FreeStyleGAN: Free-view Editable Portrait Rendering with the Camera Manifold

THOMAS LEIMKÜHLER and GEORGE DRETTAKIS, Université Côte d’Azur and Inria, France

Fig. 1. We introduce a new approach that generates an image with StyleGAN defined by a precise 3D camera. This enables faces synthesized with StyleGAN to be used in 3D free-viewpoint rendering, while also allowing all semantic editing provided by GAN methods. Our method takes as input multiple views of a person (examples in a), used to reconstruct a coarse 3D mesh (b). To render a novel view, we identify the closest camera which corresponds to an image that StyleGAN can generate (c). We lift this view to 3D and obtain free-viewpoint renderings with arbitrary camera models, which allows the integration of our renderings into synthetic 3D scenes (d). We inherit the high-quality semantic editing capabilities from StyleGAN (e) smile or aging), and enable stereoscopic rendering (f). Our method can be integrated into any rendering pipeline and - for the first time - marries generative image modeling with traditional rendering.

Current Generative Adversarial Networks (GANs) produce photorealistic renderings of portrait images. Embedding real images into the latent space of such models enables high-level image editing. While recent methods provide considerable semantic control over the (re-)generated images, they can only generate a limited set of viewpoints and cannot explicitly control the camera. Such 3D camera control is required for 3D virtual and mixed reality applications. In our solution, we use a few images of a face to perform 3D reconstruction, and we introduce the notion of the GAN camera manifold, the key element allowing us to precisely define the range of images that the GAN can reproduce in a stable manner. We train a small face-specific neural implicit representation network to map a captured face to this manifold and complement it with a warping scheme to obtain free-viewpoint novel-view synthesis. We show how our approach – due to its precise camera control – enables the integration of a pre-trained StyleGAN into standard 3D rendering pipelines, allowing e.g., stereo rendering or consistent insertion of faces in synthetic 3D environments. Our solution proposes the first truly free-viewpoint rendering of realistic faces at interactive rates, using only a small number of casual photos as input, while simultaneously allowing semantic editing capabilities, such as facial expression or lighting changes.

1 INTRODUCTION

Recent Generative Adversarial Networks (GAN) can generate stunningly realistic images of faces from a latent code [Karras et al. 2019, 2020b]. Combined with recent methods to project a real photo to such a latent code [Abdal et al. 2019; Tewari et al. 2020] they also allow semantic image editing, e.g., controlling facial expression or relighting [Abdal et al. 2021; Deng et al. 2020; Härkönen et al. 2020]. This opens the ground-breaking potential of replacing the expensive and complex process of capturing [Debevec et al. 2000], animating and rendering human faces [Seymour et al. 2017] by simply using GAN-generated images based on photos. Unfortunately, two central components are missing for this process to become reality: currently there is no way to consistently generate a realistic GAN image of a person in a Computer Graphics (CG) shot as defined by a precise and complete 3D camera (3D position, 3D rotation, and field of view), and current methods can only generate a limited set of viewpoints. We present the first solution that solves these problems, allowing highly realistic, editable faces to be correctly rendered for any given 3D camera, and thus seamlessly integrated into standard CG imagery. Our method opens the way to using such highly realistic, editable face renderings for a wide range of applications such as virtual or augmented reality, immersive video-conferencing or even immersive remote training.

We build on the StyleGAN architecture [Karras et al. 2019, 2020b] that is inherently a 2D image generator. Recent methods [Abdal et al. 2021; Deng et al. 2020; Tewari et al. 2020a] can render head poses parameterized by two angles, but have no way to generate an image for a specific and complete 3D camera, ignoring at least five degrees of freedom (DoF). This results in a small subspace of 3D camera poses; the nature and limits of this subspace have never been precisely defined, much less extended.
Our first challenge is thus to characterize the subspace of 3D camera parameters that StyleGAN can represent, enabling the design of a method for free viewpoint face rendering. We analyze this subspace, carefully defining what we call the camera manifold. This definition is inherently linked to the alignment step performed on the face database used to train the StyleGAN portrait model, severely constraining the 2D locations of eyes and mouth. We also determine the boundaries of the camera manifold, allowing us to project a free-view camera onto it.

Providing full 3D capabilities for StyleGAN requires at least some 3D information; we use a few (10-25), casually captured photos of a face as input and generate coarse face geometry and calibrated cameras to guide part of our method.

Given these photos of a person, our second challenge is finding the latent code to provide to StyleGAN so it can generate the corresponding image. To do this, we need our camera manifold: we first find the closest view on the manifold, then train a small, per-face latent representation network to find the latent vector for a given manifold view. We train the network with calibrated input views and image-based renderings using the reconstructed face geometry.

However, many free-viewpoint camera poses are not on the manifold. This raises our third and final challenge, i.e., to provide a method allowing any such view to be synthesized. We do this by warping from the closest camera on the manifold to the desired novel view, using the coarse reconstructed geometry.

These three steps allow us to introduce a fully-operational, interactive system with precise 3D camera control while fully exploiting StyleGAN-quality photorealistic face synthesis and corresponding manipulations (see Fig. 1). We consider a pre-trained and fixed StyleGAN2 model in this work, which allows us to run our per-face pre-processing pipeline within 45 minutes on a single GPU.

In summary our contributions are:

- An in-depth study and quantitative definition of the subspace of camera poses that StyleGAN can robustly synthesize, that we call camera manifold.
- A method to generate a realistic StyleGAN face based on a precisely defined 3D camera pose on the manifold.
- A warping scheme to render any camera pose freely defined in 3D, fully consistent with semantic editing methods.

Our system allows interactive synthesis of realistic faces, allowing free-viewpoint navigation in 3D while moving around a face casually captured with a handful of photos. We demonstrate interactive sessions with several captured faces, also performing semantic edits in a view-consistent manner – such as changing facial expression, opening/closing eyes/mouth – and also camera model manipulation (e.g., the Vertigo effect), seamless integration with synthetic 3D environments, and the first ever stereoscopic GAN renderings.

2 RELATED WORK

Our solution touches on several vast domains: Generative adversarial networks (GANs), image-based rendering (IBR), face models and portrait rendering, and constrained camera models. In what follows, we only review the most closely related work to ours.

2.1 Image Synthesis and Editing with GANs

Generative adversarial networks [Goodfellow et al. 2014] build a statistical model trained to mimic the distribution of training data (often images). In just a few years, GANs have evolved to produce truly photorealistic images at high resolutions. Currently, StyleGAN [Karras et al. 2020a, 2019, 2020b] marks the state of the art in unconditional image generation in narrow domains such as faces, cars or cats. Importantly, a single forward pass through the generator provides photo-realistic imagery, enabling interactive rendering.

Latent codes represent semantically meaningful and reasonably disentangled concepts for many neural models [Karras et al. 2019; Radford et al. 2016; Upchurch et al. 2017]. In StyleGAN, an informed manipulation of latents results in high-level image changes, e.g., in pose, lighting, facial expression, age, gender, etc. The manipulations can correspond to linear [Härkönen et al. 2020; Shen et al. 2020; Tewari et al. 2020a] or non-linear [Abdal et al. 2021; B R et al. 2021] paths in latent space, while disentangled controls can be jointly trained with the generator [Deng et al. 2020]. Our approach is fully compatible with semantic manipulation, such as the method of Härkönen et al. [2020], allowing us to dynamically edit our renderings.

All methods that rely on latent manipulations to change semantic attributes are necessarily bound to the span of the training data, especially when an alignment step is used [Jahanian et al. 2020]. In practice, this means that only a subset of possible camera poses can be synthesized by StyleGAN. Portrait alignment restricts cameras to 3 DoF: two rotations and field of view. While previous work considers only the two rotations, ours is the first method that allows free view navigation with complete (7+ DoF) camera models.

Recent work recovers latent codes from a single image [Abdal et al. 2019; Richardson et al. 2021; Zhu et al. 2016] enabling semantic editing of real photos with unprecedented quality [Abdal et al. 2020; Tewari et al. 2020b]. In contrast we are the first to devise a parameterized embedding of multiple views of the same face, which requires a mapping from 3D cameras to latents.

The StyleGAN latent space can be used to represent any image with high fidelity – even ones far outside the training data distribution [Abdal et al. 2019] – at the cost of low-quality edits. Consequently, researchers have investigated regularizations to enforce latents closer to the original distribution. This has been done by considering the distribution of the latents directly [Tewari et al. 2020b; Wulff and Torralba 2020; Zhu et al. 2020], or by utilizing semantic knowledge about the images generated [Richardson et al. 2021]. We use a prior as well to find a good compromise between photorealism and identity preservation.

In recent work, GANs have been used as a multi-view generator to aid inverse rendering [Zhang et al. 2021a, or to estimate 3D cameras, shape, and lighting [Pan et al. 2021; Shi et al. 2021]. A different line of work has devised GANs to incorporate 3D information directly, e.g., using 3D geometry [Zhu et al. 2018] or 3D semantic occupancy [Chen et al. 2021] followed by image-to-image translation for final output, or using volumetric generator layers followed by projection for rotations and scaling [Nguyen-Phuoc et al. 2019]. GRAF [Schwarz et al. 2020] and pi-GAN [Chan et al.
produce a 3D radiance field, which can then be rendered using volume rendering. Such GANs with explicit 3D-awareness are promising, but so far fail to generate high-quality results. In contrast to these solutions, we lift a pre-trained 2D GAN to 3D, and thus inherit the advantages of 2D GANs (high quality, high resolution, robust training procedures, etc.) while enhancing the method with precise 3D camera control. 3D-aware generative modeling can be improved by jointly learning the distribution of camera parameters [Niemeyer and Geiger 2021]. Similarly, we estimate this distribution for a pre-trained StyleGAN model to refine our camera manifold.

2.2 Multi-view Free-viewpoint Image-based Rendering
Traditional IBR usually relies on explicit proxy geometry from structure-from-motion/multi-view stereo [Schonberger and Frahm 2016]. Unstructured lumigraph rendering (ULR) [Buehler et al. 2001] uses a geometric proxy to heuristically blend multi-view images. We use this method for training. Recently, neural rendering has made significant advances in the quality of novel view synthesis [Tewari et al. 2020c]. Geometry-based methods use deep learning to improve the quality of novel view synthesis [Hedman et al. 2018; Riegler and Koltun 2020], but are restricted to static scenes. Single-object mesh-based neural rendering approaches [Thies et al. 2019a,b] also use proxy geometry for reprojection followed by neural refinement. We share the methodology of using a geometric proxy, and – like these solutions – enable true free-viewpoint rendering. However, we are the first to exploit geometry to steer a GAN for free-viewpoint face rendering.

Neural Volumes [Lombardi et al. 2019] learn a volumetric scene representation, which is rendered using ray marching. They can handle video sequences, but require synchronized and calibrated cameras. Flexible neural scene representations allow high-level scene editing [Li et al. 2020], but they are restricted to changes captured in the input views. In contrast, our approach exploits the expressive space of variations captured in a generative model.

Recent solutions include neural radiance fields [Mildenhall et al. 2020; Zhang et al. 2020], that use implicit scene representations [Sitzmann et al. 2019] and volume rendering techniques for view synthesis. Extensions allow deforming objects [Park et al. 2021] or changing lighting [Martin-Brualia et al. 2021; Srinivasan et al. 2021], offering some degree of semantic control [Zhang et al. 2021b]. These methods have long rendering times (usually tens of seconds per frame) and most of them need dense capture; in contrast, we exploit the power of GANs allowing interactive rendering and use of sparse capture (10-25 images).

2.3 Face Models and Portrait Rendering
Manually created face rigs [Seymour et al. 2017] are stunningly realistic, but require skilled visual effects artists working long hours. 3D morphable models (3DMMs) [Blanz and Vetter 1999; Egger et al. 2020] are generative models that allow fine-grained control over shape and texture of an object, often used for faces. Such models can be augmented with deformation-dependent texture maps [Matthews and Baker 2004] and photometric capture to also model appearance variation [Smith et al. 2020]. 3DMMs offer maximal control over shape, facial expression, texture, etc., but tend to lack photo-realism and typically only model the face region (hair, neck, etc. are not included) even though recent advances [Yenamandra et al. 2021] are starting to consider entire heads.

Another approach to capturing faces involves complex multi-view setups [Beeler et al. 2010; Bi et al. 2021; Ghosh et al. 2011; Lombardi et al. 2018; Wei et al. 2019]. In contrast to these methods, we only need a few casually captured images as input and can perform arbitrary semantic edits.

Face re-enactment is an active area of research. Kim et al. [2018] transfer head pose and facial expressions from a source video to target video. Single [Geng et al. 2018; Siarohin et al. 2019; Zakharov et al. 2020] or multiple [Wang et al. 2019; Zakharov et al. 2019] source images can be used in conjunction with facial feature point extraction of a target sequence to hallucinate novel views and facial expressions. Most of these methods have only been demonstrated with very limited pose variation. Our approach allows true free-viewpoint navigation while reasonably preserving identity.

Rao et al. [2020] shares some similarities with our approach, since they also process facial animations in a canonical frame, and lift the face to 3D. However, they use StyleGAN to synthesize only the mouth, and require dense capture.

Face rotation using a single image [Nagano et al. 2019; Xu et al. 2020; Zhou et al. 2020] usually proceeds by first estimating geometry, followed by a neural rendering step. While results are impressive given that only a single image is used, the variation of attainable poses tends to be limited. Subtle perspective corrections of portraits – including stereo image synthesis – can be addressed using image warping [Fried et al. 2016; Zhao et al. 2019]. We also use warping, but mostly for global transformations, while perspective effects are generated by manipulating latent codes, successfully overcoming typical problems arising from disocclusions, hair, and the lack of global consistency.

Specialized variants of neural radiance fields for portrait synthesis [Gafni et al. 2021; Wang et al. 2021] produce free-viewpoint results of high visual quality and can handle facial animations, but inherit the problems of neural radiance fields: dense capture setup and long rendering times. Gao et al. [2020] combine neural radiance fields with strong priors to allow portrait rendering from a single view, achieving some effects we handle (e.g., the Vertigo-effect), but the range of achievable poses and visual quality are limited.

2.4 Constrained Cameras
Camera control in interactive applications is important for viewpoint computation, motion planning and editing. See Christie et al. [2008] for an in-depth survey. Our camera manifold builds on fixed projected locations, related to 2-object composition problems [Blinn 1988]: How to place a camera such that two objects are at prescribed locations in the image plane? Lino and Christie [2012] observed that the solution to this problem is a manifold surface in camera parameter space; they later generalized this to the Toric space [Lino and Christie 2015], allowing interactive exploration and optimization of viewpoints with multiple geometric constraints. Similarly, we parameterize the 7 DoF camera space using a 3D manifold. However, their approach is more general, with a more complex solution space,
while our simpler problem has more constraints and is amenable to efficient convex optimization.

3 OVERVIEW
Our method takes as input 10-25 photographs of a person and allows free 3D viewpoint rendering of the face with arbitrary camera models based on a pre-trained StyleGAN2 portrait model [Karras et al. 2020b]. An overview of our method is shown in Fig. 2.

3.1 Method
Our key observation is that the StyleGAN portrait model can synthesize a quantifiable range of pose variations: All generated images have the property that the 2D locations of eyes and mouth are severely constrained. Based on this observation, we define a space of camera parameters we call the camera manifold, which results in images in the canonical configuration that respects this constraint (Sec. 4). We then analyze and delimit the range of the manifold, i.e., the subspace of camera poses within our manifold parameterization that StyleGAN can synthesize successfully.

We start from a few photos and use standard 3D reconstruction to obtain calibrated cameras and mesh of the face, as described below. To render the face from an unconstrained novel view V, we use the steps described next.

We first project V to the closest camera V on the manifold, thereby reducing the problem to an in-domain rendering task. We then use an implicit latent representation network, which maps physically meaningful coordinates on the camera manifold to a StyleGAN latent code (Sec. 5). The network functions as a parameterized embedding of the face and is trained in a supervised fashion using aligned input views and simple IBR renderings [Buehler et al. 2001] in a progressive training schedule. The entire pre-processing pipeline is face-specific and takes 45 minutes.

The final step is to move from the StyleGAN-generated manifold view V to free view V (Sec. 6). Given that both views correspond to physically meaningful cameras, we use the face mesh to establish a dense flow field which describes how the image corresponding to the manifold view V needs to be deformed to obtain the image corresponding to (unconstrained) novel view V.

3.2 Preprocessing
To incorporate 3D awareness, our method relies on calibration of the input cameras and an approximate geometric representation of the face to be rendered. Neither calibration nor geometry need to be accurate for our method to produce convincing free-viewpoint images.

We calibrate cameras and create the geometric proxy using off-the-shelf software [CapturingReality 2016], resulting in a reconstructed triangle mesh, the calibrated cameras and undistorted input images (see supplemental for details). Quality is satisfactory using 10-25 photos, despite casual sequential capture without a rig.

4 THE CAMERA MANIFOLD
To allow free-viewpoint rendering of captured faces, we first need to understand the native capabilities of StyleGAN in terms of viewpoint synthesis. To this end, we define the subset of cameras that allows the generation of valid StyleGAN2 images. The portrait model published by the authors [Karras et al. 2020b] is trained on the FFHQ dataset [Karras et al. 2019]. The alignment of this dataset is a key component of the model, greatly improving quality but limiting the variety of images that can be synthesized. This is because the photos lie on a relatively narrow image manifold [Jahanian et al. 2020], implicitly defined by this alignment process based on facial features. We define our camera manifold based on the FFHQ alignment procedure (Sec. 4.1), estimate the range of the manifold in which StyleGAN can generate realistic images (Sec. 4.2), explain how to project a free-viewpoint camera to the closest manifold camera (Sec. 4.3), and finally present the process of aligning a captured face to the canonical coordinate system we use (Sec. 4.4).

The FFHQ dataset was constructed by first collecting images from Flickr, followed by a cleanup step and alignment, where first 68 facial features are found [Kazemi and Sullivan 2014] (blue dots in Fig. 3a). Then the eye and mouth features are aggregated to obtain representative eye positions x_l and x_r, as well as a mouth position x_m in the image (green dots in Fig. 3a). From these three points a square crop window is computed by determining its center c (also using the eye midpoint x_c) as well as its orientation and size s, all illustrated in Fig. 3b. Exact formulas are given in the supplemental. Given the crop window geometry, the original image is re-sampled to obtain the final aligned output image (Fig. 3c).

4.1 Defining the Camera Manifold
The alignment procedure described above guides the definition of our camera manifold. We first write the alignment as a mapping from facial feature positions to a 2D similarity transform: F(x_l, x_r, x_m) ∈ R^6 → (R^2) → R^2. The similarity transform maps pixel locations from the unaligned to the aligned image. We observe that F is deterministic and C^0-continuous. Since F is an
alignment procedure, it exhibits an infinite number of invariances. In our model, we require two invariances for an unambiguous mapping and we consider the two (arbitrary) points (Fig. 3b)

\[ x_- = [0, 0, 5]^T = F(x_l, x_r, x_m)(c - s) =: F_-(x_l, x_r, x_m) \]

and

\[ x_+ = [1, 0, 5]^T = F(x_l, x_r, x_m)(c + s) =: F_+(x_l, x_r, x_m), \]

where the definitions of \( F_- \) and \( F_+ \) exploit the fact that \( c \) and \( s \) are themselves functions of the facial feature positions.

Our method relates the fixed positions \( x_- \) and \( x_+ \) to feasible camera parameters, thus connecting 2D image alignment to 3D-aware image formation. To map a 3D camera pose to aligned image space, we solve the following problem: Assuming fixed eye and mouth positions in 3D space, which combinations of intrinsic and extrinsic camera parameters yield the prescribed projection locations \( x_- \) and \( x_+ \) in image space?

For the purpose of defining our camera manifold, we assume that StyleGAN-generated images are modeled as being captured with a perspective pinhole camera. Note however, that we can generate images using arbitrary camera models. We further assume, without loss of generality, that the left and right 3D eye positions are

\[ p_l = [-1, 0, 0]^T \quad \text{and} \quad p_r = [1, 0, 0]^T, \quad (1) \]

respectively. We further assume we have access to the 3D mouth position \( p_m \) (see supplemental for our exact definition of a frontal pose). In Sec. 4.4 we show how to align a general pose of a face capture to this canonical setting. Note that this alignment does not impose any restrictions on face shape.

A full camera model with 7 DoF gives too much freedom for our manifold, as many camera parameters correspond to an image with the face lying outside the frame. We therefore, as a first step, restrict ourselves to a trackball camera model:

\[ \hat{A}(\theta, \phi, d, \psi) = P(\psi)T(0, 0, d)R(\theta, \phi, 0), \quad (2) \]

where \( R \in \mathbb{R}^3 \rightarrow \mathbb{R}^3 \) is a rotation parameterized by three Euler angles, \( T \in \mathbb{R}^3 \rightarrow \mathbb{R}^3 \) is a translation parameterized by three offset coordinates, and \( P \in \mathbb{R}^3 \rightarrow \mathbb{R}^2 \) is a perspective projection with field of view \( \psi \). The free parameters \( \theta, \phi, d, \) and \( \psi \) correspond to horizontal and vertical rotations around the eye midpoint, distance to the eye midpoint, and field of view, respectively (Fig. 4a). \( \hat{A} \) corresponds to an un-aligned image in the spirit of Fig. 3a: The face lies within the frame, but is not aligned in general.

In a second step, we enable fixed projection locations as follows:

\[ \hat{A}(\theta, \phi, d) = P(\psi)R(\hat{\alpha}, \hat{\beta}, \hat{\psi})T(0, 0, d)R(\theta, \phi, 0). \quad (3) \]

Here, we add an additional rotation before the projection and exclude \( \psi \) from the list of free parameters. We refer to the free parameters \( m = [\theta, \phi, d]^T \in \mathbb{R}^3 \) as manifold coordinates. We define the space of \( m \) in Sec. 4.2. In terms of the camera position, \( m \) corresponds to spherical coordinates. Intuitively, the coefficients \( \hat{c} = [\hat{\alpha}, \hat{\beta}, \hat{\psi}] \) steer a rotation (via \( \hat{\alpha}, \hat{\beta}, \hat{\psi} \)) and scaling (via \( \psi \)) without changing the camera position. Excluding degenerate cases, which we avoid as described in Sec. 4.2, for any fixed set of manifold coordinates, there exists a coefficient vector \( \hat{c}(m) \) to position two 3D points to arbitrary 2D projected image locations.

We now have all the machinery in place to solve our alignment problem (Fig. 4b), i.e., given a manifold coordinate \( m \), find a coefficient vector \( \hat{c}(m) \) that simultaneously satisfies the four equations:

\[ \begin{bmatrix} F_-(\hat{A}(m)p_l), \hat{A}(m)p_r, \hat{A}(m)p_m, F_+(\hat{A}(m)p_l), \hat{A}(m)p_r, \hat{A}(m)p_m \end{bmatrix} = x_- \quad (4) \]

where \( \hat{A}(m)p \) denotes the projection of 3D point \( p \) to the screen using camera \( \hat{A}(m) \). Due to the nonlinear nature of the equations, obtaining an analytic solution is challenging. Instead, we solve numerically for the least-squares solution using the Levenberg-Marquardt algorithm, which usually converges after 35 iterations. The optimization takes about 10 milliseconds in our Python implementation and produces temporally stable results. We therefore solve the equations on the fly.

Our model allows free positioning of the camera via \( m \). However, all rotational DoF as well as the field of view are automatically adjusted to ensure aligned portrait images. Our cameras therefore lie on a 3D manifold in the 7D parameter space. The coordinates \( \theta \) and \( \phi \) resemble yaw/pitch parameterizations of face pose in previous work, but by construction also include translational and scaling components which previous approaches do not model. Manipulation of \( d \) results in the Vertigo effect: Increasing \( d \), i.e., moving the camera...
away from the face, is compensated by a corresponding decrease in the field of view, and vice versa.

4.2 Range
The above definition does not restrict camera position \( m \), allowing arbitrary views of the head, e.g., profile or rear views. We observe, however, that StyleGAN-generated images have smaller variations in camera pose, consistently showing front or moderate oblique angle views. We next determine the boundaries or range of our camera manifold. This mainly concerns the rotation parameters \( \theta \) and \( \phi \); the Vertigo-type variations induced by \( d \) are small enough to ignore in this context, when avoiding very small \( \psi \). Therefore, we simply enforce \( d \geq 10 \), i.e., camera positions at least five inter-ocular distances away from the face.

To quantify the range of camera positions in the \( \theta\phi \)-plane, we first produce 10k random facial images using the StyleGAN2 generator with default truncation parameter \( \psi = 0.5 \). We then use the method of Bulat and Tzimopoulos [2017] to infer estimates of 3D eye and mouth positions per image, which we convert to corresponding \( \theta\phi \)-tuples (details on calibration in supplemental). When plotting the resulting distribution (dots in Fig. 5a), we see that the images indeed concentrate around the frontal pose \( \theta = \phi = 0 \), but show a strong eye-shaped anisotropy, while being close to symmetric around the axis \( \theta = 0 \). These observations suggest that we can model the boundary of this distribution using two parabolas of the form \( \phi = a\theta^2 + b \). To obtain the coefficients \( a \) and \( b \), we first convert the samples into a probability distribution \( p_{\theta\phi} \) (Fig. 5b). We use kernel density estimation on a regular grid of size 128×96 capturing \( \theta \in [-40°, 40°] \) and \( \phi \in [-30°, 30°] \), utilizing a Gaussian kernel with a standard deviation of three grid cells. We then determine the iso-lines \( p_{\theta\phi} = 0.01 \max(p_{\theta\phi}) \) (Fig. 5c) and perform two least-squares parabola fits. Below, we give the result of our fit for the upper and lower boundary curves (parabolas in Fig. 5a):

\[
c_u(\theta) = -0.024\theta^2 + 20.00 \quad \text{and} \quad c_l(\theta) = 0.010\theta^2 - 14.60.
\]

A valid coordinate in our model satisfies \( c_l(\theta) \leq \phi \leq c_u(\theta) \).

4.3 Projecting a Camera to the Manifold
Finding the closest camera on the manifold for a free-viewpoint camera is easy in our model (Fig. 6): We take the camera position in spherical coordinates as manifold coordinate \( m \) and solve Eq. 4.

However, arbitrary camera positions do not necessarily correspond to valid manifold coordinates as defined in the previous section. In these cases, we find the closest valid manifold coordinate (Fig. 6b), using an analytic solution for this projection to the valid range (details in supplemental). Note that only in this case the camera position changes and parallax is induced in the image.

4.4 Alignment to Canonical Coordinates
Our head reconstruction can have arbitrary global 3D scene scale, positioning and orientation. Since our camera manifold assumes the eyes to be at defined 3D positions \( p_l \) and \( p_r \), we perform 3D alignment to the canonical configuration (Eq. 1) [Gao et al. 2020; Rao et al. 2020], making our method independent of the reconstruction algorithm.

The locations of the eyes and mouth on the mesh are required for alignment. While 3D facial feature location algorithms exist [Bowyer et al. 2006], our multi-view data provided stable results despite possibly strong reconstruction noise. Similar to the FFHQ alignment, we use the method of Kezemi and Sullivan [2014] to obtain facial feature points on the face for each input view and compute representative 2D eye and mouth locations (Fig. 3a). We then re-project the 2D feature points onto the mesh and average the re-projected features of all input views to obtain a robust estimate of the eye and the mouth positions in 3D.

We now seek to find a similarity transform that maps the estimated 3D eye positions to their targets \( p_l \) and \( p_r \). To obtain a unique solution and remain consistent with the definition of the camera manifold, we further enforce a frontal pose in terms of vertical orientation, as detailed in the supplemental. The transformation matrix is obtained using the Levenberg-Marquardt algorithm and then used to transform both the mesh and the input camera parameters.

We also process the input images: We observe that the embedding of images into the GAN latent space (Sec. 5) gives results of higher visual quality when the background is smooth. Further, such "simple"
backgrounds are easier to integrate into a synthetic scene. We therefore blur the background of the input images. We use the method of Ke et al. [2020] to extract a foreground matte. Then we apply a strong Gaussian filter (we use a σ of one tenth the image width) to the background region [Knutsson and Westin 1993] and compose the foreground on top. Finally, we 2D-align all input images using the FFHQ alignment.

Recall, that in the canonical coordinate system, any camera’s manifold coordinate m corresponds to the camera’s position in spherical coordinates. Consequently, the manifold coordinates of the 3D-aligned input cameras – provided they are in the valid range – correspond to the 2D-aligned images in the sense of our model.

5 Mapping Manifold Coordinates to StyleGAN Latent Codes
We now have the manifold coordinates that correspond to a given camera pose. We next find the StyleGAN latent code that corresponds to a given manifold coordinate, by training a small per-face implicit latent representation network. This network allows us to move on the camera manifold.

5.1 StyleGAN Terminology
StyleGAN [Karras et al. 2019] maps normally distributed random samples \( z \in \mathbb{R}^{512} \) to an intermediate latent vector \( w \in \mathbb{R}^{512} \) using a learned mapping \( w = H(z) \). The space of \( w \)'s is commonly referred to as \( W \). The vector \( w \) controls feature normalizations in 18 layers of the generator network \( G \), which produces the final image \( I = G(w) = G(H(z)) \). It has been observed [Karras et al. 2019][Abdal et al. 2019] that the expressivity of the generator is much higher, when different \( w \) are fed to the generator layers. In the general case, 18 different sets of latents \( w^* \in \mathbb{R}^{18 \times 512} = W^* \) can be used.

5.2 Method
By construction, all views \( \hat{V} \) on the camera manifold can be rendered by finding a corresponding latent vector \( w^* \in W^* \). We therefore seek a mapping \( M : \mathbb{R} \rightarrow W^* \) from manifold coordinates to latents, such that, given manifold coordinates \( m \), we obtain \( \hat{V} = G(w_M) = G(M(m)) \).

Recall that, in contrast to all previous methods, \( m \) is an exact meaningful physical 3D quantity.

We found a face-specific mapping \( M \) to provide highest-quality results. Therefore, in analogy to recent works on implicit neural representations [Genova et al. 2019; Sitzmann et al. 2020, 2019], we refer to \( M \) as an implicit latent representation network. It has been observed that the latents fed to the earlier StyleGAN layers correspond to coarse-scale image properties including face pose, while later layers add medium- to small-scale features to the images [Karras et al. 2019]. We therefore define a constrained architecture: For layers 0-5 we parameterize \( M \) with a small multi-layer perceptron (MLP) per layer, each mapping raw manifold coordinates \( m \) to a 512-D latent vector. Due to the low-frequency behaviour of pose changes, we do not use Fourier features [Tancik et al. 2020]. For the MLPs, we found two hidden layers with 32 features and leaky ReLU activation functions to be sufficient for our application. For the remaining layers 6-17 we directly optimize for static latents, which ensures a consistent output for different \( m \) (Fig. 7).

5.3 Training Data
Thanks to our multi-view setup and the geometric reconstruction, we can train \( M \) in a supervised fashion using training tuples \( \left\{ (m_k, \hat{V}_k) \right\}^k \). We employ two complementary sources of information as training data: First, we use the aligned input images from Sec. 4.4. The images constitute a sparse set of high-quality training samples on the camera manifold. Second, we use the input views together with the face mesh to create manifold views using image-based rendering. Specifically, we use unstructured lumigraph rendering [Buehler et al. 2001] with cameras corresponding to random \( m \) on the camera manifold. We can generate any of these renderings on the fly covering the entire manifold. In the supplemental we give details on how we sample the valid manifold range. The geometric reconstruction and calibration have uncertainty, producing rendering artifacts that reduce overall image quality. We can compute a per-pixel estimate of this uncertainty by computing the color variance \( \sigma^2 \) when blending the different images; we will use this estimate to reduce the influence of incorrect IBR pixels.

5.4 Loss
Our renderings should: \( a \) be close to the training images, \( b \) preserve the identity of the depicted person from all views, and \( c \) look photorealistic. We use the following loss to achieve these goals:

\[
\mathcal{L} = \mathcal{L}_{f_1} + \lambda_{LPIPS} \mathcal{L}_{LPIPS} + \lambda_{id} \mathcal{L}_{id} + \lambda_{prior} \mathcal{L}_{prior}
\]

The first two terms address goal \( a \). Here, \( \mathcal{L}_{f_1} \) penalizes pixel differences in terms of the \( l_1 \)-norm, weighted by per-pixel IBR confidence:

\[
\mathcal{L}_{f_1} = \frac{1}{b \lambda d b l e x / b l e x / b l e x} \left( \exp \left( -\eta \sigma^2 \right) \left( \hat{V} - \hat{V}_k \right) \right) \| \hat{V} - \hat{V}_k \|_1
\]

with the falloff factor \( \eta = 100 \) in all our experiments and \( \sigma^2 = 0 \) if \( \hat{V}_k \) is an input view. \( \mathcal{L}_{LPIPS} \) is the LPIPS [Zhang et al. 2018] distance which we apply to a downsampled version of the images to a resolution of 256 × 256 pixels.

The identity loss \( \mathcal{L}_{id} \) addresses goal \( b \) and makes use of the pre-trained VGG-face [Parkhi et al. 2015] network \( \Psi \), which converts an image into face recognition features:

\[
\mathcal{L}_{id} = 1 - \langle \Psi(\hat{V}), \overline{\Psi} \rangle
\]

where \( \overline{\Psi} \) denotes the normalized mean face recognition features of all input views.

Finally, we address goal \( c \) by recognizing that StyleGAN renderings of highest quality and stability are usually obtained when the latents \( w \) follow the distribution dictated by the mapping network \( H \). Specifically, we address the problem that the extended space \( W^* \), where the styles of different StyleGAN layers are decoupled from each other, leads to an underconstrained problem and consequently images of lower realism. We enforce a certain amount of coherence between the styles using

\[
\mathcal{L}_{prior} = \frac{1}{18} \sum_{k=1}^{18} \left\| w_1 - \overline{w} \right\|_1
\]
where $w_i$ refers to the latent fed into the $i$-th generator layer, and $\overline{w} = \frac{1}{18} \sum_{i=1}^{18} w_i$. This prior encourages the network to produce latents which are similar for all generator layers, i.e., closer to the original StyleGAN latent space $W$.

5.5 Training
We found that a progressive training schedule, which splits training into two stages, produces results of highest quality. Fig. 7 summarizes our method; details and exact parameters are given in supplemental.

In the first stage, we only use the aligned input views as training data and optimize all trainable parameters (Fig. 7a). Intuitively, this training stage provides sparse anchors for the MLP, which is responsible for pose changes and at the same time optimizes the latents of the static GAN layers with the highest-possible quality training data. This stage trains in 35 min on an NVIDIA RTX6000.

In the second stage, we provide samples from the entire manifold as training data using a mixture of ULR renderings (85%) and input views (15%). We fix the latents of the static layers to prevent high-frequency IBR artifacts from impacting them (Fig. 7b). This stage fills in the pose gaps between the input views and trains in 4 min.

6 FREE-VIEWPOINT RENDERING
We have so far established a method to render a view $\hat{V}$ corresponding to a pinhole camera $\hat{A}$ on the camera manifold using StyleGAN. We now seek to move away from the manifold to synthesize free-viewpoint images $V$ with an arbitrary camera model $A$. We achieve this goal using a simple image warping strategy as follows.

First we project $A$ to the closest manifold camera $\overline{A}$ (Sec. 4.3, first column in Fig. 8). Recall that both $A$ and $\overline{A}$ correspond to physically meaningful cameras. Therefore, since we have the face geometry at our disposal, we can render an inverse flow field [Mark et al. 1997; Yang et al. 2011], which for each pixel of $V$ indicates where to lookup $\hat{V}$ (second column in Fig. 8). The desired viewpoint $V$ is then obtained by warping $\hat{V}$ according to the flow field (third column in Fig. 8). This procedure is compatible with any camera model $A$ for $V$, including physical lenses, stereoscopic setups, etc. Multi-sampled effects require multiple entries per flow field pixel [Yu et al. 2010]. We use multi-sampling by default to anti-alias occlusion boundaries arising from parallax.

Fig. 8. Free-view generation in our approach: We use StyleGAN to generate the closest view on the camera manifold (left), which is warped using a flow field (center) to obtain the final result (right). The top row shows an example of a parallax-free flow field, arising from a free camera in the valid manifold range. The configuration in the bottom row requires parallax in the flow field, as the free camera lies outside the valid manifold range.

Note that if the position of $A$ corresponds to a manifold coordinate $m$ in the valid range (Sec. 4.2), the projection to $\overline{A}$ does not change the camera’s position (Fig. 6a). Therefore, the flow field is parallax-free and corresponds to a continuous remapping (top row in Fig. 8). If $A$ is a perspective pinhole camera, the warp even reduces to a simple global affine image transformation. Parallax occurs in the flow field only if $A$ moves outside the valid manifold range (Fig. 6b; bottom row in Fig. 8). This operation is akin to view-dependent texture mapping [Debevec et al. 1996], with StyleGAN as a texture generator.

The final image $V$ is always synthesized using a coordinated interplay between latent manipulations and image warping. Even a seemingly simple lateral camera motion cannot be created by only shifting the image: The new viewpoint results in a slight variation of viewing angle that in turn leads to a change of manifold coordinates. A re-evaluation of the networks for synthesizing correct perspective is therefore required.

7 RESULTS AND EVALUATION
We implemented our method based on the original StyleGAN2 code [Karras et al. 2020b] in Python, complemented by a custom OpenGL rendering framework, including an interactive viewer. We provide all source code of our method here: https://repo-sam.inria.fr/fungraph/freestylegan.

To acquire faces we requested authorization from our institutional ethics committee, and due to restrictions imposed by their decision concerning privacy and security, we are only able to show a small number of images per subject as illustration in this paper, but cannot distribute the full datasets. We provide instructions on how to capture a face with a smartphone in the supplemental.
FreeStyleGAN: Free-view Editable Portrait Rendering with the Camera Manifold

7.1 Results

To evaluate our approach, we use our own captures. We asked participants for a series of 10-25 photographs of their head and upper body. The participants were asked to sit still, while non-professional photographers with no background in visual computing sequentially captured images from different viewing angles, using a digital (smartphone) camera. None of the authors (or their collaborators) were present during capture and no training was conducted.

To ensure both successful geometry reconstruction and disentanglement of the camera pose in the StyleGAN latents, we found it crucial to eliminate all other sources of variation (lighting, facial expression, etc.) across the input views. Yet, our technique is robust enough to allow lightweight sequential capture and does not require a multi-camera setup.

To demonstrate free-viewpoint rendering and the accuracy of our camera model, we composite our results on top of a rendering of a synthetic scene in Fig. 9 with a moving camera. We perform compositing with color correction membranes [Farbman et al. 2009] using convolution pyramids [Farbman et al. 2011]. As can also be seen from our supplemental video, our approach for the first time allows to successfully integrate a 2D GAN portrait rendering into a synthetic 3D scene with free-viewpoint control.

In Fig. 10 we show semantic editing results. We apply the method of Härkönen et al. [2020] to the output latents from our implicit representation network. We observe that the modifications stay view-consistent in our renderings (see supplemental video for interactive editing sessions). Note that the method used to perform semantic editing is independent of our approach.

In Fig. 11 we demonstrate face renderings with different camera models. The Vertigo effect (Fig. 11b) can be achieved without leaving the camera manifold, i.e., without warping, by simply varying the \(d\)-component of the manifold coordinate. We see that our latent representation network is able to synthesize the shift in perspective when moving towards the face while opening the field of view. In Fig. 11c we demonstrate spherical lens distortions, while Fig. 11d showcases a stereoscopic rendering result. Our solution generates the two images from the correct viewpoints, also allowing fine-grained control over interocular distance and screen depth.

7.2 Comparisons

We compare our algorithm to state-of-the-art StyleGAN-based rendering techniques, portrait-specific approaches, and for completeness to general-purpose free-view IBR methods. A full set of videos can be found in our supplemental material.

First we compare our approach to two state-of-the-art StyleGAN-based portrait rendering approaches: PIE [Tewari et al. 2020b] and StyleFlow [Abdal et al. 2021]. While both methods perform novel-view portrait rendering using StyleGAN, they differ from ours in two fundamental ways: First, they rely solely on manipulations of...
StyleGAN latent code $w^+$, restricting novel views to the camera manifold, i.e., a very small fraction of what our method allows, prohibiting free-viewpoint navigation. Second, their view parameterization does not correspond to a physically meaningful quantity: Latent code-induced viewpoint manipulations are not simply a camera rotation around a fixed 3D point (Sec. 4), but induce non-linear dependencies between camera position, orientation and field of view.

This is in contrast to our camera manifold formulation which translates latent codes into physically meaningful quantities. To allow a fair comparison, we thus restrict our output to the camera manifold, omitting our free-viewpoint warping. We determine image quality by re-synthesizing held-out input views. For both PIE and StyleFlow, we use a frontal view as input. Since there is no way to directly find the held-out camera with these methods, we densely sample the yaw-pitch-space for both methods and select the view minimizing mean $l_2$-distance of facial landmarks [Kazemi and Sullivan 2014] to the requested input view. Our approach generates the required views directly. Our method outperforms the others both numerically (Tbl. 1) and, arguably, in subjective image quality (Fig. 12).

We also compare against the facial re-enactment technique of Siarohin et al. [2019] and the morphable face albedo model of Smith et al. [2020] in Fig. 13. For Siarohin et al. we use their pre-trained model based on the VoxCeleb dataset [Nagrani et al. 2017], which allows a reasonable variety of poses due to a conservative crop window. We use a textured mesh [CapturingReality 2016] rendered from free viewpoints (unless stated otherwise) to drive an aligned frontal-pose view. We observe that their result quality highly depends on the viewpoint: While frame-filling portrait views are handled well, viewpoint-induced scaling of the head tends to result in distortions or severe identity shifts. Additionally, their spatial output resolution is $256 \times 256$, in contrast to our resolution of $1024 \times 1024$. For the method of Smith et al. we use their inverse rendering pipeline based on a frontal view. Their method is designed for maximum control over face shape, facial expression, and surface properties far beyond what the StyleGAN latent space captures, but it suffers from a lack of photo-realism.

Our technique allows to change the distance between face and camera by manipulating StyleGAN latent codes using our camera manifold formulation and the trained network $M$. In Fig. 14, we compare this latent-driven shift of perspective to the single-image approach of Fried et al. [2016]. Given a shot captured from a close distance (Fig. 14, left), their method uses warping to simulate a long-distance shot with the corresponding decrease in field of view.
We analyze the effectiveness of individual components of our algorithm by ablation. We consider distribution and number of input views, alternatives to our camera manifold formulation, loss terms and training procedure, and our background blur approach.

### 7.3 Ablations

We analyze the effectiveness of individual components of our algorithm by ablation. We consider distribution and number of input views, alternatives to our camera manifold formulation, loss terms and training procedure, and our background blur approach.

#### 7.3.1 Input Views

In Fig. 15a we show a typical distribution of input views in our manifold parameterization. The distributions of multiple datasets can be found in our supplemental video. We observe that our casual capture tends to result in an uneven view distribution and a significant fraction of the input views lying outside the manifold boundaries. These views are vital for successful geometry reconstruction (Sec. 3.2), as they cover the subject from a wider range of directions, but these views cannot be used in the first training stage (Sec. 5.5). We did not observe a strong effect of the exact view distribution when input views are reasonably stratified.

To investigate how the number of input views influences result quality, we progressively reduce the views used in our pipeline. We then perform an exhaustive leave-one-out image quality analysis (following the same protocol as in Sec. 7.2) for each configuration. The results are given in Fig. 15b. We observe that, unsurprisingly, image quality improves as the number of input views increases, but tends to saturate at about 20 views. Using less than 15 views has a stronger negative effect on image quality. For less than 10 views camera calibration and geometry reconstruction are unreliable.

#### 7.3.2 Camera Manifold

We analyze three alternatives to our camera manifold formulation. In the upper row of Fig. 16, we show a result obtained when omitting the manifold completely, i.e., using the trackball camera model from Eq. 2. We feed the four camera parameters to $M$ and train it to produce latents for free-viewpoint images directly. As this task is harder, we give $M$ more capacity by doubling the number of both the hidden layers and feature channels. The resulting images are severely distorted, while our approach matches the reference - a ULR rendering of the desired pose - well.

The lower row of Fig. 16 shows a result obtained with a naive manifold: We use the frontal pose to optimize for a coefficient vector $c$ and fix it while rotating the camera around the head. This naturally results in in-plane rotations and therefore out-of-distribution images...
Fig. 16. Manifold ablations: Training a mapping from free cameras to latents gives heavily distorted results (upper left). A naive manifold is better, but still does not respect training data alignment and therefore results in distortions (lower left). In contrast, our solution (middle column) is distortion-free and matches the pose reference (right column) well.

Fig. 17. Manifold boundary analysis: a) and b) Within the manifold boundaries our results are of consistently high quality. c) Directly generating a view outside the manifold (red point) using StyleGAN results in low quality compared to the held-out input view. d) We first render the closest view in the valid range (blue point) with StyleGAN. The final image (green point) is then produced using warping. Black dots denote input views.

method relies on the sparse input views only. It therefore fails to generalize to the entire camera manifold, either by generating distorted images (first row) or false perspectives (second row). If we only use the second training stage, we get blurry results with identity shifts due to IBR artifacts in the training data and the omitted LPIPS loss.

In the supplemental we also show that excluding the LPIPS term reduces image sharpness, while the identity loss preserves slight face identity shifts. The prior loss increases photo-realism.

8 DISCUSSION, FUTURE WORK, AND CONCLUSION
Our approach enables high-quality free-viewpoint synthesis of faces using StyleGAN with casually captured multi-view data as input. While our capture is lightweight, previous work on GAN embeddings require only a single image; at the expense of significantly less variation in pose and no direct way to connect to precise camera
We believe that this is an important step forward in bridging the gap between powerful 2D image-processing learning solutions and the generation of fully editable 3D content for general use, with minimal content creation effort.

ACKNOWLEDGEMENTS

This research was funded by the ERC Advanced grant FUNGRAPH No 788065 (http://fungraph.inria.fr). The authors are grateful to the OPAL infrastructure from Université Côte d’Azur for providing resources and support. The authors thank Ayush Tewari, Osh Fried, and Siddhant Prakash for help with comparisons. Adrien Bousseau, Ayush Tewari, Julien Philip, Miku Aitall, and Stavros Diolatzis for proofreading earlier drafts, the anonymous reviewers for their valuable feedback, and all participants who helped capture the face datasets.

REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. 2019. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? In ICCV. 4432–4441.
Rameen Abdal, Yipeng Qin, and Peter Wonka. 2020. Image2StyleGAN++: How to Edit the Embedded Images? In CVPR. 8296–8305.
Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. 2021. StyleXfer: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. ACM Transactions on Graphics (TOG) (2021).
Mallikarjun B R, Ayush Tewari, Abdallah Dib, Tim Weyrich, Bernd Bickel, Hans-Peter Seidel, Hanspeter Pfister, Wojciech Matusik, Louis Chevallier, Mohamad Elhabib, and Christian Theobalt. 2021. PhotoApp: Photorealistic Appearance Editing of Head Portraits. In ACM Transactions on Graphics (TOG, Proc. SIGGRAPH).
Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. 2010. High-quality single-shot capture of facial geometry. In ACM Transactions on Graphics (TOG, Proc. SIGGRAPH). 1–9.
Sai Bi, Stephen Lombardi, Shunsuke Saito, Tomas Simon, Shih-En Wei, Kevyn Mphail, Ravi Ramamoorthi, Yaser Sheikh, and Jason Saragih. 2021. Deep Relightable Appearance Models for Animatable Faces. ACM Transactions on Graphics (TOG, Proc. SIGGRAPH) (2021).
Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In Proc. SIGGRAPH. 187–194.
Yochai Blau and Tomer Michaeli. 2018. The perception-distortion tradeoff. In CVPR. 6228–6237.
J. Blinn. 1988. Where am I? What am I looking at? (cinematography). IEEE Computer Graphics and Applications 8, 4 (1988), 76–78. –http://dx.doi.org/10.1109/38.77851
Kevin W Bowyer, Kyong Chang, and Patrick Flynn. 2006. A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition. Computer vision and image understanding 101, 1 (2006), 1–15.
Chris Buehler, Michael Bosse, Leonard McCmillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In Proc. SIGGRAPH. 425–432.
Adnan Bulat and Georgios Tzimiropoulos. 2017. How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks). In ICCV.
CapturingReality. 2016. RealityCapture. www.capturingreality.com. [accessed 20-July-2020].
Eric R Chan, Marco Monteiro, Petr Kellnhofer, Iajun Wu, and Gordon Wetzstein. 2021. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. In CVPR.
Anpei Chen, Ruiyong Liu, Ling Xie, Zhang Chen, Hao Su, and Yu Jingyi. 2021. SofGAN: A Portrait Image Generator with Dynamic Styling. ACM Transactions on Graphics (TOG) 41, 1 (Article 1 July 2021), 26 pages.
Marc Christie, Patrick Olivier, and Jean-Marie Normand. 2008. Camera control in computer graphics. In Computer Graphics Forum, Vol. 27. Wiley Online Library, 2197–2218.
Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. 2000. Acquiring the reflectance field of a human face. In Proc. SIGGRAPH. 145–156.
Paul Debevec, Camillo Taylor, and Jentendra Malik. 1996. Modeling and rendering architectures from photographs: A hybrid geometry-and-image-based approach. In Proc. SIGGRAPH. 11–20.
Yu Deng, Jualong Yang, Dong Chen, Fang Wen, and Xin Tong. 2020. Disentangled and Controllable Face Image Generation via 3D Imitative-Contrastive Learning. In CVPR. 5154–5163.
Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019a. Deferred Neural Rendering: Image Synthesis using Neural Textures. *ACM Transactions on Graphics (TOG)* (2019).

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. 2019b. Image-guided neural object rendering. In ICLR.

Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. 2021. Designing an Encoder for StyleGAN Image Manipulation. In *ACM Transactions on Graphics (TOG, Proc. SIGGRAPH)*.

Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. 2017. Deep feature interpolation for image content changes. In CVPR. 7064–7073.

Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guolin Liu, Jan Kautz, and Bryan Catanzaro. 2019. Few-shot Video-to-Video Synthesis. In NeurIPS.

Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhöfer. 2021. Learning Compositional Radiance Fields of Dynamic Human Heads. In CVPR. 5704–5713.

Shih-En Wei, Jason Saragih, Tomas Simon, Adam W Harley, Stephen Lombardi, Michal Perdoch, Alexander Hipes, Dawei Wang, Hernan Badino, and Yaser Sheikh. 2019. VR facial animation via multiview image translation. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–16.

Xuan Yu, Rui Wang, and Jingyi Yu. 2010. Real-time depth of field rendering via dynamic light field generation and filtering. In Computer Graphics Forum, Vol. 29. 2099–2107.

Egor Zakharov, Aleksei Ivakhnenko, Alakadnara Shysheya, and Victor Lempitsky. 2020. Fast Bi-layer Neural Synthesis of One-Shot Realistic Head Avatars. In ECCV.

Egor Zakharov, Alakadnara Shysheya, Egor Burkov, and Victor Lempitsky. 2019. Few-shot adversarial learning of realistic neural talking head models. In IJCV 9459–9468.

Yajie Zhao, Zeng Huang, Tianye Li, Weikai Chen, Chloe LeGendre, Xinglei Ren, Ari Shapiro, and Hao Li. 2019. Learning perspective undistortion of portraits. In ICLR. 7849–7859.

Hang Zhou, Jihao Liu, Ziwei Liu, Yu Liu, and Xiaogang Wang. 2020. Rotate-and-Render: Unsupervised Photorealistic Face Rotation from Single-View Images. In CVPR. 5911–5920.

Jun-Yan Zhu, Philipp Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In CVPR. 586–595.

Yuxuan Zhang, Zeng Huang, Tianye Li, Weikai Chen, Chloe LeGendre, Xinglei Ren, Ari Shapiro, and Hao Li. 2019. Learning perspective undistortion of portraits. In ICLR. 7849–7859.

Xuan Yu, Rui Wang, and Jingyi Yu. 2010. Real-time depth of field rendering via dynamic light field generation and filtering. In *Computer Graphics Forum*, Vol. 29. 2099–2107.

Egor Zakharov, Aleksei Ivakhnenko, Alakadnara Shysheya, and Victor Lempitsky. 2020. Fast Bi-layer Neural Synthesis of One-Shot Realistic Head Avatars. In *ECCV*.

Egor Zakharov, Alakadnara Shysheya, Egor Burkov, and Victor Lempitsky. 2019. Few-shot adversarial learning of realistic neural talking head models. In *IJCV* 9459–9468.

Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. 2020. In-domain GAN Inversion for Real Image Editing. In *ECCV*.

Jun-Yan Zhu, Philipp Krahenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative visual manipulation on the natural image manifold. In *ECCV*. Springer, 597–613.

Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. 2021b. Editable Free-viewpoint Video Using a Layered Neural Representation. In *ACM Transactions on Graphics (TOG, Proc. SIGGRAPH)*.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. arXiv:2010.07492 [cs.CV].

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In CVPR. 586–595.

Yuxuan Zhang, Zeng Huang, Tianye Li, Weikai Chen, Chloe LeGendre, Xinglei Ren, Ari Shapiro, and Hao Li. 2019. Learning perspective undistortion of portraits. In ICLR. 7849–7859.

Hang Zhou, Jihao Liu, Ziwei Liu, Yu Liu, and Xiaogang Wang. 2020. Rotate-and-Render: Unsupervised Photorealistic Face Rotation from Single-View Images. In CVPR. 5911–5920.

Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. 2020. In-domain GAN Inversion for Real Image Editing. In *ECCV*.

Jun-Yan Zhu, Philipp Krahenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative visual manipulation on the natural image manifold. In *ECCV*. Springer, 597–613.

Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. 2018. Visual object networks: Image generation with disentangled 3D representations. *NeurIPS* 31 (2018), 118–129.
In this supplemental document we provide multi-view capture instructions, as well as methodology details on camera calibration and geometry reconstruction. We also present specifics of the FFHQ alignment process, how to project manifold coordinates to the valid range and how to sample it, and our definition of a frontal pose. Finally, we elaborate on our two-stage training procedure, provide additional comparisons and ablations, and show a comparison to an encoder-based embedding approach.

1 Capture Instructions
Here we provide the instructions we gave to the non-professional models and photographers who helped us with capturing the datasets.

1.1 Instructions for the Model
- You need to be as static as possible. Sit on a chair and use the backrest (make sure it is not visible above the shoulders).
- Choose a point in front of you to look at during the entire session.
- Make a neutral relaxed face, eyes open, mouth closed. No smiling please.

1.2 Instructions for the Photographer
- Don’t use a flash. Avoid casting hard shadows onto the model with your body. Lighting should not be too harsh, shades on the face are fine and even appreciated.
- Have a distance of about 1-2 meters to the model.
- Take 10-25 pictures of the face and full upper body. Frame every view such that that top of the head is slightly below the upper image boundary and the belly button marks the lower image boundary. Please capture:
  - One frontal view.
  - 5-10 views on an ellipse, about 0.25 to 0.5 meters around the frontal view (red in Fig.1).
  - 5-10 views on an ellipse, about 1.0 to 1.5 meters around the frontal view (blue in Fig.1).
  - Add more views at will.
- Capture positions don’t have to be exact.

2 3D Calibration and Reconstruction
We obtain camera calibration and the geometric proxy using off-the-shelf software [CapturingReality 2016]. First, the virtual input cameras are calibrated using structure from motion [Snavely et al. 2006], then a multi-view stereo algorithm estimates the 3D shape using dense pixel correspondences [CapturingReality 2016; Goesele et al. 2007], followed by a meshing step to obtain a triangle mesh of the face. We smooth the mesh [Sorkine 2005] to get rid of high-frequency reconstruction noise. The output of this process are the reconstructed triangle mesh, the calibrated cameras and possibly resampled input images with a lens distortion correction applied. Even though more than 10-25 cameras are usually recommended for 3D reconstruction, quality is high enough for our method, despite taking the photos casually without a rig.

3 Review of FFHQ Alignment

![Fig. 1. Figure provided to the photographer to illustrate the capture distribution, encouraging a stratified set of camera poses. Yellow dots mark example capture positions.](image)

![Fig. 2. The 2D alignment performed in the FFHQ dataset. a) Raw facial feature points (blue dots) are detected and aggregated to obtain representative eye and mouth positions (green dots). b) Geometric features are used to determine the square crop window (grey, not shown to scale) with center c and vector s giving orientation and size. c) The resulting aligned image.](image)
The FFHQ dataset was constructed by first collecting images from Flickr, followed by a cleanup step and alignment, where first 68 facial features are found [Kazemi and Sullivan 2014] (blue dots in Fig. 2a). Then the eye and mouth features are aggregated to obtain representative eye positions \(x_l\) and \(x_r\), as well as a mouth position \(x_m\) in the image (green dots in Fig. 2a). From these three points a crop window is computed as follows (Fig. 2b): The center of the window is computed as a convex combination of the eye midpoint \(x_c = 0.5 (x_l + x_r)\) and the mouth, as in \(c = \lambda x_c + (1-\lambda)x_m\), where \(\lambda = 0.9\) in the reference implementation. The eye and mouth positions are also used to determine the orientation of the square crop window:

\[
\hat{s} = x_r - x_l + \text{rot}_90(\hat{x}_m - x_c)
\]

is used as the horizontal orientation of the crop window, where \(\text{rot}_90\) denotes a counter-clockwise rotation of 90°. The size of the crop window should encompass the entire head, and the heuristic approach in the reference implementation reads as

\[
s = \max(2\|x_l - x_r\|, 1.8\|x_m - x_c\|) \frac{\hat{s}}{||\hat{s}||}
\]

Given the above crop window geometry, the original image is resampled to obtain the final aligned output image (Fig. 2c).

4 MANIFOLD RANGE PROJECTION

Given the rotational components of a manifold coordinate \(q = [\theta, \phi]^T\), the closest point \(q^*\) in the valid region of the manifold is

\[
q^* = \begin{cases} 
q & \text{if } c_1(\theta) \leq \phi \leq c_u(\theta) \\
[g, c_u(q)]^T & \text{if } c_1(\theta) > \phi > c_u(\theta) \\
[l_1, c_1(h)]^T & \text{if } c_1(\theta) < c_u(\theta) < \phi \\
[l_1, c_1(h)]^T & \text{if } c_u(\theta) > c_1(\theta) > \phi
\end{cases}
\]

where

\[
g = \text{sign}(\theta) \sqrt{\frac{b_l - b_u}{a_u - a_l}}
\]

\[
h_i = \frac{\theta}{4a_i^2} + D_i + \frac{\theta}{4a_i^2} - D_i,
\]

\[
D_i = \left(1 + 2a_i(b_l - \phi) - 6a_i^2 \right) + \left(\frac{\theta}{4a_i^2}\right)^2,
\]

with \(c_1(\theta) = a_i \theta^2 + b_i\) and \(i \in \{u, l\}\).

5 MANIFOLD SAMPLING

To sample the valid range of the camera manifold during training, we employ the inverse-CDF method. We observe that \(p(\theta)\) is proportional to the difference of the two bounding parabolas:

\[
p(\theta) = \frac{1}{2} (c_u(q) - c_l(q)) = \frac{1}{2} (\Delta a \theta^2 + \Delta b),
\]

where \(\Delta a = a_u - a_l\) and \(\Delta b = b_u - b_l\). Further, \(\theta\) is defined between the intersections of the two bounding parabolas, i.e., \(\theta \in [-g, g]\).

6 FRONTAL POSE CALIBRATION

For the definition of the manifold (Sec. 4.1) and its range (Sec. 4.2), as well as the 3D alignment of the face mesh (Sec. 4.4), we need to consistently define how 3D eye and mouth positions are related to manifold coordinates \(\theta\) and \(\phi\). For calibration, we consider the frontal pose \(\theta = \phi = 0\). The horizontal orientation is straightforward: We set \(\theta = 0\) when both eye locations have the same depth. For the vertical orientation, there exists no obvious frontal pose. We therefore set \(\phi = 0\) when the depth of the mouth is one eighth of the interocular distance smaller than the depth of the eye midpoint. This configuration is somewhat arbitrary and could be replaced by any suitable alternative. Note, however, that the exact orientation does not change results as long as we are consistent with these definitions.

7 DETAILS ON THE TRAINING PROCEDURE

We found a progressive training schedule, which splits training into two stages, to produce results of highest quality.

7.1 Stage 1: Sparse Input Views

In the first stage, we only use the aligned input views as training data and optimize all trainable parameters (Fig. 3a). Following Karras et al. [2020], we initialize all latents with \(\mu_w = \mathbb{E}_i H(z)\). This initializes the optimization with the “average face” in latent space \(W\) and is
obtained by running 1000 random codes \( z \) through the mapping network \( H \). We run our optimization for 7500 iterations using Adam [Kingma and Ba 2014] with default parameters and a batch size of 2. We start with a learning rate of 0.005 and decay it exponentially by a factor of 0.98 every 200 iterations. Again following Karras et al. [2020], we allow the optimization to escape local minima by adding stochastic Gaussian random noise to the latents in each training iteration. The optimization starts with a noise standard deviation of \( \sigma = 0.1 \) and decays as described in Karras et al. We use the following weights for our loss terms: \( \lambda_{\text{LPIPS}} = 100 \), and \( \lambda_{\text{id}} = 1 \). Following ideas from Tewari et al. [2020], we vary \( \lambda_{\text{prior}} \) over time: For the first 2500 iterations we set \( \lambda_{\text{prior}} = 10 \) to ensure a reasonable embedding close to \( W \). For the remaining iterations we set \( \lambda_{\text{prior}} = 0.1 \), which allows the optimization to explore the extended space \( W^* \). Intuitively, this training stage provides sparse anchors for the MLP, which is responsible for pose changes and at the same time optimizes the latents of the static GAN layers with the highest-possible quality training data. Training this stage takes 35 minutes on an NVIDIA RTX6000.

### 7.2 Stage 2: Dense Manifold

In the second stage, we provide samples from the entire manifold as training data using a mixture of ULR renderings (85%) and input views (15%). Now we fix the latents of the static detail layers to prevent high-frequency IBR artifacts from impacting them (Fig. 3b). Only the MLP weights are refined at this stage, essentially filling in the pose gaps between the input views. We run this stage for 750 iterations with the same optimization parameters as before, but omitting the random noise and using the following (constant) weights for our loss terms: \( \lambda_{\text{LPIPS}} = 0 \), \( \lambda_{\text{id}} = 1.5 \), and \( \lambda_{\text{prior}} = 0.1 \). Disabling the LPIPS loss term is motivated by the fact that it is very sensitive to IBR artifacts and no obvious way exists to incorporate uncertainty akin to Eq. 6 into the multiscale VGG architecture [Liu et al. 2018] without altering its perceptual prediction quality. However, we found \( \mathcal{L}_{\ell_1} \) as the sole image quality loss sufficient for this fine-tuning stage. Training this stage takes 4 minutes.

### 8 ADDITIONAL COMPARISONS

In addition to the comparisons shown in the main paper, we present more details on the comparisons to other image-based rendering methods in Fig. 4 and Tbl. 1. We compute image quality using the PSNR, SSIM [Wang et al. 2004], and E-LPIPS [Kettunen et al. 2019] metrics of the facial region. We see that - not surprisingly - the neural IBR methods win most of the competitions. This comes at the cost of static scenes, which cannot be edited. Due to time constraints, we did not train a separate NeRF++ model for each leave-one-out image set. The numerical results on facial landmarks and image quality are therefore heavily skewed in favor of the method. Please see the supplemental video for novel-view camera paths.

The method of Siarohin et al., which allows semantic editing in the form of facial expressions, does not perform well in the free-viewpoint setting for the metrics we considered. We observe that our approach allows free-view synthesis with camera accuracy in the order of magnitude of the IBR methods, while obtaining decent image quality and facial identity scores – while inheriting the full editing potential of StyleGAN. Note that the comparison to IBR is only possible because our method for the first time allows precise camera control of GAN imagery.

### 9 LOSS TERM ABLATIONS

Fig. 5 shows the effect of our individual loss terms. Excluding the LPIPS term reduces image sharpness, while the identity loss preserves slight face identity shifts. The prior loss increases photo-realism. We observe that all of our loss terms are necessary to produce results of high fidelity.

### 10 ENCODER-BASED EMBEDDINGS

General embedding strategies have been explored, which do not require face-specific optimizations to obtain latent codes [Richardson et al. 2021]. While these encoder-based approaches open up exciting research directions, the quality of the resulting embeddings is currently insufficient (see Fig. 6).
Table 1. Comparing camera accuracy (measured using facial landmarks), image quality, and face recognition error against free-view rendering methods.

| Method                  | Semantic Editing | Facial Landmarks | Image Quality | Recognition Error |
|-------------------------|------------------|------------------|---------------|-------------------|
|                         |                  | Alignment | Detection Rate | PSNR  | SSIM  | E-LPIPS  | ↓    |
| [Hedman et al. 2018]    | ✗                | .023      | 99%           | 26.4  | .875  | .010      | .07  |
| [Riegler and Koltun 2020] | ✗               | .027      | 100%          | 23.6  | .845  | .013      | .08  |
| [Zhang et al. 2020]†    | ✗                | .018      | 100%          | 30.6  | .853  | .015      | .24  |
| [Siarohin et al. 2019]  | ✓                 | .254      | 42%           | 11.7  | .531  | .061      | .23  |
| Ours                    | ✓                 | .068      | 100%          | 20.9  | .717  | .017      | .14  |

† Due to time constraints, we did not train a separate model for each leave-one-out image set, but only one model using all images per subject.

2 Editing is restricted to facial animations using a driving video.

Fig. 5. Loss term ablation. Only our full loss formulation gives best results.

Fig. 6. Comparing our optimization-based embedding strategy (center) with the state-of-the-art encoder-based method of Richardson et al. [2021] (right). We observe that our approach produces a more faithful reconstruction of the target (left).

REFERENCES

CapturingReality. 2016. RealityCapture. www.capturingreality.com. [accessed 20-July-2020].

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. 2007. Multi-view stereo for community photo collections. In 2007 IEEE 11th International Conference on Computer Vision. IEEE, 1–8.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (TOG) 37, 6 (2018), 1–15.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In CVPR. 8110–8119.

Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In CVPR. 1867–1874.

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. E-LPIPS: robust perceptual image similarity via random transformation ensembles. arXiv preprint arXiv:1906.03973 (2019).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

Guilin Liu, Fisum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image Inpainting for Irregular Holes Using Partial Convolutions. In Proceedings of the European Conference on Computer Vision (ECCV). S. Milborrow, J. Morkel, and F. Nicolls. 2010. The MUCT Landmarked Face Database. Pattern Recognition Association of South Africa (2010). http://www.milbo.org/muct.

Elad Richardson, Yuval Alaluf, Or Pataut, Yaron Nitzan, Yaniv Azan, Stav Shapiro, and Daniel Cohen-Or. 2021. Encoding in style: a stylegan encoder for image-to-image translation. In CVPR. 2287–2296.

Gernot Riegler and Vladlen Koltun. 2020. Free view synthesis. In ECCV. 623–640.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. 2019. First Order Motion Model for Image Animation. In NeurIPS.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. In ACM Siggraph 2006 Papers. 835–846.

Olga Sorkine. 2005. Laplacian mesh processing. Eurographics (STARs) 29 (2005).

Ayush Tewari, Mohamed Elgharib, Malikkarjun BR, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. 2020. PIE: Portrait Image Embedding for Semantic Control. ACM Transactions on Graphics (TOG, Proc. SIGGRAPH Asia) 39, 6.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing 13, 4 (2004), 600–612.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. arXiv:2010.07492 [cs.CV]