Creating a photo-realistic Avatar using pre-trained machine learning models - decrypting AvatarGen

Lukas Mieth
Technische Universität Berlin
Berlin, Germany
lukas.mieth@campus.tuberlin. de

Deniz Mert Tecimer
Technische Universität Berlin
Berlin, Germany
tecimer@campus.tu-berlin.de

***Abstract*—Abstract**
***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The Covid-19 pandemic in 2020 and 2021 has forced a change of paradigms in regard to the impact of digital tools in work and private life. With this, it confirmed the need to develop such digital tools and offer solutions to everyday needs and works. In addition, rising developments of the Metaverse and similar kinds of virtual worlds offer a whole new field of ways within this digitalization to tackle said problems. In this context combined with the generally high need for individuality, it makes only sense to create a digitized fully modeled 3D representation to represent oneself in such meta and fully digital environments. For this project, we wanted to find a solution to the question of how can such a representation be created in a highly personalized way, based on as little input data as possible, which then can be animated and is easy to work with.

One of the most promising solutions created in the last years is the AvatarGen Model [1], which will be the main focus of this paper.

II. Introducing AvatarGen

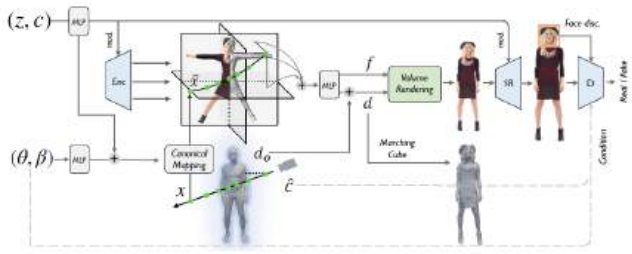

Fig. 1. Pipeline and general structure of AvatarGen Paper, taken from [1] published in November

AvatarGen [1] is a very recent paper that came into preprint in November 2022 with improvements on the first paper published in August. It promised really good results, which can be seen in figure 3, using a very intuitive and effective pipeline, which is shown in figure 2. In its core, the pipeline consists of three different elements: the *canonical human creation*, in which a human in the same canonical space is created by

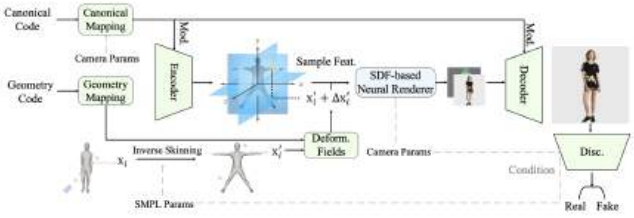

Fig. 2. Pipeline and general structure of AvatarGen Paper, taken from [1] published in August

using StyleGan latent code, might be derived from an input picture. In order to disentangle the geometry from the style, the canonical space is generated via GNARF [2]. GNARF takes a latent code and creates a canonical triplane with the canonical pose. The triplane representation has been proven to be more efficient then implicit representations like NERF [3] or voxel representations, especially in terms of its more easier scaling with the resolution, rather small storage within the code and easy feature extraction [4].

The next section is the *SMPL guided mapping*, in which the very common vertex-based skinned model for animatable human generation [5]. SMPL is used to create an animatable mesh using the pose and shape parameters of a human. The resulting SMPL model is used as observational space and a certain amount of points is sampled from there and deformed into the canonical space using linear blend skinning. The deformed points are used to query the triplane to obtain the features for the corresponding point.

The third section is the *rendering* in which the sampled features from the previous sections are fed to an MLP to get color and SDF values. SDF values are fine-tuned with SMPL guidance. The predictions are given to the volume rendering, which yields a low-resolution output image from a desired angle. Via the StyleGAN-based super-resolution module, the low-resolution image is converted to a high-resolution image.

The AvatarGen was trained using the SSHQ dataset introduced with StyleHuman [6]. However, the dataset is not publicly available and must be requested.

III. Implementation

The Pipeline of AvatarGen is end-to-end trained, meaning that any individual MLP and model were treated as a single in-

Fig. 3. End results of fully trained AvatarGen with novel poses, taken from [1]

tegrated system and their weights and parameters are adjusted in one go. Since we were heading for a rather lightweight implementation and were unable to spend the same resources as the original authors, we decided to use as many pre-trained models as possible, especially on the actual processing of the picture.

A. Canonical Human creation

As mentioned before the first step is the creation of the canonical human, which means that based on the input image the aforementioned triplane representation of the human in a canonical pose is created. The canonical pose is the standard rest-pose or T-pose widely used in the animation of human characters.

1) Image Alignment using Openpose, Paddleseg and StyleHuman: To make sure any model is able to focus on only the human on the picture and its pose the first step in our image processing pipeline is an alignment module. We implemented this referring to [6] by using a combination of the pose detection framework Openpose [7] and the image segmentation module Paddleseg [8], which is based on PaddlePaddle. While Openpose is able to detect human pose key points using a heatmap structure, Paddleseg creates individual image segments based on the labeling of the input image pixels. Both models have a pre-trained human model which they use to accurately predict the needed features. The pose parameters and segmentation values are then used to crop the human from the background and align it into the center of the image. Openpose, as well as Paddleseg are two very popular and easy-to-obtain open source frameworks, which make them very suitable for our purposes.

2) Latent Code creation using Pivotal Tuning Inversion: The next step in our image processing pipeline is the creation of StyleGan latent code, also following [6]. For this, we use a technique called Pivotal Tuning Inversion (PTI) [9]. PTI has the goal to create an easily editable latent code, which still is able to preserve the id of the input image as well as possible. For more classical approaches to creating StlyeGan latent code like [10], this has proven to be relatively hard. To solve this PTI is first inverting the input image into latent code using the shelf methods, in our case a pre-trained e4e encoder [11]. This results in an image that is similar but not exactly the same as the input image. After the inversion, the pivotal tuning is performed, in which the pre-trained StyleGan generator is lightly tuned so the input image is generated when using the latent code created by the encoder. With this PTI is able to create an for the concrete image much improved generator, which is able to create a very highly optimized latent code. This is very useful for the canonical human creation, since the quality of the latent code is crucial for any further steps

3) Triplane Creation using GNARF: As already mentioned in the introduction sections, we are using a framework called GNARF to create the canonical space. GNARF stands for Generative Neural Articulated Radiance Fields [2], which was developed to be able to create a high-quality 3D human body, which is also easily editable, using GANs. The editability of GNARfs output however is not feasible for our goal, since it is not possible to create an animatable file, like a .obj, with it, but only the pose of the created 3D human representation can be adjusted using SMPL pose parameters. It is heavily building on and improving the highly popular Neural radiance fields (NERF) [3] and is using a StyleGAN2 backbone. The full pipeline for GNARF can be seen in figure 4, however, for the purposes of recreating AvatarGen, we were only interested in the triplane creation. For us, this meant using the mapping module of the loaded pre-trained module to obtain the model's weights, but then cutting off the process after the creation of the triplane based on the latent code.

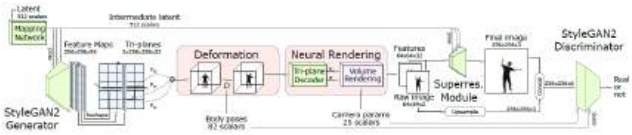

Fig. 4. Pipeline of GNARf. Taken from [2]

With this, we have created an effective three-dimensional representation of the human in the input image. It is now possible to sample from this representation and get the color and density features used to create the actual 3D mesh.

B. Observational Space and STAR Guided Mapping

In addition to the creation of the canonical human, the pipeline of AvatarGen, shown in fig 2, shows in the bottom left the observational space, including the SMPL-guided canonical mapping. Since one of our goals was to improve the AvatarGen paper, one easy improvement was to get a better model than SMPL. SMPL was published in 2015 [5] and has since been improved and enhanced multiple times, with one of the most recent ones being STAR, Sparse Trained Articulated Human Body Regressor [12]. STAR is much more sparse than SMPL, meaning there are fewer in total pose blend shapes, 207 to 93, as well as a much higher disentanglement between the

individual joints. The latter one means that the adjustment of one joint has much less impact on all other joints. If for example a joint on the arm was moved in SMPL it was offsetting the model on locations not related to this body part. The differences in corrective offset are visualized in figure 16. We, therefore, decided to use STAR instead of SMPL for the guided mapping

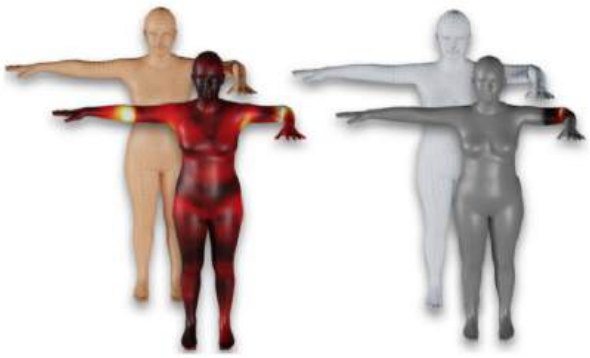

Fig. 5. SMPL Model on the left and STAR model on the right. The heatmap shows the corrective offset created by the rotation of the individual elbow joint. Taken from [12]

1) Pose Estimation using PARE: To be able to create an according SMPL model, it is necessary to obtain the pose and shape parameters from the input image. For this task [1] has been using a model named PARE, Part Attention Regressor for 3D Human Body Estimation [13]. Even though we already use Openpose for an earlier application PARE has some advancements over Openpose, especially when it comes to input pictures where some parts of the body are not visible. To come to the challenge of occlusion handling PARE uses an attention-based model, which allows it to selectively focus on the most relevant information provided in the picture, leading to an untangled model, which is able to estimate the poses of an occluded joint. Since we wanted to be able to create the photo-realistic avatar on as little input data as possible, it just makes sense to be able to get a highly accurate SMPL model based on pictures, which everyone has of themselves, which are not created in an artificial laboratory environment. Off the shelf, PARE uses YOLOv3 (You Only Look Once, version 3) as its first step to detect the Human subject on the input image [14]. However, Yolov3 has been released in 2018 and is not able to get the best possible results. A simple way to improve our results and the results of PARE, in general, would be to use the most recent YOLO version, which is version eight.

2) Transformation to STAR: As mentioned before, we wanted to use STAR instead of SMPL for the guided mapping to improve on the AvatarGen. Due to this, the next step in our image processing pipeline is the transformation of the PARE output SMPL to STAR. Within their Github repository, the authors of [12] are providing a conversion tool, which takes the SMPL vertices, combined with model configurations as input, and is then using limited BFGS algorithms to create the according to STAR parameters.

With this, we are able to transform the input image into a STAR model which can now be transformed into canonical space and used to sample information.

C. Translation to canonical space

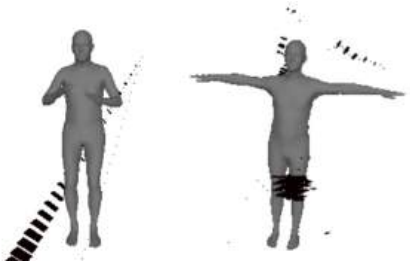

Fig. 6. Visualisation of STAR model in observational space and in canonical form, with a selection of sampled points

1) Canonical Space converter: Neither the Triplane representation nor the posed and shaped STAR model are however useful if we are unable to connect the accordingly and gather the relevant information. To do so we created our own canonical mapping class, based on the information provided in [1]. The class is initialized on the Parameters of the created STAR model and the desired camera parameters for point sampling. Based on the camera parameters, mainly the azimuth angle, elevation, and field of view, a starting point on the unit circle is created. From this starting point, a cone is created, representing rays going from the source point through the model in the direction of the coordinate center, with the field of view being the cone's angle. Based on the basis of the cone a grid of 64 times 64 points is created, which serve as the endpoints of the ray. Along the individual rays, 24 points x are sampled resulting in a final array of 98304 three-dimensional coordinates, which then can be shaped into the desired 64x64x24x3 shape. The converter class then needs to produce the basis for the signed distance field d_0. The d_0 values are calculated as the signed distance between the individual points x and the model in the observational pose. To calculate this we have been using Python trimesh library [15], to create a mesh based on the vertices and faces of the STAR model. Using the properties of the mesh the intersection point of a ray emitted from x in the direction of the nearest vertex can be calculated, as well as if x is within the model or not. Then using the information of the containment of x and the distance between x and the intersection point d_0 can be calculated among all points. The next step in the converter class is the transformation of the sampled points x into the canonical rest pose. For this linear blend skinning is used based on the skinning weights of the closest vertex, defined in [1], as

$$\mathbf{v}^* = \underset{\mathbf{v}\in\mathcal{V}}{\operatorname{argmin}} ||x_i - \mathbf{v}||_2$$

with the corresponding skinning weights $\mathbf{s}^*$. Using this we can transform any x_i to its corresponding point in canonical space

$\bar{x_i}$ with the following transformation:

$$T_{is}(\mathbf{x_i}, \mathbf{s}^*, \mathbf{p}) = \sum_j s_j \cdot (R_j \mathbf{v}^* + t_j)$$

Where $\mathbf{p}$ is the STAR model, R_j is the rotation and t_j is the translation of the models joint. Within our implementation, we are deriving the translation as the difference between the neural joint and the posed joint. For the R_j we use rotation matrices based on the Euler angles calculated between the joint in rest position and in posed position.

2) Feature Sampling: With the obtaining of d_0 transformation of all x as well as the model based on $T_{is}(\mathbf{x_i}, \mathbf{s}^*, \mathbf{p})$ the conversion to canonical space is completed and we can use the $\bar{x}$ to sample features from the triplane. To do so we first project the coordinates onto the individual planes and then use a grid sample method to get the individual feature values from the plane. The function to obtain features is then defined as in [2] as

$$F(\bar{x_i}) = F_{xy}(\bar{x_i}) + F_{yz}(\bar{x_i}) + F_{xz}(\bar{x_i})$$

Where F_{ij} is the before-mentioned projection. Figure 7 is also showing the sampling process, and that a simple neural network, only consisting of two fully connected layers is enough to extract color and density values.
The outputs of the translation to canonical space are then in the next steps used to create the volume rendering.

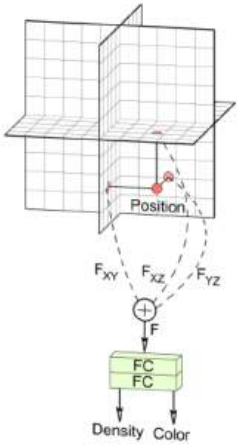

Fig. 7. Visualisation of sampling and creation of color and density values on a triplane representation taken from [4]

D. Volume Rendering using StyleSDF

1) StyleSDF: The StyleSDF gets the sampled points given the camera parameters and a latent code and produces a 3D animatable mesh and a high-resolution image taken from a desired angle [16].

In their implementation, the authors sample a latent code and use it to predict the color and SDF values with a camera in a unit sphere directed at the origin for simplicity. The point cloud is in the shape of 64x64x24x3, where 64x64 is the dimensions of the 2D feature maps created by aggregating the rays, 24 is the number of points sampled on each ray and 3 corresponds to the 3D location of the point.

By using the styles (W in the figure), the latent code of each point is predicted via the MLP network. Following that, through the fully connected layers, SDF and feature values are predicted. The marching cube can be accessed at this point. To generate an image, density values, and color information are inferred using the generated features and SDF values. By aggregating the rays with the corresponding feature maps on the volume aggregation module, the resulting low-resolution image is created and given to the StyleGAN generator to increase the resolution.

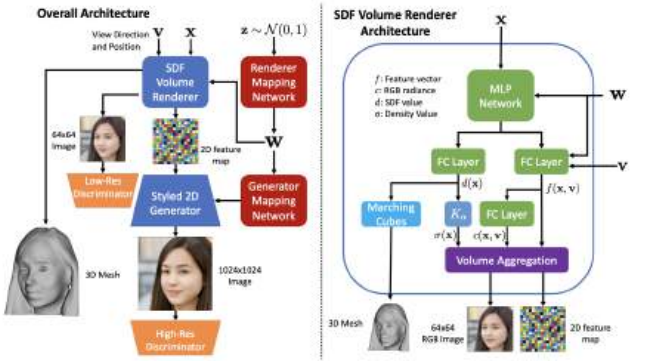

Fig. 8. Structure of StyleSDF taken from [16]

2) Adapting StyleSDF for AvatarGen: The features of each point are already created after querying the triplane. Therefore, we omit the MLP network used for that. The shape of the features of points is 64x64x12x32, where 64x64 is the feature map dimensions, 12 is the number of points sampled per ray, and 32 corresponds to the feature size. In the early release of AvatarGen, they concatenate the SDF values (coarse SDF values) sampled from the SMPL and the sampled features to obtain the residual SDF values. By summing the coarse and residual SDF, the final SDF values are calculated, which fine-tunes the predictions.

The models used so far have a pre-trained version, suitable for reuse on this task as they were trained on images containing a human body. However, the StyleSDF has pre-trained versions trained on FFHQ and AFHQ datasets, respectively.

Training the pipeline end-to-end is an intensive operation. The authors of AvatarGen denote that model converges after 4 days of training. To overcome this limitation, the pre-trained models are used and only the volume renderer and the StyleGAN generator were planned to be trained with the default parameters of StyleSDF and the camera parameters used for sampling the points.

While adapting the StyleSDF into the AvatarGen pipeline, some problems emerged with the dimensions of the model input and output used in the StyleSDF model. The difference between styles and the latent is not clearly explained and requires further and deeper investigation of the model. Debugging deep neural network applications can get really hard, especially when there are many submodels used and they

depend on each other's input and output. In the AvatarGen paper released in August, the figure below is presented. The coarse SDF values are concatenated with the sampled features and fed together for residual SDF prediction.

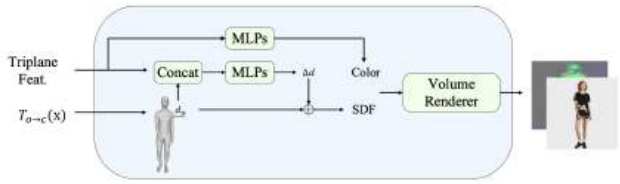

Fig. 9. Volume rendering section of AvatarGen Paper, taken from [1] published in August

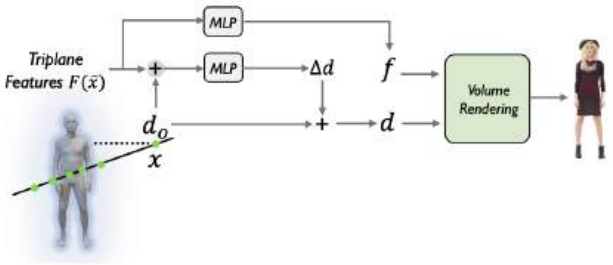

Fig. 10. Volume rendering section of AvatarGen Paper, taken from [1] published in November

Some notations and methods used in the paper released in August and November do not match. In the paper published in August plus sign corresponds to concatenation of the features and coarse SDF values and in the other one, it means aggregation via summation. Besides that, some MLPs and the face discriminator do not exist in the previous paper. The inconsistency causes confusion about reproducing their methods as we discovered the previous version of the paper recently. As mentioned before, StyleSDF gets a random style vector as noise and produces the output based on that, it samples the features using the style, given the sampling points and their directions. At this stage, we could not restructure the SDF prediction linear layer in order to perform this operation by utilizing the coarse SDF values guided by the STAR model.

Instead, the StyleSDF was trained to examine its performance on human images.

3) Training StyleSDF: The StyleSDF model was trained on the Deep-Fashion Multi-Modal Dataset [17], which contains 44,096 fashion images. Only the pictures taken from the front and containing the full body are used for training as they mention in the StyleSDF paper. To ensure consistency with the other models, the same alignment and preprocessing procedure (human segmentation, alignment, etc.) from the StyleHuman paper was used. The preprocessed images were manually examined to remove the images with distortions, extreme poses and not containing the full body from the dataset. The images were padded to the size of the bigger dimension, thus converting the image to a square format to fit the input dimensions of the StyleSDF. The final dataset contains 9244 images.

Fig. 11. Examples from Deep-Fashion Multi-Modal Dataset [17]

Fig. 12. Examples of removed images after preprocessing

Fig. 13. Examples of padded images after preprocessing

The model was trained for more than 30 hours in total with NVIDIA A100 Tensor Core GPU with 40 GB RAM. First, the volume renderer and afterward the StyleGAN decoder was trained. As the authors recommend, the training of the volume renderer was stopped when the Beta values decreased below $3 * 10^{-}3$. During training the model consumes around 25 GB for a batch size of 16 and chunk size of 8. The chunk size determines how many samples are used for training in parallel The output size was set as 512x512, the default feature map size as 64x64, and the default style dimension as 256.

IV. Results

Initially, our purpose was to compare the performance of the AvatarGen when the underlying models were pre-trained and trained separately with the performance published in the paper.

Due to the problems in integrating the StyleSDF into the pipeline and the costly training of StyleGAN Decoder in StyleSDF, we are not able to present the results for our initial goal. Instead, we tested the models individually and share the achieved results in the following sections.

A. PARE and STAR

As already discussed in section III-B1, our off-the-shelf PARE implementation uses the 5-year-old detection module YOLOv3, which is already pretty outdated. Since the attention algorithm is focusing on the area detected by YOLO, this can worsen the results of PARE significantly, even though the prediction of occluded joints, which might also be joints laying outside of the by YOLO created boundary, is still able to return some reasonable results. Figure 14 is showing an output STAR model next to the inputted Image on the right. It is clearly visible that the algorithm has been predicting the location of the lower joints around the ankles and feet, leading to the model standing more wide-legged than it actually is. On the other side, however, it did a pretty good job on the position of the hands, upper arms, and position of the head. The predicted model shape, meaning the STAR betas, is also visible on the mesh's arms. There it ignores the outline of the suit on behalf of the rather skinny predicted body parameters of the inputted image, leading to the position of the arms appearing a bit further away from the body.

Fig. 14. Created STAR Model compared to Input Image

In the end, however, our experiments on the AvatarGen implementation showed that PARE even with the outdated image detection has shown reasonable results for the needed requirements within the full pipeline, before the rendering module. The created model on its own on the other hand has shown significant improvements when YOLOv8 was used to detect the boundaries, without really making the whole model more heavy.

B. GNARF

Since there already exist pre-trained versions of GNARF on the SHHQ as in [6] dataset, which was primarily used in [1] as well as the Deep-Fashion dataset [17], which we used for training, and also in the regard of us not being able to create an output in AvatarGen, we decided to not individually train this model. With training, we would be able to get three-dimensional human representations, which then could be compared to StyleSDF volumetric rendering and our AvatarGen result. Also since in our pipeline we are only using the triplane generator, valuable information on its performance would have only been visible on the downstream tasks. To nevertheless show an output of the full GNARF pipeline, we used a pre-trained model with a random latent code to get a general output within the triplane representation, as seen in figure 15.

Fig. 15. Output of pre-trained GNARF

Since the created image needs to be decoupled from the triplane, and the latent code used to create the image, it is visibly distorted. However, the color and density values could still be obtained from the associated triplane.

C. StyleSDF Volumetric Rendering

In figure 7, two different example outputs of the volume renderer trained on the Deep-Fashion Multi-Modal Dataset are presented. The model can not segment the human body from the background. After preprocessing, the images were padded and examples of that can be viewed in figure 16. As a result of the padding the image with black pixels, which has a white background might imply to the model that the white space is not a part of the background. In the output image examples, the edge due to the transition from black to white was kept

and also projected onto the 3D space. The model can create a shade of a human body with a very smooth surface texture. StyleSDF might not have the complexity enough to model a human body from random noise as the human body has a really complex structure due to many edges and vertices.

The authors of AvatarGen also mention that the SMPL-guided SDF prediction yields much better results in comparison to other state-of-the-art models they compare in the paper. In contrast to the structure of the AvatarGen, there is no disentanglement of the style and geometry in StyleSDF while predicting the necessary features.

The dataset used for training the volume renderer contains 9244 images and the AvatarGen was trained on the SSHQ dataset containing over 44000 images. The discriminator performance has a big impact when training generative networks via adversarial training. The low amount of data might cause an underfitting discriminator, which affects the performance of the generator.

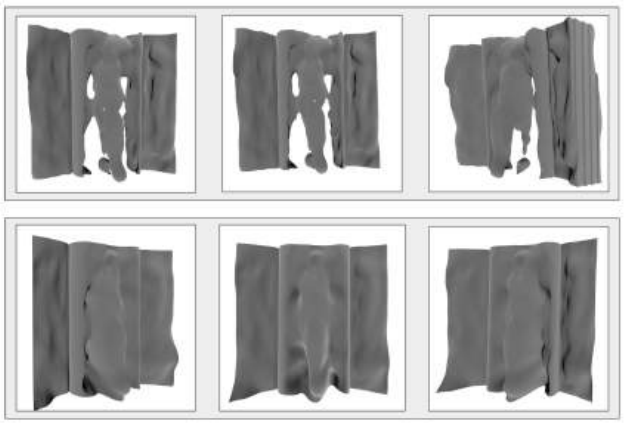

Fig. 16. Output examples of the volume renderer

Fig. 17. From top to bottom inference output of the checkpoint StyleSDF model saved on the 10000th iteration to the 70000th

After each 10000 iterations of the decoder training, a checkpoint model was saved. Their inference output can be viewed in figure 17.

During the training validation images were generated. Due to the limitations, the environment rebooted on the 78129th iteration. Thanks to the validation images saved, we were able to get the validation images generated after the 75000th iteration. The last generated validation image is shown in figure 18. Validation samples are generated for 8 identities for 8 different azimuth angles -0.35, -0.25, -0.15, -0.05, 0.05, 0.15, 0.25, and 0.35, respectively.

The challenges mentioned in the interpretation of the volume renderer results are also valid for evaluating the performance of the full pipeline (volume renderer and StyleGAN decoder together).

The model performance improved until the 60000th iteration and after that, it decreased significantly. However, based on the validation sample shown in figure 18, we can assume that the performance started to increase again.

When the training data is observed that the performance decrease occurred when the discriminator had a very low loss. Based on that, the generator loss increased and the model might have converged in a different direction.

V. Limitations

When tackling the encryption of the AvatarGen, we faced multiple limitations, of various kinds. One which was very present over the whole process was in regard of computational resources. All of the used models are bound to the usage of GPUs in their application. Within the context of this project, we were unable to access powerful machines with multiple high-level GPUs, RAM, and general high computational power. To overcome this limitation we resorted to Google's so-called Colab service [18]. Even though this service is free and publicly available, only rather slow GPUs and low-RAM runtimes are available without paying. To be able to train StyleSDF in the background and to get faster point sampling and better GNARF representation we spend an overall amount of 55€ for computational resources. As also already mentioned in the discussion of results, another limitation was model interoperability. Even though within [1] it was displayed as if the individual models are able to work together

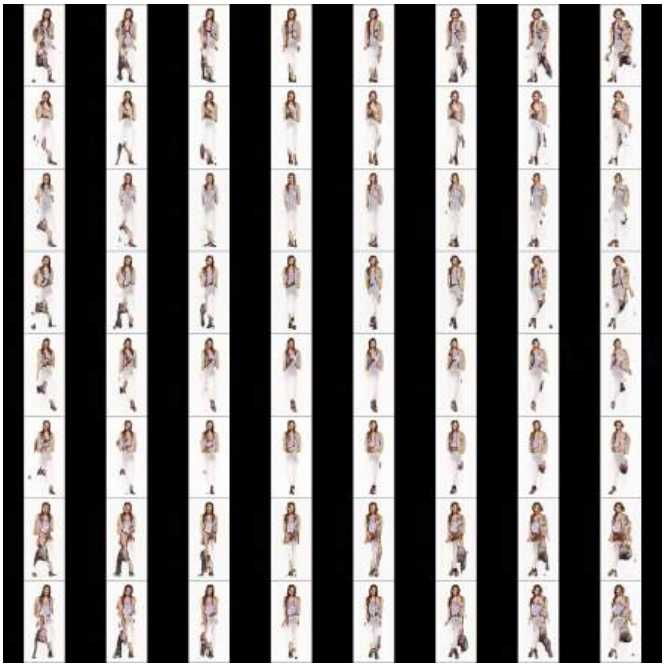

Fig. 18. Validation sample generated after the 75000th iteration

seamlessly. In reality, a lot of work went into adjusting the parameters, input-output dimensions, and most significantly the structure of the individual models, and adapting them to each other, which makes reusing pre-trained models a complex problem. Up until the point, where we were not able to feed the output of sampled points and signed distances into the training pipeline of StyleSDF, to train the model on our data. In regard to training, there were also limitations on time, which come naturally with such a project, leading us to not being able to solve the problem of connecting the individual parts of the pipeline and fully train our generative network. Another limitation for us was also the sheer amount of knowledge needed to fully grasp and completely encrypt StyleGan. The understanding of this paper requires knowledge of computer vision, generative adversarial networks, general three-dimensional machine learning, and computer graphics.

VI. Conclusion

We had the goal to create the best photorealistic animatable avatar possible with the state of machine learning as of today, which is also quick in creating the avatar, needs only one image as input data, and primarily uses pre-trained easy-to-obtain models. To achieve this goal we opted for AvatarGen as a highly promising paper, which had a pipeline we found intuitive and which was also similar to other pipelines for other models creating 3D outputs, like the already mentioned GNARF [2] or ARCH [19]. ARCH is a model for creating clothed humans in 3D space in 2020. The structural differences between the two different publications of AvatarGen and the sometimes really shallow explanation of used models and methods made the decryption of AvatarGen and the final creation of an Avatar very difficult. Nevertheless, we were able to understand and encrypt their methods and find pre-trained models with high performance which we could also utilize. Within the process of this project, we were almost able to reach the goal we set, but failed on time and the connection of the rendering module with the canonical human and mapping module. However, we came to the conclusion that with more time and resources, some further optimization, and deep diving into the individual knowledge fields and models. It is indeed possible to create photorealistic and animatable avatars in a short time, based on only one input image with the usage of off-the-shelf pre-trained models.

With this knowledge and this realization, ethical concerns may come to mind. With technology further enhancing and machine learning models getting better and better, a publicly available model with an easy-to-use interface, like ChatGPT, for photorealistic avatars could be created. This would give any individual the power to let any other person, of which they can obtain a picture, let do anything within an animated context. On the one hand, this is exciting for for example the creation of movies but could also be misused to spread fake information.

Overall we could say that we were able to encrypt the AvatarGen to almost its full extent, but failed on reproducing a possible output. AvatarGen might be a powerful tool, which is able to solve problems in the Metaverse and general applications in animation, but also opens a new field of concern if the easiness of usage and the realism of the output is further improving.

References

[1] J. Zhang, Z. Jiang, D. Yang, H. Xu, Y. Shi, G. Song, Z. Xu, X. Wang, and J. Feng, "Avatargen: A 3d generative model for animatable human avatars," in *Arxiv*, 2022.

[2] A. W. Bergman, P. Kellnhofer, W. Yifan, E. R. Chan, D. B. Lindell, and G. Wetzstein, "Generative neural articulated radiance fields," in *NeurIPS*, 2022.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[4] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, *et al.*, "Efficient geometry-aware 3d generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

[5] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, pp. 248:1–248:16, Oct. 2015.

[6] J. Fu, S. Li, Y. Jiang, K.-Y. Lin, C. Qian, C.-C. Loy, W. Wu, and Z. Liu, "Stylegan-human: A data-centric odyssey of human generation," *arXiv preprint*, vol. arXiv:2204.11823, 2022.

[7] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[8] P. Contributors, "Paddleseg, end-to-end image segmentation kit based on paddlepaddle." https://github.com/PaddlePaddle/PaddleSeg, 2019.

[9] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or, "Pivotal tuning for latent-based editing of real images," *ACM Trans. Graph.*, 2021.

[10] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4432–4441, 2019.

[11] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for stylegan image manipulation," *arXiv preprint arXiv:2102.02766*, 2021.

[12] A. A. Osman, T. Bolkart, and M. J. Black, "Star: Sparse trained articulated human body regressor," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 598–613, Springer, 2020.

[13] M. Kocabas, C.-H. P. Huang, O. Hilliges, and M. J. Black, "Pare: Part attention regressor for 3d human body estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11127–11137, 2021.

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[15] D.-H. et al., "trimesh."

[16] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman, "Stylesdf: High-resolution 3d-consistent image and geometry generation," 2022.

[17] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Alphabet Inc, "Google colaboratory."

[19] 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*, pp. 3093–3102, 2020.