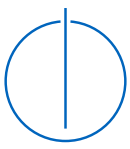
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Learning Neural Parametric Head Models with 2D Adversarial Objectives

Tathagata Bandyopadhyay



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

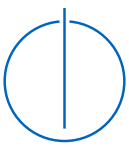
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Learning Neural Parametric Head Models with 2D Adversarial Objectives

Lernen neuronaler parametrischer Kopfmodelle mit 2D-Adversarialzielen

Author:	Tathagata Bandyopadhyay
Supervisor:	Prof. Dr. Matthias Nießner
Advisor:	Prof. Dr. Matthias Nießner
Submission Date:	15.12.2023



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.12.2023 Tathagata Bandyopadhyay

Acknowledgments

This thesis is an outcome of a long and challenging academic journey which would not have been possible without enormous assistance and endless support from many people along the way.

First, I would like to thank my supervisor Prof. Dr. Matthias Nießner for offering me this thesis and guiding me with patience and care through out the whole journey. His valuable advice and constructive feedback helped me to quickly navigate through the vast fields of face reconstruction and 3D deep learning starting from scratch.

I would also like to thank Christoph Weiler for supporting me setting up and correctly configuring the GPU servers for training heavy deep learning models. My sincere gratitude goes to Simon Giebenhain and Tobias Kirschstein for their dedicated effort to create and share the dataset with me and answering many related questions.

I am grateful to Ananta Bhattarai, Anurag Singh, Artem Sevastopolsky, Manuel Dahnert and all fellow researchers of Visual Computing Lab for time-to-time insightful discussions and practical tips.

Finally, I would like to thank my parents, friends and family for their constant motivation, inspiration and moral and emotional support along the way.

Abstract

Modeling human head is a long standing problem in computer graphics and has got many practical applications in computer games, animation and film industry. Parametric head models try to approach this problem by providing low dimensional and ideally disentangled control parameters to change the identity and expression of a human face without requiring to explicitly model each face and expression combination. While a simple PCA based parametric model can be easily obtained directly from the 3D face scans, such model is restricted to fixed template mesh topology and thus limited in capacity to represent high frequency details of any real face due to inherent linearity of the model. Recent progress in neural networks and neural implicit surface representation has enabled the computers to learn more complex parametric head models with better generalization capability. However, they still struggle to fit sparse and possibly noisy point-clouds or depth maps of real faces due to under constrained latent space.

In this thesis, we propose to learn a neural parametric head model which utilizes 2D adversarial objective on multi-view differentiable renderings of 2D normal maps along side existing 3D point-cloud based objectives to properly constrain the latent spaces, thus enabling it to fit noisy point-clouds of real face scans. Additionally, we use tri-plane based hybrid neural surface representation to leverage its 3D aware rich features for representing high-frequency details and faster convergence. We empirically show that the proposed model not only out performs most of the existing models in reconstruction and fitting, but also is more robust to scanning noise. To our knowledge, we are the first to use tri-plane based hybrid surface representation in the context of neural parametric head models.

Contents

1 Introduction

Human face or head is one of the most important aspects of human visual perception of identity and expression. Thus, in this digital age, 3D computer modeling of human head has found a wide range of interesting applications starting from computer games, animation movies, virtual reality, digital avatars and so on. Manually sculpting such digital heads requires tremendous amount of time and effort by skilled professionals and thus it is not a scalable and generalizable solution. To this end, modern 3D head scanning techniques [1] using depth cameras or 3D laser scanners have come to the rescue and thus 3D reconstruction of human heads from sparse inputs like point clouds or depth maps is the key to the scalable approach of computerized human head modeling. However, general surface reconstruction from point clouds [2, 3] without any prior is an extremely ill-posed and under constrained problem especially in the context of human face or head.

Human heads have a common structure, for instance the relative positioning of the hair, eyes, nose, mouth, ears and so on. Parametric head or face models in general leverage this idea as strong prior and conceptually model each individual face as some kind of deviation from an average face in a low dimensional parametric space. To this end, Blendshape [4, 5] and 3D Morphable Models (3DMM) [6–11] have demonstrated reasonably well performance to provide compact 3D face representation from sparse point clouds while regularizing out noise.

Most simplistically a 3DMM can be obtained using Principal Component Analysis (PCA) based low rank approximation of the underlying 3D mesh geometry over a fixed template mesh [6–8, 12]. In this approach, a template face model with fixed topology (i.e. fixed number of vertices and faces) is non-rigidly registered to each 3D scans. On these registered templates, a dimensionality reduction technique like PCA can be used to compute a 3DMM. Such PCA based models can handle noisy sparse input well due to their strong regularizing property. However, inherent linearity of such models and their reliance on a fixed topology template, makes such 3DMMs extremely dependent on the quality of scans and the quality of non-rigid registrations and ultimately inhibits them in capacity to represent high frequency surface details and diverse hair styles.

Advent of deep neural networks enabled the formulation of more complex non-linear 3DMMs [9–11, 13–16], commonly and collectively referred as Neural Parametric Face Models (NPFM), with better representation and generalization capabilities. Usually, such solutions [14–16] use an Multi Layer Perceptron (MLP) based deep implicit surface representations in the form of Signed Distance Field (SDF) [17] or Occupancy Field (OF) [18] and learn disentangled latent codes for identity (shape) and expression spaces to achieve the capability of independent control over identity and expression manipulation. To this end, some of such approaches [14, 15] model both individual identities and individual expressions as a deviation or deformation field over an average template identity and a neutral (canonical) expression respectively. Some other approaches [16, 19, 20] model the identity directly with the identity space latent codes without requiring the concept of average template identity, but model the individual expressions as a forward or backward deformation field over the neutral (canonical) expression. Some of them [20] also employ ensemble of multiple local MLPs instead of one single global MLP for the implicit surface representation.

NPFMs, as outlined above, are extremely good at representing diverse face and hair styles and high frequency surface details due to high non-linearity of the neural networks, infinite resolution surface representation capability of the implicit SDF or OF networks and no reliance on a fixed topology template mesh. However, such powerful expressive capacity comes at the cost of lower inherent regularization property as such models can easily overfit to noise present in the input point clouds or depth maps, thus producing noisy reconstructions in the inference time. To mitigate this issue, such models can be explicitly regularized by adding some normal consistency or smoothness terms in the objective function, but such explicit regularization is prone to produce over smooth results compromising on the high frequency details due to lack of adaptive discernability between surface vs noise.

Generative Adversarial Network (GAN)s [21, 22] are excellent at learning complex distributions of real data as it leverages another neural-network namely discriminator, which classifies real vs generated samples, as an implicit loss function instead of using explicit reconstruction and regularization based loss functions. Such capability of GANs can be utilized in different hybrid generative contexts even alongside explicit reconstruction losses to implicitly regularize the latent spaces to make the model immune to input noise without compromising with high frequency details in the data.

In this thesis, we propose a parametric head model which uses 2D GAN loss on the multi-view differentiable renderings of normal maps along side the existing 3D

reconstruction losses to constrain the latent spaces to make the model robust to noisy inputs. We leverage a tri-plane based powerful hybrid surface representation [23, 24] to model the human heads and use a forward deformation field over the open mouth neutral (canonical) expression to model the individual expressions. To this end, a Convolutional Neural Network (CNN) is used to generate the tri-planes from the shape codes and then a small MLP decoder is used to obtain the SDF values of every (x, y, z) locations of the volume from the tri-plane features. Thus, the identity model along with the identity codes (shape codes) are jointly trained in auto-decoder [17] fashion with the 3D reconstruction loss on predicted SDF values and predicted surface normals of the training point clouds along with the 2D GAN loss on the multi-view volumetric renderings of normal maps of the hybrid surface representation. To learn the expression model along with the expression latent codes, another MLP is used to capture the deviation deltas of the surface points from the canonical expression to target expression and trained in auto-decoder [17] fashion with 3D dense correspondence loss. In summary, the main contributions of this thesis are:

- We propose a neural parametric head model using 2D adversarial loss to implicitly regularize the latent spaces.
- To our knowledge, we are the first to utilize tri-plane hybrid representation in neural parametric head models.
- We develop a multi-stage training strategy to effectively use the 2D adversarial loss along with the 3D reconstruction losses without leading to instability in training.

Rest of this thesis is structured as follows: Chapter 2 discusses related and existing works. Chapter 3 covers some theoretical backgrounds of some concepts used in this thesis. In Chapter 4 we discuss the proposed framework in detail. Then, Chapter 5 covers different experimental setups followed by the results in Chapter 6. Finally, we conclude in the Chapter 7.

2 Related Work

In this chapter we will discuss and review existing works relevant to different parts of this thesis project. In particular, we will talk about 3D Morphable Models, Neural Field Representation, Differentiable rendering and Generative Adversarial Networks.

2.1 Explicit 3D Morphable Models

2.1.1 Linear Models

First model-based approach to capture variations in human faces was proposed by Blanz and Vetter [6] in 1999. They used PCA to construct a low rank approximation of underlying mesh geometry of 200 head scans. As those scans lacked variation due to controlled scanning environment, this model was limited in expressiveness. This model was improved in [7] by using a better non-rigid registration technique. Further improvements were done by capturing faces in the wild to add more variation in the data [8, 25, 26]. To improve the disentanglement between identity and expression space, multi-linear models were proposed [12, 27]. Li et al. [28] used articulated jaw, neck, eyeballs and pose-dependent corrective blend shapes to bridge the gap between high-end and low-end face models. Localized deformation model was proposed in [29].

2.1.2 Neural Non-Linear Models

Face models discussed so far were all linear. With increasing success of deep learning, deep non-linear face models were also proposed. Tran et al. [11] used CNN and GAN based multi-view reconstruction approach to learn non-linear face models from images. A number of methods followed convolution based approaches [10, 30–32]. Use of attention [9] and graph convolutions [13, 33] were also proposed in this context.

2.2 Neural Field Representation

3DMMs discussed above (sec. 2.1) used some form of explicit representation for 3D shapes in learning face models and thus limited by their shortcomings (see sec. 3.6.1). Recently, neural field based implicit representations (see sec. 3.6.2) have become quite

popular choice for 3D deep learning due to their efficiency and capacity to represent high level of details. There are mainly three fundamental lines of work have been done in implicit representations as follows:

2.2.1 Occupancy Field

Traditionally, in a voxelized 3D representation, each voxel is marked as either occupied or not occupied. Mescheder et al. [18] extended this idea to use a neural network (MLP) to predict whether a queried (x, y, z) point is inside (occupied) or outside (not-occupied) of a shape. Thus, they represent the shape implicitly with the decision boundary of the binary classifier network over the occupancy field. As the decision boundary is continuous, the 3D scene can be represented and queried at infinite resolution. Iso-surface extraction (see sec. 3.6.4) algorithms like marching cubes [34] can be used to extract a mesh at a desired resolution. Peng et al. [24] later proposed to use convolution operation in the occupancy network to represent larger scenes with the translation invariant property of convolution filters.

2.2.2 Signed Distance Field

Signed distance fields are wildly used in 3D scene representation in computer graphics due to its interesting properties. Quite similar to occupancy network [18], Park et al. [17] proposed to use a MLP to predict the SDF value at a queried (x, y, z) location i.e the network will tell what is the distance to the nearest surface from the queried point. The authors showed that such network can be conditioned with some latent code to represent different class of shapes. This idea is widely used in many followup works (see sec. 2.3) and we also use the similar idea (with a little different network architecture) to represent the 3D faces.

2.2.3 Neural Radiance Field

Although not directly related to this thesis, but one of the most influential inventions in the recent history of neural field representation is Neural Radiance Field (NeRF) [35]. Here, the idea is to overfit a MLP to a 3D scene to predict the RGB color (r, g, b) and volume density (σ) of a queried point (x, y, z) along a queried viewing direction (θ, ϕ). This MLP is trained using a photometric loss on the rendered images obtained from different camera poses. NeRFs can produce visually appealing photo-realistic novel views of a scene with view dependent effects. Many followup works have been done in last few years improving different aspects of NeRF. Zhang et al. [36] proposes to use separate foreground and background sampling to represent unbounded outdoor

scenes. Martin-Brualla et al. [37] uses unconstrained 'in the wild' images to train NeRF. In [38], NeRF is extended to represent dynamic scenes. KiloNeRF [39] speeds up the inference time by using ensemble of many small MLPs. In a seminal work Müller et al. [40] proposed to train NeRF in seconds using multi-resolution hash encoding where original NeRF usually takes hours or days to train.

The only relation to NeRF with this thesis is that we use some of the volume rendering formulations (see sec. 3.7.2) proposed in vanilla NeRF [35] for rendering normal maps from our hybrid SDF shape representation.

2.3 Implicit 3D Morphable Models

With the advent of neural field based implicit representations (see sec. 2.2 & sec. 3.6.2), many neural implicit 3DMMs have been proposed. Such approaches usually use two networks: one to represent the shape in canonical pose, another to model the complex deformations due to change in expression or pose. SDF based parametric models for human bodies have been proposed in [19, 41, 42]. In [43, 44], implicit generative head models are proposed, but they do not consider to model expressions. Yenamandra et al. [14] models both identity and expression as a deformation over a reference shape in neutral expression implicitly represented as a SDF. ImFace [15] extended the idea by using pseudo SDF for water-tightness and facial keypoint based localized approach to capture local surface details. Zheng et al. [16] uses video based correspondences to reconstruct an implicit head morphable model. NPHM [20] improves over ImFace [15] by using higher quality 3D scans and big ensemble of local MLPs. Amongst many of such recent implicit parametric models, two are most relevant to this thesis and thus we have elaborated more on those as follows:

2.3.1 NPMs

Although, originally introduced [19] for full body parametric model, this forms the baseline of our parametric head model. It represents the shape with DeepSDF [17] based implicit SDF. Shape codes are learnt jointly along with the SDF shape network.

Pose is modeled as forward deformation over canonical T-pose and pose codes are jointly learned with pose deformation network with dense correspondence.

Although, this thesis is mostly based on NPMs [19], there are few fundamental differences: 1) We use tri-plane based hybrid representation instead of DeepSDF like representation, 2) We additionally use 2D GAN loss on the multi-view normal renderings.

2.3.2 NPHM

NPHM [20] is one of the state-of-the-arts and also same data set is used in this thesis. NPHM is also inspired by NPMs [19], but instead of DeepSDF [17] based implicit representation, it uses ensemble of local MLPs centered around facial keypoints or *anchors* to capture more local details and use weight sharing between some of the local MLPs to exploit the symmetry based geometric priors.

Once again, the main differences of NPHM [20] to our approach are: 1) We try to take care of local details using tri-plane based representation instead of ensemble of MLPs, 2) We use 2D adversarial loss on the 2D rendering of the normal maps.

2.4 Differentiable Rendering

Differentiable rendering is at the core of any formulation that uses a 2D objective function for reconstructing or manipulating 3D shapes. Although there are numerous amounts of work done in this domain, we will mostly keep the discussion limited to image order differentiable rendering that can be used for implicit functions. Groce et al. [45] used model based 2D objective for 3D hand pose estimation from videos. Li et al. [46] introduced differentiable monte carlo ray tracing. Müller et al. [47] introduced neural importance sampling to efficiently sample the volume while ray marching. This concept was later used in NeRF [35]. DIST [48] proposed differentiable sphere tracing with gradient approximation for shape fitting with pre-trained DeepSDF [17] based implicit representation. However, this doesn't work for training the network from scratch (see sec. 3.7.1). VolSDF [49] and NeuS [50] were able to successfully use differentiable volume rendering to reconstruct (train) SDF based neural implicit surfaces. They used two different (Laplace and Logistic) density formulations, to convert the SDF values to volume densities required for volume rendering. VolSDF also used errorbound sampling instead of importance sampling to complement their density formulation. Many followup works have been proposed mostly either improving the surface geometry [51–53] or reducing the training time [40, 54] or both [55]. For the sake of simplicity, in this thesis we have followed MonoSDF [53] volume rendering approach (see sec. 3.7.2) which intern is mostly inspired by VolSDF [49] formulation. For a comprehensive review on differentiable or neural rendering please refer to the recent surveys [56–58].

2.5 Generative Adversarial Networks

GANs, first introduced in [21], have taken the generative modeling to another level in recent history. In last few years, numerous amount of GAN papers have been proposed in different domains. However, in the context of this thesis we will restrict our discussion on GANs in the domain of computer vision.

2D GANs

Original GAN paper [21], used MLP based generator and discriminator (see sec. 3.4) to generate images of hand written digits. DCGAN [59] used strided convolutions in generator and convolutions in discriminator to stabilize training for unsupervised image generation. WGAN [60] and WGAN-GP [61] used 1-Wasserstein distance instead of Jensen-Shanon divergence to alleviate vanishing gradient problem and thus improving training stability even further. PGGAN [62] used *progressive growing* strategy to train GANs with gradually increasing image resolutions. BigGAN [63] proposed scalability and stability for high resolution image generation. StyleGAN [64] and its followups [65–67] proposed highly optimized GAN architectures for human face image generation with controllable styles and facial features. Pix2pix [68] used conditional GAN to translate images from one domain to another domain and showed the idea of using GAN loss along with other reconstruction loss(es). CycleGAN [69] extended this concept by relaxing the requirement of pair-wise training images by using *cycle consistency* loss.

3D or 3D aware GANs

GANs have been explored in 3D generative modelling as well. PlatonicGAN [70] used 3D generator to generate 3D shapes in voxel representation with 2D discriminator on the rendered images. HoloGAN [71] improves on it using 3D convolution based intermediate 3D feature representation. GRAF [72] and GIRAFFE [73] uses radiance field formulation with 2D discriminator to learn 3D representation. π-GAN [74] used similar approach for 3D aware face image synthesis. EG3D [23] proposed a novel *triplane* based hybrid representation to improve the reconstructed 3D geometry. In this thesis we have used the concept of *triplane* from EG3D and discriminator from π-GAN.

3 Theoretical Background

In this chapter, we will give short overview of different key concepts used in this thesis to provide a basic pre-requisite understanding. In particular, we will touch upon different types of deep neural networks, concepts of 3D surface representation, differentiable rendering and generative adversarial networks.

3.1 Deep Neural Networks

Deep Neural Network (DNN)s are the core of any deep learning applications. At a very high level, they can be thought of a set of matrices sandwiched together with some kind of non-linear functions in between to loosely mimic the concept of biological neural networks present in human brain. Mathematically, DNNs can be considered as universal function approximators due to their ability to learn any complex non-linear mappings reasonably well between some inputs and outputs. Depending on the low level fundamental architecture, DNNs can be of many types, out of which we would discuss only about MLP and CNN in relevance to the context of this thesis.

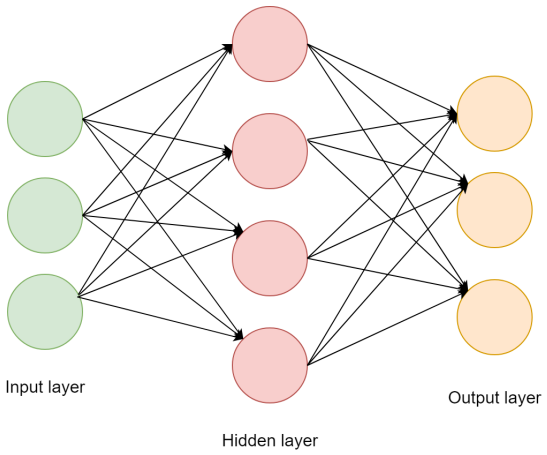


Figure 3.1: A MLP with single hidden layer

3.1.1 MLP

Multi Layer Perceptron (MLP), also known as fully connected networks, refers to a DNN where each node of a layer is connected to all the nodes of next layer. Typically, this architecture has an input set of neurons followed by one or more fully connected hidden layers followed by a set of output neurons (see Fig 3.1). The operation of a fully connected layer on input vector $\mathbf{x}$ can be mathematically described as:

$$f(\mathbf{x}) = w_0 + \sum_{n=1}^{N} w_n x_n \tag{3.1}$$

where $w_0, w_1, ..., w_N$ denotes the weights of a single fully connected layer. Output of this operation is usually passed through some non-linear activation function before passing it to the next layer of neurons to add non-linear expressive power to the network.

3.1.2 CNN

MLPs are not efficient when it comes to recognize some repeated or translation invariant structure, say from images, due to lack of built-in concept of weight sharing. CNNs address this issue by implementing small learnable filter kernels which slide over the images to produce feature maps (filter responses). For each position of the filter kernels, convolution operation is performed by taking dot product between the filter weights and the small portion of the input data (see Fig 3.2). Please note, if we don't use padding, the convolution operation shrink the spatial dimension, thus it can be applied to gradually compress data in spatial dimension even without using pooling layer.

Transposed Convolution

As we just observed, regular convolution operation shrinks the spatial input size by using filter kernels. Transposed convolutions, on the contrary, broadcasts input elements through the kernels and thus expand the spatial size of the input. This is usually used in image generator or decoder modules. In the context of this thesis we have used it in the tri-plane generator along with regular convolutions.

3.2 Auto-encoder

This a special type of neural network consisting of two modules namely encoder and decoder. Encoder gradually compress the high dimensional input into a low dimensional latent code (bottle neck). Decoder use this low dimensional latent code and try to reconstruct the input by gradually expanding it (See Fig 3.3). Thus, jointly

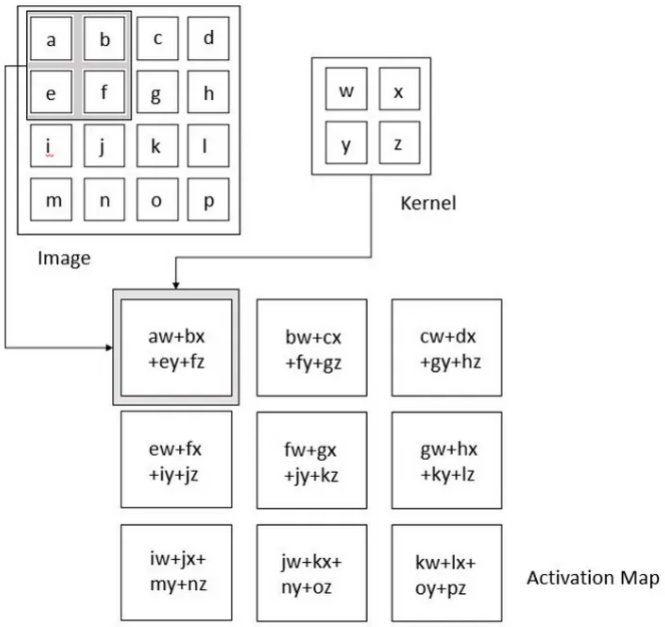


Figure 3.2: Visualization of 2D convolution operation. Image taken from [75]

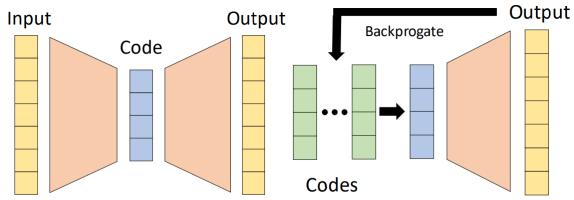
training both encoder and decoder with input reconstruction loss, compact latent representations of the inputs can be learnt. However, once such model is trained, many a times the encoder becomes useless. To address this issue of resource waste, researchers came up with an encoder free version of representation learning known as auto-decoder as discussed next.

3.3 Auto-decoder

This is an encoder free version of representation learning introduced in [17]. Here, instead of generating the latent code from encoder, it is randomly initialized from standard Gaussian distribution and jointly trained along with the decoder. Fig 3.3 shows a conceptual comparison between auto-encoder and auto-decoder. We employ this concept in training both of our identity and expression network.

3.4 Generative Adversarial Networks

GANs [21, 22] are one of the most successful generative models in recent history of deep learning. Traditionally, GANs consist of two neural network modules namely



(a) Auto-encoder **(b)** Auto-decoder

Figure 3.3: Conceptual comparison of auto-encoder with auto-decoder. In auto-encoder latent code is generated by encoder module, where as in auto-decoder the latent code is randomly initialized and then jointly trained with the decoder module. Image taken from [17]

generator (G) and discriminator (D). G generates image (or some data) from noise or latent codes and D tries to identify whether the sample is real or fake (see Fig 3.4). They are trained jointly in an adversarial mini-max game with D having the goal of correctly distinguishing real vs generated samples and G having the goal of fooling D. Mathematically, this can be formulated as:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))] \tag{3.2}$$

where x denotes the real samples where as z denotes the noise or latent code. In practice, the second term is implemented as $\mathbb{E}_{z \sim p_z(z)}[\log D(G(z))]$ with label flipping, to get better behaved gradients.

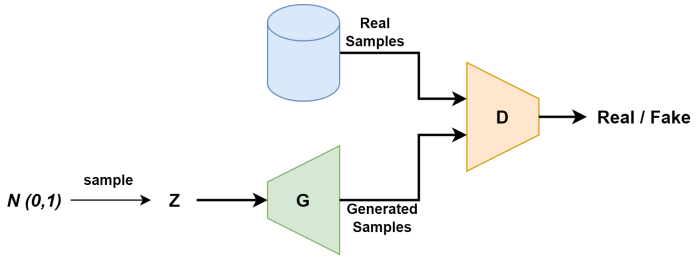


Figure 3.4: Overview of a GAN. Generator (G) generates fake samples from a randomly sampled latent code from standard Gaussian distribution. Discriminator (D) tries to distinguish between real and fake samples.

Wasserstein GAN

Under some generic assumptions on training procedure, minimizing Eq 3.2 becomes asymptotically equivalent to minimizing Jensen-Shanon (JS) divergence between real

data distribution and generated sample distribution. However, in initial stages of training this JS divergence can be undefined (or simply infinite) due to no overlap between the two distributions, thus providing bad or vanishing gradients. This issue is mitigated using 1-Wasserstein distance [60] (or Earth Mover Distance) instead of JS divergence. The modified objective, known as Wasserstein GAN (WGAN), can be formulated as follows:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))] \tag{3.3}$$

Wasserstein GAN with Gradient Penalty

The above formulation (Eq 3.3), needs the discriminator to be 1-Lipschitz. Originally it was achieved using weight clipping [60], but later it was found penalizing the discriminator gradient [61] is a better approach to produce over all well behaved gradients. This formulation, known as WGAN-GP, is used in our discriminator loss.

3.5 Signed Distance Fields

Signed Distance Field, in short SDF, is one of the most crucial concepts of this thesis as we represent the shapes with SDFs using a hybrid representation as described in 3.6.3. Theoretically, SDF value at a point **p** in a signed distance field refers to the distance of the nearest surface from **p**.

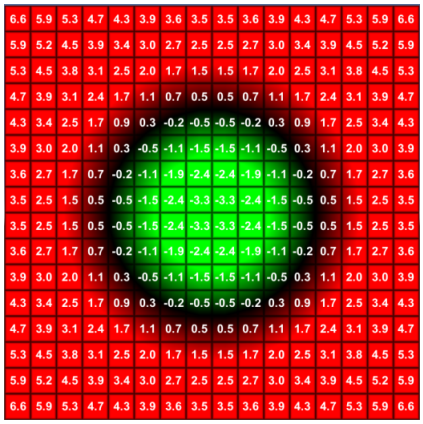


Figure 3.5: 2D SDF visualization of a circle. Image taken from [76]

Now, if the underlying surface is closed or *watertight* (i.e. it has a well defined inside) then a negative SDF value implies **p** is inside the shape, positive SDF value implies **p** is

outside the shape and the value is 0 exactly on the surface. However, if the underlying surface is not watertight, then positive SDFs refers to free spaces, but negative SDFs represents unknown values [77]. Fig 3.5 shows a 2D SDF field of a circle.

3.6 3D Representation

A 2D digital image is almost ubiquitously represented as a 2D (grayscale) or 3D (color) tensors of pixel values. However, when it comes to 3D shapes, there exists a multitude of digital representations each with their own pros and cons. Fig 3.6 shows different 3D representations.

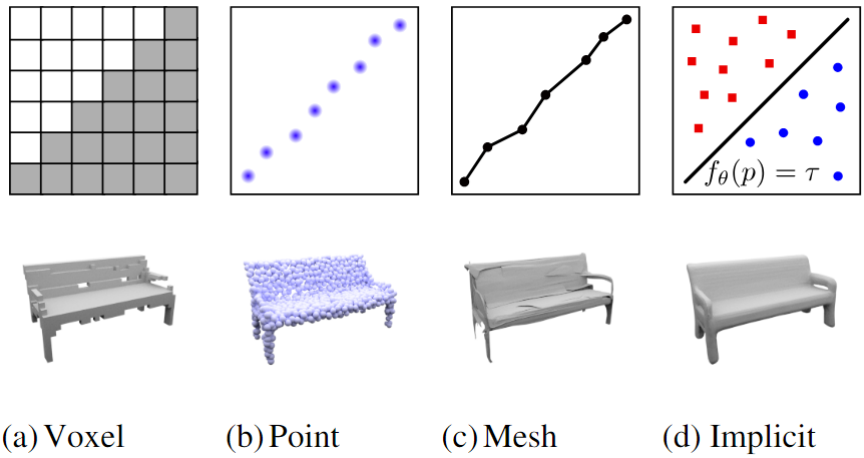


Figure 3.6: Different representations of a 3D surface along with their 2D analogues. Image taken from [18]

3.6.1 Explicit

Such representations explicitly store some form of information of the 3D shape into some data structure. Based on the data structure used and information stored, such representations can be of following types:

Voxel

This is direct 3D extension of the 2D pixel concept. This represents a volume with a 3D grid of 3D unit volumes (cubes). In the simplest variant each voxel stores binary occupancy value thus representing the 3D shape (or scene) with a dense n^3 occupancy

grid where n denotes the grid resolution. This can represent arbitrary topology, but it is highly limited by smaller grid resolution as memory requirement grows in $O(n^3)$.

Point

This represents a 3D shape with a set of 3D point co-ordinates sampled from the surface of the shape. This is efficient in memory and can represent arbitrary shapes, but it doesn't contain any information of the shape topology and thus it is quite challenging to extract accurate surface from such representation.

Mesh

This represents a 3D shape with a bunch of connected triangles (or polygons in general). It is memory efficient and ready-to-use for existing computer graphics pipelines. However, it can't represent variable topology and thus difficult to be used in deep generative modeling.

3.6.2 Implicit

This is one of the most suited representation for deep generative modeling as it employs a neural network to represent an arbitrary 3D shape as the continuous decision boundary of a classifier or regressor. When queried with a (x, y, z) 3D co-ordinates such networks outputs the SDF or OF values and thus the shape is represented as a particular Iso-Surface of the underlying volumetric field. As this approach doesn't discretize the volume, it can be queried at infinite resolution.

3.6.3 Hybrid

This method utilizes concepts of voxel as well as implicit representation to combine the best of both the worlds. There can be many different hybrid representations, but in the context of this thesis we consider *tri-plane* representation introduced in [23]. In tri-plane representation, three orthogonal feature planes are used to represent 3D aware rich features along three canonical axis aligned planes. With a (x, y, z) co-ordinate input, first the point is projected to three orthogonal planes and tri-plane features are computed by interpolating over the corresponding feature planes. Then the implicit network is queried with extracted and aggregated tri-plane features to obtain SDF or color and density values. Fig 3.7 shows tri-plane representation.

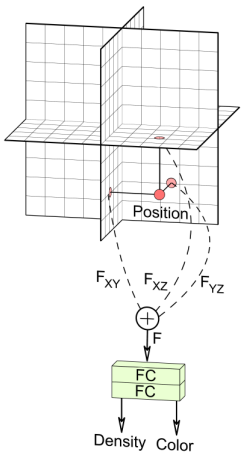


Figure 3.7: Tri-plane representation. A 3D input point is first projected to three orthogonal planes to extract axis aligned features namely F_{xy}, F_{xz} and F_{yz} using *bilinear* interpolation. Then such features are aggregated by summation. Finally, the network is queried with the aggregated features to get the SDF (not shown) or color and density values of the corresponding input point. Image taken from [23]

3.6.4 Iso-Surface Extraction

Implicit and hybrid (tri-plane) representations do not directly store the shape surface information, instead represent surface as a particular iso-value level-set of the volumetric field. Therefore, special algorithm is required to extract the iso-surface. To this end, we use popular Marching Cubes algorithm introduced in [34]. At a high level, this algorithm individually carves out the triangles from each voxels of a volumetric grid by comparing the eight corners with the predefined iso-value (0 for SDF and 0.5 for OF) and thus classifying each voxels in one of fifteen patterns as depicted in Fig 3.8. These individual triangles are finally unified to obtain the full triangle mesh representation of the shape.

3.7 Differentiable Rendering

In computer graphics, rendering in general refers to the process of generating 2D views of a 3D scene with a camera model. Differentiable rendering algorithms can back propagate gradients from 2D to 3D such that 2D losses on the views can be used for 3D reconstruction.

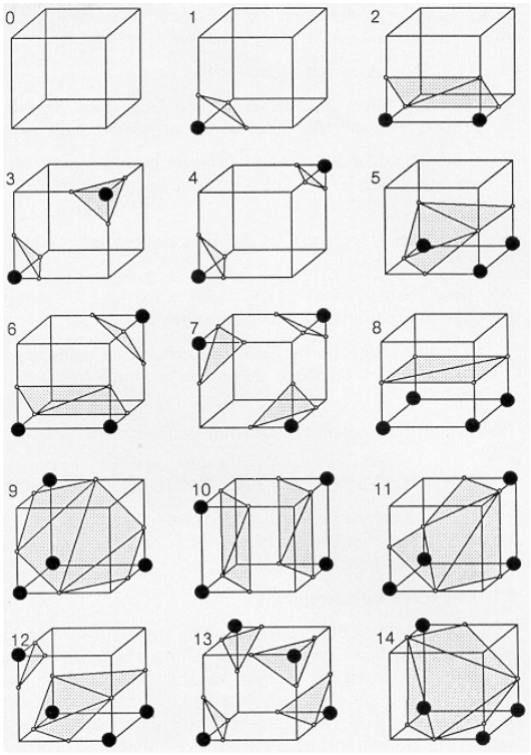


Figure 3.8: Fifteen unique voxel patterns for marching cubes algorithm. Bold black corners represents values less than predefined iso-value, there by denoting the points inside the 3D surface. Such triangles from each voxels are finally unified to extract the full triangle mesh. Image taken from [34]

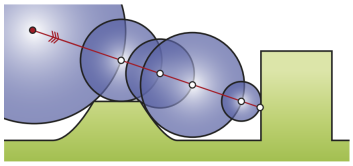


Figure 3.9: Illustration of the classical sphere tracing algorithm: The intersection point with a surface is determined by traversing along the ray, using the distance to the closest surface at each iteration step until the distance is below a threshold. Green represents the surface, blue spheres represent SDF values encoded in their radii. Image taken from [78]

If the rendering algorithm starts with the 3D scene and tries to figure out iteratively which objects would be visible in a particular camera view, such rendering algorithms are known as *object order* rendering. On the other hand, if the algorithm starts from the 2D pixel grid and shoots ray in the scene, such algorithms are known as *image order*image order rendering. There exists a multitude of differentiable rendering algorithms, thus discussing all of them is out of scope. Below we briefly discuss two image order rendering algorithms in the context of rendering normal maps relevant to this project.

3.7.1 Sphere Tracing

This is an image order surface rendering technique applicable specifically to SDFs. Here we shoot rays from camera to the scene through the image pixels and iteratively march into the 3D scene to get the surface interaction. Now, as the SDF value at any (x, y, z) coordinate tells us the distance of the closest surface from that point, at each ray marching step the ray can safely march a max distance provided by the current SDF value without hitting any surface. Thus actual ray marching can be significantly accelerated to quickly find the ray intersections. Once the ray intersection is obtained, we can calculate the surface normals (which are none other than the normalized gradients of the SDF at those points) at the intersection points either by automatic differentiation or finite difference methods. Fig 3.9 illustrates the concept of sphere tracing.

This works great with correct SDF values, i.e. in our context with already trained SDF networks. However, when the model is not trained and SDF values are mere random guess, this algorithm not only just fails to render normal maps, but also produces random gradients in the back propagation and there by doesn't help in analysis by synthesis mechanism of 3D reconstruction. Hence, we use volume rendering as discussed next.

3.7.2 Volume Rendering

This is also an image order rendering technique which shoots ray from camera to the scene through each pixel of the 2D image grid. However, unlike sphere tracing it doesn't explicitly stop at ray intersection and instead continues to go through the scene and sample density and color (or normals) values. While integrating along the ray weights are adjusted such a way that density or the opacity becomes maximum at the first hit. In other words if we have the normal n and density σ in continuous co-ordinates, we can render the normal $N(r)$ of the camera ray $r(t) = o + td$ with near and far bounds t_n and t_f using a volume rendering technique:

$$N(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))n(r(t), d)\, dt \tag{3.4}$$

where $T(t)$ is given by:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\, ds\right) \tag{3.5}$$

SDF to density conversion

However, in our context the model outputs SDF values and not a σ density. So, we need a mechanism to convert the SDF to density values. To this end, we use similar formulation as of VolSDF [49] and MonoSDF [53] to transform SDF values s to corresponding density values σ as follows:

$$\sigma_\beta(s) = \begin{cases} \frac{1}{2\beta} \exp\left(\frac{s}{\beta}\right) & \text{if } s \leq 0 \\ \frac{1}{\beta}\left(1 - \frac{1}{2}\exp\left(-\frac{s}{\beta}\right)\right) & \text{if } s > 0 \end{cases}, \tag{3.6}$$

where β is a learnable parameter. In practice, we finally compute the normals $\hat{N}(r)$) of the surface intersecting the current ray r using the discretized numerical integration as proposed in NeRF [35] as follows:

$$\hat{N}(\mathbf{r}) = \sum_{i=1}^{M} T_{\mathbf{r}}^i \alpha_{\mathbf{r}}^i \hat{\mathbf{n}}_{\mathbf{r}}^i \qquad T_{\mathbf{r}}^i = \prod_{j=1}^{i-1}\left(1 - \alpha_{\mathbf{r}}^j\right) \qquad \alpha_{\mathbf{r}}^i = 1 - \exp\left(-\sigma_{\mathbf{r}}^i \delta_{\mathbf{r}}^i\right) \quad , \tag{3.7}$$

where $T_{\mathbf{r}}^i$ and $\alpha_{\mathbf{r}}^i$ denote the transmittance and alpha value of sample point i along ray r, respectively, and $\delta_{\mathbf{r}}^i$ is the distance between neighboring sample points.

4 Method

In this chapter we will discuss in detail the network architectures and design of multiple objective functions to train the networks.

4.1 Overview

We implicitly represent a face identity in open mouth neutral expression as a SDF modeled as a neural network conditioned with a latent code for identity. We use another neural network as a forward deformation field over the neutral expression to model the expression space. This deformation network is conditioned with two separate latent codes for identities and expressions. We will discuss each of these components in detail in the following sections.

4.2 Modeling Identity

We propose to use a tri-plane representation [23] based implicit network and a 2D adversarial loss along with existing 3D losses [19, 20] to model the face identity. Fig 4.1 shows a schematic of the proposed identity model which is discussed in detail in the following sub sections.

4.2.1 Input and Output

We model face identity (face in open mouth neutral expression) with an implicit neural network $\mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id}) : \mathbb{R}^3 \times \mathbb{R}^{d_{id}} \rightarrow \mathbb{R}$, which takes a 3D co-ordinate $\mathbf{x} \in \mathbb{R}^3$ along with a latent code $\mathbf{z}_{id} \in \mathbb{R}^{d_{id}}$ and predicts corresponding SDF value $\hat{s} \in \mathbb{R}$. Identity latent code $\mathbf{z}_{id}$ is initialized from a standard normal distribution and learnt in an auto-decoder (see sec. 3.3) fashion.

4.2.2 Model Architecture

For modelling identity we use tri-plane representation (see sec. 3.6.3), so the identity network $\mathcal{F}_{\mathbf{id}}$ is internally composed of two modules namely *tri-plane generator*, which generates the tri-planes from latent codes and *tri-plane decoder*, which produces the

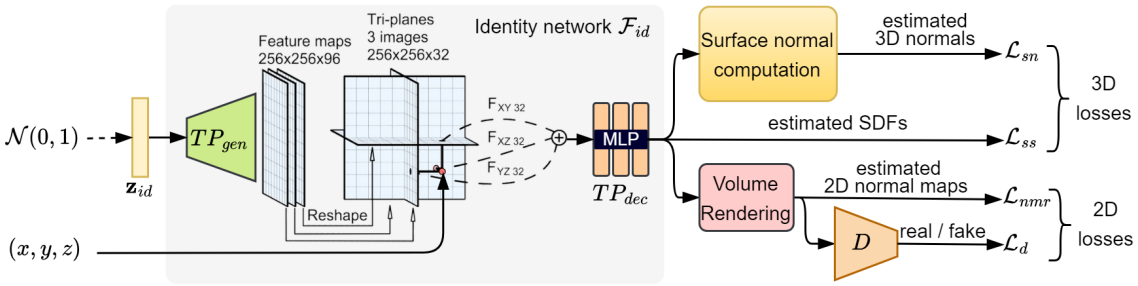


Figure 4.1: Schematic diagram of the proposed identity model. Identity latent code $\mathbf{z}_{id}$ is randomly initialized from standard normal distribution and is used to generate the identity specific tri-planes using tri-plane generator TP_{gen}. Then tri-plane features are extracted using *bilinear* interpolation (see sec. 3.6.3) for every randomly sampled points (x, y, z) from the ground truth shapes. These features are aggregated by summation and fed to tri-plane decoder TP_{dec} to predict the corresponding SDF values which is directly used to calculate the 3D losses $\mathcal{L}_{ss}$ and $\mathcal{L}_{sn}$. On another branch, volume rendering is used to render 2D normal maps from the implicit identity network $\mathcal{F}_{\mathbf{id}}$. These 2D normal maps are used to compute 2D normal reconstruction loss $\mathcal{L}_{nmr}$ and discriminator loss $\mathcal{L}_d$. Finally, all the losses are back propagated to jointly train discriminator D, volume renderer $\mathcal{R}$, identity network $\mathcal{F}_{\mathbf{id}}$ and identity code $\mathbf{z}_{id}$ in auto-decoder fashion. Please note, this diagram doesn't show the regularization losses for simplicity. All losses are listed in Table 4.1.

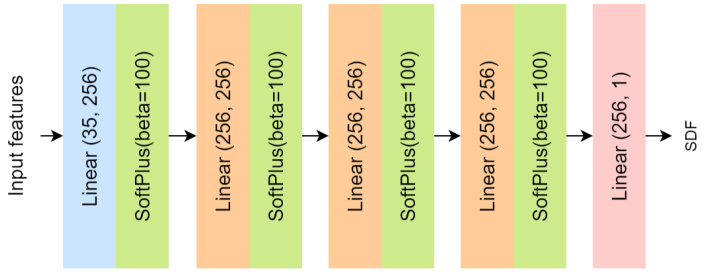


Figure 4.2: Tri-plane decoder (TP_{dec}) architecture. It is a five layer MLP that takes the sampled tri-plane features corresponding to a 3D point co-ordinate (x, y, z) and predicts the corresponding SDF value of that point. We concatenate the original points to the tri-plane features for a slightly better (observed) performance.

SDF value at queried (x, y, z) location using tri-plane features. Other than these two components, we also need a *volume rendering* module and a *discriminator* module for computing the 2D adversarial objectives.

Tri-plane Generator

It is a convolutional neural network $TP_{gen}(\mathbf{z}_{id}) : \mathbb{R}^{d_{id}} \rightarrow \mathbb{R}^{3 \times d_f \times t_{res} \times t_{res}}$ which produces three feature planes from a given identity code $\mathbf{z}_{id}$. Here, d_f is the feature depth and t_{res} is the spatial resolution of each feature planes. For all of our experiments, we typically set $d_f = 32$ and $t_{res} = 256$ following EG3D [23]. TP_{gen} is implemented as a slightly modified DCGAN [59] generator where we remove the last *tanh* and add an upsample layer with $scale\_factor = 4$ followed by a conv2d layer with $kernel\_size = 3$, $padding = 1$ and $out\_channels = 96$. Finally, we split the output tensor across the channel dimension to produce 3 planes having 32 channels each. We have also tried with StyleGAN2 [65] generator as used in EG3D [23], however we observed similar performance while DCGAN generator was easier to train. Hence, for the sake of simplicity we stick to DCGAN generator in this context.

Tri-plane Decoder

Fig. 4.2 shows tri-plane decoder architecture. It is a five layer MLP modeled as $TP_{dec}(\mathbf{x}, \mathbf{F}) : \mathbb{R}^{d_f+3} \rightarrow \mathbb{R}$ which takes the aggregated tri-plane features $\mathbf{F}$ concatenated with 3D co-ordinate $\mathbf{x}$ and produces the SDF value for that input co-ordinate. All the hidden layers have 256 nodes (neurons) and *softplus* non-linearity with $beta = 100$. A 3D co-ordinate $\mathbf{x} := (x, y, z)$ is first projected into the three orthogonal feature planes and tri-plane features $\mathbf{F}_{XY}$, $\mathbf{F}_{YZ}$, $\mathbf{F}_{XZ}$ are calculated from each of the feature planes using *bilinear* interpolation. Then, these features are aggregated by sum i.e. $\mathbf{F} = \mathbf{F}_{XY} + \mathbf{F}_{YZ} + \mathbf{F}_{XZ}$. We observed slightly better performance by concatenating the input co-ordinate (x, y, z) with $\mathbf{F}$ to form a combined input to the tri-plane decoder.

Volume Renderer

Given, an implicit shape representation network $\mathcal{F}_{\mathbf{id}} \in \mathscr{F}$, camera intrinsic $\mathbf{K} \in \mathbb{R}^{3\times3}$, camera pose (extrinsic) $\mathbf{Rt} \in \mathbb{R}^{4\times4}$ and image resolution $res \in \mathbb{R}$, volume rendering can be modeled as $\mathcal{R}(\mathcal{F}_{\mathbf{id}}, \mathbf{K}, \mathbf{Rt}, res) : \mathscr{F} \times \mathbb{R}^{3\times3} \times \mathbb{R}^{4\times4} \times \mathbb{R} \rightarrow \mathbb{R}^{res\times res\times3}$, which produces a $(res \times res)$ 2D view (image) of the 3D implicit shape based on the camera intrinsic and extrinsic parameters. We shoot one ray per pixel (see Fig. 4.3) and perform error bound sampling like VolSDF [49] within unit sphere (see Fig. 4.4). We render normal maps instead of RGB images and use MonoSDF [53] formulation for volume rendering with only one learnable parameter β. See sec. 3.7.2 for more details.

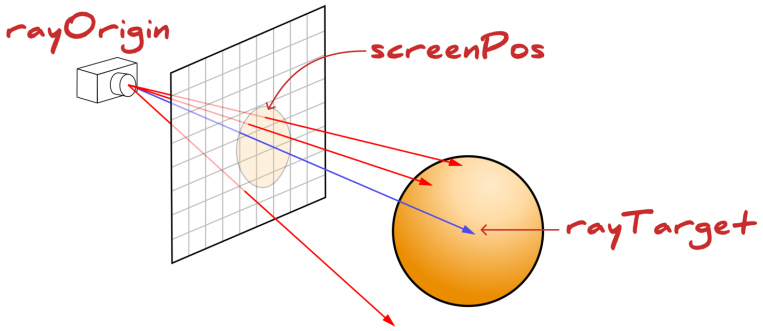


Figure 4.3: A schematic diagram for ray shooting. For volume rendering we shoot rays from camera to the scene through each pixel of the image (only 4 rays are shown), there by shooting only one ray per pixel. Image taken from [79]

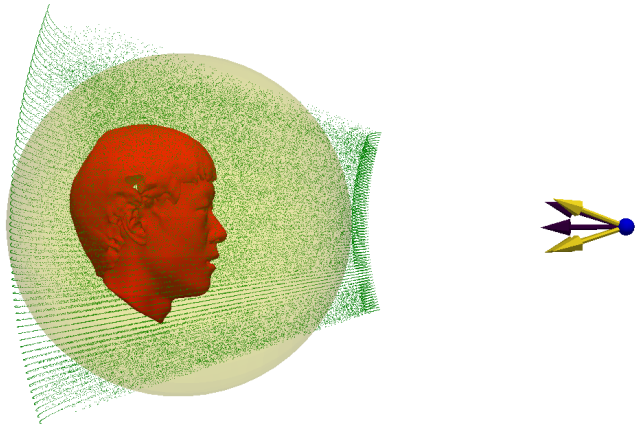


Figure 4.4: Visualization of actual ray samples for volume rendering. Blue dot denotes the camera location or ray origin and the arrows show ray directions (only 5 rays are shown for simplicity). As all the shapes are bounded within unit sphere, we adjust the near and far planes such that maximum samples are concentrated inside the unit sphere. Please note: the face shape inside sphere is shown just for visualization purpose, actually the explicit shape doesn't exist, instead it is encoded in the latent code $\mathbf{z_{id}}$ and the parameters of network $\mathcal{F}_{\mathbf{id}}$.

Discriminator

Discriminator $D : \mathbb{R}^{res \times res \times 3} \rightarrow \{0, 1\}$ is used to classify the normal maps as real or generated to compute the discriminator loss. To this end, we use π-GAN [74] discriminator implementation with only one modification. We just disable the *progressive growing* mechanism as we render the normal maps at fixed resolution $res = 64$.

Table 4.1: Objective functions, types, notations and weights used in modeling identity

Objective function	Type	Notation	Weight
Surface SDF	3D	$\mathcal{L}_{ss}$	$\lambda_{ss} = 2.0 \times 10^{1}$
Surface Normal	3D	$\mathcal{L}_{sn}$	$\lambda_{sn} = 3.0$
Eikonal	3D reg	$\mathcal{L}_{eik}$	$\lambda_{eik} = 2.0$
Non-surface penalty	3D reg	$\mathcal{L}_{nsp}$	$\lambda_{nsp} = 1 \times 10^{-1}$
Explicit Density	3D reg	$\mathcal{L}_{edr}$	$\lambda_{edr} = 1 \times 10^{-5}$
Normal Map	2D	$\mathcal{L}_{nmr}$	$\lambda_{nmr} = 2.0$
Generator loss	2D reg	$\mathcal{L}_{\mathrm{g}}$	$\lambda_{g} = 1.0$
Discriminator loss	2D reg	$\mathcal{L}_{\mathrm{d}}$	$\lambda_{d} = 1.0$
Total Variation	reg	$\mathcal{L}_{tv}$	$\lambda_{tv} = 1 \times 10^{-4}$
Tri-plane Penalty	reg	$\mathcal{L}_{tp}$	$\lambda_{tp} = 1 \times 10^{-4}$

4.2.3 Design of Training Objectives

To train our neural parametric head (identity) model a number of objective functions are used roughly grouped into *3D objectives*, *2D objectives* and *regularization objectives*. 3D objectives are computed with 3D data, i.e point clouds in our case, whereas 2D objectives are computed on multi-view normal map renderings of the implicit shape. Regularization objectives are used to obtain well behaved latent space and well behaved implicit shape representation. Table 4.1 lists all the objective functions for modeling identity and below we discuss their mathematical formulation.

Surface SDF

By definition SDF value should be zero on the surface (manifold) of the shape. To apply this constraint we sample 3D points from surface of the 3D scans and formulate the objective as follows:

$$\mathcal{L}_{ss} := \sum_{\mathbf{x} \in \delta \mathbf{X}} |\mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id})| \tag{4.1}$$

where $\delta\mathbf{X}$ denotes the 3D point samples on the surface of the shape and $\mathbf{z}_{id}$ is the identity latent code.

Surface Normal

To represent the shape correctly, the surface normals from the implicit SDF representation should match the surface normals from the ground truth mesh, or in other words their scalar product should be 1. To this end, surface normals from the implicit shape can be computed from the gradient of the SDF on surface points. Thus, we model this objective as follows:

$$\mathcal{L}_{sn} := \sum_{\mathbf{x}\in\delta\mathbf{X}} (1 - \langle \nabla \mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id}), n_{id}(\mathbf{x}) \rangle) \tag{4.2}$$

where ∇ denotes the gradients with respect to $\mathbf{x}$, $n_{id}(\mathbf{x})$ denotes the ground truth surface normals at $\mathbf{x}$ for a particular identity and $\langle ., . \rangle$ denotes the scalar (dot) product.

Eikonal

This is a property of a true SDF that everywhere within the field, the gradient of the field will have unit norm. This idea was introduced as a regularization for training implicit shapes in [80, 81] to avoid *everywhere zero* solution. We also use this constraint modelled as:

$$\mathcal{L}_{eik} := \sum_{\mathbf{x}\in\mathbf{X}\cup\delta\mathbf{X}} \left(||\nabla \mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id})||_2 - 1 \right) \tag{4.3}$$

where $\mathbf{X}$ denotes the set of random 3D co-ordinates of free space within unit sphere and $\delta\mathbf{X}$ denotes the 3D point samples on the surface of the shape as earlier.

Non-surface Penalty

This objective penalizes the free space (non-surface) points from producing zero SDF values. Mathematically, this is modeled as exponential penalty:

$$\mathcal{L}_{nsp} := \sum_{\mathbf{x}\in\mathbf{X}} \exp(-\alpha |\mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id})|) \tag{4.4}$$

where α is a constant and typically we set this value to 10.

Explicit Density Regularization

Inspired from [82], this regularization is used to learn a smooth outside-of-shape volume. The idea here is that for a small random perturbation on the non-surface

points the output of the implicit network should not change much. Mathematically:

$$\mathcal{L}_{edr} := \sum_{\mathbf{x} \in \mathbf{X}} ||\mathcal{F}_{\mathbf{id}}(\mathbf{x}, \mathbf{z}_{id}) - \mathcal{F}_{\mathbf{id}}(\mathbf{x} + \omega, \mathbf{z}_{id})||_2^2 \tag{4.5}$$

where ω is a small random noise vector sampled from $\mathcal{N}(0, 0.0001)$.

Normal Map Reconstruction

Here, the idea is that 2D normal map obtained through volume rendering of the implicit shape from a particular camera pose should match the ground truth normal map obtained from rendering the 3D ground truth mesh in that same camera pose. Let's assume there are M number of different view normal renderings available for a particular identity *id* denoted as $N_{id}^{(v)}$, where $v \in \{1, 2, \ldots, M\}$. Then, this reconstruction loss can be modeled with L1 loss as follows:

$$\mathcal{L}_{nmr} := \sum_{v=1}^{M} ||\hat{N}_{id}^{(v)} - N_{id}^{(v)}||_1 \tag{4.6}$$

where $\hat{N}_{id}^{(v)}$ are the predicted normal maps:

$$\hat{N}_{id}^{(v)} = \mathcal{R}(\mathcal{F}_{\mathbf{id}}, \mathbf{K}^{(v)}, \mathbf{Rt}^{(v)}, res) \tag{4.7}$$

where $\mathcal{R}$ is the volume rendering function defined in sec. 4.2.2.

Adversarial Objectives

We use the discriminator D defined in sec 4.2.2 in an adversarial setting with a goal to make the volume rendered normal maps indistinguishable from the ground truth normal maps. We train the discriminator to classify the ground truth samples to class 1 and estimated samples to class 0, where as the generator i.e the identity network is trained to fool the discriminator. We use a non saturating GAN loss with R1 penalty [83] formulated as follows:

$$\mathcal{L}_{\mathrm{d}}(D; G) := \mathbb{E}_{(\mathbf{z}_{id}, v)} \left[f(D(G(\mathbf{z}_{id}, v))) \right] + \mathbb{E}_{(N_{id}^{(v)})} \left[f(-D(N_{id}^{(v)})) \right] + \lambda \mathcal{L}_{gp} \tag{4.8}$$

$$\mathcal{L}_{\mathrm{g}}(G; D) := \mathbb{E}_{(\mathbf{z}_{id}, v)} \left[f(-D(G(\mathbf{z}_{id}, v))) \right] \tag{4.9}$$

$$\mathcal{L}_{gp} := \mathbb{E}_{(N_{id}^{(v)})} \left[|\nabla D(N_{id}^{(v)})|^2 \right] \tag{4.10}$$

$$f(u) := \log(1 + \exp(u)) \tag{4.11}$$

where λ is a constant and we typically use $\lambda = 5$. Generator G can be considered as the composition of the renderer $\mathcal{R}$ and identity network $\mathcal{F}_{\mathbf{id}}$ i.e. $G := \mathcal{R} \circ \mathcal{F}_{\mathbf{id}}$. More explicitly this is defined as:

$$G(\mathbf{z}_{id}, v) := \hat{N}_{id}^{(v)} = \mathcal{R}(\mathcal{F}_{\mathbf{id}}(\mathbf{z}_{id}), \mathbf{K}^{(v)}, \mathbf{Rt}^{(v)}, res) \tag{4.12}$$

(we have omitted the co-ordinates input $\mathbf{x}$ for simplifying the notation.)

Total Variation

To simplify the data manifold we regularize the tri-plane feature planes using Total Variation (TV) as proposed in [82]. Let, $f_h : \mathbb{R}^{d_f \times t_{res} \times t_{res}} \rightarrow \mathbb{R}^{d_f \times t_{res} \times t_{res}}$ and $f_v : \mathbb{R}^{d_f \times t_{res} \times t_{res}} \rightarrow \mathbb{R}^{d_f \times t_{res} \times t_{res}}$ are two functions which flip the input horizontally (across second dimension) and vertically (across third dimension) respectively. Then mathematically TV penalty can be computed as:

$$\mathcal{L}_{tv} := \sum_{p \in \{xy,yz,xz\}} \left(\sqrt{\sum (\mathbf{t}_p - f_h(\mathbf{t}_p))^2} + \sqrt{\sum (\mathbf{t}_p - f_v(\mathbf{t}_p))^2} \right) \tag{4.13}$$

where $\mathbf{t}_{xy}$, $\mathbf{t}_{yz}$, $\mathbf{t}_{xz}$ three orthogonal feature planes generated from an identity code $\mathbf{z}_{id}$ using tri-plane generator TP_{gen}.

Tri-plane Penalty

To avoid very high values (or outliers) in the feature planes we use a L2 penalty as well on the tri-planes as follows:

$$\mathcal{L}_{tp} := \sum_{p \in \{xy,yz,xz\}} ||\mathbf{t}_p||_2 \tag{4.14}$$

Combining All the Objectives

We combine all but the discriminator loss $\mathcal{L}_{\mathrm{d}}$ using weighted sum with the weights defined in Table 4.1. Summing up this individual combined loss for all identities in training set we get total identity loss $\mathcal{L}_{id}$:

$$\mathcal{L}_{id} := \sum_{j \in J} \sum_{s \in S} \lambda_s (\mathcal{L}_s)_{(j)} \tag{4.15}$$

where J is the set of all training identities and S is the set of all sub scripts except d, i.e.

$$S := \{ss, sn, eik, nsp, edr, nmr, g, tv, tp\} \tag{4.16}$$

$\mathcal{L}_{id}$ is back propagated to train the renderer $\mathcal{R}$, identity network $\mathcal{F}_{\mathbf{id}}$ and identity latent codes $\mathbf{z}_{id}$ whereas $\sum_{j \in J} (\mathcal{L}_d)_{(j)}$ is back propagated to train the discriminator D.

4.2.4 Multi-stage Training

We use the 2D GAN loss to constrain or implicitly regularize the latent space to learn the distribution of the real faces. However, as we only use a very small rendering resolution of 64×64 due to high computational cost, we observe a saturation in the loss after few thousand epochs with reconstructions lacking fine surface details like eye-lines or so. So, we come up with a multi-stage training strategy where we train with all the objectives upto 5k epochs, then we shut off the 2D objectives ($\mathcal{L}_{nmr}$, $\mathcal{L}_g$, $\mathcal{L}_d$) and train another 3k epochs. After that we turn off $\mathcal{L}_{edr}$, $\mathcal{L}_{tp}$ and $\mathcal{L}_{tv}$ and keep training untill convergence (usually around 12k-15k epochs) only with the 3D losses. Here, the intuitive understanding is that GAN based 2D loss acts as an adaptive regularizer in the early training phase to help the model avoid bad local minima, whereas at later training phase only $3D$ losses are used to recover the fine surface details.

4.3 Modeling Expression

This thesis is mostly focused at modeling identity. So, for modeling expression we have used similar forward deformation approach as of [19, 20]. The high level idea here is to train another neural network to deform the faces in neutral (canonical) expression to match the target expression while preserving the face identity.

4.3.1 Input and Output

We model expression with a deformation network $\mathcal{F}_{\mathbf{ex}}(\mathbf{x}, \mathbf{z}_{ex}, \mathbf{z}_{id}) : \mathbb{R}^3 \times \mathbb{R}^{d_{id}+d_{ex}} \rightarrow \mathbb{R}^3$ which takes a 3D co-ordinate $\mathbf{x} \in \mathbb{R}^3$ (in neutral pose) along with a pre-trained identity code $\mathbf{z}_{id} \in \mathbb{R}^{d_{id}}$ and a randomly initialized expression code $\mathbf{z}_{ex} \in \mathbb{R}^{d_{ex}}$ and predicts the point wise deformation $\Delta\mathbf{x} \in \mathbb{R}^3$ such that $\mathbf{x}_{posed} = \mathbf{x} + \Delta\mathbf{x}$ is the corresponding point of $\mathbf{x}$ in the target expression. Identity code $\mathbf{z}_{id}$ is pre-trained in identity model and frozen, whereas $\mathbf{z}_{ex}$ is initialized from a standard normal distribution and is trained in auto-decoder (see sec. 3.3) fashion along with the deformation network $\mathcal{F}_{\mathbf{ex}}$.

4.3.2 Model Architecture

Following [19, 20], we use a MLP as the deformation network $\mathcal{F}_{\mathbf{ex}}$. In particular, we use a DeepSDF [17] network with $input\_dim = 3$, $output\_dim = 3$, $n\_layers = 8$, $hidden\_dim = 1024$ and $latent\_dim = d_{id} + d_{ex} = 512 + 200 = 712$ as we use $d_{id} = 512$ and $d_{ex} = 200$ as the latent code lengths for *identity* and *expression* spaces respectively.

4.3.3 Design of Training Objectives

To train the expression (deformation) network $\mathcal{F}_{\mathbf{ex}}$ we use two 3D objectives and one regularization on the expression latent codes $\mathbf{z}_{ex}$ as described below:

Correspondence Loss

Here the idea is that if we add the deformation deltas $\Delta\mathbf{x}$ to the unposed (neutral expression) co-ordinates $\mathbf{x}$ it should be close to the corresponding points $\mathbf{x}_{posed}$ in the target expression. Formally this can be defined as:

$$\mathcal{L}_c := \sum_{\mathbf{x}\in\mathbf{X}} \big((\mathbf{x} + \Delta\mathbf{x}) - \mathbf{x}_{posed}\big)^2 \tag{4.17}$$

$$\Delta\mathbf{x} := \mathcal{F}_{\mathbf{ex}}(\mathbf{x}, \mathbf{z}_{ex}, \mathbf{z}_{id}) \tag{4.18}$$

where $\mathbf{X}$ is the set of all (surface) points in the facial region of neutral (canonical) expression for an identity and $\mathbf{x}_{posed}$ denotes the ground truth correspondence (or image) of a point $\mathbf{x}$ in a particular expression pose for that same identity.

Deformation Penalty

Deformation field should only be locally active in the face region i.e. it should not deform the global structure of the head while changing expression. That means deformation deltas should be zero for any random points outside the face region. Mathematically this can be modelled as:

$$\mathcal{L}_{dp} := \sum_{\mathbf{x}\notin\mathbf{X}} ||\mathcal{F}_{\mathbf{ex}}(\mathbf{x}, \mathbf{z}_{ex}, \mathbf{z}_{id})||_2^2 \tag{4.19}$$

Latent Regularization

To have a well-behaved expression latent space and avoid outliers or very high values in the expression latent code $\mathbf{z}_{id}$ we add a L2 regularization on it:

$$\mathcal{L}_r := ||\mathbf{z}_{ex}||_2^2 \tag{4.20}$$

Combining All the Objectives

All of the above losses are calculated for each of L expressions for each of J identities and combined with weighted sum as follows:

$$\mathcal{L}_{ex} := \sum_{j,l\in J,L} \lambda_c(\mathcal{L}_c)_{(jl)} + \lambda_{dp}(\mathcal{L}_{dp})_{(jl)} + \lambda_r(\mathcal{L}_r)_{(jl)} \tag{4.21}$$

where $\lambda_c = 1.0 \times 10^2$, $\lambda_{dp} = 1.0 \times 10^{-5}$, $\lambda_r = 5.0 \times 10^{-5}$ are the weights for the corresponding three objectives. $\mathcal{L}_{ex}$ is back propagated to train the expression (deformation) network $\mathcal{F}_{\mathbf{ex}}$.

4.4 Model Fitting

Model fitting refers to the inference time process of finding the identity $\mathbf{z}_{id}$ and expression $\mathbf{z}_{ex}$ codes from a single depth frame observation of a previously unseen face. We can use this process to use the learned face priors to reconstruct new faces from single frame depth observation. Below we discuss model fitting for identity and expressions for our parametric head model.

4.4.1 Identity Fitting

Let's assume we have a single view depth map $\mathbf{X}_p \subset \mathbb{R}^3$ of a previously unseen person in neural expression. Ideally we need to jointly optimize for $\mathbf{z}_{id}$ and $\mathbf{z}_{ex}$ keeping $\mathcal{F}_{\mathbf{id}}$ and $\mathcal{F}_{\mathbf{ex}}$ frozen. Expression code $\mathbf{z}_{ex}$ is involved here to capture minor deviations from perfectly neutral expression. Now, jointly optimizing for identity and expression with a forward deformation expression model is not trivial. To this end, we follow SNARF [84] and NPHM [20] which used *iterative root finding* to get $\mathbf{X}_c$ points in perfect neutral expression and then optimize for $\mathbf{z}_{id}$. Mathematically,

$$\mathbf{x}_c = \arg\min_{\mathbf{x}} |\mathbf{x}_p - \mathcal{F}_{\mathbf{ex}}(\mathbf{x}, \mathbf{z}_{ex}, \mathbf{z}_{id})| \tag{4.22}$$

and then optimize for $\mathbf{z}_{id}$ and $\mathbf{z}_{ex}$ in the following:

$$\sum_{\mathbf{x}_p \in \mathbf{X}_p} |\mathcal{F}_{\mathbf{id}}(\mathbf{x}_c, \mathbf{z}_{id})| + \lambda_{id}^{fit} ||\mathbf{z}_{id}||_2^2 + + \lambda_{ex}^{fit} ||\mathbf{z}_{ex}||_2^2 \tag{4.23}$$

However, for the ablation study (see sec. 6.3) we assume that the input depth observation is already in perfect neutral expression i.e. $\mathbf{X}_p$ and $\mathbf{X}_c$ are interchangeable and thus optimize only equation 4.23.

4.4.2 Expression Fitting

We already get $\mathbf{z}_{id}$ and a good initial estimate for $\mathbf{z}_{ex}$ from identity fitting. So, here we optimize further only for $\mathbf{z}_{ex}$ using the same equation 4.23 keeping everything else fixed to fit a particular expression for a particular identity.

5 Experiment

As the name suggests, in this chapter practical details of the experiments will be discussed. In particular, we will talk about dataset, data pre-processing, experimental setup, evaluation metrics and comparison baselines.

5.1 Dataset

We used the same 3D head scan dataset acquired by NPHM [20] for training our identity and expression models. Dataset has 3D face scans of 255 different subjects out of which 188 are male and 67 are female. Each subject has roughly 19 to 23 different expressions, thereby having 5200 scans in total.

5.1.1 Data Scanning

Human subjects were scanned by the NPHM [20] authors at Technical University of Munich (TUM) Visual Computing Lab using two Artec Eva scanners attached at a complementary viewing angle to the two opposite ends of an inverted U-shape structure rotating around the subjects' heads with the help of a robotic actuator.

Each of the two scanners uses structured light projection approach for range measurement and produces 95 frames over 6 seconds for a full 360° rotation. All of these frames are processed, aligned and finally fused into 3D mesh followed by hole filling and noise removal. For more details on this, please refer to Appnedix B of NPHM [20] paper.

5.1.2 Data Pre-processing

NPHM dataset provides ground truth 3D mesh scans, their registration meshes and FLAME [28] fitting meshes. However, to train our implicit identity network we need 3D point clouds with ground truth SDF and normal values for the 3D losses and 2D ground truth multi-view normal maps for the 2D losses.

Torso Separation

NPHM [20] raw meshes contains significant portion of torso roughly until chest. However, as we are interested in head model, we separate the torso portion from the head using a plane defined by three vertices in the corresponding FLAME [28] fitting following NPHM [20] approach for the sake of simplicity.

Sampling Points

We sample sample (x, y, z) point co-ordinates on and around the surface of the ground truth 3D meshes following DeepSDF [17] sampling strategy. To this end, we first sample N_{surf} points on the mesh whose SDF values are zero by definition and normals are calculated using *barycentric interpolation* of adjacent face normals. We further divide N_{surf} points into two sets namely N_{surf}^{face} and $N_{surf}^{non\_face}$ depending on whether they are in face or non-face region of the head. Now, we perturb these two sets of points twice with standard deviations $\sigma_1 = 0.0025$ and $\sigma_2 = 0.00025$ to create another four sets of near surface points namely $N_{near\_surf\_1}^{face}$, $N_{near\_surf\_1}^{non\_face}$, $N_{near\_surf\_2}^{face}$ and $N_{near\_surf\_2}^{non\_face}$. We compute their ground truth SDF values by using virtual laser scanning model. Finally, we sample N_{ball} points uniformly from a unit sphere centered at origin. Please note here, scanned meshes in NPHM dataset are already normalized to be centered at origin and bounded within unit sphere, so scaling and translation is not required. In practice, for speed we pre-compute point samples in bulk (25M samples) and randomly sub-sample $|N_{surf}^{face}| = |N_{near\_surf\_1}^{face}| = |N_{near\_surf\_2}^{face}| = 750$, $|N_{surf}^{non\_face}| = |N_{near\_surf\_1}^{non\_face}| = |N_{near\_surf\_2}^{non\_face}| = 250$ and $|N_{ball}| = 300$ points per head per iteration.

Normal Maps Rendering

We need multi-view ground truth 2D normal maps for computing 2D losses. To this end, we employ the concept of *Fibonacci lattice* [85, 86] to uniformly place 128 virtual cameras, each looking at the object at origin, on a sphere of radius 2.6 and render the 2D normal maps in *OpenGl* convention using *pyrender*. It is noteworthy, that the MonoSDF [53] based differentiable volume rendering approach used in our training pipeline follows *OpenCV* camera convention instead of *OpenGl* convention. We resolve this discrepancy by computing an extra transformation on the volume renderings to bring them in *OpenGl* format.

Sampling Deformations

As we model different expressions as deformation fields over neutral expression, we need to sample neutral and posed point pairs. Sampling such pair-wise point correspondences from the ground truth raw scans is not trivial due to different number of vertices or faces. So, following NPHM [20], we sample the deformation point pairs from non-rigid registrations of the faces in neutral and different expression. To this end, we first sample N_{def} points from the registration mesh of the neutral expression and find their face ids and *barycentric* coordinates. Then we use this information to find corresponding points on the registered mesh posed with some expression. Then, we repeat the process by sampling on the posed mesh first followed by correspondence finding on the neutral mesh. Thus, we sample two way correspondences and reduce directional sampling bias. To make this process even robust, we add two levels of Gaussian noise with $\sigma_1 = 0.01$ and $\sigma_2 = 0.002$ for loose and tight correspondence pairs respectively. We typically pre-compute such point pairs in bulk (1M samples per head per expression) and randomly sub-sample 1000 point pairs per head per expression per iteration.

5.2 Experimental Setup

Here we will talk about the system configurations and hyperparameters used in our experiments in the following sub-sections.

5.2.1 Hardware and Software Configuration

We have performed all of our experiments on a Linux server running Ubuntu 20.04.4 LTS with 16 core Intel(R) Xeon(R) W-2245 CPU (3.90GHz), 64 GiB DDR4 RAM and a NVIDIA RTX3090 GPU with CUDA 11.6. To run our experiments in default setting, minimum 24 GiB of GPU VRAM is required.

We have used Python 3.10.9 as the programming language and PyTorch 1.12.1 as the deep learning framework. To mention some more important specific python libraries, we have used *trimesh* for mesh operation, *pyvista* for visualization, *pyrender* for rendering ground-truth normal maps, *numpy* to compute and store ground-truth SDFs and surface normals and *wandb* for tracking training progress.

5.2.2 Hyper-parameters and Design Choices

We make quite a few design choices and set a lot of hyper-parameters in different portions of our proposed approach. We discuss them as follows:

Architecture

For the identity model, the first design choice we make for the tri-plane generator (TP_{gen}) architecture amongst Style GAN2 vs DCGAN approach. From single shape over-fitting experience we have empirically observed Style GAN2 approach takes longer to reach same performance as of DCGAN approach. Hence, we use DCGAN generator model for the tri-plane generator and set identity latent code dimension $d_{id} = 512$, tri-plane resolution $t_{res} = 256$ and tri-plane feature depth $d_f = 32$ following EG3D [23]. For the tri-plane decoder (TP_{dec}), we use a MLP with 5 layers and 256 neurons in each hidden layer with *SoftPlus* non-linearity with $beta = 100$. For the discriminator (D), we use π-GAN [74] discriminator with *progressive growing* disabled. For NPM [19] baseline, we use a MLP with 8 hidden layers with 1024 neurons in each layer with *SoftPlus* non-linearity with $beta = 100$ and skip connection in layer 4 following DeepSDF [17]. For expression model, we use similar architecture as of NPHM [20].

Rendering

For rendering multi-view normal maps we place the virtual cameras on a sphere of radius 2.6 centered at origin, set $near = 0.2$, $far = 5.0$ and use rendering resolution $res = 64$. Additionally, for the differentiable volume rendering we use *error bound sampling* with 32 initial samples and 32 evaluation samples, $\epsilon = 0.1$, initial $\beta = 0.001$, $\beta_{min} = 0.0001$, beta-iteration $\beta_{iter} = 10$ and $max_{iter} = 10$ following MonoSDF [53].

Objective Functions

Weights of the different objective functions are some of the crucial hyper-parameters for the proposed model. For the proposed identity model, these hyper-parameters are already listed in Table 4.1. For NPM [19] baseline and ablation with only 3D losses we use slightly different weights as follows: $\lambda_{ss} = 2.0$, $\lambda_{sn} = 0.3$, $\lambda_{eik} = 0.1$, $\lambda_{nsp} = 0.01$.

Training

For training NPM [19] baseline and ablations with only 3D losses we use batch size $bs = 32$, learning rate $lr = 5 \times 10^{-5}$, learning rate for identity codes $lr_{lat} = 1 \times 10^{-3}$, $grad\_clip = 0.1$, $grad\_clip_{lat} = 0.1$, $lr\_decay = 0.75$, $weight\_decay = 0.02$, $decay\_interval = 1000$ and $decay\_interval_{lat} = 2000$. For the full proposed model and ablations involving 2D losses we use batch size $bs = 4$ and for the experiments involving discriminator we set discriminator learning rate $lr_{disc} = 2 \times 10^{-4}$, keeping other hyper-parameters same as mentioned before. We use *AdamW*, *SparseAdam*, and *SGD* optimizers for training identity network, identity latent codes and discriminator

respectively. For training expression model we use same hyper-parameters as of NPHM [20].

Mesh Extraction

For mesh extraction from the implicit surface, in all our experiments, we query the network at 3D grid points within $(-1, 1)$ with grid resolution $res_{grid} = 256$ and then run marching cubes algorithm with $threshold = 0.0$.

5.3 Evaluation Metrics

For quantitative comparison of the proposed model with the baselines and ablations, we have used three evaluation metrics, similar to NPHM [20], namely *Chamfer-L_1* distance, *Normal Consistency (N.C.)* and *F-Score @ t*. We discuss these formulations in the following sub-sections.

5.3.1 Chamfer Distance

It measures the similarity between two point clouds P_{gt} and P_{pred} and defined as average distance between the pairs of nearest neighbours between the two point clouds. To this end, we sample 2.5 M points from each of reconstructed and ground-truth meshes to create P_{pred} and P_{gt} respectively and then mathematically define Chamfer-L_1 distance as:

$$\text{Chamfer-}L_1(P_{pred}, P_{gt}) := \frac{1}{|P_{gt}|} \sum_{\mathbf{x} \in P_{gt}} \min_{\mathbf{y} \in P_{pred}} ||\mathbf{x} - \mathbf{y}||_1 + \frac{1}{|P_{pred}|} \sum_{\mathbf{y} \in P_{pred}} \min_{\mathbf{x} \in P_{gt}} ||\mathbf{x} - \mathbf{y}||_1 \quad (5.1)$$

Quite intuitively, by construction, lower value of this measure refers to higher similarity between P_{gt} and P_{pred} and thereby corresponds to better reconstruction.

5.3.2 F-Score

It is used to explicitly evaluate the distance between the ground-truth (gt) and reconstructed (pred) surfaces and defined as the harmonic mean of precision and recall. Precision p_t is the percentage of the reconstructed points that lie within a certain distance threshold t to the ground truth points and it is a measure of accuracy. On the other hand recall r_t measures the completeness of the reconstruction by calculating the percentage of ground-truth points that lie within a certain distance threshold t to the reconstructed points. Mathematically, this can be defined as:

$$\text{F-Score @ t}(P_{pred}, P_{gt}) := \frac{2 * p_t * r_t}{p_t + r_t} \quad (5.2)$$

$$p_t := \frac{1}{|P_{pred}|} |\{\mathbf{y} : \mathbf{y} \in P_{pred}, \mathrm{d}(\mathbf{y}, \tilde{\mathbf{x}}) <= t\}| \tag{5.3}$$

$$r_t := \frac{1}{|P_{gt}|} |\{\mathbf{x} : \mathbf{x} \in P_{gt}, \mathrm{d}(\mathbf{x}, \tilde{\mathbf{y}}) <= t\}| \tag{5.4}$$

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) := ||\mathbf{p} - \mathbf{q}||_1 \tag{5.5}$$

where $\tilde{\mathbf{x}} = \min_{\mathbf{r} \in P_{gt}} ||\mathbf{r} - \mathbf{y}||_1$ and $\tilde{\mathbf{y}} = \min_{\mathbf{r} \in P_{pred}} ||\mathbf{x} - \mathbf{r}||_1$ are the corresponding nearest neighbours from the other set. By construction, higher value of this measure corresponds to better reconstruction. We compute F-Score at $t = 1$ mm with 2.5M points sampled from both ground-truth and reconstructed meshes.

5.3.3 Normal Consistency

While Chamfer-L_1 distance and F-Score @ t consider the spatial position of the two meshes, N.C. takes into account the orientation of the meshes. It is defined as the average *cosine* of the angles between the normal vectors of adjacent points in the predicted and ground-truth meshes. Mathematically,

$$\text{N.C.}(P_{pred}, P_{gt}) := \frac{1}{|P_{gt}|} \sum_{\mathbf{x} \in P_{gt}} \cos(\mathrm{n}(\mathbf{x}), \mathrm{n}(\tilde{\mathbf{y}})) + \frac{1}{|P_{pred}|} \sum_{\mathbf{y} \in P_{pred}} \cos(\mathrm{n}(\mathbf{y}), \mathrm{n}(\tilde{\mathbf{x}})) \tag{5.6}$$

where $\tilde{\mathbf{y}} = \min_{\mathbf{r} \in P_{pred}} ||\mathbf{x} - \mathbf{r}||_1$, $\tilde{\mathbf{x}} = \min_{\mathbf{r} \in P_{gt}} ||\mathbf{r} - \mathbf{y}||_1$ and $\mathrm{n}(\mathbf{p})$ is the normal at point $\mathbf{p}$. As *cosine* attains *maximum* at 0 (i.e. two vectors are perfectly oriented in the same direction), a higher value for N.C. corresponds to better reconstruction. We compute N.C. with 2.5M points sampled from both ground-truth and reconstructed meshes.

5.4 Baselines

As briefly discussed in the Introduction (see Chapter 1) and Related Work (see Chapter 2), there are broadly three major lines of research in parametric head modeling namely *linear explicit models*, *neural implicit models* and *localized models*. Considering these three categories, we choose one approach from each group to compare with our results. The chosen baselines are discussed below:

5.4.1 FLAME

FLAME [28] is one of the most recent and advanced *linear explicit models* which seamlessly incorporates articulated head components in linear shape spaces. It models disjoint parametric spaces for shape (identity), expression, head pose, neck and jaw

movements with separate PCA for each. However, as we only model identity and expression, we compare our results with *zero pose* FLAME fittings without any head or neck movement. As we already use same co-ordinate system (subject to a scale factor of 4 as of NPHM [20]) as of FLAME, we can easily fit the point clouds to obtain the FLAME parameters using the code from [87].

5.4.2 NPMs

NPMs [19], though originally introduced for full human body, is a recent simple *neural implicit model* on top of which the proposed model is built. So, it is a natural choice for a comparison baseline. Here, instead of tri-planes a big co-ordinate based MLP is used like DeepSDF [17] and it is trained only with 3D losses without explicit density and total variation regularization due to its inherent strong inductive bias. For more details on similarity and difference of NPMs to our approach please see sec. 2.3.1.

5.4.3 NPHM

NPHM [20] is the most recent state-of-the-art in parametric head models and it falls under both the categories of *neural implicit models* and *localized models* as it employs ensemble of small local MLPs for the implicit shape representation. As we use same data and co-ordinate system as of NPHM [20], we directly use their code from official repository[1] for generating comparison result. Although our proposed approach is not directly a *localized model*, it is still comparable to NPHM as both use implicit model and tri-plane representation of our approach can have indirect local effect. For more details on approach comparison please see sec. 2.3.2.

[1] `https://github.com/SimonGiebenhain/NPHM`

6 Results

In this chapter we will report the results of our proposed model and compare them with the baselines and state-of-the-arts in this field. Additionally, we will show ablation study to analyze the effects of different hyper-parameters and design choices.

6.1 Identity Space

This section describes the results of our proposed identity model. In particular we will talk about inference reconstruction, latent interpolation and unconditional sampling of face identities.

Table 6.1: Inference time identity fitting with partial point clouds obtained from single view depth maps. The metrics are separately computed for both only face region and full head region. We compare our method to widely-used state-of-the-art parametric head models including FLAME [28], NPM [19] and NPHM [20]. Our model performs significantly better at constructing overall head geometry including hair regions.

Method	L_1-Chamfer ↓		N.C. ↑		F-Score @ 1 mm ↑	
	face	head	face	head	face	head
FLAME [28]	0.643	5.829	0.975	0.894	0.998	0.636
NPM [19]	0.451	2.037	0.991	0.897	0.999	0.901
NPHM [20]	**0.320**	1.360	**0.994**	0.924	**1.000**	0.957
Ours	0.359	**0.820**	0.993	**0.948**	**1.000**	**0.989**

6.1.1 Reconstruction

Reconstruction at the inference time refers to the idea of fitting (see sec. 4.4) a partial depth observation to our identity model i.e obtaining the full 3D face shape of the corresponding person. Fig 6.1 and Table 6.1 jointly show our identity reconstruction results at inference time. As compared to baselines and state-of-the-arts, clearly our

Figure 6.1: Identity fitting comparison. At inference time we fit partial point clouds obtained from single view depth maps to reconstruct full head geometry. We compare our method to widely-used state-of-the-art parametric head models including FLAME [28], NPM [19] and NPHM [20]. Our proposed approach is able to capture more high frequency surface details especially in the hair regions.

model is able to capture significantly more detailed geometry in the overall head region while slightly compromising in the face region. Also, from Table 6.2 we can see mesh extraction from our approach is significantly faster from other approaches due to convolution based tri-plane hybrid representation instead of big MLPs.

Table 6.2: Given an identity latent code $\mathbf{z}_{id}$, we compare the mesh extraction time (at 256 grid resolution) of our method with other two implicit parametric models NPM [19] and NPHM [20]. Our approach is significantly faster due to convolution based tri-plane hybrid representation and small MLPs.

Method	Time (Sec.)
NPM [19]	18.319
NPHM [20]	33.119
Ours	**03.717**

6.1.2 Latent Interpolation

Let, $\mathbf{z}_{id_1}, \mathbf{z}_{id_2} \in \mathbb{R}^{d_{id}}$ are two latent codes representing two identities. Now, for an $\alpha \in [0, 1]$, linear interpolation between the latent codes can be defined as:

$$\mathbf{z}_{id_i} := \alpha \mathbf{z}_{id_1} + (1 - \alpha)\mathbf{z}_{id_2} \tag{6.1}$$

By varying α from 0 to 1, we can get several intermediate latent codes. Now, if the learnt latent space is smooth, theses intermediate codes will produce shapes which smoothly transitions from source to target shape. We can see this effect in Fig 6.2 confirming that our identity latent space is smooth.

Figure 6.2: Interpolation in identity latent space. Smooth transition from source (extreme-left) to target (extreme-right) shapes shows the smoothness of the latent space of the proposed identity model.

Figure 6.3: Unconditional samples generated from the our identity model latent space.

6.1.3 Unconditional Sampling

As we model our identity latent space from Gaussian distribution, at inference time new latent codes can easily be sampled from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu} \in \mathbb{R}^{d_{id}}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{d_{id} \times d_{id}}$ are the learned mean and co-variance respectively of the identity latent codes from training set. Such newly sampled latent codes can be used to generate new shape identities unconditionally. We show some of such unconditional samples in Fig 6.3.

6.2 Expression Space

In this section we will discuss the results of our expression model in terms of inference time fitting or reconstruction and latent interpolation between different expressions.

Table 6.3: Inference time expression fitting with partial point clouds obtained from single view depth maps. The metrics are separately computed for both only face region and full head region. We compare our method to widely-used state-of-the-art parametric head models including FLAME [28], NPM [19] and NPHM [20].

Method	L_1-Chamfer ↓		N.C. ↑		F-Score @ 1 mm ↑	
	face	head	face	head	face	head
FLAME [28]	0.769	6.016	0.972	0.882	0.999	0.636
NPM [19]	0.416	1.659	0.988	0.888	**1.000**	0.934
NPHM [20]	**0.368**	1.313	**0.991**	0.909	**1.000**	0.965
Ours	0.650	**1.179**	0.981	**0.915**	0.998	**0.984**

6.2.1 Reconstruction

Table 6.3 and Fig 6.4 show the inference time reconstruction results of our expression space. Like identity model here also our approach consistently performs better while the metrics are calculated on full head region. However, when metrics are calculated only in the facial region we observe a drop in the performance. This could probably be explained by the fact that we do not use GAN loss based regularization on the deformations and thus unregularized deformation field sometimes over deforms the facial region obtained from nicely regularized identity model, thereby badly affecting the evaluation metrics.

Figure 6.4: Expression fitting comparison. At inference time we jointly optimize for identity and expression latent codes while fitting partial point clouds obtained from single view depth maps to reconstruct full head geometry. We compare our method to widely-used state-of-the-art parametric head models including FLAME [28], NPM [19] and NPHM [20].

Figure 6.5: Interpolation in expression latent space. Smooth transition from source (extreme-left) to target (extreme-right) shapes shows the smoothness of the latent space of the proposed expression (deformation) model.

6.2.2 Latent Interpolation

This idea is quite similar to identity interpolation (see sec. 6.1.2). However, as expression can not exist without a base identity (or in other words deformation filed can not work without a base shape), we keep a base identity code fixed and interpolate over two expression latent codes. Fig 6.5 shows this interpolation result as a smooth transition between two expressions on a single identity which confirms the smoothness of our expression deformation field.

6.3 Ablation Study

We perform ablation study to determine the effects of different design choices for the proposed parametric head model. As our primary focus is modeling identity, we ablate only in the identity space keeping the expression neutral.

6.3.1 Ablation Setup

We start with a NPM [19] baseline and gradually increase complexity either by changing the architecture or adding more loss terms as described below:

- **NPM:** Same big MLP architecture for $\mathcal{F}_{\mathbf{id}}$ as of [19], trained with only $\mathcal{L}_{ss}$, $\mathcal{L}_{sn}$, $\mathcal{L}_{eik}$ and $\mathcal{L}_{nsp}$.
- **TP_3D:** $\mathcal{F}_{\mathbf{id}}$ network is implemented with tri-plane representation, all other things remain unchanged.
- **TP_3D_reg:** $\mathcal{L}_{edr}$, $\mathcal{L}_{tv}$ and $\mathcal{L}_{tp}$ loss terms are added to regularize the tri-planes.
- **TP_2D_reg:** Here, we introduce the renderer $\mathcal{R}$ and discriminator D and train the model with only $\mathcal{L}_{nmr}$, $\mathcal{L}_{g}$, $\mathcal{L}_{d}$, $\mathcal{L}_{edr}$, $\mathcal{L}_{tv}$ and $\mathcal{L}_{tp}$ without any direct 3D supervision.
- **Full model:** Finally, here we activate all the loss terms as listed in Table 4.1 along with the tri-plane based architecture.

6.3.2 Ablation Results

Figure 6.6 and Table 6.4 show the ablation results. Just replacing the big MLP in NPM [19] with tri-plane representation can capture more surface details especially in the hairs. However, if not regularized, it also creates random artifacts in the surrounding. Attempt of regularizing it with traditional density and weight norm based approaches

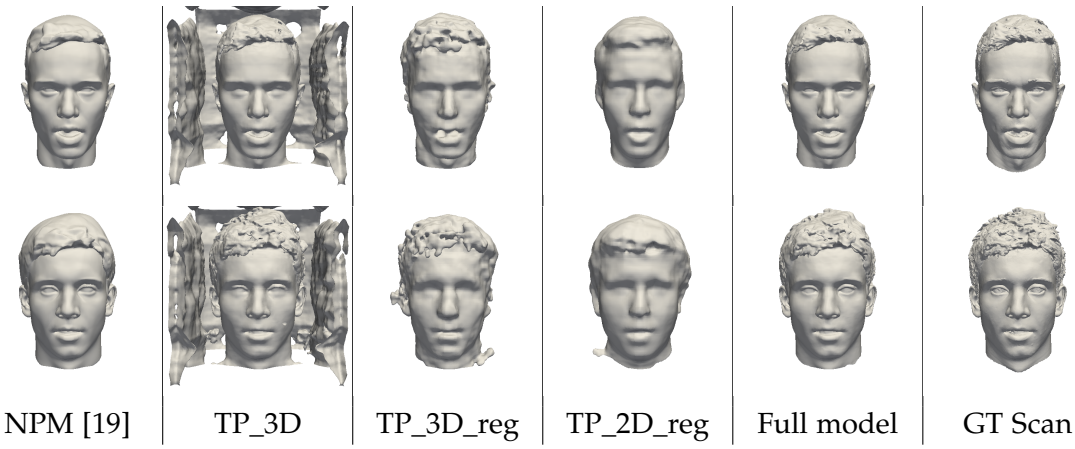


Figure 6.6: Identity fitting ablation results. Tri-plane (TP) representation is able to capture more details, but creates extra artifacts without regularization. While traditional 3D or 2D regularization produces over-smooth results, proposed GAN based 2D regularization along with 3D losses preserves high frequency details without creating any random artifacts.

produce over-smooth results. On the other hand, training only with 2D losses on the rendered normal maps can roughly reconstruct the shapes, however it still misses the fine surface details probably due to small resolution of the normal maps. Finally, the proposed approach which incorporates both 3D and 2D losses can reconstruct the surface details without creating random artifacts. The 3D losses help the model to learn more detail geometry where as the 2D adversarial losses act as an adaptive regularizer to constrain the latent space to avoid random artifacts.

6.4 Applications

Parametric head models can have a multitude of applications starting from 3D face tracking [88–91], face reenactment [92–94], voice puppetry [93, 95] and so on. We explore below one such application namely *expression transfer* which forms the basis of face reenactment and voice puppetry.

6.4.1 Expression Transfer

Here, the high level idea is to transfer the facial expression of a person to the face of another person. To this end, we fit two depth scans of two different person in different facial expressions to obtain corresponding identity and expression codes

Table 6.4: Effects of different losses and architectures on identity model. Results are calculated on the entire head region for fitting identities in neutral expression. Partial pointclouds are obtained by back-projecting random view near-frontal renderings.

Method	L_1-Chamfer ↓	N.C. ↑	F-Score @ 1 mm ↑
NPM [19]	1.361	0.924	0.957
TP_3D	17.226	0.869	0.642
TP_3D_reg	3.150	0.857	0.791
TP_2D_reg	3.785	0.866	0.747
Full model	**0.819**	**0.948**	**0.989**

$(\mathbf{z}_{id}^{(1)}, \mathbf{z}_{ex}^{(1)})$ and $(\mathbf{z}_{id}^{(2)}, \mathbf{z}_{ex}^{(2)})$. Now, as the identity and expression latent spaces are disjoint by construction, we can simply swap the expression codes $\mathbf{z}_{ex}^{(1)}$ and $\mathbf{z}_{ex}^{(2)}$ and query the networks with $(\mathbf{z}_{id}^{(1)}, \mathbf{z}_{ex}^{(2)})$ and $(\mathbf{z}_{id}^{(2)}, \mathbf{z}_{ex}^{(1)})$ to transfer the facial expression from the first person to the second person and vice versa. Fig 6.7 shows the expression transfer result. The expression code obtained from the source (left column) is used to deform the neutral targets (middle column) to the source expression pose. In the right column we show the result of the expression transfer.

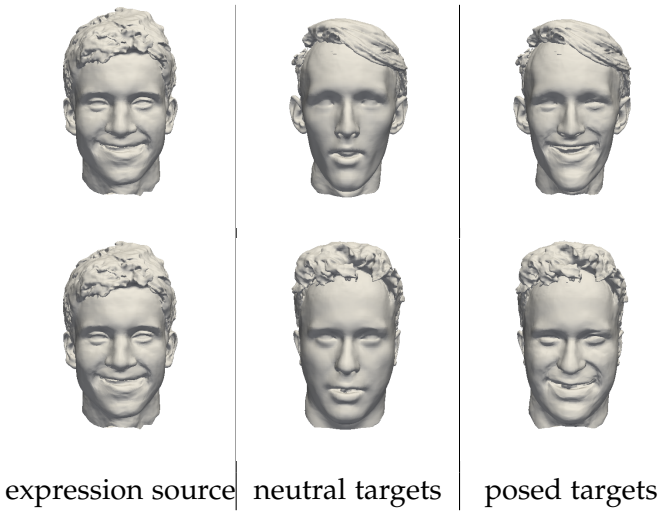


Figure 6.7: Expression transfer: left most column shows the source expression that needs to be transferred to the neutral pose identities in the middle column. Rightmost column shows the result of expression transfer.

7 Conclusion

In this thesis we present a neural parametric head model trained with 2D GAN losses along side the existing 3D losses. Our tri-plane based hybrid shape representation is able to nicely capture high frequency surface details when 2D GAN loss based adaptive regularization is imposed. In other words, our 3D losses help the model to learn high-frequency surface details whereas the 2D GAN loss puts the necessary guardrails to restrict the model falling into bad local minima and generate random artifacts. Both of our identity and expression models generates smooth latent spaces as observed from the latent interpolation experiments. Our identity model also runs faster as compared to other implicit models due to convolution based tri-plane representation instead of big MLPs.

However, as of now, the expression deformation model is not regularized with GANs and that results in occasional over deformation to the extent that can destroy the facial structures. Additionally, the rendering resolution for the GAN loss is a limiting factor as the current per-pixel ray shooting implementation of the volume renderer grows in quadratic order of the resolution. These two issues could probably be taken up in future directions of research on this. Also, instead of GAN based regularization on the 2D renderings, the latent codes could probably be directly refined using recent state-of-the-art denoising diffusion models to obtain more robust parametric head models.

Abbreviations

TUM Technical University of Munich

3DMM 3D Morphable Models

PCA Principal Component Analysis

NPFM Neural Parametric Face Models

MLP Multi Layer Perceptron

SDF Signed Distance Field

OF Occupancy Field

GAN Generative Adversarial Network

CNN Convolutional Neural Network

DNN Deep Neural Network

WGAN Wasserstein GAN

NeRF Neural Radiance Field

TV Total Variation

N.C. Normal Consistency

List of Figures

List of Tables

Bibliography

[1] M. Daneshmand, A. Helmi, E. Avots, F. Noroozi, F. Alisinanoglu, H. S. Arslan, J. Gorbova, R. E. Haamer, C. Ozcinar, and G. Anbarjafari, "3d scanning: A comprehensive survey," *arXiv preprint arXiv:1801.08863*, 2018.

[2] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer graphics forum*, Wiley Online Library, vol. 36, 2017, pp. 301–329.

[3] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva, "State of the art in surface reconstruction from point clouds," in *35th Annual Conference of the European Association for Computer Graphics, Eurographics 2014-State of the Art Reports*, The Eurographics Association, 2014.

[4] P. Joshi, W. C. Tien, M. Desbrun, and F. Pighin, "Learning controls for blend shape based realistic facial animation," in *ACM Siggraph 2006 Courses*, 2006, 17–es.

[5] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng, "Practice and theory of blendshape facial models.," *Eurographics (State of the Art Reports)*, vol. 1, no. 8, p. 2, 2014.

[6] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1999)*, ACM Press, 1999, pp. 187–194.

[7] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *2009 sixth IEEE international conference on advanced video and signal based surveillance*, Ieee, 2009, pp. 296–301.

[8] J. Booth, E. Antonakos, S. Ploumpis, G. Trigeorgis, Y. Panagakis, and S. Zafeiriou, "3d face morphable models" in-the-wild"," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 48–57.

[9] Z. Chen and T.-K. Kim, "Learning feature aggregation for deep 3d morphable models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 164–13 173.

[10] L. Tran, F. Liu, and X. Liu, "Towards high-fidelity nonlinear 3d face morphable model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1126–1135.

[11] L. Tran and X. Liu, "Nonlinear 3d face morphable model," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7346–7355.

[12] A. Brunton, T. Bolkart, and S. Wuhrer, "Multilinear wavelets: A statistical shape space for human faces," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 297–312.

[13] G. Bouritsas, S. Bokhnyak, S. Ploumpis, M. Bronstein, and S. Zafeiriou, "Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7213–7222.

[14] T. Yenamandra, A. Tewari, F. Bernard, H.-P. Seidel, M. Elgharib, D. Cremers, and C. Theobalt, "I3dmm: Deep implicit 3d morphable model of human heads," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 803–12 813.

[15] M. Zheng, H. Yang, D. Huang, and L. Chen, "Imface: A nonlinear 3d morphable face model with implicit neural representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 343–20 352.

[16] Y. Zheng, V. F. Abrevaya, M. C. Bühler, X. Chen, M. J. Black, and O. Hilliges, "Im avatar: Implicit morphable head avatars from videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 545–13 555.

[17] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[18] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[19] P. Palafox, A. Božič, J. Thies, M. Nießner, and A. Dai, "Npms: Neural parametric models for 3d deformable shapes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 695–12 705.

[20] S. Giebenhain, T. Kirschstein, M. Georgopoulos, M. Rünz, L. Agapito, and M. Nießner, "Learning neural parametric head models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 003–21 012.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[22] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[23] 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*, 2022, pp. 16 123–16 133.

[24] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, Springer, 2020, pp. 523–540.

[25] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway, "A 3d morphable model learnt from 10,000 faces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5543–5552.

[26] S. Ploumpis, H. Wang, N. Pears, W. A. Smith, and S. Zafeiriou, "Combining 3d morphable models: A large scale face-and-head model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 934–10 943.

[27] T. Bolkart and S. Wuhrer, "A groupwise multilinear correspondence optimization for 3d faces," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3604–3612.

[28] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, "Learning a model of facial shape and expression from 4d scans.," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 194–1, 2017.

[29] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt, "Sparse localized deformation components," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 1–10, 2013.

[30] L. Tran and X. Liu, "On learning 3d face morphable model from in-the-wild images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 157–171, 2019.

[31] R. Li, K. Bladin, Y. Zhao, C. Chinara, O. Ingraham, P. Xiang, X. Ren, P. Prasad, B. Kishore, J. Xing, *et al.*, "Learning formation of physically-based face attributes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3410–3419.

[32] H. Yang, H. Zhu, Y. Wang, M. Huang, Q. Shen, R. Yang, and X. Cao, "Facescape: A large-scale high quality 3d face dataset and detailed riggable 3d face prediction," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 601–610.

[33] S. Gong, L. Chen, M. Bronstein, and S. Zafeiriou, "Spiralnet++: A fast and highly efficient mesh convolution operator," in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019.

[34] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

[35] 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.

[36] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.

[37] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[38] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

[39] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345.

[40] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[41] X. Chen, T. Jiang, J. Song, J. Yang, M. J. Black, A. Geiger, and O. Hilliges, "Gdna: Towards generative detailed neural avatars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 427–20 437.

[42] P. Palafox, N. Sarafianos, T. Tung, and A. Dai, "Spams: Structured implicit parametric models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 851–12 860.

[43] E. Ramon, G. Triginer, J. Escur, A. Pumarola, J. Garcia, X. Giro-i-Nieto, and F. Moreno-Noguer, "H3d-net: Few-shot high-fidelity 3d head reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5620–5629.

[44] D. Wang, P. Chandran, G. Zoss, D. Bradley, and P. Gotardo, "Morf: Morphable radiance fields for multiview neural head modeling," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9.

[45] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3d hand pose estimation from monocular video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1793–1805, 2011.

[46] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018.

[47] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling," *ACM Transactions on Graphics (ToG)*, vol. 38, no. 5, pp. 1–19, 2019.

[48] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "Dist: Rendering deep implicit signed distance function with differentiable sphere tracing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2019–2028.

[49] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021.

[50] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[51] Y. Wang, I. Skorokhodov, and P. Wonka, "Hf-neus: Improved surface reconstruction using high-frequency details," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1966–1978, 2022.

[52] Y. Wang, I. Skorokhodov, and P. Wonka, "Pet-neus: Positional encoding tri-planes for neural surfaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 598–12 607.

[53] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction," *Advances in neural information processing systems*, vol. 35, pp. 25 018–25 032, 2022.

[54] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu, "Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3295–3306.

[55] R. A. Rosu and S. Behnke, "Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8466–8475.

[56] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, *et al.*, "State of the art on neural rendering," in *Computer Graphics Forum*, Wiley Online Library, vol. 39, 2020, pp. 701–727.

[57] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Differentiable rendering: A survey," *arXiv preprint arXiv:2006.12057*, 2020.

[58] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, *et al.*, "Advances in neural rendering," in *Computer Graphics Forum*, Wiley Online Library, vol. 41, 2022, pp. 703–735.

[59] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[60] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, PMLR, 2017, pp. 214–223.

[61] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[62] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[63] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[64] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[65] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.

[66] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *Advances in neural information processing systems*, vol. 33, pp. 12 104–12 114, 2020.

[67] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 852–863, 2021.

[68] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[69] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[70] P. Henzler, N. J. Mitra, and T. Ritschel, "Escaping plato's cave: 3d shape from adversarial rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9984–9993.

[71] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "Hologan: Unsupervised learning of 3d representations from natural images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7588–7597.

[72] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 154–20 166, 2020.

[73] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 453–11 464.

[74] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[76] chriscummingshrg. "Signed distance fields part 2: Solid geometry." (2018), [Online]. Available: `https://shaderfun.com/2018/03/25/signed-distance-fields-part-2-solid-geometry/` (visited on 11/19/2023).

[77] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[78] B. K. H. S. J. Korndörfer, U. G. M. Stamminger, and B. Keinert, "Enhanced sphere tracing," *STAG: Smart Tools & Apps for Graphics*, vol. 8, no. 4, 2014.

[79] V. VACHHAR. "Iridescent crystal with raymarching and signed distance fields." (2023), [Online]. Available: `https://varun.ca/ray-march-sdf/` (visited on 11/27/2023).

[80] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.

[81] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.

[82] J. R. Shue, E. R. Chan, R. Po, Z. Ankner, J. Wu, and G. Wetzstein, "3d neural field generation using triplane diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 875–20 886.

[83] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" In *International conference on machine learning*, PMLR, 2018, pp. 3481–3490.

[84] X. Chen, Y. Zheng, M. J. Black, O. Hilliges, and A. Geiger, "Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 594–11 604.

[85] E. B. Saff and A. B. Kuijlaars, "Distributing many points on a sphere," *The mathematical intelligencer*, vol. 19, pp. 5–11, 1997.

[86] D. P. Hardin, T. Michaels, and E. B. Saff, "A comparison of popular point configurations on $\mathbb{S}^2$," *arXiv preprint arXiv:1607.04590*, 2016.

[87] T. Bolkart and T. Li. "Flame: Articulated expressive 3d head model." (2022), [Online]. Available: `https://github.com/Rubikplayer/flame-fitting` (visited on 12/07/2023).

[88] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3d face reconstruction, tracking, and applications," in *Computer graphics forum*, Wiley Online Library, vol. 37, 2018, pp. 523–550.

[89] Z. Wen *et al.*, "Capturing subtle facial motions in 3d face tracking," in *Proceedings Ninth IEEE International Conference on Computer Vision*, IEEE, 2003, pp. 1343–1350.

[90] T. S. Jebara and A. Pentland, "Parametrized structure from motion for 3d adaptive feedback tracking of faces," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 144–150.

[91] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 2610–2617.

[92] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2387–2395.

[93] R. Huang, P. Lai, Y. Qin, and G. Li, "Parametric implicit face representation for audio-driven facial reenactment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 759–12 768.

[94] K. Yang, K. Chen, D. Guo, S.-H. Zhang, Y.-C. Guo, and W. Zhang, "Face2face ρ: Real-time high-resolution one-shot face reenactment," in *European conference on computer vision*, Springer, 2022, pp. 55–71.

[95] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, and M. Nießner, "Neural voice puppetry: Audio-driven facial reenactment," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, Springer, 2020, pp. 716–731.