconstruction of humans wearing clothing (Phorhum) of Alldieck et al. (6). The main reasons for doing so are that the work is state of the art but more importantly allows to construct a realistic avatar from only a single image, enabling easy access to create a digital clone for almost every human. Since neither the code nor the dataset of their work is available to the public, we will showcase in the following how to collect and prepare a free dataset and also rebuild parts of their end-to-end trainable model. We will provide some improvements and ideas to extend the default model. Further, we will display a prototype of a viewing tool, that could be used to deploy avatar creation as an easy-to-use service for everyone. This work is part of a research internship at the human-computer interaction department of the university Tübingen. It should be noted that we only spend two months on this project, therefore the qualitative results of this work could be improved with more time.

2 Background

There are many approaches to create photorealistic avatars, Prokudin et al. use poseable 3D models and human images to create avatar images, Alldieck et al. create 3D people with video-based input and Gafni et al. use monocular input sequence to reconstruct 4D facial avatars (7; 8; 9). While these approaches show qualitative great results they impose methods that are unpractical for quick and easy avatar creation. Singular image to avatar reconstruction is another sub-field of realistic avatar reconstruction and is most likely the simplest method. Saito et al. showed great success with their pixel-aligned implicit function and corresponding follow-up work to construct avatars from single images (10; 11; 12). With ARCH++ Tong et al. improved their work and introduced an unposed mechanism to create animation-ready avatars (13; 14). While those methods were pioneers in this field, they are very complex to implement and maintain. Phorhum by Alldieck et al. tackles those issues by introducing an end-to-end trainable model, that operates on signed distances and not only returns the color but also the albedo, shading, and surface information (6). In addition, it reduces the preprocessing overhead by getting rid of center cropping and image masking (6). With Phorhum being the state of the art method for avatar 3D reconstruction from a single image, we will take a deeper look at how they built up their dataset and implement their architecture.

2.1 Dataset

Alldieck et al. used expensive commercial websites to acquire 3D scanned posed and unposed humans and also used their own scans with a total of 217 avatars (6; 15). For the background environments of the image dataset they used free available HDRs, although they did not mention it, they probably ground projected them (6; 16). They rendered over 190k images with random HDR backgrounds and color and pose augmented avatars (6). To train the network, they use random points, on and near the surface of the avatar mesh, to probe the distance, color and normal.

2.2 Network

The architecture of Alldieck et al. contains three sequential networks, a feature extractor network that receives a $512 \times 512 \times 3$ input image and produces pixel-aligned features for all points within the space, an implicit signed distance network that computes the distance to the closest surface point, and the corresponding color for a point with pixel aligned feature vector, and a shading network that predicts the shading for surface points given the previously predicted surface normal and latent space from the feature extractor (6). This architecture is similar to ours (Fig. 6) for further implementation details we refer to their work (6).

3 Methods

The goal of this work is to predict the 3D colored surface of a human presented in an image. To do so we present a full pipeline. First a dataset generator for the various datasets that are needed for training, mainly the image dataset and the signed distance field (SDF) dataset. And second, a end-to-end trainable network, that is inspired by Phorhum by Alldieck et al., including a few changes and an idea to improve the work (6).

3.1 Dataset

Predicting the surface from a single image is a relatively new task and therefore there aren't any dedicated datasets available containing images and corresponding SDF with color information. State of the art solution build their image dataset by rendering scanned humans or 3D modeled avatars over a realistic background and a corresponding SDF colored dataset by probing the mesh and the surrounding space (6; 10; 14). None of these made their image and colored SDF dataset public available, therefore we build our own custom dataset. We first selected an open source human avatar library and collected a small indoor environment dataset as well as a high dynamic range (HDR) image dataset. After that, we rendered the collected data into images of the avatars within varying environments and HDRs. In a final step, we probed the surface and surroundings of the mesh, to evaluate the SDF and store the corresponding color and normal information.

3.1.1 Avatar Dataset

A 3D human model dataset was needed that fits the properties of the project, which are that the models have high fidelity with a realistic look, have an albedo texture map, are rigged, come in a compatible file format, and are affordable. While there are many available datasets for 3D human models, there are only a few that satisfy those properties. The people snapshot dataset by Alldieck et al. for example does not satisfy the realistic look, while the human dataset, which is used by the original paper, would fit perfectly but is too expensive (17; 15). Other datasets such as metroploy 3D people, humano 3D people, 3D people dataset, or 3D people have been evaluated but did not fit the criteria (18; 19; 20; 21). We settled with the Microsoft Rocketbox dataset (Fig. 1), although the avatars could look more realistic it fits the needs of the project and is under the MIT license (22). In an additional step, the retrieved FBX file had to be converted to OBJ files to fit into the pipeline, a script was written that converted the FBX files with Blender (23). The dataset contains a total of 44 unique avatars, but since each base model has 1 to 2 profession, there are a total of 116 avatar variations.

Figure 1: Snapshot of the Microsoft Rocketbox dataset - woman, men, children, profession (22)

3.1.2 Environment Dataset

To create an image dataset of realistic-looking pictures a background is needed, recent methods use ground projected HDRs (6). While ground projected HDRs are an easy, computational cheap, and fast way to generate realistic-looking scenes and augment the background, it only works with scenes with a wide open area and a background that is far away. Therefore they are not suited for indoor HDRs. Since the goal of this project is to open source avatar creation, the task has to be solved from a user perspective. Since most users will probably take pictures inside their homes or offices for convenience, a dataset is needed that is constructed of indoor environments. Therefore we collected models from photogrammetry scanned homes, offices, and hotels from Sketchfab (24). The models have been cleaned inside Blender and environment scenes were constructed for every variation of the models (23). A plane clipping shader was built (Fig. 2) to cut a viewing window into the environments and allow to hand-select different viewing angles. We collected and prepared a total of 11 unique environments with 20 angle variations(Fig. 2). Additionally, we collected 18 HDRs from Poly Heaven to render the avatars and environment under different lighting conditions (16).

Figure 2: Environment dataset (left) and clipping plane shader (right)

3.1.3 Image Dataset

The image dataset was constructed by rendering the avatar models from the avatar dataset within the environments of the environment dataset. For this purpose a small pipeline was created, loading the avatar into the corresponding scene. Since the avatar and environment dataset is relatively small compared to recent methods, a random augmentation step was introduced in the pipeline to create a greater variety (6; 10; 14). We allow for a augmentation of the camera position and rotation, HDR lighting, avatar position and rotation, as well as avatar shoulder and elbow bone rotation (Fig. 3). We decided against the usage of pose variation, since a unified pose, such as the A-Pose, is easier to learn and could serve as confirmation, such that not every picture of a human could be turned into a digital avatar, making digital identity theft harder.

Figure 3: Image dataset (exaggerated) augmentation possibilities and variability

Each of the avatars is rendered in each of the environments with subtle random augmentations to an image with dimensions 512×512 (Fig. 4), resulting in 116 images per environment with a total of 2320 images, which is relatively small dataset compared to recent methods (6; 10; 14). We decided to take a close-up shot of the avatars to keep as much detail as possible and make only small augmentation changes to ensure pixel-aligned features.

Figure 4: Snapshot of the image dataset

3.1.4 SDF Dataset

While the image dataset is used as the input for the task, a SDF dataset is needed to compare the predicted distance, color, and normal against. We collected a dataset of points (Fig. 5) around and

near the surface and store the corresponding distance to the closest surface point of the avatar as well as the color and normal information. Recent methods sample these points in an online manner while training, we did not have the computational capacities to do so (6). Therefore we decided to preprocess the avatars and collected large point clouds, containing needed information, for each avatar. We augmented the Mesh To SDF package, which produces SDF point clouds for a given model, to also include the normal and color data (25). We then proceeded by storing two different versions of the dataset for each avatar. One version contains points that directly lie on the surface of the avatar, while the other version includes points near the surface and points randomly sampled inside the unit sphere. We collected 500k points per version meaning that we have a total sample size of 1 million points per avatar.

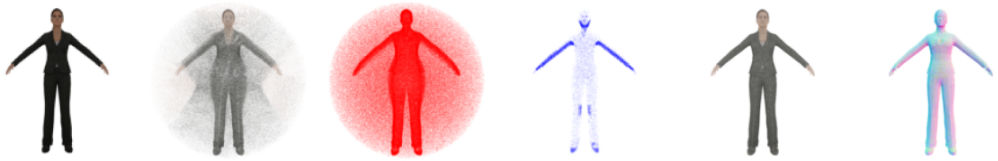


Figure 5: Sample of the SDF dataset, from left to right (avatar, points far color, points far outside, points far inside, points near color, points near color)

3.2 Network

For solving the task of inferring the surface and its color from a single image, we use an end-to-end learnable neural network model, that is inspired by the work of Alldieck et al. (6). Given the time and computational constraints of the project, we couldn't reproduce the full model and used subsets of their implementation. While the paper describes the network architecture in detail, there are some smaller ambiguities regarding the implementation. In the following, we will go deeper into how we interpreted those ambiguities and where we took different approaches.

3.2.1 Architecture

The architecture of the model (Fig. 6) is almost identical to Alldieck et al. therefore we denote S_0 as the zero-level-set of the signed distance function f for a input image $\boldsymbol{I}$ (6).

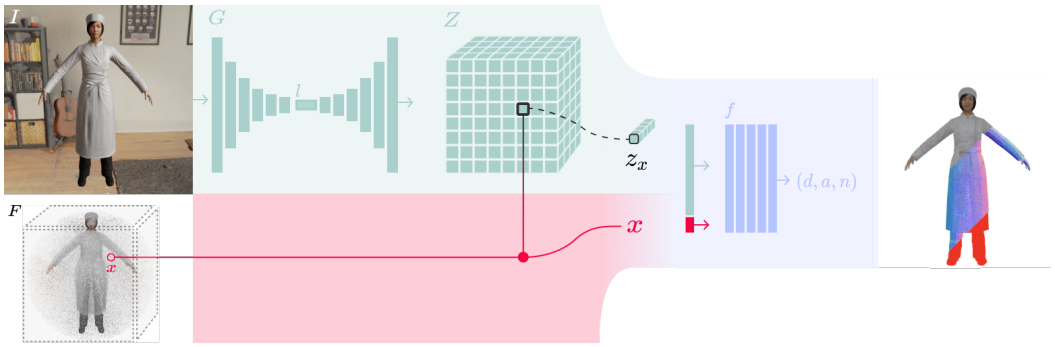


Figure 6: Overview of our method. The feature extractor network G receives input $\boldsymbol{I}$ and produced a feature vector $\boldsymbol{z_x}$ with the selection of the point $\boldsymbol{x}$ from point set F. The implicit signed distance network f computes the distance d, the color $\boldsymbol{a}$ and the normal $\boldsymbol{n}$ for the input of the $\boldsymbol{z_x}$ and x. On the right we show what the reconstruction of the $\boldsymbol{I}$ should look like, split into color, normal and distance.

$$\mathcal{S}_0(\boldsymbol{I}) = \{\boldsymbol{x} \in \mathbb{R}^3 | f(g(\boldsymbol{I}, \boldsymbol{x}; \boldsymbol{\theta}), \gamma(x); \boldsymbol{\theta}) = (0, \boldsymbol{a}, \boldsymbol{n})\} \tag{1}$$

Here $\boldsymbol{\theta}$ is denoted as the superset of all learnable parameters and γ the positional encoding function (26). In difference to Alldieck et al. we do not only make predictions for the depth d and color $\boldsymbol{a}$ but

also for the normal $\boldsymbol{n}$, since they don't explain how to infer it (6). The geometry predictor g produces a pixel encoded vector $\boldsymbol{z_x}$ for the given position $\boldsymbol{x}$ drawn from the feature extractor network G.

$$g(\boldsymbol{I}, \boldsymbol{x}; \boldsymbol{\theta}) = b(G(\boldsymbol{I}; \boldsymbol{\theta}), \pi(\boldsymbol{x})) = \boldsymbol{z_x} \tag{2}$$

Where b defines a pixel access with bilinear interpolation. While Alldieck et al. define $\pi(\boldsymbol{x})$ as the pixel location of the point $\boldsymbol{x}$ projected using the perspective camera π, we assume for simplicity an orthographic camera (6).

$$b(G(\boldsymbol{I}; \boldsymbol{\theta}), \pi(\boldsymbol{x})) = G(\boldsymbol{I}; \boldsymbol{\theta})_{(\lfloor x_0 \rfloor, \lfloor x_1 \rfloor)} = \boldsymbol{z_x} \tag{3}$$

Lastly we ignored the shading network s since it adds a vast amount of overhead to the implementation and also only improves the results minimal.

3.2.2 Losses

For the losses, we follow again the definitions of Alldieck et al. and only slightly alter them or add new ideas (6). For the introduced color loss we assume that the mean distance value is meant rather than the absolute value for each color channel.

$$\mathcal{L}_p = \frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} ||(\boldsymbol{x}_i - \boldsymbol{n}_i d_i) - (\boldsymbol{x}_i - \bar{\boldsymbol{n}}_i \bar{d}_i)|| \tag{4}$$

We define a surface projection loss $\mathcal{L}_p$ which takes the projection of the true normal $\boldsymbol{n}$ and distance d and compares it to the projection of the predicted $\bar{\boldsymbol{n}}$ and distance $\bar{d}$ for points around and near the surface $\mathcal{F}$.

3.2.3 Attention Lookup

As described before Alldieck et al. use b as pixel access with bilinear interpolation and $\pi(\boldsymbol{x})$ as a point to pixel projection to retrieve the feature vector $\boldsymbol{z_x}$ from the feature extractor G. While this is a computationally efficient way, it assumes that the true color $\boldsymbol{a}$ and distance d for a point $\boldsymbol{x}$ are perfectly lined up with the rendered image $\boldsymbol{I}$. This also means that for distanced points $x \in \mathcal{F}$ it is hard to infer the correct surface color since the lookup occurs in a different position.

$$\boldsymbol{z_x} = \sum_{i=0}^{n} \sum_{j=0}^{n} G(\boldsymbol{I}; \boldsymbol{\theta})_{i,j} * r(a(\boldsymbol{x}, \boldsymbol{l}; \boldsymbol{\theta}))_{i,j} \tag{5}$$

As another version of the network, we introduce a learnable attention lookup (Fig. **??**) that is predicted for each point. While this lookup could be fully learnable, we try to steer the network to do a ray lookup with a multivariate normal distribution $\mathcal{N}$.

$$a(\boldsymbol{x}, \boldsymbol{l}; \boldsymbol{\theta}) = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, s) \tag{6}$$

Where a is defined as the lookup predictor, taking a point $\boldsymbol{x}$ and the latent vector $\boldsymbol{l}$ and estimates the multidimensional mean $\boldsymbol{\mu}$ the covariance matrix $\boldsymbol{\Sigma}$ and a scaling factor s for the multivariate normal distribution.

$$r(a(\boldsymbol{x}, \boldsymbol{l}; \boldsymbol{\theta})) = s\boldsymbol{A} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{7}$$

We define r as the distribution function, that builds the linear system $\boldsymbol{A} \in \mathbb{R}^{n \times n \times d}$ and distributes it after the predicted multivariate normal distribution. Here $\boldsymbol{A}$ could be chosen to either function as a 2D pixel lookup similar to Alldieck et al. or as a 3D ray lookup (6).

4 Experiments

We present our experiments, in the form of the training, visualisation implementation and qualitative result.

4.1 Training

We implemented the network using Tensorflow as described in the previous section and with the implementation details from Alldieck et al. (27; 6). While building up the network from scratch, we added the non-vanilla parts of the implementation details from Alldieck et al. such as the swish activation function, blur pooling, positional encoding, and others one after another, all of these additions showed either performance improvements or increased the training speed (6). The recommended linear learning rate decay from 1×10^{-4} with a factor of 0.9 over 50k steps had a noticeable impact, meaning that changing the values prevents the SDF loss from decreasing. Since we had memory limitations, we carefully selected a batch size of 4 images for the feature extractor network and 512 points per batch for the signed distance function network. The 512 points are divided equally into 256 points from the far and 256 points from the near dataset. Our custom-designed projection loss decreased the loss and the training time of the model. We couldn't test our attention lookup proposal since we could not fit it into the memory. We trained the network on a machine with 45 GiB RAM, 8 CPUs, and an A6000 GPU with 48GiB for roughly 2 hours for about 6200 steps (Fig. 7).

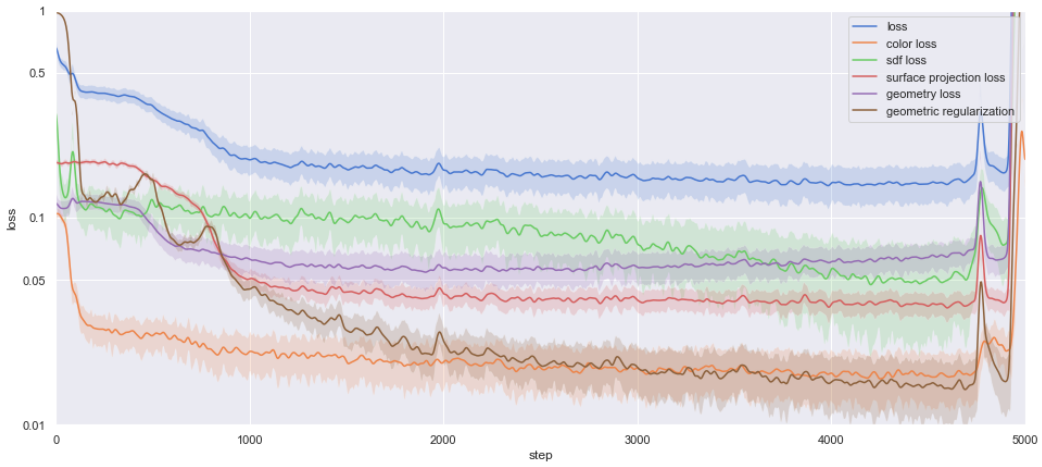


Figure 7: Training loss, where the loss is combined from weighted other losses see Alldieck et al. for details (6). Notable is that, the SDF loss takes a long time to start declining, the surface projection loss and the geometry loss proceed the same. After around 4800 the losses diverges, we used early stopping here. Beware that we use log-scaling for the Y-axis.

4.2 Results

We could not reproduce the results with the quality from Alldieck et al. (6). Our results suggest that our model learns an estimator for basic avatar geometry but fails to extract information from the input images and restore color (6). We only show qualitative results from the trained model (Fig. 8), since to compare our quantitative result with previous work, we would need to render the resulting SDF, but we did not have the time to build an SDF inference viewer and therefore can only refer to or point cloud renders.

4.2.1 Color

Viewing the results (Fig. 8) for given images and previously stored SDF points from our near and far dataset one can see, that the color reconstruction for all images looks similar, while one can differentiate body parts with various shading, the model is not able to retrieve a good color reconstruction from the presented input images. This becomes more clear when looking at input

images with different environments for the same avatar (Fig. 9). The predictions for surface points show no difference in color or normal. The general goal of the model is to infer the same albedo color for different lighting and background conditions, therfore seeing no difference in color should be a good sign. In this particular case with bad color prediction and no difference it could be suggested, that the model fails to make a connection between the image and presented points. The observed color prediction could be interpreted as, that the network simply outputs the mean color for every avatar and is not able to draw color information from the feature extractor network.

Figure 8: Qualitative results for input images (left) and queried training points, for the near row from left to right (normal, predicted normal, color, predicted color), and for the far row from left to right (colors, surface projected colors, surfaces projected outside points, surface projected inside points). The reconstructed normal does not fully resemble the true normal but is consistent and similar. The predicted surface color only contains some information about the true color, mainly contrast. The estimated surface projection reveals parts of the geometry, although the section between the arms is not predicted correctly.

4.2.2 Geometry

The reconstruction of the normals for points on the surface (Fig. 8) looks decent with consistent normals, although some detail is lost as the normal image looks blurry compared to the true normals. The projection of the far points to the surface with predicted normal and distance reveal the rough structure of the avatars. It can be clearly stated that the area between the arms and the body of the avatar isn't well estimated. For new points sampled randomly within the unit sphere of the avatar

(Fig. 10) the surface projection estimation show almost identical results for different input images. This is a different result than for the surface projection of points presented during learning (Fig. 8). Interpreting this in combination with the bad estimation of color (Fig. 9), one could state that the network is remembering the points and the rough structure of the avatars without learning the correlation to presented images.

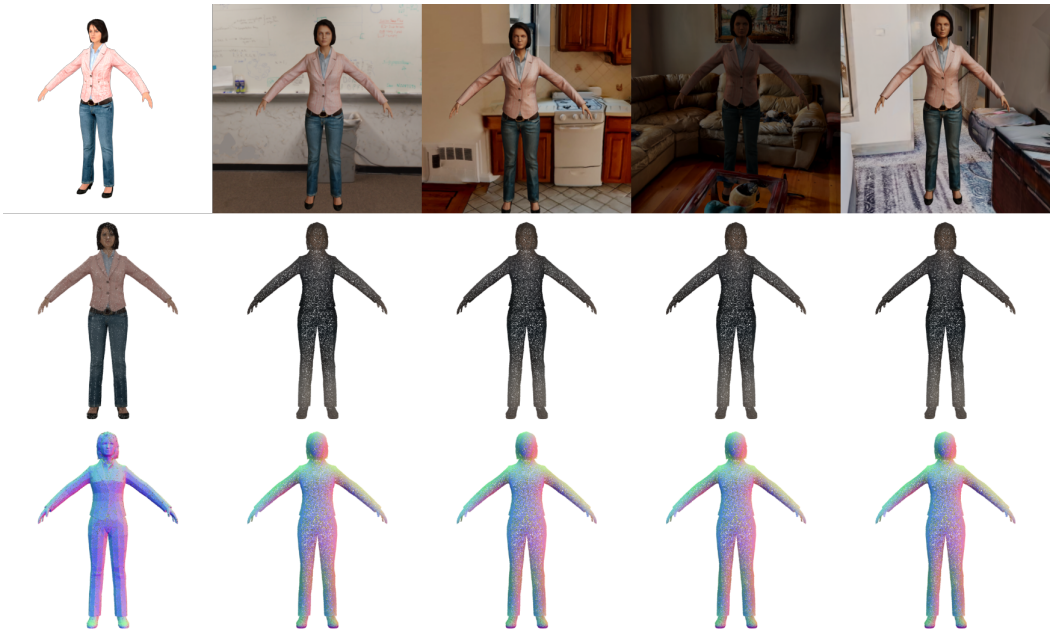


Figure 9: Qualitative results for environment variation, the first row contains the input images, the second row the true color (left) and the predicted color for surface points and, the third row the true normal (left) and the predicted normal. The predicted color and normal are not influenced by the input image.

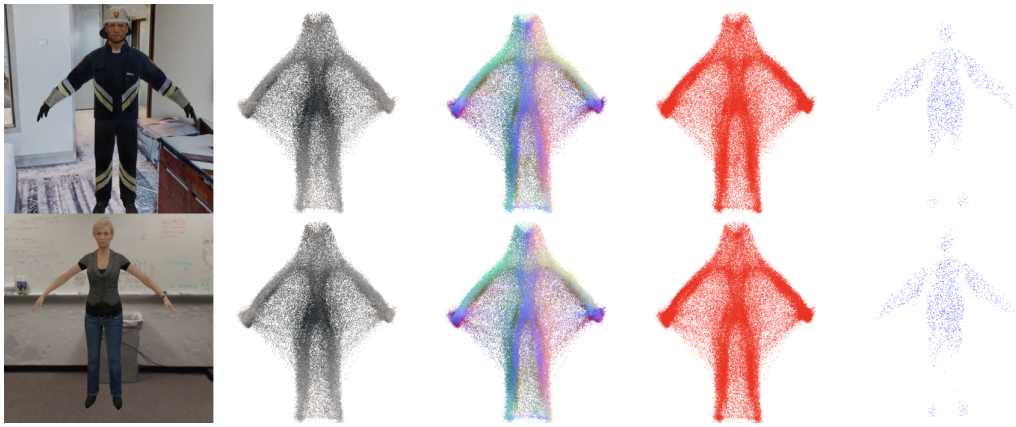


Figure 10: Qualitative results for new random uniform points with two different input images (left). The surface projection of color, normal, outside, and inside points are almost identical. The predicted normal and distance are not influenced by the image.

4.3 Visualisation

For visualization purposes, a custom real-time 3D viewer was built rendering millions of points efficiently and enabling the developer to better identify prediction errors (Fig. 11). A client-server architecture was chosen with the server running a Flask application directly interacting with the

trained Tensorflow model and communicating with a React Three Fiber (R3F) client to render the data (28; 27; 29). For the R3F client, an extra system was built to render the points efficiently with buffer geometry and shaders. The viewer has multiple modes, such as viewing the dataset, viewing predictions, and comparing the prediction against the dataset. Additionally for each mode a visualization type can be selected, similar to (Fig. 5) with color, normal, and distance for each dataset near and far.

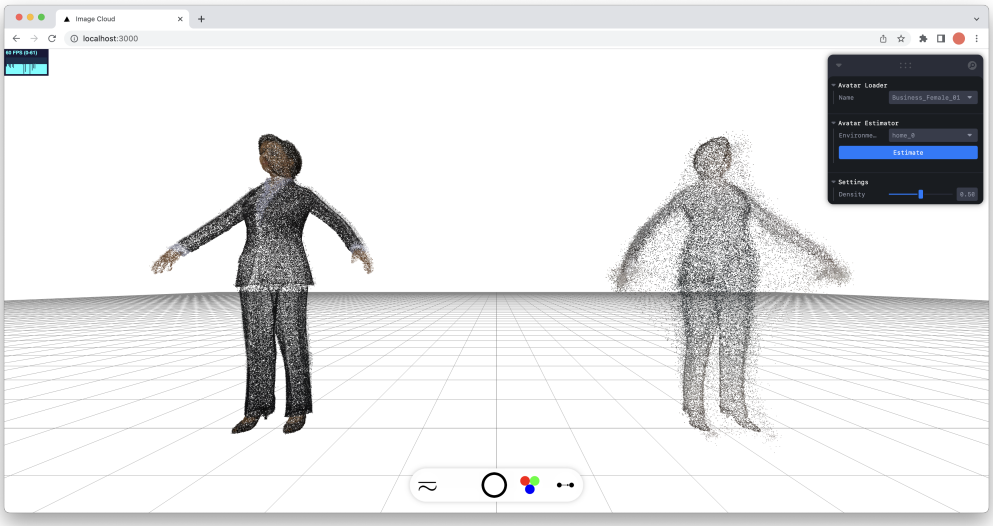


Figure 11: Visualizer, showing the true sampled points with color (left) and the estimated points with color (right).

5 Outlook

While this results show that the trained model may only be able to remember presented points and not be able to correlate the presented points with the presented images, we would like to give a future outlook on how this problems could be solved and how this work could be continued in a true open source manner. In the following we will show which parts of the pipeline could be improved.

Dataset From the start, it was clear, that we will not reach the quality of previous work since there are no datasets available. The work could be drastically improved by adding more data since our dataset is really small compared to the work of Alldieck et al. (6). One could add more scanned avatars from different sources, such as the Unreal Metahuman generator, to increase variety or increase the number of environments by preparing more scanned environments (4). The environment dataset could be also extended by adding ground projected HDRs or building upon other work e.g. radiance fields of indoor scenes (30). While the model seems to remember the points presented, using an infinite amount of points by using online point sampling could solve the problem. Also, the image augmented rendering could be part of an online mechanism rendering images on the fly.

Network The network architecture could be extended to also include the shading network of Alldieck et al. (6). Of course, new ideas such as the attention lookup could also be included in the architecture.

Training Our main problem with training was the vast amount of memory that is used by the network, one could either reduce the size of the network, which may also solve the remembering problem, or bring the architecture and dataset allocation to a better standard.

Viewer + Product For developing purposes an SDF inference renderer could be added to the viewer, to see the true underlying predicted surface and not only points that are surface projected.

Since the viewer is written with web technology it could easily be included within Tensorboard to view the SDF while training (27). The viewer could become a product by uploading it to a server and extending it to the point where a user can simply drag and drop an image of itself to generate an avatar that is also rigged and fully usable for the next XR experience.

6 Conclusion

We presented a pipeline that allows us to take a step in the direction of open-sourcing digital avatar reconstruction. We showed how to set up needed datasets to solve the task of 3D avatar reconstruction from a single image. Further, we laid the groundwork to open source a current state of the art model by implementing the architecture. The resulting trained model allows to roughly estimate the geometry of an avatar but fails to find a correlation between the image and the color. We assume that the model is remembering the presented points due to its vast structure and the very small dataset. In addition, we previewed a version of a viewer, that is built upon web technology and could easily be turned into a product. We want to state again, that this project was part of a research internship with a fixed time limit of two months. With more time named issues could be improved. We hope this work will be continued.

References

[1] A. Abbasi, "What we can learn from top brands already in the metaverse," July 2022, [Online; accessed 05-August-2022]. [Online]. Available: https://www.forbes.com/sites/forbesagencycouncil/2022/07/15/what-we-can-learn-from-top-brands-already-in-the-metaverse/?sh=5f8b561a3a70

[2] E. DeFilippis, S. M. Impink, M. Singell, J. T. Polzer, and R. Sadun, "The impact of covid-19 on digital communication patterns," *Humanities and Social Sciences Communications*, vol. 9, no. 1, p. 180, May 2022. [Online]. Available: https://doi.org/10.1057/s41599-022-01190-9

[3] S. Ma, T. Simon, J. Saragih, D. Wang, Y. Li, F. De La Torre, and Y. Sheikh, "Pixel codec avatars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 64–73.

[4] Unreal, *MetaHuman*. [Online]. Available: https://www.unrealengine.com/en-US/metahuman

[5] H. Engine, *Human Engine*. [Online]. Available: https://www.human-engine.com/

[6] T. Alldieck, M. Zanfir, and C. Sminchisescu, "Photorealistic monocular 3d reconstruction of humans wearing clothing," 2022. [Online]. Available: https://arxiv.org/abs/2204.08906

[7] S. Prokudin, M. J. Black, and J. Romero, "Smplpix: Neural avatars from 3d human models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1810–1819.

[8] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll, "Video based reconstruction of 3d people models," 2018. [Online]. Available: https://arxiv.org/abs/1803.04758

[9] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 8649–8658.

[10] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2304–2314.

[11] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020.

[12] T. He, J. Collomosse, H. Jin, and S. Soatto, "Geo-pifu: Geometry and pixel aligned implicit functions for single-view human reconstruction," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9276–9287, 2020.

[13] Z. Huang, Y. Xu, C. Lassner, H. Li, and T. Tung, "Arch: Animatable reconstruction of clothed humans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3093–3102.

[14] T. He, Y. Xu, S. Saito, S. Soatto, and T. Tung, "Arch++: Animation-ready clothed human reconstruction revisited," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 046–11 056.

[15] Renderpeople, *Human Dataset*. [Online]. Available: https://humandataset.com/

[16] P. Heaven, *Poly Heaven HDR*. [Online]. Available: https://polyhaven.com/

[17] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll, "Video based reconstruction of 3d people models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2018, pp. 8387–8397, CVPR Spotlight Paper.

[18] AXYZ-Design, *Metroploy 3D People*. [Online]. Available: https://secure.axyz-design.com/de/geschaft/kategorie/3d-people-rigged

[19] Humano, *Humano 3D People*. [Online]. Available: https://humano3d.com/

[20] A. Pumarola, J. Sanchez, G. Choi, A. Sanfeliu, and F. Moreno-Noguer, "3DPeople: Modeling the Geometry of Dressed Humans," in *International Conference in Computer Vision (ICCV)*, 2019.

[21] 3DPeople, *3D People*. [Online]. Available: https://3dpeople.com/en/

[22] M. Gonzalez Franco, E. Ofek, Y. Pan, A. Antley, A. Steed, B. Spanlang, A. Maselli, D. Banakou, N. Pelechano, S. Orts-Escolano, V. Orvalho, L. Trutoiu, M. Wojcik, M. V. Sanchez-Vives, J. Bailenson, M. Slater, and J. Lanier, "The rocketbox library and the utility of freely available rigged avatars," November 2020. [Online]. Available: https://github.com/microsoft/Microsoft-Rocketbox

[23] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[24] Sketchfab, *Sketchfab Models*. [Online]. Available: https://sketchfab.com/

[25] M. Kleineberg, *Mesh to SDF*. [Online]. Available: https://github.com/marian42/mesh_to_sdf

[26] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[27] Google, *Tensorflow*. [Online]. Available: https://www.tensorflow.org/

[28] Pallets, *Flask*. [Online]. Available: https://flask.palletsprojects.com/en/2.2.x/

[29] pmndrs, *React Three Fiber*. [Online]. Available: https://github.com/pmndrs/react-three-fiber

[30] Apple, *Learning to Generate Radiance Fields of Indoor Scenes*. [Online]. Available: https://machinelearning.apple.com/research/learning-to-generate-radiance-fields