Latent Space Manipulation of GANs for Seamless Image Compositing

Dissertation by
Anna Frühstück

In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia

© April, 2023
Anna Frühstück
All rights reserved
https://orcid.org/0000-0002-3870-4850
The dissertation of Anna Frühstück is approved by the examination committee

Committee Chairperson: Dr. Peter Wonka
Committee Members:  Dr. Bernd Bickel
                   Dr. Helmut Pottmann
                   Dr. Bernard Ghanem
**ABSTRACT**

Latent Space Manipulation of GANs for Seamless Image Compositing

Anna Frühstück

Generative Adversarial Networks (GANs) are a very successful method for high-quality image synthesis and are a powerful tool to generate realistic images by learning their visual properties from a dataset of exemplars. However, the controllability of the generator output still poses many challenges. We propose several methods for achieving larger and/or higher visual quality in GAN outputs by combining latent space manipulations with image compositing operations: (1) GANs are inherently suitable for small-scale texture synthesis due to the generator’s capability to learn image properties of a limited domain such as the properties of a specific texture type at a desired level of detail. A rich variety of suitable texture tiles can be synthesized from the trained generator. Due to the convolutional nature of GANs, we can achieve large-scale texture synthesis by tiling intermediate latent blocks, allowing the generation of (almost) arbitrarily large texture images that are seamlessly merged. (2) We notice that generators trained on heterogeneous data perform worse than specialized GANs, and we demonstrate that we can optimize multiple independently trained generators in such a way that a specialized network can fill in high-quality details for specific image regions, or insets, of a lower-quality canvas generator. Multiple generators can collaborate to improve the visual output quality and through careful optimization, seamless transitions between different generators can be achieved. (3) GANs can also be used to semantically edit facial images and videos, with novel 3D GANs even allowing for camera changes, enabling unseen views of the target. However, the GAN output must be merged with the surrounding image or video in a spatially and temporally consistent way, which we demonstrate in our method.
ACKNOWLEDGEMENTS

The path to my PhD degree has been a wonderful, exciting, strange, and unexpected adventure. It would not have been possible without the support of my family, friends, and colleagues along the way.

My journey in life and research has taught me that the conjunction of education, internal choices, and external influences weave together to shape one's perception of the world - like training set bias! Consequently, every human's model of how the world looks and functions is highly individualized by their life experiences. Indeed, I often marvel at how different the same color, view, or experience must look and feel through the senses of the person next to me. Mindful of this caleidoscope of life, I count myself so lucky for the multitude of encounters and influences that shaped me to perceive the world the way I see it today:

I would like to thank my advisor Peter who has encouraged me to pursue a PhD degree in the first place. His passion for research is a huge inspiration and he was both patient when things didn't work out and encouraging when they did. I'm grateful for his advice, guidance, and patience in this journey.

I also express my gratitude to my PhD committee members Bernd Bickel, Bernard Ghanem, and Helmut Pottmann for their time and valuable comments on my research and dissertation.

A huge Thank You to the collaborators I encountered during my internships at Weta, Adobe, and Meta: Despite never meeting some of them in person due to the pandemic, they have become mentors and friends that were always ready to impart their wisdom and I feel very grateful that they placed their trust in me. I learned so much from them! Thank you Nikos Sarafianos, Tony Tung, Yuanlu Xu, Cynthia Lu, Niloy Mitra, Krishna Kumar Singh, Eli Shechtman, Navi Brouwer, Antoine Bouthors, and Andrew Glassner.

Without my amazing friends in the Visual Computing Center, I would have never been able to make it through the roller coaster ride of a PhD: Peter, Ali, Ronell, Matej, Mohammed, Michael, Ondřej, Alberto, Amani, and all my other friends and colleagues. Thank you for listening to me complain, for encouraging me when I needed it, for cheering me on, and for celebrating the wins – and thanks for all the coffee breaks!

I feel also incredibly grateful for the support of my family and friends both in KAUST and back home, whose belief in me gave me strength and perseverance. Seeing me move far away certainly wasn't easy, yet you encouraged me to go, explore, and find my own path! Thanks especially to my parents Martha and Anton for their unconditional love and for generously giving me everything and asking nothing in return.

Most of all, thank you to Markus, who believed in me even when I didn't, going through the ups and downs with me, and, as needed, provided me with advice, encouragement, consolation, guidance, support, and love. Thank you, for all of it! ♡
# Table of Contents

Examination Committee Page 2

Abstract 3

Acknowledgements 4

List of Figures 9

List of Tables 10

1 Introduction 11
  1.1 Thesis Outline ................................................. 13
  1.2 List of Publications ........................................... 15
  1.3 Contributions .................................................. 15

2 Fundamental Concepts and Related Work 17
  2.1 Traditional Image Synthesis ................................. 17
  2.2 Generative Adversarial Networks ............................ 19
  2.3 Latent Spaces of GANs ......................................... 24

3 Synthesizing Large-Scale Texture Images by Tiling Latents 31
  3.1 Introduction .................................................... 31
  3.2 Previous Approaches ........................................... 33
    3.2.1 Non-parametric Texture Synthesis ......................... 33
    3.2.2 Parametric Texture Synthesis ............................ 34
    3.2.3 GANs Trained for Texture Synthesis ....................... 34
    3.2.4 Selected Applications of GANs ........................... 35
  3.3 Overview ...................................................... 36
  3.4 Methodology ................................................... 37
    3.4.1 Preprocessing .............................................. 38
    3.4.2 Latent Field Synthesis .................................... 39
    3.4.3 Texture Synthesis .......................................... 43
4 Improving Generated Image Regions through Insets

4.1 Introduction

4.2 Previous Approaches

4.2.1 Unconditional Image Generation

4.2.2 Image Outpainting

4.2.3 Conditional Generation of Full-Body Humans

4.3 Methodology

4.3.1 Full-Body GAN

4.3.2 Multi-GAN Optimization

4.3.3 Optimization Objectives

4.3.4 Face Refinement versus Face Swap

4.3.5 Body Generation for an Existing Face

4.3.6 Face Body Montage

4.3.7 Optimization Process

4.4 Dataset and Implementation

4.4.1 Training Details

4.4.2 Unconditional Generation and Adaptive Truncation

4.5 Evaluation and Discussion

4.5.1 Quantitative Evaluation

4.5.2 Baseline Comparison

4.5.3 User Study

4.5.4 Limitations

4.5.5 Dataset Bias and Societal Impact

4.6 Conclusion

4.7 Additional Results and Discussion

4.7.1 Two Insets

4.7.2 Face Orientations

4.7.3 Face-Body Montage

4.7.4 Latent Space Walk

4.7.5 Custom Face Generator
5 Seamless 3D Editing of Humans in the Context of Videos 95

5.1 Introduction ......................................................... 95

5.2 Previous Approaches .............................................. 98

5.2.1 GAN Inversion .................................................. 98

5.2.2 GAN-based Latent Space Editing ......................... 98

5.2.3 3D-aware GANs ............................................... 99

5.2.4 Video Synthesis and Editing ............................... 99

5.2.5 GAN-based Video Editing ................................. 99

5.3 Methodology ....................................................... 100

5.3.1 Personalized 3D-Aware Generator ....................... 101

5.3.2 Frame-by-frame Video Inversion ......................... 105

5.3.3 Attribute Editing and Novel View Synthesis ........... 105

5.3.4 Compositing with Source Video ......................... 107

5.4 Experiments and Evaluation ................................... 110

5.4.1 Inversion Quality ............................................. 111

5.4.2 Image Quality ................................................ 113

5.4.3 Face Fidelity .................................................. 114

5.4.4 Resource Usage .............................................. 116

5.4.5 Ablation Study ............................................... 116

5.4.6 Evaluation of Semantic Edits ............................. 118

5.4.7 Evaluation on Novel View Synthesis .................... 120

5.4.8 Compositing with Challenging Boundaries ............ 121

5.4.9 Experimental Edits ......................................... 122

5.4.10 Limitations .................................................. 125

5.5 Optimization Details and Parameter Settings ............. 125

5.6 Social Impact ..................................................... 127

5.7 Conclusion ....................................................... 128

6 Conclusions ......................................................... 129

6.1 Impact of Our Work on Seamless Compositing ......... 130

6.2 Future Work ..................................................... 131

References .......................................................... 134
LIST OF FIGURES

2.1 Basic GAN Architecture ........................................... 19
2.2 StyleGAN2 Generator Architecture ............................... 21
2.3 FFHQ Dataset Alignment Strategy ................................. 22
2.4 EG3D Architecture ................................................ 23
2.5 StyleGAN2 Latent Spaces ......................................... 25

3.1 Example of Large-Scale Synthesized Aerial Textures .......... 31
3.2 Texture Synthesis Merging Problem ............................. 33
3.3 TileGAN Synthesis Intermediate Output ......................... 36
3.4 TileGAN Synthesis Tile Merging .................................. 38
3.5 Latent Merging at Different Layers .............................. 42
3.6 Artistic Control .................................................. 44
3.7 TileGAN Comparison to State of the Art ......................... 48
3.8 Undesirable Transitions Occurring in TileGAN ................. 50
3.9 Artifacts in TileGAN Synthesis ................................ 52
3.10 Results Generated Using TileGAN .............................. 53
3.11 Additional Results Generated Using TileGAN .................. 54

4.1 Application of InsetGAN in Different Scenarios ................. 56
4.2 InsetGAN Pipeline ............................................... 60
4.3 Unconditional Generation of Full-Body Humans ................ 61
4.4 Simultaneous Optimization of Two Distinct Insets ............. 63
4.5 Face Refinement Using InsetGAN ................................ 65
4.6 Body Generation for an Existing Face Using InsetGAN ......... 68
4.7 Face/Body Montage Using InsetGAN ............................ 69
4.8 Multimodal Face Improvement ................................... 71
4.9 Full-body Human Dataset Generation ........................... 75
4.10 Effect of Adaptive Truncation on Unconditional Synthesis .. 77
4.11 Comparison of Faces to Evaluate Dataset Quality ............. 82
4.12 Face Refinement Comparison of InsetGAN with CoModGAN .... 83
# List of Tables

3.1 Quantitative Evaluation of TileGAN .............................................. 47

4.1 Lambda Weight Terms for InsetGAN Losses ................................. 73
4.2 Quantitative Evaluation of InsetGAN ........................................... 79
4.3 InsetGAN Precision and Recall Evaluation ................................. 80
4.4 InsetGAN User Study Results .................................................... 87

5.1 Video Quality Metrics Comparison to Related Work .......................... 112
5.2 Image Quality Metrics Comparison to Related Work ......................... 113
5.3 VIVE3D Detailed Reconstruction and Image Quality Analysis ............... 113
5.4 Face Similarity Metrics Comparison to Related Work ........................ 114
5.5 VIVE3D In-Depth Face Similarity Analysis ..................................... 115
5.6 Timings and Memory Requirements for VIVE3D and Related Methods .... 116
5.7 VIVE3D Quantitative Ablation Evaluation ..................................... 117
Chapter 1

Introduction

The success of deep learning techniques in the last decade, combined with the widespread availability of large-scale image databases which can be gathered from online sources has caused a surge of interest in generative models in both academia and industry. Image synthesis is a useful tool for a wide variety of applications in game development, film production, as well as image creation and editing. In recent years, Generative Adversarial Networks (GANs), proposed by Goodfellow et al. [47] in 2014, have been established as the de-facto standard for deep learning-based domain-specific image synthesis. Concurrently developed unsupervised learning techniques exhibiting powerful representation capabilities include Variational Auto-Encoders (VAEs) [79, 22], Autoregressive Models [139, 106], Energy Based Models [34, 58] and Normalizing Flows [31]. Certain of these approaches have advantages over GANs, e.g. simpler training objectives or decreased likelihood of mode collapse, which is a failure state of training where the network overfits to one specific training sample rather than attempting to capture the full complexity of the target domain. In particular, diffusion models [149] have recently shown very promising results for general-purpose image synthesis tasks. However, while the quality gap between GANs and other generative models was shrinking towards the end of this PhD thesis, GANs still achieve unsurpassed results in generative quality and expressivity [14] for human image editing.
The availability of suitable high-quality training data is a crucial factor in the success of any particular deep learning technique [117], since even the best available model will not be able to represent the domain of images it should reflect, no matter the complexity of the model and training resources, if the dataset it is trained with does not properly describe the manifold of possible images that should be learned. While image data has increasingly become available in the last two decades with the advent of smartphone cameras and social media, gathering, curating and preprocessing data to form a suitable dataset remains an extremely challenging task. Many generative models, therefore, rely on the same tried and tested datasets as ImageNet [28] or LSUN [155] and additionally often limit their target domain to a specific subject such as human faces, like CelebA [94, 72] and FFHQ [75] or animal faces [24].

One big factor in the success of generative modeling techniques is the controllability and editability of the output image. The underlying nature of training (unconditional) GANs is stochastic, which means that during the training phase, a mapping from a random input vector to a synthesized output image is learned. The learned space of the input vector’s domain is usually high-dimensional and difficult to visualize and understand. State-of-the-art GAN architectures [76, 73, 74] introduced a mapping stage to the pipeline that transforms the input latent to a more disentangled intermediate space. In order to understand and navigate the various latent spaces of current GAN architectures, many concurrent research projects [2, 3, 115, 137] concern themselves with the exploration of these latent spaces to achieve mechanisms of control over the image output. A particularly important task is to synthesize a generated output that adheres closely to a desired target image, a process that is often called "encoding" or "projecting" an image into the latent space. In the last few months, text-to-image synthesis and editing [109, 44] have also gained extreme popularity. Given the powerful generative capabilities of GANs, applications can then include semantic image editing operations, which are time-intensive and very challenging to perform even by skilled
artists using traditional image editing techniques. Additional possible applications include image improvement, object removal or addition, inpainting and extrapolation, as well as neural upsampling techniques.

Despite all these advances, current generative models are limited both in their output size and generative capability, where state-of-the-art output sizes typically do not exceed 1024×1024, or rarely 2048×2048px resolution. The limiting factor for generating images at large resolutions stems partially from the limitations of current graphics hardware and training cost and duration, but is largely also caused by the exploding complexity when attempting to scale up neural architectures to yield high-quality result sizes without deteriorating the image quality. This makes it important to develop techniques to interconnect generative content with adjacent or surrounding images, which can be both generated or real. Unlocking the ability to composite generative content is the main focus of our line of work, which we apply in a variety of scenarios.

1.1 Thesis Outline

This thesis is structured in the following sections: After introductory remarks, we provide an overview of the fundamental concepts used in the following chapters in Chapter 2. To that end, we first introduce image synthesis in Section 2.1 followed by the concept and design of Generative Adversarial Networks in Section 2.2. We then describe how the usage of different latent spaces of GANs can achieve different levels of output control in Section 2.3. Furthermore, we discuss the state of the art in latent-space control and editing techniques in this section.

Then, we describe our published work contributing to these domains:

We describe our work on neural texture tiling in Chapter 3, where we discuss how we can leverage the convolutional nature of GANs to use a pre-trained GAN for tiling
intermediate latent blocks to create a larger, coherently merged output image. This is particularly useful in the domain of texture synthesis where a generator trained on texture blocks can hence be (theoretically) tiled endlessly, yielding seamless output textures at a very large scale. In practice, we demonstrate results up to Gigapixel scale. This chapter is based on our work published at SIGGRAPH 2019 [41].

We also discuss our work on using specialized GANs to improve insets in larger context images in Chapter 4. This approach shows that two or more independently trained generator networks can be optimized to achieve a coherent output image, where one generator provides the general layout of the output and the remaining generators improve specific regions of the canvas by providing high-quality details. These generators can produce coherent outputs through joint optimization techniques even in challenging circumstances. This chapter is based on our work published at CVPR 2022 [43].

Generative editing is also a useful tool in the context of video editing, and we discuss our method for this application in Chapter 5. We leverage a novel 3D-aware GAN architecture to edit parts of images and seamlessly merge the edited generated results with the surrounding video frame in a consistent fashion, even when changing the camera angle of the generator. Our technique is able to do this in a natural-looking and temporally-coherent fashion, enabling video editing using 3D GANs. This chapter is based on our work published at CVPR 2023 [42].

Finally, we conclude with a summary of the thesis in Chapter 6. We discuss our contributions and their impact to the domain of deep-learning-based image synthesis and composition and provide an outlook on future avenues of research in this exciting and diverse domain in Section 6.2.
1.2 List of Publications

In the line of research for this dissertation, we have published the following three papers:

(1) Anna Frühstück, Ibraheem Alhashim and Peter Wonka, **TileGAN: Synthesis of Large-Scale Non-Homogeneous Textures** [41], published in the *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 2019

(2) Anna Frühstück, Krishna Kumar Singh, Eli Shechtman, Niloy Mitra, Peter Wonka, and Cynthia Lu, **InsetGAN for Full-Body Image Generation** [43], published in the *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022

(3) Anna Frühstück, Nikolaos Sarafianos, Yuanlu Xu, Peter Wonka and Tony Tung, **VIVE3D: Viewpoint-Independent Video Editing using 3D-Aware GANs** [42], accepted to the *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023

1.3 Contributions

Our work in the area of deep learning for image synthesis contributes to the state of the art in the following areas:

- We propose a novel technique to explore the usage of Generative Adversarial Networks for texture synthesis. For that purpose, we demonstrate that GANs are a suitable tool to learn a generative representation of a specific texture domain. We show that we can leverage the convolutional nature of the GAN upsampling stages to extract and merge intermediate texture tiles and use the remainder of the network to seamlessly create very large texture outputs.
• We show that GANs still struggle to capture highly complex domains at a large scale with sufficient detail. In order to improve upon the state of the art, we explore the usage of specialized generator networks to enhance image regions of a base canvas generator. This is useful if the canvas domain is too challenging for a single GAN. We show that we can use specialized GANs capable of generating high-quality details to improve specific regions of another GAN by jointly optimizing two or more networks simultaneously to create coherent outputs with seamless transitions.

• We demonstrate that by ensuring temporal coherence, GANs can be a useful tool for semantic video editing of human subjects. Our work is the first that leverages the advantages of novel 3D GANs for this purpose, specifically enabling the possibility of viewpoint-independent face editing through the added flexibility of an altered camera position within the context of 3D GANs. We discuss the challenge of compositing such edits with the surrounding static video frames caused by changes in the physical layout of the head of the target person. To alleviate this problem, we propose a novel optical flow-based correction strategy in order to allow for the natural compositing of the edited images, yielding both spatially and temporally consistent results.
Chapter 2

Fundamental Concepts and Related Work

In this chapter, we aim to give an overview of the basic concepts of synthetic image generation and we will discuss important methods that are widely used and referenced in our work on machine-learning-based image synthesis.

2.1 Traditional Image Synthesis

While research efforts in the area of image synthesis have multiplied with the advent of deep learning, traditional image synthesis techniques have been established from the early days of computer graphics. With the advent of 3D graphics in the late 1970s [39], one important application of computer-generated image synthesis was texture synthesis, which is a field of research that developed orthogonally to rendering [36]. Textures are images used to simulate the surface of graphical objects, elevating computer graphics from wireframe models to textured and colorized 3D objects, thereby greatly enhancing the realism of rendered generated models. While high-quality textures were (and still are) often hand-painted and adjusted by skilled artists, many research efforts have been concerned with automating, improving, and upscaling texture synthesis techniques. Typically, such texture synthesis techniques are concerned with the problem of creating larger or multiple varying textures based on a given exemplar image, or with the arrangement of specific features within the 2D mapping of a texture.
Non-parametric Texture Synthesis. Classic texture synthesis algorithms are usually reliant on a single exemplar image to provide the context of desired visual features. Often, the complexity of synthesizing textures that are similar to a given input exemplar, is tackled through the usage of patch-based techniques, where new textures are generated such that each patch in the generated result has an approximate match in the input texture [84]. In order to enable a fast and reliable correspondence between such patches, PatchMatch [10, 11] and several of its extensions [26, 77, 165] have been established as the de-facto standard for such lookups, and these techniques are employed in most state-of-the-art non-parametric texture synthesis algorithms, e.g. [77, 60]. Despite these remarkable advances, classic texture synthesis techniques are constrained by the limited information present in the exemplar image, making it difficult to scale the algorithms to large or diverse outputs. Additionally, synthesis speed can become a bottleneck in these algorithms, as very large inputs and/or outputs yield increasing complexity in the lookup process.

Parametric Texture Synthesis. In contrast to patch-based techniques, another approach is to view texture synthesis as a statistical problem [52]. Many successful approaches developed the concept of extracting feature statistics from the exemplar in order to produce similar features in an output image [55, 27, 112]. The advent of neural networks for image recognition [82] revisited this idea, leveraging neural networks trained on image collections for feature extraction, typically referred to as "deep features". In the context of texture synthesis, picking specific layers of the deep features and calculating their dot product allows for the creation of a texture descriptor [46, 45]. This idea can be expanded to incorporate correlation between deep features [123] in order to capture regular patterns and similarities, describing structural properties of the texture.
2.2 Generative Adversarial Networks

**Architecture.** Generative Adversarial Networks (GANs) [47] are an instance of machine learning models popularized in the last decade [50]. In this architectural model, two initially untrained neural networks are pitted against each other in a min-max game, as shown in Figure 2.1. One model, the generator, attempts to create an image – which initially is just random noise – from a random input through a series of learned reshaping and upsampling steps. The second network, a discriminator, or critic is trained simultaneously with the generator to distinguish between real and fake by passing the example images through a classification network. This is achieved by providing the discriminator with examples from both a dataset of real images as well as the images produced by the generator network. Starting with random choices for each image, the discriminator learns to refine its classification guess for each real and fake image it is shown, while the generator is simultaneously encouraged to produce increasingly plausible output images. Both networks are trained through backpropagation to improve upon their respective tasks. The success of GANs relies on achieving the delicate balance in this dual-network setup, where no network should overpower the other, while both are able to improve simultaneously.
GAN training can behave somewhat erratically when compared to other machine learning training tasks such as classification, where the training losses do not necessarily clearly reflect an improvement over time. Since the goal of training GANs is to get an optimally trained generator network, after training completion, in most instances, the discriminator is discarded. Efforts to reuse the discriminator for other purposes have had mixed success.

The initially proposed GAN architecture as depicted in Figure 2.1 has been redesigned and refined simultaneously, incorporating several new insights and tricks. Among the most successful and important insights in GAN training were the use of convolutional and deconvolutional layers [114], progressive growing [72], Adaptive Instance Normalization (AdaIN) [61]. Current GANs are, in ideal cases, able to produce generated images at an almost photorealistic quality that are in many cases difficult to distinguish from real images for human observers. The excellent generative quality can be largely attributed to several consecutive architectural refinements [72, 75, 76, 73, 74] as well as careful domain selection, combined with dataset curation and alignment of the input data [76]. The carefully designed architecture of StyleGAN2, which is still the state of the art of image quality on domain-specific generation, is depicted in Figure 2.2. Despite big advances in the area [15, 120], very good general-purpose generative models at large scale still remain elusive.

**Challenges.** Several challenges are still prevalent in current GAN architectures:

1. The output size of GANs is currently limited both by the architectural choices made in state-of-the-art network architectures, and also, to an extent, by the capacity of current Graphics Processing Units. Current GANs typically don't exceed an output image resolution of 1024×1024px, because larger networks are increasingly hard to train and results tend to suffer from image artifacts stemming
Figure 2.2: **Architecture of a StyleGAN2 Generator.** In the StyleGAN architecture, Karras et al. [75] introduced the usage of an MLP as a mapping network $f$ that translates the input latent space $z$ to an intermediate latent space $w$, which is able to capture a higher dimensionality. This latent space is used as layer-wise input for a generator network composed of several consecutive identical blocks consisting of convolutional and upsampling operations. Random noise $n_i$ is injected at each layer $i$ of the network for additional stochasticity. The depicted StyleGAN2 [76] architecture introduced several refinements of the original StyleGAN architecture.

from the increased complexity of large-scale images. While state-of-the-art Deep Learning GPU generations have significantly larger memory sizes, upscaling current architectures by simply increasing output image sizes and/or training batch sizes does not yield satisfactory results.

(2) State-of-the-art GAN architectures take notoriously long to train to achieve good generative performance, often requiring weeks or even months of training time even on state-of-the-art GPU clusters. This is especially true for challenging data domains and/or when training networks at large scale.

(3) The generative quality of a trained GAN is highly dependent on the quality, preprocessing, and complexity of the input data. Despite research efforts on the choices of the network architecture and data augmentation [73], current GANs still need a large amount of training data to produce a powerful and high-quality generator network. Additionally, the trained networks yielding the best visual results are dependent on input data that is well-aligned, e.g. by positioning the important visual features such as specific keypoints of the subject in similar locations for all samples in the dataset [75], as depicted in Figure 2.3. However,
For complex input domains or highly varied data, such alignment strategies may be challenging or impossible.

(4) The high-dimensional latent space of GANs is still not well understood. When generating samples from a random input, gaining control of the appearance of specific features in the output is highly complex and challenging. The stochasticity of the random input, or, in certain architectures, an intermediate latent space exhibiting a higher-dimensional latent manifold, allows for smooth interpolations between adjacent samples within the latent manifold, however, the location of desired samples within the manifold is difficult to approximate and a reliable mechanism of control of the appearance of the generated output is the subject of many research projects [2, 3, 115, 137, 6].
Figure 2.4: **Architecture of EG3D.** High-level sketch of the architecture of EG3D [20], one of several concurrently developed 3D-aware GANs attempting to capture 3D information from 2D image collections. EG3D leverages a StyleGAN2 backbone and reshapes the output into a tri-plane representation, which is rendered using a neural renderer given specific camera parameters. This step also produces a depth representation of the target subject, which is implicitly learned from the dataset. Finally, the image output is super-resolved to yield a high-quality output image. Altering the camera angles allows for a 3D examination of the target subject.

**3D-aware GANs.** The tremendous success of Neural Radiance Fields (NeRFs) in many different contexts of deep learning for Computer Vision tasks has inspired a wave of works attempting to infer spatial information from 2D image collections, utilizing NeRFs as the internal shape representation [20, 107, 30, 59, 162, 122]. Typically, these concurrently developed architectures rely on tried and tested GAN architectures [76] as a generative backbone, however, the output of the generator is reshaped into a 3D representation that is rendered, given specific camera parameters, using classic volume rendering techniques. Due to the complexity of volume rendering, output resolutions are limited, hence the rendering stage is often combined with upsamplers to achieve state-of-the-art output resolutions. One of the most popular 3D-aware architectures, EG3D [20], is depicted in Figure 2.4. This particular architecture has inspired follow-up work, for example, the integration of a segmentation branch [132]. A recent survey [146] gives a more comprehensive overview of recent 3D GAN architectures.
2.3 Latent Spaces of GANs

The initial GAN architecture proposed by Goodfellow et al. [47] already introduced the concept of generating an output image given an input latent \( z \), also sometimes called "feature vector", which is typically defined as a vector of random uniform noise in the range of \([-1, 1]\). In some architectures, the latent \( z \) is supplemented with additional noise, or, as in the case of conditional GAN architectures [101], with a vector of conditional information \( y \) or encoded contextual information.

Early GAN architectures often used a \( 100 \times 1 \)-dimensional vector for \( z \); later architectures increased the dimensionality of \( z \) to \( 512 \times 1 \), which is the most common choice for current architectures.

**Latent Manifolds.** Similar to the intuition underlying autoencoders, the latent space of a GAN can be seen as a sort of "encoding" or "condensing" of the information the Generator is supposed to represent, where each value of the feature vector can be viewed as a control over some learned image characteristics in the output. Later refinements of the GAN architecture discovered that a latent space defined by a single input is not expressive enough to yield a desirable level of control over the output features of the generator. To provide a manifold that can be navigated more easily, Karras et al. [75] introduced a fully-connected mapping network \( f \) that translates the input space \( z \) to another latent space \( w \) that contains \( 2 \times (\log_2(\text{resolution}) - 1) \) distinct \( 512 \)-dimensional latent vectors that directly control the appearance of different levels in the network, as depicted in Figure 2.2. The architectural choice of prepending the generator's architecture with this MLP achieves a certain disentanglement of the generator's latent space by allowing the MLP to learn a mapping of the input features to a more meaningful feature space, as shown in Figure 2.5. In practice, many approaches attempt to control the intermediate space \( w \) rather than \( z \) due to this disentanglement. The actual generator network \( G \) has a learned, but static \( 4 \times 4 \) input \( c_1 \). The information
and stochasticity are injected into the network throughout the different convolution and upsampling stages by adding the respective $w$ vector using an affine transformation $A$ and by injecting some noise $n_i$. During training, subsets of the vectors in $w$ are sometimes mixed and matched between different instances within the minibatch, mixing what the authors call "coarse", "medium" and "fine" features to encourage disentanglement of the different latents. The latent space $w$ also allows for the evaluation of an "average" output of the trained generator by mapping a large number (e.g. 10000) of random latents $z$ to $w$-space and calculating the arithmetic mean of all mapped $w$ vectors, yielding an averaged vector $w_{avg}$.

**Intermediate GAN Latents.** While the latent spaces $z$ and $w$ do not explicitly encode spatial information of features present in the output image, the different blocks in the generator network $G$ do, since they are, starting from a $4 \times 4$px size block, always upsampled by a factor of 2 until reaching the final output resolution. Some GAN papers aimed at stabilizing the training process through a progressive growing strategy [72]. The intuition behind this process is that lower resolutions of the same output image are less complex, and therefore easier to train, so low-resolution outputs are trained earlier and each next resolution level is only added to the generator network after the
previous resolution level has been trained up to a certain extent. Whereas the output learned at lower resolutions becomes useless after another resolution level is added in the initial Progressive Growing method, retaining meaningful lower-resolution outputs was also pursued in other approaches that attempt to progressively train a GAN capable of synthesizing multi-resolution results [71].

**Conditioning GANs.** In all instances of generative modeling, the controllability of the output is a clear priority. A possible direction to control the output of GANs is to design Conditional GANs, i.e., to train the generator by supplying additional information about desired output features such as object class, or semantic information such as color or gender to the generator, penalizing unsuitable outputs with losses during training [101, 111, 105]. However, this direction has several shortcomings: (1) describing complex controls as conditional vectors is very difficult, and (2) the training needs to be supervised, i.e. the dataset needs to exhibit the appropriate features for conditional training.

**Projection through Optimization.** As an alternative to conditioning a GAN during training, many research efforts have attempted to recover an approximate latent space representation for a given target image using a process typically referred to as "projection" or "inversion" [25, 2, 3, 142], traversing the latent space of a fully-trained unconditional GAN. Projecting into a trained generator typically entails the usage of an optimization strategy attempting to find a latent vector that generates an output exhibiting some desired features. This is usually achieved by using either a random starting point in the manifold of $\mathbf{w}$ or starting from an average latent vector $\mathbf{w}_{\text{avg}}$, repeatedly evaluating the output $G(\mathbf{w})$ and using gradient descent to approach an output that best approximates one or more optimization targets. Any given projection strategy needs to employ one or multiple losses encouraging these optimization targets, how-
ever, this is significantly easier to define than generating an annotated dataset. Several losses have been proposed for projecting real images at full or reduced resolution [100] into the latent space of StyleGAN or comparable GANs. For optimizing towards a target image, typically pixel-based $L_1$ or $L_2$ losses are combined with feature-based losses such as Perceptual Similarity (LPIPS) [160].

Early attempts at latent-space projection [2, 3] have discovered that using the same latent vector for all inputs in $w$ space (as is the default behavior when generating a random sample from a given latent $z$) does not yield very good result quality, particularly when attempting to project through optimization. However, removing this constraint and allowing all $r$ latent vectors in $w$ to optimize independently yields a much more expressive manifold and allows for significantly closer optimization results. This space is usually called $w^+$ in current literature. In fact, projecting into $w^+$ space can force a generator network to reproduce samples that are completely out of the domain of the original dataset, e.g. generating cats from a GAN trained on human faces [2].

However, most projection-based approaches do not only care about finding a faithful projection of the desired input but also about the possibilities of control and editing after finding a projected target image. Editing capabilities, however, can be significantly inhibited when producing out-of-domain samples from a trained generator network, as navigating the manifold around the projected latent may not contain meaningful other samples belonging to the target domain. To that end, many different variants of latent spaces and constraints have been explored. Refer to [147] for a comprehensive survey of latent space choices and inversion techniques.

**Encoder Networks.** An alternative strategy to find latents corresponding to desired image properties is to train an encoder network that maps an input image (or feature descriptor) to a target latent $w$ or $w^+$ [115, 137, 6]. This encoding is typically achieved by extracting some feature maps from an input image through a pre-trained feature
extraction backbone like ResNet [54] or VGG [128] and then training a per-layer mapping network to generate appropriate latent vectors for the desired features. The main advantage of an encoder network over optimization is that after training, inference of any given image of interest is very fast, whereas optimizing each particular target image is a computationally costly process. For a speed/quality trade-off, encoding and optimization-based approaches can be combined to form a hybrid strategy [167], where the encoder gives an initial estimate and improved correspondence to the input is achieved through additional optimization.

**Generator Fine-Tuning.** In order to retain the generalization ability of the w-space while providing a high-quality inversion, Pivotal Tuning [118] has successfully shown that trained generators can overfit to target images while still maintaining a navigable latent space. This is a useful design strategy to achieve results with very good visual coherence to the target but has also been shown to successfully generate out-of-domain samples such as unusual attributes or heavy makeup, and can be used for transfer-learning to domains with limited training data, such as cartoon faces, animated characters, or specific visual styles [131]. Importantly, when the change in generator re-training is regularized, previously discovered latent space editing directions are typically still applicable even in the updated generator space.

**Inversion in 3D.** With the advent of 3D GANs, recent works also investigate inversion [92] in the context of 3D GANs, attempting to discover a latent space representation for a given reference image or a set of reference images of a target. 3D inversion techniques need to deal with the added complexity of having to either provide or implicitly estimate the camera parameters used in the rendering step and should yield outputs that hold up with respect to correspondence to the input and image quality when altering the camera angle after the inversion. Since optimization techniques are in
danger of overfitting to the 2D input, current approaches rely on early stopping and using deep generative priors in order to retain 3D generalizability [159, 152, 153].

**Latent Space Editing Techniques.** Given an approximate latent space representation of a target image, GANs allow for semantic edits of the image by finding directions in the latent space manifold surrounding the latent code that yield the desired changes in the output image. This is an extremely powerful tool, as it can enable seamless semantic changes such as editing age, hair color, facial features, expressions, etc in the context of human faces. As such, significant research effort has gone into navigating and understanding the latent space of state-of-the-art GANs.

Editing directions can be discovered without having to provide labeled data by exploring unsupervised and self-supervised techniques [64, 140, 134, 135, 23]. As an example, GANSpace [53] performs PCA on the latent space $w$, yielding a basis $V$. This can be used to edit an input latent by varying the PCA coordinates, given a control vector $x$, such that $w' = w + Vx$. To further disentangle the effect of the edits, the weights of the control vector can be controlled per layer in $w^+$ space. SeFa [126] uses eigenvector decomposition on the affine layers. Unsupervised methods, however, are often less effective at discovering meaningful and disentangled interpretable editing directions.

Supervised methods [170, 150] often rely on off-the-shelf classifiers or segmenters in order to provide semantic labels or segmentations for specific input latents. A popular supervised method used in many application scenarios is InterfaceGAN [124, 125], a simple and robust supervised technique where latent space directions are discovered by classifying the outputs of large amounts of latents given pre-trained classifiers for specific binary attributes, e.g. gender or eyeglasses. Based on the latents which are classified with the largest confidence for the target attribute, an SVM is used to estimate a hyperplane in the latent space, yielding a latent space direction that can
be used to modify a source latent given a weight determining the strength of the edit. StyleFlow [4] uses normalizing flow in conjunction with off-the-shelf classifiers to learn a reversible mapping in the latent space.

Given recent advances in natural language-based supervision [113], text-based image synthesis and editing has recently gained immense popularity [44, 109].
Chapter 3

Synthesizing Large-Scale Texture Images by Tiling Latents

Figure 3.1: Aerial Texture Synthesis. TileGAN can synthesize large-scale textures with rich details. We show aerial images produced from a latent-space tiling of texture tiles at different levels of detail generated using our framework, which allows for interactive texture editing. Our results contain a broad diversity of features at multiple scales and can be several hundreds of megapixels in size.

3.1 Introduction

Example-based texture synthesis is the task of generating textures that look similar to a given input example. The visual features of the input texture should be faithfully reproduced while maintaining both small-scale as well as global characteristics of the exemplar.

In this chapter, we are interested in synthesizing large-scale textures that consist of multiple megapixels (see Figure 3.1). The first challenge in large-scale texture synthesis
is to process a large amount of input data. This is crucial because without a considerable amount of reference data, any generated output will not have a lot of variability and lack features at multiple scales. Such a synthesized output could be large-scale, but will be very homogeneous and boring or repetitive. Recent work in parametric texture synthesis using generative adversarial networks (GANs) seems ideally suited to tackle this challenge and we build on a recent GAN architecture that can generate high-quality results when trained on natural textures [72]. The second challenge in large-scale texture synthesis is how to generate large-scale output data. This is the core topic of this chapter and we have identified two important sub-problems that we tackle in our work.

First, assuming that the selected GAN can only generate tiles of limited resolution, it is necessary to make these tiles match. There are multiple possible solutions to this problem that were explored in previous work. An elegant and powerful method is to compute graph cuts between overlapping tiles [85]. While this method works well in some cases, very often it leads to artifacts when the blended tiles are not similar enough. Another possibility is to use a pixel-based texture synthesis algorithm like PatchMatch [10] to repair seams between textures. A very simple method is to use blending. We propose a solution that is based on manipulating latent codes of lower resolution levels of the GAN to obtain nice transition regions. See Figure 3.2 for a comparison of our method illustrated by blending four neighboring tiles. Second, we need to be able to incorporate user input to control the visual appearance of the synthesized output. A major challenge for most existing texture synthesis methods is the artistic control over the final result. While patch-batch based texture synthesis techniques can be constructed to provide artistic control, such as painting by numbers [56, 116, 96, 95], most existing GAN-based texture synthesis approaches provide no (or minimal) artistic control. In this work, we propose a solution based on latent brushes and other intuitive editing tools that allow easy global control on large scale texture maps.
Technically, the major contribution of our work is to provide a framework to take a GAN of limited resolution as a building block and produce a possibly infinite output texture. As a practical result, we are able to significantly improve the quality and speed of the state of the art in large-scale texture synthesis.

### 3.2 Previous Approaches

#### 3.2.1 Non-parametric Texture Synthesis

Existing non-parametric texture synthesis algorithms try to synthesize a new texture such that each $k \times k$ patch in the output texture has an approximate match in the input texture [84]. A very important ingredient for these algorithms is a fast correspondence algorithm such as PatchMatch [10] that is employed in most state-of-the-art texture synthesis algorithms, e.g. [77, 60]. PatchMatch can be extended to create faster queries [11] or additional error metrics [26, 77, 165]. While existing methods provide strong visual results, we propose to build on recent work in deep learning that shows a much greater promise with regards to the scalability of the considered input data or the size of the output due to faster synthesis speed. A notable earlier algorithm proposed a

---

**Figure 3.2: Texture Tiling Problem.** We show the difficulty of seamlessly merging a grid of four texture tiles, which is challenging even for roughly matching adjacent tiles. (a) Four input texture tiles. (b) Tiles are combined using graph cuts [85]. (c) Tiles are combined using our method.
hierarchical extension using an earlier version of non-parametric synthesis [51], but this algorithm is specific to the texture synthesis algorithm it employs [87] and it cannot be easily adapted to a deep learning framework.

### 3.2.2 Parametric Texture Synthesis

A popular early approach to texture synthesis was to extract features and feature statistics from an input texture and then try to create a new texture that would match these feature statistics [55, 27, 112]. This idea has now been revisited using features extracted by neural networks. Gatys et al. [46, 45] proposed the idea to use inner products between feature layers at different levels of the network as a texture descriptor. For each layer of the network, each pair of features gives one inner product to compute. This idea was expanded by Sendik and Cohen-Or [123], who introduced a structural energy term based on correlations between deep features, thus capturing self-similarities and regularities in the structural composition of the texture. The technique proposed by Snelgrove et al. [130] presents an early effort to increase the maximum size of texture features that can be synthesized using the method of Gatys et al. [46]. This is accomplished by matching a small number of network layers across many scales of a Gaussian pyramid leading to improved synthesized textures. Instead of using gradient-based optimization to compute new textures, it is also possible to train a generator using the feature statistics for the loss function [33].

### 3.2.3 GANs Trained for Texture Synthesis

Generative adversarial networks (GANs) were introduced in a seminal paper by Goodfellow et al. [47]. Over the years, the architecture of GANs has improved significantly and state-of-the-art GANs can produce results of stunning visual quality [72, 15, 75, 158]. In our work, we have chosen to build on the framework proposed by Karras et al. [72]. They introduced a progressively growing architecture that starts the training on a
low-resolution exemplar and slowly increases the size of the networks, as well as the exemplars. Their network is able to produce semantically coherent image content at a significantly higher resolution than previous work. Zhou et al. [166] introduced a technique to expand textures while preserving challenging structural arrangements by iteratively training a GAN on sub-blocks of the input textures. While this work uses GANs, it only uses a single image as input by generating many different crops during training. The convolutional nature of GANs can be exploited to synthesize images and textures of output sizes different from the image resolution the GAN was trained on [67, 13]. GANosaic [66] extends such methods to generate textures by optimizing the latent noise space to produce textures that match the overall content of a given guidance image. However, such methods are limited in the type of textures they support, the expected size, and the overall variability and quality of the output. The image stylization method FAMOS [68] improves on the quality by training the texture GAN and the guidance image styling network at the same time. While this method produces smoother transitions between texture patches, it still suffers from issues relating to scalability and variability, which we try to address in our method.

### 3.2.4 Selected Applications of GANs

GANs have been successfully adapted to classical image processing problems, such as inpainting [110, 151, 156] and super-resolution [143]. While traditional GANs generate images starting from a random vector, the GAN training can be extended to the problem of image-to-image translation using either paired or unpaired training data [63, 168, 169, 62]. In computer graphics, recent papers apply GANs to the synthesis of caricatures of human faces [18], the synthesis of human avatars from a single image [102], texture and geometry synthesis of building details [78], surface-based modeling of shapes [12] and the volumetric modeling of shapes [141]. The most related problem to our work is the problem of terrain synthesis [49].
Figure 3.3: **Intermediate output of Generator Network.** Our generator network takes a random latent vector of size 512 as input. At latent level $l = 3$, the latent tile (a) is of spatial resolution $8 \times 8$. For simplicity, the depth of the feature maps in each layer is not shown. We sample a large set $S$ of latent tiles for processing. For each sampled latent tile we also generate the corresponding output image (b) and save a downsampled version $d_r$ (c) (here we set $r = 16$) for retrieval during neighborhood similarity matching.

### 3.3 Overview

In this section, we present the three main components of our framework for seamless neural texture synthesis in a high-level overview:

**Generative models.** Our framework requires a generative model that produces novel images. State-of-the-art GANs typically consist of two networks: a generator and a discriminator. The generator network produces sample images which match the training distribution using convolutional layers that gradually increase the spatial resolution of a random latent vector to a full-size image. The discriminator network assesses how well the generated samples match the training distribution. The two networks are constructed to be differentiable and their gradients are used to guide the training of the full generative model. Our main focus in this work is on combining multiple outputs of the generator network of a standard GAN for large scale texture synthesis. More details on the GANs and data sets we use in our experiments are given in Section 3.6.
Synthesis. Our key contribution is a method to synthesize plausible large-scale non-homogeneous textures using a pretrained generator network. This is accomplished by generating a tiling of compatible intermediate latent tiles, which we call the latent field $F$, that the generator network $G$ uses to produce a coherent large-scale texture $I$ (see Figure 3.4). The intermediate latent tiles can be efficiently sampled and stored for analysis and online processing. In order to ensure that the synthesized textures are globally coherent, we optimize the latent field to satisfy two main objectives. First, the expected synthesized output should follow an initial small target guidance map $M$ for the expected large-scale synthesized image. This map can be randomly generated or specified by the user. Second, in order to afford local coherence and minimize abrupt texture changes between neighboring texture tiles, we optimize the latent field by replacing problematic tiles with better candidates that are more compatible with their neighbors. The details of our entire synthesis pipeline will be presented in Section 3.4.

Artistic Control. We propose a set of tools to facilitate user control over our texture synthesis process. The key idea behind the control our method affords during synthesis lies in modifying the latent field. To that end, we utilize operations such as painting, shuffling, copying, and target image matching, all of which enable different ways of artistic control. We will discuss details about our interactive tool in Section 3.5.

3.4 Methodology

We first describe the general notation used in this chapter before describing the different phases of our framework. We start by redefining the generator network, from a standard deep convolutional GAN, as:

$$G(z) = G_{B_i}(G_{A_i}(z)),$$  \hspace{1cm} (3.1)
where $z$ is a randomly sampled latent tile and $l$ specifies the intermediate level at which we plan to perform our latent field synthesis. Lastly, $G_{B_l}$ and $G_{A_l}$ are two parts of $G$ that split the set of convolutional layers at the level $l$ (see Figure 3.3). For a GAN with $n$ levels, $G_{A_l}$ takes the latent vector at level 1 as input and produces a $k \times k$ tile at level $l$ and $G_{B_l}$ takes a tile at level $l$ as input and produces a color image at level $n$, the final level.

Our large-scale non-homogeneous texture synthesis framework is divided into three phases: (1) a one-time preprocessing phase, (2) an online latent field synthesis phase, and (3) an online texture synthesis phase.

### 3.4.1 Preprocessing

The first step of preprocessing is to create a large set of texture samples $S$ that are generated using the generator network from a standard deep convolutional GAN. Each sample $s_i$ comprises two components: (I) an intermediate tensor $t_l = G_{A_l}(z)$, which we refer to as a latent tile where $l$ is the level at which we will synthesize the latent field,
and (2) $d_r$ a downsampled version of the texture map $G_{B_i}(t_i)$, where $r$ represents its spatial resolution. The greater the number of samples in $S$, the more texture variability is afforded by our framework. The second step in this phase is to cluster the texture samples in $S$ by their visual appearance, using $d_r$, in order to enable fast lookup of visually similar latent tiles. We perform the clustering using $k$-means and assign cluster centers $c_k$ as representative texture samples.

**Algorithm 1** The TileGAN Algorithm

```plaintext
1: procedure GENERATE_TEXTUREMAP
   input: $G_{B_i}, S, M$
   output: $F, I$
2:   $U ← NEXT_UNASSIGNED_PATCH(F)$
3: while COUNT($U$) > 0 do ▷ Initial tiling
4:     $i ← U(0)$
5:     $F_i ← TopMatch(i, F, I, S, M)$ ▷ Single top match
6:     $I_i ← G_{B_i}(F_i)$
7: end while
8: while $E(F) > \theta$ do ▷ Optimizing the entire texture
9:     $i ← Random(F, I)$
10:    $F_i ← BetterMatch(i, F, I, S, M)$
11:    $I_i ← G_{B_i}(F_i)$
12: end while
13: return $F, I$
14: end procedure
```

### 3.4.2 Latent Field Synthesis

**Markov Random Fields.** The second phase of our framework is the synthesis of large compositions of GAN-generated textures with no apparent visual artifacts, seams, or obvious repetition. We use a variant of the Markov Random Fields (MRF) model for texture synthesis applied on the latent field $F$. While the MRF model has been applied to texture colors [145] as well as texture statistics [89], we are the first to propose an application to GAN latent vectors. The goal of such an MRF model can be redefined
for our framework as follows: given a large set of individual texture tiles sampled from a single distribution, synthesize a large-scale output of texture tiles so that for each output tile, its spatial neighborhood is similar to some neighborhood from the input distribution. With this MRF assumption, the similarity of the local neighborhood between input and output helps ensure an overall coherent texture map with minimal boundary artifacts.

**Two-Step Process.** In order to efficiently generate textures at a large scale, we perform the latent field synthesis in two steps: an initialization step and an iterative refinement step. Algorithm 1 formalizes the entire process of our framework for latent field synthesis. Splitting the computational task of synthesizing the texture facilitates interactive editing. The first step is typically computed on the order of seconds and is immediately presented to the user. The refinement step is computed on a background process that regularly updates the latent field and displays the final output. Algorithm 2 represents how we find better candidates in the refinement step.

**Algorithm 2** Refine Latent Tile

1: **procedure** BETTERMATCH
2: **input:** $i, F, I, S, M$ ▷ Relevant neighboring regions
3: **output:** $F_i$
4: **if** $E(F_i) \leq \theta$ **then**
5: **return** $F_i$
6: **end if**
7: $T \leftarrow \text{TopMatches} \left( i, F, I, S, M \right)$ ▷ Set of top matches
8: **return** $\arg\min_i E(T_i)$
9: **end procedure**

**Initialization.** In the initializing step, we aim to efficiently generate a tiling of the latent field that approximately satisfies the guidance map $M$. The map $M$ provides global content control. At this stage, we assume that the texture samples $S$, generated at
the latent level \( l \), and its clustering result were computed in a prior preprocessing step. We start by performing a latent-tile-based texture synthesis to cover all unassigned tiles in the output latent field \( F \). For each unassigned tile, we find the single top matching \( F_i \) using the unary energy term defined below. We repeat this processing until no unassigned latents remain.

**Refinement.** Refinement steps are performed until the total latent field’s energy is lower than our set threshold. This stopping criterion is currently set empirically to a value that ensures that a desirable variety of visual features in the tiling is preserved during the MRF refinement. In each step, we randomly sample a latent tile in \( F \) and check for candidate tiles that minimize the local energy. We define the optimal latent field as the field that minimizes the following energy of weighted unary and binary terms:

\[
E = E_m + E_n.
\]  

(3.2)

The unary term \( E_m \) is the sum of visual similarities of \( d_r \) of a candidate tile \( F_i \) with its corresponding region in \( M \). In our experiments, we consider the Euclidean distance of the two images as the similarity measure in order to accelerate this computation. The binary term \( E_n \) considers the 4-connected neighboring latent tiles for each tile by the following weighted dissimilarity terms:

\[
E_n = \frac{1}{2} \sum_i \sum_{j \in N_i} (\lambda_V D_V(F_i, F_j) + \lambda_L D_L(F_i, F_j) + \lambda_C D_C(F_i, F_j)).
\]  

(3.3)

These terms represent the dissimilarity between a tile \( F_i \) and another tile in the set \( T_i \) of its 4-connected neighbors: visual appearance \( D_V \), latent vector representation \( D_L \), and cluster membership \( D_C \). For every pair of latent tiles in the 4-connected neighborhood, we approximate \( D_V \) and \( D_L \) using the Euclidean distance of their overlapping region. The dissimilarity measure \( D_V \) is computed on the corresponding
\( d_r \) of each tile while \( D_L \) is computed on the corresponding latent tensor \( t_l \). The last term \( D_C \) is the average agreement of cluster membership where we assign a 0 to pairs with matching clusters and 1 to non-matching pairs. The different energy terms are weighted by the corresponding \( \lambda_x \) weight parameter. We set \( \lambda_V \) as 1 and \( \lambda_L \) and \( \lambda_C \) as 0.5 in our experiments. When finding the top matches in the refinement step, we first return the top 10 matching tiles using \( E_m \) and then compute the entire energy after placing each candidate tile.

**Boundary and latents tiles.** An important aspect when combining tiles from different samples of \( S \) is the unpredictability at their joining region (see Figure 3.8). In our experiments, we have noticed that latents that fall on the outermost regions of the latent tile exhibit lots of instability. This is likely due to a bias caused by the zero

![Unmerged Result](image)

Figure 3.5: **Latent merging.** Our network architecture creates transitions between very different tiles when merging latent tiles at various depths. Here we show merging results for layers two to six. The impact region of the transition decreases in size when processing the latents at later stages in the network.
padding that is applied in \( G_{Bl} \). In order to minimize this effect, we only consider a cropped version for each sample of \( S \). The size of the cropped latent tile influences the overall visual coherence of neighboring output regions, where smaller latent tiles exhibit a smoother feature blending than larger tiles. In our experiments, we have typically used latent tile sizes of \( 2 \times 2 \) to \( 4 \times 4 \) regardless of the merge level \( l \). While we crop the latent blocks, we use the entire representative image \( d_r \) for comparison with the guidance map, which creates an overlapping sliding-window effect when finding tile matches, thereby further enhancing the coherence of neighboring tiles.

**Choice of parameter \( l \).** Selecting the level at which to split the GAN is a trade-off between the quality of the transition region and the scope of the region impacted by the transition changes (see Figure 3.5). We have mainly experimented with splitting at earlier levels \( l = 2 \) to \( l = 5 \) in our work because these parameters yield the best visual results according to our judgment.

### 3.4.3 Texture Synthesis

The final synthesized image is generated by taking a latent field and applying the trained generator network \( G_{Bl} \). Using this multi-stage process, the network is able to output arbitrarily large results. A latent field of size \( w \times h \) will result in an RGB texture of size \( 2^{(n-l)}w \times 2^{(n-l)}h \), where \( n \) is the number of levels in the pyramid. We use \( l = 3, n = 9 \) for most of our experiments. Since the generating function \( G_{Bl} \) is convolutional, we can profit in two ways. First, it can be efficiently applied for local re-synthesis. Second, arbitrarily large latent fields can be processed by multiple overlapping applications of \( G_{Bl} \), where the overlapping parts of the output of each application of \( G_{Bl} \) are discarded.
Figure 3.6: **User Interaction Controls.** Our interface allows users to manipulate latent fields interactively. The user can select images as guidance maps and edit the image using various latent-space painting techniques. On the bottom of the interface, representative tiles show clustered latents, which can be used to manipulate the image content. The main part of the UI consists of a preview of the final texture that is locally updated as soon as changes to the latent field are made. More details on user interaction are explained in Section 3.5.

### 3.5 Artistic Control

Our texture synthesis framework can fully automatically generate plausible results. However, as with most texture synthesis scenarios, user control and interactive editing are highly desirable. We have developed an interactive tool with different editing operations for our GAN-based image synthesis approach, see Figure 3.6.

We provide two major sets of editing operations: (1) directly manipulating the latent field, (2) editing the guidance map.
The first editing operation for manipulating the latent field allows the user to drag and drop a tile from a list of clusters (Figure 3.6, bottom) onto an existing latent field. The inserted latent tile is randomly sampled from the latent tiles belonging to the selected cluster. In order to visualize the expected tile appearance, we show the user a representative image corresponding to the cluster centers $c_k$. This tool can be generalized as a GAN-based paintbrush of variable size, which offers a high degree of user control. The second editing operation is a cloning tool, where we take parts of existing content from the synthesized image and clone the respective latents onto other regions. We provide an option to spatially shuffle the cloned tiles to add more diversity to the cloned region. Moreover, we can add small amounts of noise or interpolate between two latent tiles to allow for additional degrees of variability. These simple latent manipulation tools provide local control of the resulting output.

Finally, the appearance of the output texture can be influenced by modifying the guidance map using traditional image manipulation techniques. This capability provides virtually limitless variations in the size of the output, shading, placement of features, etc.

3.6 Implementation Details

Learning. Our GAN architecture and training are based on the approach of Karras et al. [72], called Progressive Growing of GANs (ProGAN). We have slightly modified the generator architecture to extract the intermediate latents at any arbitrary layer of the GAN. These latent tiles can be modified and merged to generate large-scale non-homogeneous textures. We have chosen ProGAN over other GAN architectures because it consistently produces high-quality output images and because the architecture consisting of a stack of identical building blocks facilitates the division into the two parts that allow us to manipulate the intermediate latent field.
While the training process may take on the order of days of training on multi-GPUs to reach tiles of acceptable quality, the synthesis and editing steps of the texture generation are possible at interactive rates running on a machine with a single GPU. The preprocessing step is done only once and typically requires 30 minutes to sample a set $S$ of size $100K$ latent tiles and then cluster the corresponding representative $d_r$ images into $k = 10$ clusters. For each data set, we train the GAN on four NVIDIA v100 GPUs for $16K$ iterations for around 4 days. We use the default optimizer and training schedule provided by the authors’ official TensorFlow [1] implementation [72].

**Training Data.** We have compiled various training data sets for experimentation with our method. Popular existing data sets like CelebA or LSUN are not suitable for large-scale texture synthesis. Therefore, we have curated our own test data from several sources of publicly available large-scale imagery, all of which are processed as image tiles of size $512 \times 512$:

- **Terrain map.** We collect $18K$ tiles of the terrain base map provided by Google Maps.

- **Satellite imagery.** We use $65K$ samples from the tiles of Landsat satellite images.

- **Oil canvas.** We also consider high-resolution images of smaller objects including $30K$ tiles from the detailed Gigapixel image of Vincent van Gogh’s The Starry Night provided by the Google Art Project.

- **Night sky.** We sample a total of $19K$ high-resolution tiles from the European Southern Observatory and the Hubble Space Telescope image repository.

For all data sets, we do not perform any alignment steps or augmentation of the input training tiles.
Table 3.1: **Quantitative evaluation.** At the top of the table, we show an evaluation of the results of our system presented in Figures 3.10 and 3.11. Below is a comparison between results generated using TileGAN and the state-of-the-art techniques of Self-Tuning Texture Optimization and Non-Stationary Texture Synthesis.

<table>
<thead>
<tr>
<th>Result</th>
<th>Training Data Set</th>
<th>Output</th>
<th>Merge Level</th>
<th>Latent Tile Size</th>
<th>Synthesis</th>
<th>Edits</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medieval Island</td>
<td>4700 MEGAPIXELS</td>
<td>94</td>
<td>2</td>
<td>2×2</td>
<td>15.0</td>
<td>1</td>
<td>16.0</td>
</tr>
<tr>
<td>(Figure 3.10 top)</td>
<td>2 MEGAPIXELS</td>
<td>1</td>
<td>1</td>
<td></td>
<td>16.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dinosaur Park</td>
<td>17000 MEGAPIXELS</td>
<td>22</td>
<td>2</td>
<td>2×2</td>
<td>4.5</td>
<td>5</td>
<td>9.5</td>
</tr>
<tr>
<td>(Figure 3.10 bottom)</td>
<td>2 MEGAPIXELS</td>
<td>5</td>
<td>1</td>
<td></td>
<td>9.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mountain Painting</td>
<td>7800 MEGAPIXELS</td>
<td>620</td>
<td>2</td>
<td>2×2</td>
<td>25.0</td>
<td>0</td>
<td>25.0</td>
</tr>
<tr>
<td>(Figure 3.11 top)</td>
<td>2 MEGAPIXELS</td>
<td>1</td>
<td>1</td>
<td></td>
<td>25.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space Panorama</td>
<td>5000 MEGAPIXELS</td>
<td>6.5</td>
<td>5</td>
<td>1×1</td>
<td>1.2</td>
<td>10</td>
<td>11.2</td>
</tr>
<tr>
<td>(Figure 3.11 bottom)</td>
<td>2 MEGAPIXELS</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STTO</td>
<td>0.27 MEGAPIXELS</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>9.4</td>
<td>—</td>
<td>9.4</td>
</tr>
<tr>
<td>(Figure 3.7 top)</td>
<td>2 MEGAPIXELS</td>
<td>2</td>
<td>—</td>
<td></td>
<td>9.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSTS</td>
<td>0.27 MEGAPIXELS</td>
<td>1.08</td>
<td>—</td>
<td>—</td>
<td>2000</td>
<td>—</td>
<td>2000</td>
</tr>
<tr>
<td>(Figure 3.7 top)</td>
<td>2 MEGAPIXELS</td>
<td>2</td>
<td>—</td>
<td></td>
<td>2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TileGAN</td>
<td>17000 MEGAPIXELS</td>
<td>9.8</td>
<td>2</td>
<td>2×2</td>
<td>3.0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>(Figure 3.7 top)</td>
<td>2 MEGAPIXELS</td>
<td>2</td>
<td>—</td>
<td></td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**3.7 Results**

In this section, we present a qualitative and quantitative analysis of the results created using our method. All synthesis results are generated using our interactive tool written in Python and running on a desktop machine equipped with an Intel Xeon 3.00GHz CPU with 32GB RAM and a single NVIDIA TITAN Xp GPU with 12GB memory. Note that in order to handle results with hundreds of megapixels, we resort to the caching of the latent field and synthesizing the image in chunks by the maximum supported block that fits on the GPU one block at a time.
Figure 3.7: **Comparison.** We provide an informal comparison to the current state-of-the-art texture synthesis methods Self-Tuning Texture Optimization (b) and Non-Stationary Texture Synthesis (c), using (a) as input. We can observe that the current state of the art has difficulty in reproducing multi-scale features while our result (d) exhibits interesting variations. Note that our method does not use (a) as input, but only as guidance for synthesis based on pre-trained content.

**Visual quality.** We use our framework on the data sets described in Section 3.6 and showcase selected results in Figures 3.10 and 3.11 to demonstrate the quality and variability afforded by our method. Table 3.1 shows the corresponding statistics for each result. We argue that our method can generate large-scale non-homogeneous textures with high visual quality that surpasses the quality achieved by any other published method.

**Visual comparison to the State of the Art.** We compare our method to two other state-of-the-art algorithms. We selected self-tuning texture optimization (STTO) [77], which we believe to be the state-of-the-art texture synthesis algorithm not using neural networks and non-stationary texture synthesis (NSTS) [166], a recent neural network-based algorithm. We take a single texture tile of resolution $628 \times 425$ as input for the STTO and NSTS algorithms. For both techniques, we used the recommended settings
provided by the authors. To generate our results, we use the fully trained network and apply the input image as a guidance map. We present a comparison of the different methods in Figure 3.7. Even though we could verify that STTO generates excellent results for a large variety of textures, in these tests we can see that STTO is unable to handle the multi-scale features present in the aerial image and produces a highly repetitive output. This demonstrates that challenging issues related to multi-scale texture synthesis have not been fully explored. We also verified our installation of the NSTS code released by the authors by replicating the results in their paper before running it on aerial images. However, NSTS is also not able to produce high-quality results when taking aerial images as input. We believe that multi-scale texture synthesis requires a large amount of input and that previous work is inherently not suitable to generate multi-scale content of high quality. By contrast, we can generate images of high visual quality with few artifacts.

**Quantitative comparison.** We also provide a comparison of the running time and scale for the synthesis of various examples and methods (see Table 3.1). As shown in the table, self-tuning texture optimization does not scale as well with respect to the input data size and the output data size as our method. Non-stationary texture synthesis spends a lot of time analyzing a small input texture, but this is not a suitable strategy for large-scale and multi-scale texture synthesis. Our method is faster than the competing methods if we exclude our preprocessing times with the justification that the preprocessing time would be amortized over many applications of a single trained GAN generator.

### 3.8 Limitations

Our framework is able to generate high-resolution textures on multiple data sets. However, there are still several limitations to be considered for future work.
Figure 3.8: **Undesired transitions.** When tiling latents with apparently similar visual appearance, such as this extreme case depicting snow and ice (a), our framework can produce unexpected regions of very different appearance (b). While these transitions look plausible, they can introduce unintended salient boundaries. We believe that this is due to the fact that the ice samples were learned from different continents and that no transition between these regions is available in the training data.

The largest output that we are able to generate on our testing machine is about 1.6 Gigapixels. Synthesizing and viewing larger images would require the implementation of additional memory management procedures.

Not all latent tiles generate good-looking results, see Figure 3.9 for a failure example where our result contains visual artifacts. Such failures might be attributed to training, sampling of the data set, or inherent limitations of the chosen GAN implementation. Since our framework is fairly modular, we believe that we can integrate new GANs easily.

In addition, the blending of two tiles can lead to unpredictable results. For example, two forest tiles next to each other can generate a visible boundary of different color in the transition region that was not visible in either tile when generated by itself. Figure 3.8 depicts this failure case, where a merging of apparently visually similar tiles (such as the depicted mixture of arctic ice and continental ice) generates undesirable
transition artifacts when attempting to merge them. Our framework tries to minimize these unpredictable results at the refinement step, but it is not able to eliminate them completely. Furthermore, such refinement may lead to decreased diversity due to increased repeated similar tiles if the MRF optimization is run till convergence [77]. To help avoid outputs with repeated visually similar latent tiles we stop the optimization early. When matching features during the optimization, visual artifacts occurring by directly selecting the highest ranked feature could be reduced by incorporating implicit diversity in the MRF as a regularization loss [144]. Alternative strategies to the MRF optimization considering latent tile usage [65] or incorporating diversity-encouraging feature distance metrics [98] are worth exploring in future work.

Our results can occasionally exhibit grid-like artifacts, as illustrated in Figure 3.9. This undesirable effect is especially noticeable when initializing a grid tiling completely randomly or when tiling challenging regions where the MRF doesn't manage to select good-fitting neighbors and the edge transitions become overly visible. The tiling may also exhibit some repetitiveness when the guidance map contains large regions of uniform color, which are tiled by very similar latent tiles, as visible in the background of Figure 3.6.

### 3.9 Conclusions and Future Work

In this work, we have tackled the problem of texture synthesis in a setting where many input images are given and a large-scale output is required. We have built on recent advances in high-quality generative adversarial networks and proposed a fast algorithm to tile outputs of GANs to produce large plausible texture maps with virtually no boundary artifacts. We have also proposed an interface that enables local and global artistic control of the output image. Our early quantitative and qualitative results demonstrate the fast generation of high-quality textures consisting of hundreds of
Figure 3.9: **Artifacts.** When choosing randomized or mismatching neighboring regions, as in this randomized tiling, our textures can exhibit a visible grid-like structure (yellow squares). This effect is more prominent for larger latent tile sizes and can be controlled by choosing appropriate values for the merge level, latent tile size, and MRF. Occasionally, the quality of our generated tiles is showing artifacts or unnatural patterns (blue circles). This effect is especially noticeable if multiple defective tiles are selected nearby.

megapixels. As far as we know, our work is the first to attempt to seamlessly combine intermediate latent tiles at different levels of a GAN to interactively generate such large texture synthesis results.

One interesting venue for future work is to experiment on datasets from other celestial bodies (e.g., Mars, Pluto, Sun) and close-ups of everyday objects captured at Gigapixel levels. We are also interested in applying our technique to data with depth information or multiple channels. Another venue for future work is adopting a stacking of multi-layer GANs in order to generate more realistic guide maps that can also be possibly created from an upper-layer GAN. Furthermore, the idea of simply manipulating a latent field to produce large textures can be exploited to quickly and consistently modify global appearance including applying color transformation or global patterns. We believe that the idea of latent vector manipulation can lead to many innovations in the future of texture synthesis.
Figure 3.10: **Results generated using TileGAN.** The results were generated using GANs trained on the Google terrain map (top) and the Satellite imagery (bottom) data sets. For each result, the first column shows samples from the training data, the second column contains sample tiles generated by the trained GAN, the third column features the low-resolution guidance map input to our method, and the final large-scale output. In the last column, we show two cropped and zoomed regions scaled by the zoom value given on the bottom right. Image credits: top, first column © Google; bottom, first column © ESRI.
Figure 3.11: **Further TileGAN results.** These results were generated by our method using GANs trained on the Oil canvas *(top)* and the Night sky *(bottom)* data sets. See Figure 3.10 for an explanation of the image columns. Image credits: top, low-resolution input © Vincent Brady; bottom, first column © ESO and ESA/Hubble.
Chapter 4

Improving Generated Image Regions through Insets

4.1 Introduction

Generative adversarial networks (GANs) have emerged as a very successful image generation paradigm. For example, StyleGAN [76] is now the method of choice for creating near photorealistic images for multiple classes (e.g., faces, cars, landscapes). However, for classes that exhibit complex variations, creating very high quality results becomes harder. For example, full-body human generation still remains an open challenge, given the high variability of human pose, shape, and appearance.

How can we generate results at both high resolution and high quality? One approach is to break the target image into tiles and train a GAN to sequentially produce them [41]. Such methods, however, are unsuited for cases where the coupling between the (object) parts are nonlocal and/or cannot easily be statistically modeled. An alternate approach is to aim for collecting very high resolution images and train a single GAN, at full resolution. However, this makes the data collection and training tasks very expensive, and variations in object configuration/poses cause further challenges. To the best of our knowledge, neither such a high resolution dataset, nor a corresponding high resolution GAN architecture has been published.

We propose InsetGAN towards solving the above problems. Specifically, we propose to combine a generator to provide the global context in the form of a canvas, and a set of specialized part generators that provide details for different regions of interest.
Figure 4.1: InsetGAN application. Our full-body human generator is able to generate reasonable bodies at state-of-the-art resolution (1024×1024px) (top left). However, some artifacts appear in the synthesized results, most visibly in hands and faces. We make use of a second, specialized generator to seamlessly improve the face region (1). More than one inset generator can be used to improve multiple regions at once, such as face and shoes (2). We can also use a given face as an input for unconditional generation of bodies (3). Furthermore, we can specify both specific faces and bodies and compose them in a seamlessly merged output (4).

The specialized results are then pasted, as insets, on to the canvas to produce a final generation. Such an approach has multiple advantages:

(I) the canvas GAN can be trained on medium quality data, where the object parts are not necessarily aligned. Although this results in the individual parts in the canvas
being somewhat blurry (e.g., fuzzy/distorted faces in case of human bodies), this is sufficient to provide global coordination for later specialized parts to be inserted;

(II) the specialized parts can be trained on part-specific data, where consistent alignment can be more easily achieved; and

(III) different canvas/part GANs can be trained at different resolutions, thus lowering the data (quality) requirements.

CollageGAN [91] has explored a similar idea in a conditional setting. Given a semantic map which provides useful shape and alignment hints, they create a collage using an ensemble of outputs from class-specific GANs [91]. In contrast, our work focuses on the unconditional setting, which is more challenging since our multiple generators need to collaborate with one another to generate a coherent shape and appearance together without access to a semantic map for hints.

The remaining problem is how to coordinate the canvas and the part GANs, such that adding the insets to the canvas does not reveal seam artifacts at the inset boundaries. This aspect is particularly challenging when boundary conditions are nontrivial and the inset boundaries themselves are unknown. For example, a face, when added to the body, should have consistent skin tone, clothing boundaries, and hair flow. We solve the problem by jointly seeking latent codes in (pretrained) canvas and part GANs such that the final image, formed by inserting the part insets on the canvas, does not exhibit any seams. In this chapter, we investigate this problem in the context of human body generation, where the human faces are created by a face-specific GAN.

We evaluate InsetGAN on a custom dataset, compare with alternative approaches, and evaluate the quality of the results with quantitative metrics and user studies. Figure 4.1 shows human body generation applications highlighting both seamless results, across face insets, as well as having diversity of solutions across face insertion boundaries.
**Contributions.** We summarize our findings on the subject of multi-generator optimization in the following:

(1) We propose a multi-GAN optimization framework that jointly optimizes the latent codes of two or more collaborative generators such that the overall composed result is coherent and free of boundary artifacts when the generated parts are inserted as insets into the generated canvas.

(2) We demonstrate our framework on the highly challenging full-body human generation task and propose the first viable pipeline to generate plausible-looking humans unconditionally at 1024×1024px resolution.

### 4.2 Previous Approaches

#### 4.2.1 Unconditional Image Generation

The unconstrained generation of image content via Generative Adversarial Networks, or GANs [47], has shown a lot of promise in recent years. In this context, the StyleGAN architecture was developed over a sequence of papers [75, 76, 73, 74] and is widely considered the state of the art for synthesizing individual object classes. For class-conditional image generation on the ImageNet dataset, BigGAN [15] is often the architecture of choice. In our work, we are building on StyleGAN2-ADA, since this architecture yields better FID [57] and Precision&Recall [86] scores on our domain when compared to StyleGAN3. In addition, generating complete human body images using StyleGAN2 is a baseline we would like to improve upon in our work.

#### 4.2.2 Image Outpainting

In contrast to inpainting methods, where an area of pixels within an image is missing or should be replaced, in image completion or outpainting problems the missing pixels are not surrounded by available pixels. Recent papers build on the ideas to use
generative adversarial networks [133] and the explicit modeling of structure [90, 163]. Though these two papers specialize in human bodies, we find that the GAN architecture CoModGAN [161] has even more impressive results for image outpainting (see the comparison in Section 4.5).

### 4.2.3 Conditional Generation of Full-Body Humans

Whereas unconditional generation doesn’t receive guidance or input beyond a stochastic latent, conditional generation is guided through additional input such as pose, segmentation mask, a reference image, or a textual description. First, conditional generation enables more control. Second, conditional generation can help in controlling variability and improve visual quality. In the context of humans, a natural idea is to condition the generation on the human pose [97, 127, 83, 99, 80, 119, 8] or segmentation information [69].

As many conditional architectures are not able to handle the same high resolution (1024×1024px) of unconditional StyleGAN, an alternative to developing new architectures is conditional embedding into an unconditional generator’s latent space. Two approaches used in this context are StyleGAN embedding using optimization [2, 3] or StyleGAN embedding using an encoder architecture [115, 137, 6]. Our work also makes use of embedding algorithms.

### 4.3 Methodology

We propose a method for the unconditional generation of full-body human images using one or more independent pretrained unconditional generator networks. Depending on the desired application and output configuration, we describe different ways to coordinate the multiple generators.
Figure 4.2: **InsetGAN Pipeline.** Given two latents $\mathbf{w}_A$ and $\mathbf{w}_B$, along with pretrained generators $G_A$ and $G_B$, that generate two images $I_A := G_A(\mathbf{w}_A)$ and $I_B := G_B(\mathbf{w}_B)$, respectively, we design a pipeline that can optimize either only $\mathbf{w}_A$ (a), or iteratively optimize both $\mathbf{w}_A$ and $\mathbf{w}_B$ (b) in order to achieve a seamless output composition of face and body. We use a set of losses $L_{\text{coarse}}$ and $L_{\text{border}}$ to describe the conditions we want to minimize during optimization. On the right, we show that given an input body, mere copy and pasting of a target face yields boundary artifacts. We show an application of one-way optimization (top right) and two-way optimization (bottom right) to create a seamlessly merged result. Note that when the algorithm can optimize in both inset-face and canvas-body generator spaces, it produces more natural results at the seam boundary – notice how the hair and skin tone blend from the head to the body region. The joint optimization is challenging as the bounding box $B(I_A)$ is conditioned on the variable $\mathbf{w}_A$. 
4.3.1 Full-Body GAN

The naive approach to generate a full-body human image is to use a single generator trained on tens of thousands of example humans (see Section 4.4 about the dataset). We adopt the state-of-the-art StyleGAN2 architecture proposed by Karras et al. [73]. Most previous full-body generation or editing work [88, 8, 164, 91] generate images at 256×256px or 512×512px resolution. We made the first attempt to unconditionally generate full-body humans at 1024×1024px resolution. Due to the complex nature of our target domain, the results generated by a single GAN sometimes exhibit artifacts such as weirdly-shaped body parts and non-photorealistic appearance. These artifacts are most visible in faces and extremities, as shown in Figure 4.1(a). Due to the vast diversity of human poses and appearances and the associated alignment difficulty, hands and feet appear in many possible locations in the training images, making them harder for a single generator to learn. Faces are especially hard since we humans are
ultra-sensitive to artifacts in these areas. They, therefore, deserve dedicated networks and special treatment. Figure 4.3 shows a variety of unconditional generation results. Our results exhibit correct human body proportions, consistent skin tones across the face and body, interesting garment variations, and plausible-looking accessories (e.g. handbags and sunglasses) whereas artifacts can be present when viewed in detail.

4.3.2 Multi-GAN Optimization

To improve the problematic regions generated by the full-body GAN, we use other generators trained on images of specific body regions to generate pixels to be pasted, as insets, into the full-body GAN result. The base full-body GAN and the dedicated body part GANs can be trained using the same or different datasets. In either case, the additional network capacity contained in the multiple GANs can better model the complex appearance and variability of the human bodies. As a proof of concept, we show that a face GAN trained with the face regions cropped from our full-body training images can be used to improve the appearance of the body GAN results. Alternatively, we can also leverage a face generator trained on other datasets such as FFHQ [76] for face enhancement as well. Similarly, specialized hands or feet generators can also be used in our framework to improve other regions of the body. We show that we can also use multiple part generators together in a multi-optimization process, as depicted in Figure 4.4.

The main challenge is how to coordinate multiple unconditional GANs to produce pixels that are coherent with one another. In our application, we have a $G_A$ that generates the full-body human where $I_A := G_A(w_A)$ and another $G_B$ that generates a sub-region or inset within the human body where $I_B := G_B(w_B)$. In order to coordinate the specialized part GAN with the global/canvas GAN, we need a bounding box detector to identify the region of $I_A$ that corresponds to the region our part GAN generates. We crop $I_A$ with the detected bounding box and denote the cropped pixels as $B(I_A)$. The
Figure 4.4: **Two Insets.** These results are improved using a dedicated shoe generator trained on shoe crops from our original dataset, and also using our face generator. All three generators (full-body canvas and two insets) are jointly optimized to produce a seamless coherent output with improved faces and shoe regions. The circular closeups show the shoes before *(bottom)* and after *(top)* the improvement optimization (please zoom).

The problem of inserting a separately-generated part $I_B$ into the canvas $I_A$ is equivalent to finding a latent code pair $(w_A, w_B)$ such that the respective images $I_A$ and $I_B$ can be combined without noticeable seams in the boundary regions of $B(I_A)$ and $I_B$. To generate the final result, we directly replace the original pixels inside the bounding box $B(I_A)$ with the generated pixels from $I_B$,

$$\min_{w_A, w_B} \int_{\Omega} L(G_A(w_A), G_B(w_B))$$

(4.1)

where, $\Omega := B(G_A(w_A))$ and, with slight abuse of notation, $L$ captures the loss both along the boundary of $\Omega$ measuring seam quality and inside the region $\Omega$ measuring similarity of $I_A$ and $I_B$ inside the respective faces. The full optimization is complex as the region of interest $\Omega$ depends on $w_A$.

Our multi-GAN optimization framework can support various human generation and editing applications. Depending on the application scenario, we optimize either $w_A$ or $w_B$ or jointly optimize both for the best results.
4.3.3 Optimization Objectives

When optimizing the latent codes $\mathbf{w}_A$, $\mathbf{w}_B$ or both, we consider multiple objectives: (I) the face regions generated by the face GAN and body GAN should have a similar appearance at a coarse scale so that when the pixels generated by the face GAN are pasted onto the body GAN canvas, attributes match (e.g., the skin tone of the face matches that of the neck); (II) the boundary pixels around the face crops match up so that a simple copy-and-paste operation does not result in visible seams; and (III) the final composed result looks realistic.

To match the face appearance, we downsample the face regions and calculate a combination of $L_1$ and perceptual loss [160] $L_{\text{LPIPS}}$:

$$
L_{\text{coarse}} := \lambda_1 L_1(I_{A}^\downarrow, I_{B}^\downarrow) + \lambda_2 L_{\text{LPIPS}}(I_{A}^\downarrow, I_{B}^\downarrow),
$$

(4.2)

where $I_{A}^\downarrow = D_{64}(B(I_A))$ and $I_{B}^\downarrow = D_{64}(I_B)$ and $D_{64}$ refers to downsampling the image to 64×64px resolution.

For the boundary matching, we also apply a $L_1$ and perceptual loss to the border pixels at full resolution:

$$
L_{\text{border}} := \lambda_3 L_1(\mathcal{E}_8(B(I_A)), \mathcal{E}_8(I_B)) + \lambda_4 L_{\text{LPIPS}}(\mathcal{E}_8(B(I_A)), \mathcal{E}_8(I_B))
$$

(4.3)

where $\mathcal{E}_x(I)$ is the border region of $I$ of width $x$ pixels.

To maintain realism during the optimization, we also add two regularization terms:

$$
L_{\text{reg}} := \lambda_{r1} \left\| \mathbf{w}^* - \mathbf{w}_{\text{avg}} \right\| + \lambda_{r2} \sum_i \left\| \mathbf{\delta}_i \right\|
$$

(4.4)

The first term prevents the optimized latent code from deviating too far from the average latent. We compute $\mathbf{w}_{\text{avg}}$ by randomly sampling a large number of latents in
Figure 4.5: **Face Refinement.** Given generated humans, we use a dedicated face model trained on the same dataset to improve the quality of the face region. We jointly optimize both the face and the human latent codes so that the two generators coordinate with each other to produce coherent results. The two inset face crops show the initial face generated by the body GAN (bottom) and the final face improved by dedicated face GAN (top).

$Z$ space, mapping them to $W$ space, and computing the average. The second term is to regularize the latent code in $w^+$ latent space. During StyleGAN2 inference, the same 512 dimensional latent code $w$ is fed into each of the $n$ generator layers ($n$ is dependent on the output resolution). Many GAN inversion methods optimize in this $n \times 512$ dimensional $w^+$ latent space [147] instead of the 512 dimensional $w$ latent
space. We follow recent work to decompose the $w^+$ latent into a single base $w^*$ latent and $n$ offset latents $\delta_i$. The latent used for layer $i$ is $w + \delta_i$. We use the $L_2$ norm as a regularizer to ensure that the $\delta_i$s remain small. Based on our visual analysis of the results, we use larger weights for the body generator than the face generator for this regularizer.

We mix and match the various losses depending on the specific application at hand.

4.3.4 Face Refinement versus Face Swap

Given a randomly generated human body $G_A(w_A)$, we can keep $w_A$ fixed and optimize for $w_B$ such that $G_B(w_B)$ looks similar to $B(G_A(w_A))$ at a coarse scale and matches the boundary pixels at a fine scale (Figure 4.2 top right). We have:

$$\min_{w_B} (L_{\text{coarse}} + L_{\text{border}})$$

While this almost produces satisfactory results, boundary discontinuities show up at times. For further improvement, we can optimize both $w_A$ and $w_B$ so that both generators coordinate with each other to generate a coherent image free of blending artifacts (Figure 4.2 bottom right). To keep the body appearance unchanged during the optimization of $w_A$, we introduce an additional loss term:

$$L_{\text{body}} := \lambda_5 L_1(R^O(I_A), R^O(I_{\text{ref}})) + \lambda_6 L_{\text{LPIPS}}(R^O(I_A), R^O(I_{\text{ref}}))$$

where $I_{\text{ref}}$ is the input reference body generated by $G_A$ that should remain unchanged during the optimization, $R^O$ defines the body region outside of the face bounding box. We also use the mean latent regularization term $L_{\text{reg}}$ to prevent generating artifacts.
The final objective function becomes:

$$\min_{w_A, w_B} \left( \mathcal{L}_{\text{coarse}} + \mathcal{L}_{\text{border}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{body}} \right)$$  \hspace{1cm} (4.7)

Figure 4.1(b) and Figure 4.5 show face refinement results using a dedicated face model trained on faces cropped from the same data used for training the body generator. We show this dataset in Figure 4.11. Our refinement results when using the pretrained FFHQ face model exhibit similar visual quality, as shown in Figures 4.7, 4.18 and 4.19.

### 4.3.5 Body Generation for an Existing Face

Given a real face or a randomly-generated face $G_B(w_B)$, we can keep $w_B$ fixed and optimize for $w_A$ such that $G_A(w_A)$ produces a body that looks compatible with the input face in terms of pose, skin tone, gender, hairstyle, etc. In practice, we find that to best maintain boundary continuity, especially when generating bodies to match faces of complex hairstyles, it is often to discourage large changes in $w_B$, such that the face identity is mostly preserved but the boundary and background pixels can be slightly adjusted to make the optimization of $w_A$ easier. To preserve the face identity during the optimization, we use an additional face reconstruction loss:

$$\mathcal{L}_{\text{face}} := \lambda_7 \mathcal{L}_1(R^I(I_B), R^I(I_{\text{ref}})) + \lambda_8 \mathcal{L}_{\text{LPIPS}}(R^I(I_B), R^I(I_{\text{ref}}))$$  \hspace{1cm} (4.8)

where $R^I$ defines the interior region of the face crop and $I_{\text{ref}}$ denotes the referenced input face. For more precise control, face segmentation can be used instead of bounding boxes. Our objective function becomes:

$$\min_{w_A, w_B} \left( \mathcal{L}_{\text{coarse}} + \mathcal{L}_{\text{border}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{face}} \right)$$  \hspace{1cm} (4.9)
Figure 4.6: **Body Generation for an Existing Face.** For each face generated by the pretrained FFHQ model (*middle column*), we use joint optimization to generate three different bodies while maintaining the facial identities from the input faces.

With different initialization for $w_A$, we can generate multiple results per face as shown in Figure 4.6. Note that our model can generate diverse body appearances compatible with the input face. The generated body skin tone generally matches the input face skin tone (e.g., the dark-skinned women in the top and bottom rows of Figure 4.6). Figure 4.1(c) shows another example.
Figure 4.7: **Face Body Montage.** Given target faces (top row) generated by the pretrained FFHQ model and bodies (left column) generated by our full-body human generator, we apply joint latent optimization to find compatible face and human latent codes that can be combined to produce coherent full-body humans. Notice how the face and skin colors get synchronized, and zoom in to observe the (lack of) seams around face insets.
4.3.6 Face Body Montage

We can combine any real or generated face with any generated body to produce a photo montage. With a real face, we need to first encode it into the latent space of $G_B$ as $w_B$ using an off-the-shell encoder [137]. Similarly, a real body could be encoded into the latent space of $G_A$, but due to the high variability of human bodies, it is difficult to achieve a low reconstruction error. All montage results are created from synthetic bodies generated by $G_B$. We use the following objective function:

$$\min_{w_A, w_B} (L_{\text{coarse}} + L_{\text{border}} + L_{\text{reg}} + L_{\text{face}} + L_{\text{body}})$$

Figure 4.7 shows the result of combining faces (top row) generated by the pretrained FFHQ model with bodies (leftmost column) generated by our full-body generator $G_A$. With minor adjustments of both the face and body latent codes, we achieve composition results that are coherent and identity-preserving. While we do not have any explicit loss encouraging skin tone coherence, given faces with different skin tones, our joint optimization slightly adjusts the skin tone of the body’s neck and hand pixels to minimize appearance incoherence and boundary discrepancy in the final results. Figure 4.1(d) shows two more examples. Our joint optimization is able to slightly adjust the shoulder region of the lady to extend her hair to naturally rest on her right shoulder. The rightmost column in Figure 4.2 shows the improvement the joint optimization makes to the final result quality (bottom) compared to only optimizing $w_B$ given an input body (top).

4.3.7 Optimization Process

While the difference is subtle, we observe a slightly better visual performance when using $L_1$ over $L_2$ losses. We apply many of our losses to downsampled versions of the images $D_{64}(B(I_A))$ and $D_{64}(I_B)$, reducing the risk of overfitting to artifacts from the
source image (e.g., the body’s face region, which lacks realistic high-frequency details) in a strategy similar to PULSE [100].

One challenge in the joint optimization of $w_A$ and $w_B$ is that the boundary condition $\Omega$ depends on the variable $w_A$. We address this by alternately optimizing for $w_A$ and $w_B$, and reevaluating the boundary after each update of $w_A$. We stop the process when the updates converge.

**Optimization Initialization.** The default choice of initialization for either $w_A$ or $w_B$ is their corresponding average latent vector $w_{avg}$. This typically leads to reasonable results quickly. However, it is desirable to generate a variety of results for applications like finding matching bodies $I_A$ for an input face $I_B$. In this case, we start from truncated latent codes $w_{trunc} = w_{rand} \ast (1 - \alpha) + w_{avg} \ast \alpha$. Due to the introduced randomness and the interpolation with the average latent code, we can generate diverse yet realistic results (see Figure 4.6). In Figure 4.8, given humans generated by our full-body model trained on DeepFashion, we use the pretrained FFHQ face model to swap in multiple better-looking faces. Different initialization of $w_B$ yields different results. In the cases where either the face region or the body region should remain fixed during the joint

![Figure 4.8: Multimodal Face Improvement.](image)

Figure 4.8: **Multimodal Face Improvement.** To improve humans generated by a full-body model trained on DeepFashion, we use the pretrained FFHQ model to synthesize a variety of results that all look compatible with the input body.
optimization of both latent codes, we initialize the optimization with the latent code initially used to generate the synthetic reference image or the latent code encoded from a real image.

**Optimization Details.** All our results were optimized using ADAM. We usually stop the optimization when the edge loss falls below a certain threshold (usually defined as $\mathcal{L}_1(border)(w_B) < 0.09$) or after the number of iterations exceeds a threshold (typically 1000 optimization steps). When performing joint optimization, we define two distinct optimizers for $w_A$ and $w_B$ and switch the optimization target every 50 iterations. Depending on the application, we can start with the canvas optimizer or the inset optimizer. We choose different learning rates for the canvas optimizer: $lr = 0.05$ and for the inset optimizer: $lr = 0.002$. We reevaluate the bounding box every 25 iterations during optimization for a certain number of iterations (typically 150 iterations during body generation, 75 iterations during face refinement.) before keeping the bounding box fixed. We observe that reevaluating the bounding box too often or too long makes the optimization unstable.

**Lambda weights for Losses.** We report the $\lambda$ weight combination we use for the face body montage application. In this use case, we have losses for improving the coherence of $G_A$ and $G_B$ from the perspective of each GAN, as well as losses for controlling the appearance of each output, either by constraining closeness of the center image region (face) to a target or some outer image region (body) to adhere to a specific body.
We define several different optimization targets with custom parameters and setup:

(1) **Improving the face area of a given human image.**
We either run one-way optimization or take a small learning rate for the human optimizer to allow for a small wiggling of the canvas area around the inset, which generally improves the coherence of inset and canvas. We start with optimizing the inset from a random starting point for a certain number of iterations (e.g. \( n=100 \)) and then fix the inner face area by adding an additional loss constraint keeping the face close to the remembered state. This allows more iterations for the boundary area to improve but prevents overfitting to unwanted artifacts in the face region of the input canvas or deterioration of the facial quality due to over-optimization.

(2) **Finding suitable bodies for a given input face.**
We start with optimizing the canvas from a random starting point for a certain amount of iterations (e.g. \( n=150 \)), allowing the body generator to roughly hal-
lucinate a person with similar facial structure as the target. Then, we switch to an alternating optimization schedule, allowing the appearance of body and face to gradually resemble each other in the boundary regions. We can regularize the appearance of the body using a similar strategy as described above to avoid over-optimization.

(3) **Seamlessly combine a given face and body.**

In order to maintain the original appearance of both the input face as well as the body, we constrain the joint optimization so that the face GAN result stays close to the input face and the body GAN result outside of the face stays close to the input body.

### 4.4 Dataset and Implementation

We curate a proprietary dataset of 83,972 high-quality full-body human photographs at 1024×1024px resolution. These images stem from a dataset of 100,718 diverse photographs in the wild purchased from a third-party data vendor. The dataset includes hand-labeled ground-truth segmentation masks. We apply a human pose-detection network [19] on the original images and filter out those that contain extreme poses causing pose detection results to have low confidence. Figure 4.9 shows some sample training images. Feature alignment plays an important role in high-quality image generation, as can be seen in the qualitative difference between models trained on FFHQ data vs. other face datasets. Therefore, we carefully align the humans using their pose skeletons. We define an upper body axis based on the position of the neck and hip joints. We position humans so that the upper body axis is aligned with the center of the image. As the variance in perspective and pose is very large, choosing the appropriate scale for each person within the context of their image frame is challenging. We scale the humans based on their upper-body length and then evaluate the extent of the face
We create a dataset from photographs of humans in the wild. The images are automatically preprocessed, aligned, and cropped to 1024×1024px resolution using the ground-truth segmentation masks and detected pose skeletons.

region as defined by the segmentation mask. If the face length is smaller (larger) than a given minimum (maximum) value, we rescale so that the face length is equal to the minimum (maximum).

Lastly, we enlarge the backgrounds using reflection padding and heavily blur them using a Gaussian kernel of size 27 to focus the generator capacity on modeling only the foreground humans. The huge variation in background appearance in these in-the-wild photos poses extreme challenges for the GAN, especially with limited data.

We also considered completely removing the background but did not do it for two reasons: (1) Human-labeled segmentation masks are still imperfect around boundaries and (2) We observe that current GAN architectures do not handle large areas of uniform color well.

We also show our method on DeepFashion [93], which consists of 66,607 fashion photographs, including garment pieces and garments on humans. Using the same alignment strategy as above, we extract 10,145 full-body images at 1024×768 resolution. Since the backgrounds are already uniform, we do not blur them.
4.4.1 Training Details

We trained our main human body generator network at 1024×1024px resolution using the StyleGAN2-ADA architecture using all augmentation schemes proposed in the paper [73] for 28 days and 18 hours on 4 Titan V GPUs, using a batch size of 4, processing a total of 42M images. After experimenting with different $R1 \gamma$ values between 0.1 and 20, we chose a value of 13. Similarly, we trained our DeepFashion human generator network at 1024×768px resolution for 9 days on 4 v100 GPUs, using a batch size of 8, processing a total of 18M images. We use a pretrained FaceNet [121] to detect and align the bounding boxes of the face regions in our generated bodies and faces. The running time of our optimization algorithm for jointly optimizing two generator latents at 1024×1024px output resolution is about 75 seconds on a Titan RTX GPU. If $G_B$ has a smaller resolution of 256×256px, the optimization time decreases to around 60 seconds.

4.4.2 Unconditional Generation and Adaptive Truncation

Since our generator is trained on very diverse data, we can observe a wide range of image quality when generating untruncated output. When truncating the generated results as described in the original StyleGAN2 paper [76] by linearly interpolating from the sample position in $w$ space to the average latent $w_{avg}$ we can drastically reduce artifacts in pose and details. However, this trick also reduces the diversity in the sample output, and notably reduces the color vibrancy of the output images, as outfit colors are interpolated towards an averaged greyish hue. In our approach, whenever possible (i.e. whenever we are not constrained to operate in the $w$ space), we use a layer-adaptive truncation scheme to generate a visually pleasing result of improved perceptual quality while preserving as many diverse features as possible from the untruncated samples, as shown in Figure 4.10.
Figure 4.10: **Adaptive Truncation.** We show a set of untruncated samples from our human generator exhibiting unrealistic poses and unwanted artifacts. Standard truncation \((t=0.6, \text{ bottom row})\) reduces artifacts, but also removes desirable clothing details and reduces the color vibrancy. Our adaptive truncation (center row) better preserves colors, texture details, and accessories.

To achieve this, when generating unconditional samples, we use the \(w^+\) space and define a separate truncation value for each layer. In our generator, we have 18 layers, and we define the layer-wise truncation values as

\[
t = [0.35, 0.25, 0.25, 0.70, 0.75, 0.65, 0.65, 0.40, 0.40, 0.35, 0.25, 0.15, 0.15, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05]
\]

The values were chosen through experimentation where we truncate individual layers
separately to identify the ones that cause the most artifacts. Note that we apply almost no truncation on later layers, as they can be used to generate desirable clothing details, vibrant colors, and accessories and do not cause significant artifacts. We observe that latent codes for the middle layers (4-7) are most responsible for artifacts, so we truncate them the most.

We also measure the Fréchet Inception Distance (FID) of 4K random results generated using our adaptive truncation scheme and observe a significantly lower FID (53.26) as compared to using regular truncation at $t=0.6$ (71.89). We would like to point out that we did not use the adaptive truncation trick when we performed the quantitative evaluations in Section 4.5.1, both for clarity and simplicity and because we were optimizing in $w + \delta_1$ space, which restricts the effect of adaptive truncation.

### 4.5 Evaluation and Discussion

#### 4.5.1 Quantitative Evaluation

**Image Quality.** We follow the standard practice to calculate FID (Fréchet Inception Distance) to measure how closely our generated full-body results follow the training distribution. Many previous papers including CoModGAN point out that FID statistics are noisy and do not correlate well with human perception of visual quality. We also observe that FID is more sensitive to result diversity than quality and increases significantly as we truncate the generated results, which reduces variation but is crucial for generating natural-looking images with fewer artifacts. While the FID for untruncated results is 13.96, it rises to 26.67 for $t=0.7$ and 71.90 for $t=0.4$ (more truncation). We compare FID values of several alternative approaches for our face refinement application. We use two different truncation settings, $t=0.7$ and $t=0.4$, and evaluate both the full-body images and image crops that include the refined face and the boundary pixels after copy&pasting.
Table 4.2: **Quantitative evaluation.** We evaluate the Fréchet Inception Distance of our improved results with respect to the unconditional generation at two different truncation values.

<table>
<thead>
<tr>
<th>FID Score (lower is better)</th>
<th>BODY</th>
<th>FACE</th>
<th>BODY</th>
<th>FACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconditional Generation</td>
<td>26.67</td>
<td>27.14</td>
<td>71.90</td>
<td>66.61</td>
</tr>
<tr>
<td>InsetGAN</td>
<td>25.33</td>
<td>31.61</td>
<td>69.58</td>
<td>61.57</td>
</tr>
</tbody>
</table>

The differences in FID are small, indicating that the face refinement using joint optimization does not modify the distribution learned by the unconditional generator and therefore does not decrease the result diversity. However, large differences in perceptual quality are still possible despite similar FID values, as demonstrated in our user study.

**Precision and Recall Scores.** In addition to calculating the FID scores to quantitatively evaluate our InsetGAN improved results, we also followed Kynkanniemi *et al.* [86] and evaluated the precision and recall score in Table 4.3. Precision and Recall provide a more disentangled way of mapping the quality and variability of samples. These metrics are a very intuitive quantitative evaluation tool for GANs. Of the two calculated values, precision describes a measurement of image quality (higher=better quality), and recall quantifies the variability of the generated images (high=more variability). Both metrics are scaled between 0 and 1.

We evaluate these scores on more ($t = 0.4$) and less ($t = 0.7$) truncated results to observe the impact of improving overall full-body generation quality at the cost of lowering the variability. All evaluations are performed both on the full-body image as well as on a crop area around the face region that includes a border around the pasted region to evaluate the image coherence. These generated images are compared to the dataset to evaluate a precision and recall score. We calculate the baseline as the precision & recall of unconditional generation of our model (1) and then evaluate the
scores for two different datasets used for improving the face region: (2) the pretrained FFHQ face generator and (3) our custom face dataset trained on the same data as the full-body generator as shown in Figure 4.21.

We can see that our method is able to achieve a significant improvement in precision throughout all experiments. The increase in precision is particularly large for the less-truncated \( t = 0.7 \) experiments exhibiting more artifacts in the unconditional generator, where we are able to improve the precision by a large margin using our own model. We can also achieve a comparable improvement using the FFHQ model, which is somewhat surprising since the training data is not based on the same input distribution as the full-body generator. This shows (a) that the generative capabilities of well-trained GANs are providing powerful generalizable models of their domain and (b) that our method is able to encourage good results even for specialized part generators that are trained on a completely different distribution. Note that our improvements come at a cost of a small drop in recall, which denotes that the variability of the samples goes down a little in most instances. The drop is insignificant when measured on the full body, yet noticeable when calculating the metrics on the cropped face area.

Table 4.3: Precision and Recall evaluation. We evaluate the Precision and Recall scores [86] of our improved results for two datasets with respect to the unconditional generation at two different truncation values for both the whole image and a crop area of the face region.

<table>
<thead>
<tr>
<th>( t = 0.7 )</th>
<th>Full-body Image</th>
<th>Face Crop Area</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRECISION</td>
<td>RECALL</td>
</tr>
<tr>
<td>(1) unconditional</td>
<td>0.6958</td>
<td>0.3280</td>
</tr>
<tr>
<td>(2) FFHQ</td>
<td>0.8293</td>
<td>0.3126</td>
</tr>
<tr>
<td>(3) our dataset</td>
<td><strong>0.8364</strong></td>
<td>0.3076</td>
</tr>
<tr>
<td>( t = 0.4 )</td>
<td>Full-body Image</td>
<td>Face Crop Area</td>
</tr>
<tr>
<td></td>
<td>PRECISION</td>
<td>RECALL</td>
</tr>
<tr>
<td>(1) unconditional</td>
<td>0.9247</td>
<td>0.0334</td>
</tr>
<tr>
<td>(2) FFHQ</td>
<td><strong>0.9333</strong></td>
<td>0.0336</td>
</tr>
<tr>
<td>(3) our dataset</td>
<td>0.9298</td>
<td><strong>0.0362</strong></td>
</tr>
</tbody>
</table>
We attribute this drop in recall to the fact that we reduce artifacts caused by outliers and unusual generated samples in the images, which decreases the variability in the samples. Our results on larger truncation ($t = 0.4$) paint a similar picture, albeit with smaller margins in the improvement, as the generator is significantly more restricted in this setting. We can also see that the Recall values at this truncation level are already extremely low.

**Issues of FID and Training Data.** As shown in Section 4.5.1, the FID score [57] of unconditionally generated samples is similar to that of InsetGAN improved samples. We think there are two main reasons:

1. Even though FID is the most commonly used metric for evaluating image generation quality, it does not correlate well with human perceptual quality and cannot effectively capture subtle visual differences as discussed in [161]. Our user study results also contradict results based on FID because in the user study, our InsetGAN results are clearly preferred by the users but FID cannot properly reflect the quality improvements.

2. Our dataset contains photographs of varying quality and resolution, ranging from high-resolution studio-quality photographs to low-lighting cellphone snapshots. Additionally, human subjects might only occupy small regions of the original photographs. After cropping and resampling, artifacts can be quite visible and sometimes magnified. For instance, as shown in Figure 4.11 left, we observe JPG artifacts, motion blur, and noisiness caused by low-lighting conditions. Our face GAN trained on cropped face regions from this dataset can alleviate some of these artifacts when used to improve the generated faces from the trained human GAN as shown in Figure 4.11 right. We notice a good number of our randomly-sampled 4K training images used for FID evaluation contain artifacts. The human GAN-generated faces that contain more artifacts might accidentally have more similar
distribution to the training set than the nice clean faces generated by the face
GAN used in our joint optimization.

4.5.2 Baseline Comparison

To the best of our knowledge, no other prior work generates full-body humans uncondition-ally or inpaints/outpaints humans at 1024×1024px resolution without requiring conditioning other than reference pixels of the known regions. Previous works have attempted to generate plausible human bodies, but they require segmentation masks as input [91, 90]. The best state-of-the-art method that can be repurposed for our body generation and face refinement applications is CoModGAN [161]. Figure 4.12 shows that our InsetGAN (top right) outperforms CoModGAN (bottom right) in replacing the initial face generated by our body generator (left). We trained CoModGAN with square

![Comparison of faces](image)

Figure 4.11: Dataset Quality. We show a comparison of faces cropped from our dataset (left) with faces sampled from unconditionally generated and InsetGAN-improved humans (right). Zoom in to observe the variable quality of the input data.
Figure 4.12: **Face Refinement Comparison with CoModGAN [161]**. Given generated humans (left), InsetGAN improves the face quality (top right), producing more convincing results than CoModGAN (bottom right). CoModGAN results are generated by defining rectangular holes around the face regions.

(with small random offsets for generalization) holes around faces using the official implementation, training data, and default parameters for two weeks on four V100 GPUs. Similarly, we train CoModGAN with rectangular holes around the bodies to compare with our InsetGAN for the body generation task. In Figure 4.13, we show the two best results of CoModGAN obtained by using several random initializations per input face. Compared to our results in Figure 4.6, CoModGAN produces less realistic and diverse image completions.

Figure 4.13: **Body Generation with CoModGAN [161]**. We show results generated by CoModGAN trained to fill a rectangular hole covering the body in a given image. Inputs with holes are shown in the insets. We generate several results per input and show the best-looking two here. Please refer to Figure 4.6 for our results on the same input faces. We observe that CoModGAN creates seamless content, but worse visual quality compared to ours.
Figure 4.14: **Unconstrained Comparison with CoModGAN.** We remove the conditional constraint on the face and allow for unconditional (only edge-conditional) face insertion. In contrast to Figure 4.12, we see that the face is allowed to diverge from the face input. We also keep the body latent fixed so that the body pixels in both our results and CoModGAN results remain unchanged.

In Figure 4.14, we show an additional comparison of our method to CoModGAN where we evaluate the quality of our "inpainting" capabilities after removing the constraint that the output face needs to be similar to the underlying input face. We only optimize the face latent code based on the edge coherence term and keep the body latent code fixed. This makes the comparison fairer since CoModGAN invents completely new faces based only on the context pixels outside the input bounding box and does not alter the pixels of the body. We show that we are able to generate plausible and coherent results without using the input face as guidance and without joint optimization.

### 4.5.3 User Study

We performed a user study to better evaluate the perceptual quality of our method. We aggregated 500 generated humans from our full-body generator and 500 random training images. We then applied either our joint optimization method or CoModGAN to replace the face regions in the generated samples. We showed several sets of image pairs to volunteer participants on Amazon Mechanical Turk and asked them to pick
“in which of the two images the person looks more plausible and real”. Per image pair, 5 votes were collected and aggregated towards the majority votes. The study shows that in 12.4% of image pairs, users prefer our unrefined results over the training images when given only 1 second to look at each image. This shows that our results get the basic human proportion and pose right, being able to confuse people about being real. In 98% of cases, users prefer our joint optimization results over the unrefined images. In contrast, only 7% of the CoModGAN samples were picked over the unrefined images, which is consistent with our observation from Figure 4.12.

Figure 4.15: **User Study Interface.** We show the interface provided to users during the experiment where users were asked to pick their preferred result from the vanilla StyleGAN2 generated result and the InsetGAN improved result. Each experiment consisted of 10 image pairs.
**User Study Details.** We provide additional details about the user studies we conducted on Amazon Mechanical Turk. We adopt a forced choice paired comparison procedure where the participant is shown a pair of images at a time and is asked to “select in which of the two images the person looks more plausible and real” as seen in Figure 4.15. For each HIT, we randomize the pair orders and randomly choose whether the left or the right is our result.

We performed four different independent studies:

1. Compare unconditionally generated samples (truncated with $t = 0.4$) with images in our training set.
2. Compare unconditionally generated samples ($t = 0.4$) with the results of joint InsetGAN optimization for face refinement.
3. Compare unconditionally generated samples ($t = 0.4$) with the results of using CoModGAN for face regeneration.
4. Compare our InsetGAN results with CoModGAN results directly.

In study (1) the images are unpaired since there is no correspondence between any generated image and any training image. Given two images, we show the first one for a second and then the other one for another second. The images are rendered at 384×768px resolution so that they fit into the browser without the need for scrolling. For studies (2), (3), and (4), we show 512×1024px center-cropped images side-by-side to the participants so that they can focus on the differences in the image details. The selection buttons above the image pairs are faded in after a 6-second delay so that users are encouraged to carefully study the image differences before making their selections. We collect 5 votes per image pair and choose the winning image that receives 3 or more votes. We summarize the results in the following table:
<table>
<thead>
<tr>
<th>Study</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COUNT</td>
<td>PERCENT</td>
</tr>
<tr>
<td>(1)</td>
<td>Real</td>
<td>Generated</td>
</tr>
<tr>
<td>(2)</td>
<td>Generated</td>
<td>InsetGAN</td>
</tr>
<tr>
<td>(3)</td>
<td>Generated</td>
<td>CoModGAN</td>
</tr>
<tr>
<td>(4)</td>
<td>InsetGAN</td>
<td>CoModGAN</td>
</tr>
</tbody>
</table>

### 4.5.4 Limitations

Our work has multiple limitations that could benefit from improvements. First, the joint optimization approach may change details such as hairstyle, neckline, or clothing details. In many cases the changes are minor, but in some cases, changes can be larger, e.g. the woman's hair in the middle row of Figure 4.6 and the man's collar in the top row of Figure 4.7. Second, as shown in Figure 4.3, our full-body GAN has other problems not improved by InsetGAN: symmetry, e.g. noticeable in the hands and feet and on outfits (the woman with a light blue shirt), and the consistency of the fabric used for the clothing (the rightmost person). Third, the generated results have limited variations in body type and pose as discussed next in more detail. The vast majority of our generated results have a slender body type due to the training data distribution.

### 4.5.5 Dataset Bias and Societal Impact

Both DeepFashion and our proprietary dataset contain biases. DeepFashion contains a limited number of unique identities. The same models appear in multiple images. An overwhelming majority of the images are female (around 9:1) and the range of age, ethnicity, and body shape does not represent the real human population. As a result, models trained on it can only generate a limited range of identities (mostly young white females) as shown in Figure 4.8. We made our best attempt to look for
a diverse dataset and purchased one from a data vendor in East Asia but noticed the over-representation of young Asian females in the images. Also, many images appear to portray slim street fashion models; as a result, the vast majority of them contain slender body types and formal attire. Models trained on biased datasets tend to learn biased representations of the human body. Due to the over-representation of Asians, our results on other ethnicities contain more artifacts in the face region (see the rightmost four results in Figure 4.3). We encourage future research efforts to diversify the training dataset to better serve our diverse society. The unconditional nature of our generation process combined with the limitation of some optimization losses operating at a low resolution implies that our generated human outputs might not preserve the attributes in the input image exactly. In addition, since the age distribution in our data is almost exclusively adult humans, we are not able to faithfully produce bodies for the faces of children. Similar to other human domain generative models, our approach can be exploited by malicious users to produce deep fakes. However, as we have seen in the user study, even in a short second, users identify most of our generated results as fake compared to real human images. As we further improve the result qualities, we hope, and encourage other researchers, to investigate deep fake detection algorithms.

4.6 Conclusion

We presented InsetGAN, the first viable framework to generate plausible-looking human images unconditionally at 1024×1024px resolution. The main technical contribution of InsetGAN is to introduce a multi-GAN optimization framework that jointly optimizes the latent codes of two or more collaborative generators. In future work, we propose to extend the multi-generator idea to 3D shape representations, such as 3D GANs or auto-regressive models based on transformers. We also plan to demonstrate the InsetGAN framework on other image domains and investigate coordinated latent editing in the multi-GAN setup.
4.7 Additional Results and Discussion

4.7.1 Two Insets

We demonstrate in Figure 4.16 that our technique is able to generate good results for insets of another domain beyond faces. To that end, we trained a shoe generator at 256×256px resolution on shoes cropped from the same dataset and use the generated shoe outputs to improve problematic areas in the canvas domain with higher-quality shoe insets exhibiting more detailed and natural features than the full-body GAN. These results also demonstrate that our technique can jointly optimize more than one inset: in this example, we select a target face, and target shoes, and find an appropriate body, optimizing all three generators to create a seamless output.

4.7.2 Face Orientations

While our results exhibit somewhat limited body poses due to the entanglement of body pose variability and deterioration in image quality, we show that we are able to still capture a variety of face orientations and our technique can match the face orientation with a plausible-looking body oriented correctly based on the face target, as shown in the results in Figure 4.17.

Figure 4.16: Two Inset Optimization. We show some additional results of our technique successfully merging a canvas with two independent insets (shoes and face).
Figure 4.17: **Orientations.** We demonstrate that our technique can capture a wide range of face orientations and generate natural-looking face-body compositions for each respective input face. The body poses are oriented accordingly.

### 4.7.3 Face-Body Montage

We show some additional results using a human generator trained on our custom dataset as well as another generator trained on the DeepFashion dataset as seen in Figures 4.18 and 4.19. The generator trained on DeepFashion is able to synthesize bodies and garment details with good quality, but the generator is overfitted to the limited quantity and variety of the input data and is thus not very flexible in harmonizing skin tone differences (columns 2 and 3 in Figure 4.19). Note how the results in Figure 4.18 adapt the hair and body composition, even generating reasonable results for the short-haired woman in the rightmost column.
Figure 4.18: **Face Body Montage.** Given target faces (top row) generated by the pretrained FFHQ model and bodies (leftmost column) generated by our full-body human generator trained on our custom dataset, we apply joint latent optimization to find compatible face and human latent codes that can be combined to produce coherent full-body humans.
Figure 4.19: **Face Body Montage.** Given target faces (top row) generated by the pretrained FFHQ model and bodies (leftmost column) generated by our full-body human generator trained on DeepFashion [93], we apply joint latent optimization to find compatible face and human latent codes that can be combined to produce coherent full-body humans.
4.7.4 Latent Space Walk

We are able to create a joint latent space walk as shown in Figure 4.20 by linearly interpolating both the face and the body latents and using them as initialization in our joint optimization framework to combine the face and body seamlessly in a temporally coherent way. We explain the process to obtain a seamless interpolation in $n$ steps based on two keyframes $K_{\text{start}}$ and $K_{\text{end}}$ and their corresponding optimized face and body latent pairs $(w_{A\text{start}}, w_{B\text{start}})$ and $(w_{A\text{end}}, w_{B\text{end}})$.

The naive solution is to simply linearly interpolate $n$ times between these latent pairs to obtain the in-betweens. However, interpolating in each of the two latent spaces independently does not yield a seamlessly merged boundary region between the canvas and the inset. In order to improve this boundary, we consider the latent space walk of the canvas as fixed and define the optimization of the inset as an interpolation problem where we optimize frame by frame and do the following at frame $i$:

1. Consider the previous frame (initially $K_{\text{start}}$) and obtain the next frame as the linear interpolation given by $f = \frac{1}{n-i}$ and $w_{B\text{next}} = (1-f) \times w_{B\text{previous}} + f \times w_{B\text{end}}$.
2. To avoid unwanted jittering, we no longer reevaluate the face bounding box per frame but linearly interpolate from $B_{\text{start}}$ to $B_{\text{end}}$ to obtain a smooth transition from one inset position to the next.

Figure 4.20: Latent Space Walk. We can optimize for a joint latent space walk of multiple generators by interpolating in both latent spaces and optimizing for the boundary of the resulting interpolation sequences.
(3) Optimize $w_{B\text{next}}$ for a small number of iterations (e.g., 100 optimization steps) with a set of losses optimizing for (a) the edge coherence with the canvas, (b) the identity preservation with the starting point of $w_{B\text{next}}$, and (c) the minimization of the edge region changes with respect to the last frame $K_{\text{previous}}$.

We use this method to insert about 20 to 40 interpolated frames between two given keyframes and render the resulting animation to a video at 16-20fps. By replicating the first keyframe as the last, the latent space walk can loop infinitely.

### 4.7.5 Custom Face Generator

Most results in this chapter use a face generator trained on the same data used to train our human GAN. We crop the faces and resample them to $256 \times 256$ px resolution and train a face generator using the StyleGAN2 architecture. We show some generated face samples in Figure 4.21. The visual quality is much higher than that of the faces generated by our full-body generator. Compared to the FFHQ face model, our custom face generator can be used to obtain nicer joint optimization results that better preserve the input face characteristics (ethnicity, skin tone, etc.) without distribution shift.

![Figure 4.21: Face Synthesis. We show unconditionally generated results at 256×256px resolution from a face generator trained on the same data as our full-body human generator.](image)
Chapter 5

Seamless 3D Editing of Humans in the Context of Videos

Figure 5.1: Our method VIVE3D is a novel technique that creates a powerful personalized 3D-aware generator using a low number of selected images of a target person. Given a new video of that person, we can faithfully modify facial attributes as well as the camera viewpoint of the head crop. We seamlessly composite the edited face with the source frame in a temporally and spatially consistent manner while retaining a plausible composition with the static components of the frame outside of the generator’s region. The dotted squares in the center frame denote the reference regions for the three different camera poses in the column below.

5.1 Introduction

Semantic image editing has been an active research topic for the past few years. Previous work [47] uses Generative Adversarial Networks (GANs) to produce high-fidelity results in the image space. The most popular backbone is StyleGAN [76, 75, 73, 74] as it generates high-resolution domain-specific images while providing a disentangled
latent space that can be utilized for editing operations. To edit real photographs, there are typically two steps: The first step maps the input image to the latent space of a pre-trained generator. This is usually accomplished either through encoder-based embedding or through optimization, such that generator can accurately reconstruct the image from the latent code [147]. The second step is semantic image manipulation, where one latent input representation is mapped to another to obtain a certain attribute edit, (e.g. changing age, facial expression, glasses, or hairstyle). While existing approaches produce impressive results on single images, extending them to videos is far from straightforward. Among the challenges that arise are: (1) people tend to move their heads freely in videos (instead of assuming frontal image inputs), (2) the inversion of multiple frames should be coordinated, (3) the inverted face and edits need to be temporally consistent and (4) the compositing of the edited face with the original frame must maintain boundary consistency.

A recent set of approaches has focused on 3D-aware GANs where a 2D face generator is combined with a neural renderer. Given a latent code, a 2D image and the underlying 3D geometry are generated, thus allowing for some camera movement while rendering the head of the person.

In this work, we tackle the problem of viewpoint-independent face editing in videos. The edited face is rendered from novel views in a temporally-consistent manner. Specifically, we use a 3D-aware GAN in the temporal domain and apply facial image editing techniques per frame that are temporally smooth regardless of the rendered view. Compared with other GAN-based video editing approaches [138, 7], our method is the first to perform viewpoint-independent video editing while showing the full upper body of the person in the video with high fidelity.

VIVE3D takes a video of a person captured from a monocular camera as input. The captured person can move freely across time, talk, and make facial expressions while their body can be visible. Unlike all prior work that learns a generator and performs
edits on the exact same video, we disentangle these steps. Hence the output of our approach can be a different video of the same person or the same video. In both cases, the face has undergone one or more attribute edits and is rendered from a novel view. To accomplish this challenging task, we introduce several novel components, each addressing one challenge of the problem at hand. Specifically, we first propose a simple yet effective technique to create a personalized generator by inverting multiple frames at the same time. The simultaneous inversion of $N$ frames exposes the generator to a variety of facial poses and expressions, which results in a larger capacity that we can then utilize. Our generator can generalize to new unseen videos of the same identity where the person might be wearing a different shirt, a result not demonstrated in the literature so far. In addition, we propose to optimize the camera pose of the 3D-aware GAN during inversion to obtain an accurate estimate of which angle the face was captured from. Finally, we introduce an optical flow-based compositing method to properly place the novel view of the edited face back into the original frame while ensuring that the end result is temporally and spatially consistent. Our experimental work provides a wide range of qualitative and quantitative results to demonstrate that VIVE3D accomplishes semantic video editing with changing camera poses faithfully.

In summary, our contributions are:

- A new 3D GAN inversion technique that jointly embeds multiple images while optimizing for their camera poses.
- A complete attribute editing framework and an optical flow-based compositing technique to replace the edited face in the original video.
- VIVE3D is the first 3D GAN-based video editing method and the first that can change the camera pose of the face.
5.2 Previous Approaches

5.2.1 GAN Inversion

GANs are a powerful tool for semantic editing. Most editing techniques are tailored to StyleGAN, the state-of-the-art of 2D GANs [75, 76, 73, 74]. Several editing techniques [43, 41, 70, 21, 108] build upon StyleGAN as it uses an intermediate disentangled latent space, usually referred to as \( w \)-space. Before editing, a latent space representation of the input image has to be recovered using a process typically referred to as Inversion or Projection [25, 2, 3, 142]. Refer to [147] for a survey of inversion techniques. In contrast to optimization-based inversion techniques, learning-based approaches attempt to obtain faster latent space correspondences by training encoders [115, 137, 6]. In order to retain the generalization ability of the \( w \)-space while providing a high-quality inversion, Pivotal Tuning [118] has successfully shown that trained generators can overfit to target images while still maintaining a navigable latent space. Recent works study 3D GAN inversion [92, 81], attempting to infer a 3D representation for a reference image.

5.2.2 GAN-based Latent Space Editing

Once an appropriate latent space representation of an input image has been recovered, semantic edits can be applied by navigating the latent space manifold surrounding the inverted latent code. Unsupervised techniques attempt to find interesting edits without labeled data [64, 53, 126, 140]. InterfaceGAN [124, 125] is a simple and robust supervised technique that is highly recommended for practical applications, and as such we also employ it in our work. While there is a plethora of other techniques [170, 150, 23, 4, 135, 134] the development of related latent space manipulations itself is not the focus of our work. Another line of work is text-based editing which gained immense popularity during the last year [44, 109].
5.2.3 3D-aware GANs

Recent GAN papers attempt to discover 3D information from large collections of 2D images using Neural Radiance Fields (NeRFs) as shape representations [20, 107, 30, 59, 162, 122, 17]. While most of these papers share similar architectural ideas, EG3D [20] has emerged as a popular basis for follow-up work (e.g. integration of a segmentation branch [132]). We chose to build upon EG3D, but our work is also applicable to other generators with a similar latent space. For more information on 3D GAN architectures, we refer the interested reader to a recent survey [146].

5.2.4 Video Synthesis and Editing

One branch of research attempts to leverage 2D GANs to generate video sequences [40, 136, 157, 129]. These ideas can be extended to create 3D videos [9], which also rely on 3D NeRFs.

5.2.5 GAN-based Video Editing

GAN-based video editing is the core topic of this project. Duong et al. [35] employ deep reinforcement learning for automatic face aging. LatentTransformer [150] encodes frames into the StyleGAN latent space using an encoder. They train a transformer to do attribute editing on single frames and blend the result with Poisson blending. The main competitor to our work is Stitch it in Time (StiiT) [138], which crops the faces from a video, edits them with 2D GAN techniques, and merges the edited result back to the video with some blending. However, StiiT does not learn a 3D model of the human head, overfits to a particular video, and is unable to provide edits to the viewpoint of the human head. Recently, VideoEditGAN (VEG) [148] attempted to improve the temporal consistency of StiiT by running a two-step optimization approach focused on localized temporal coherence. Alaluf et al. [7] use StyleGAN3 for video editing, to leverage its
Figure 5.2: **VIVE3D Pipeline.** To create an edited video, we first need to create a personalized generator by jointly inverting selected faces and fine-tuning a pre-trained generator. We then invert the cropped face regions from a source video (which could be the same or a different video) into our personalized generator and recover the latent codes and camera poses for each target frame. We are able to perform semantic editing on the inverted stack of latent codes using previously discovered latent space directions and we can freely change the camera path around the face region. In order to composite the face with the source frame in a consistent fashion, we use optical flow to correct the position of the inset within the frame, which allows us to composite the result in a seamless and temporally consistent fashion.

Inherent alignment capabilities and reduce texture sticking artifacts. Since this is an active area of research, all these techniques are concurrent work to our method, yet we do provide comparisons to showcase the benefits of our proposed approach.

5.3 **Methodology**

In this section, we introduce the key components of VIVE3D to perform frame-by-frame video editing while allowing for rendering the edited face from new views. We leverage a 3D-aware generator that infers 3D geometry and camera positions while being trained solely on 2D images. We build a personalized 3D-aware generator by performing joint...
inversion on multiple frames and then use it to perform attribute editing, apply camera viewpoint changes, and finally composite the edited face rendered from a new view back into the original frame. An overview of our proposed approach is depicted in Figure 5.2 while the personalized generator architecture is shown in Figure 5.3.

5.3.1 Personalized 3D-Aware Generator

**Face Selection and Cropping.** To create a personalized 3D-aware GAN model, we start by processing a short range from the input video where \( N \) frames are selected such that they cover a range of orientations and facial expressions of the target person, as shown in Figure 5.2 (top). We detect the facial keypoints within these frames using an off-the-shelf facial keypoint detector [16] and use them to determine the face bounding box within the frame. This is achieved by calculating a rigid transformation from the facial keypoints in the frame to the facial keypoints in a generated example image, thereby aligning the keypoints at the center of the crop in the same way as the generator's original training data. We pick a specific field of view for cropping the faces and optimizing the generator, but the field of view remains a flexible parameter that can be adapted during any later stage in the pipeline.

**Simultaneous Inversion.** We propose to perform multiple inversions simultaneously, as illustrated in Figure 5.3. EG3D has two major components in its generator. The first component uses a mapping network to map random vectors into a semantically meaningful space, called \( \mathbf{w} \)-space. Vectors in this space control a 3D neural field that defines a 3D representation that is rendered using volumetric rendering. The second component is a 2D upsampler that performs a \( 4 \times \) super-resolution on the original output. We invert all selected faces simultaneously into the \( \mathbf{w} \)-space following a strategy similar to [118] that we discuss in detail below.

In order to find a representation in \( \mathbf{w} \)-space, we define a “global” \( \mathbf{w}_{ID} \) aiming at
Figure 5.3: **Personalized Generator.** First, we run a joint inversion on $N$ selected target faces, where we optimize a shared target person latent $w_{ID}$ and an offset $o_n$ for each face. This ensures the inversions share information about the target. Simultaneously, we jointly optimize for the camera pose $c_n$. We then fine-tune the generator to ensure it captures the fine details of the target identity. Note that the “default” latent (left column) implicitly captures the identity of the target person without being explicitly optimized.
capturing the global identity features of the target person, and a “local” offset vector \( o_n \) for each input expression \( F_n \) that encodes the differences of each individual facial expression and position from the default \( w_{ID} \). The length of each \( o_n \) is regularized using an \( L_2 \) loss, aiming to keep the difference as small as possible, and capturing all similarities between the input images within the default person latent \( w_{ID} \). We use a combination of a perceptual loss \( L_{LPIPS} \) and a pixel loss \( L_{L1} \) for the inversion. Note that during this stage, we calculate these losses on the raw output \( G_{raw}(w_{ID} + o_n) \) of the EG3D neural renderer at 128×128 resolution because we observed that it yields sharper result quality rather than evaluating the loss at the output of the super-resolution network. We downsample our target images to the same resolution \( D_{128}(F_n) \) to compare. To ensure that we can faithfully capture the target person’s identity and expression, we use BiSeNet [154] to obtain a segmentation \( S_{exp}(F_n) \) of the facial regions encoding the expression (eyes, mouth, eyebrows, and nose) and add an additional feature loss on this area to encourage consistent facial expressions (e.g. closed eyes).

To obtain the inversion, in each optimization step, we sum up the losses for each face image \( F_n \), hence jointly optimizing all targets simultaneously, yielding a total loss \( L_{inv} \).

\[
L_{inv} = \sum_{n=0}^{N} \lambda_{LPIPS} L_{LPIPS}(G_{raw}(w_{ID} + o_n), D_{128}(F_n)) + \\
\lambda_{L1} L_{L1}(G_{raw}(w_{ID} + o_n), D_{128}(F_n)) + \\
\lambda_{seg} L_{LPIPS}(S_{exp}(G(w_{ID} + o_n)), S_{exp}(F_n)) + \\
\lambda_{reg} L_{L2}(o_n)
\]

Due to the 3D awareness of the EG3D generator, the quality of the inversion into the latent space is highly sensitive to the camera parameter settings. Hence, in addition to optimizing for \( w_{ID} \) and \( o_n \), we propose to also allow the inversion to optimize for the camera parameters \( c_n \) (yaw \( n \) and pitch \( n \)) for each input expression \( F_n \), which reliably estimates the camera position that the face is captured from.
A key advantage of this joint optimization is that the facial characteristics of the person preserve their high fidelity even when seen from novel views. When inverting a single image of a side-facing person into the EG3D latent space, exploring other viewpoints of the inverted latent can lead to significant distortions. Often, unseen features (e.g., hidden ears) can be blurry or distorted, and the identity no longer resembles the input from a different viewpoint. The joint inversion, however, ensures that the different views are embedded closely enough in latent space such that even unseen views yield consistently identity-preserving outputs.

**Generator Fine-tuning.** We propose a variant of Pivotal Tuning [118] to jointly fine-tune the weights of the generator $G_{EG3D}$ on all input faces $F_n$ while keeping the detected $w_{ID}$, $o_n$ and camera poses $c_n$ fixed. Here, we do not allow the weights of the upsampler of the generator to be updated as we want to preserve the generalization capabilities of the super-resolution network and prevent it from overfitting to our target images. During this fine-tuning stage, we employ perceptual and pixel losses described as follows:

$$L_{tune} = \sum_{n=0}^{N} \lambda_{LPIPS} L_{LPIPS}(G_{ID}(w_{ID} + o_n), F_n) + \lambda_{L1} L_{L1}(G_{ID}(w_{ID} + o_n), F_n)$$

Finally, we obtain a personalized EG3D generator $G_{ID}$, fine-tuned to a set of facial expressions of the target person. We verify that the fine-tuned generator indeed provides a good generalized latent space for the target person even though it was inverted and tuned based on a low number of frames by exploring the person created by the “global” latent code, which was not explicitly fine-tuned for, as well as through a latent space walk in the fine-tuned latent space.
5.3.2 Frame-by-frame Video Inversion

With the personalized 3D-aware generator in hand, we are now given a video of the same person as input which can be different from the one the generator was trained on. To process our new target video, we extract the facial keypoints from each frame to determine the location of the box to indicate the face crop within the frame. In order to stabilize the crop over time, which supports the temporal coherence of the inversion, we perform a Gaussian smoothing on the extracted facial keypoints along the temporal axis after extraction. However, it is important to not over-smooth, because fast motions in the video would yield distorted keypoint locations, deteriorating the inversion quality.

We then perform a frame-by-frame inversion of the extracted face regions $F_f$ into the space of the fine-tuned generator $G_{ID}$. Like before, we optimize for an offset $\mathbf{o}_f$ for each input frame $F_f$, as well as regularizing the offset length. After inverting the first frame, each consecutive frame $F_{f+1}$ is inverted starting from the previous inversion and only needs a low number (~50) of optimization steps.

\[
\mathcal{L}_{vid} = \lambda_{LPIPS} \mathcal{L}_{LPIPS}(G_{ID}(\mathbf{w}_{ID} + \mathbf{o}_f), F_f) + \\
\lambda_{L1} \mathcal{L}_{L1}(G_{ID}(\mathbf{w}_{ID} + \mathbf{o}_f), F_f) + \\
\lambda_{reg} \mathcal{L}_{L2}(\mathbf{o}_f)
\]  

(5.2)

This inversion yields a stack of offsets $\mathbf{o}_f$ from the identity latent $\mathbf{w}_{ID}$ as well as camera parameters $c_f$ ($\text{yaw}_f, \text{pitch}_f$) encoding the per-frame expression and camera position.

5.3.3 Attribute Editing and Novel View Synthesis

Since EG3D is built on top of StyleGAN2, we can leverage existing latent space editing techniques in order to discover semantic editing directions in the latent space of EG3D.
Figure 5.4: **InterfaceGAN edits.** We show InterfaceGAN editing directions discovered in the latent space of the pretrained EG3D generator by applying them on variants of our personalized generator models. The attribute edits are plausible and consistent in 3D.

As a proof of concept, we implemented InterfaceGAN [125] to find meaningful latent space direction vectors. We re-trained classifiers on the CelebA dataset for several facial attributes such as age, smile, gender, glasses, beard, and hair color and use these classifiers to classify the reference outputs of a set of randomized latent codes from our generator. Finally, we used an SVM to recover editing boundaries from these classified latent codes, which allows us to perform attribute editing in the EG3D latent space, as shown in Figure 5.4. For a given latent space direction $w_{dir}$, we apply an edit as a linear
combination of the person latent $w_{ID}$ with the direction, multiplied by an empirically chosen weight $\alpha_{dir}$, which can be positive or negative. For our video sequence, we use the edited person latent $w_{ID}'$ as the new identity to which we apply our video offsets $o_f$.

$$w_{ID}' = w_{ID} + \alpha_{dir} \times w_{dir}$$

In addition, we explore our temporal stack of latent encodings from novel views, diverging from the input views the inversion discovered. This allows us, to generate frontalized videos of the person by fixing the camera position or to define arbitrary camera trajectories around the subject.

5.3.4 Compositing with Source Video

After editing the inverted video, we want to recomposite the edited faces back with the source frames such that the edited video is temporally and spatially consistent. We accomplish this by running an optimization to ensure that the boundaries between the compositing of the edited face and the background of the source frame are smooth. Since cluttered video backgrounds are hard to reproduce consistently and without artifacts – especially for novel views – we define a compositing boundary region in a similar manner to [138]. To accomplish this, we need an accurate segmentation of the face and hair for both the original frame as well as the edited face. Hence, we use BiSeNet [154], a semantic segmentation technique, that accurately provides such face semantics. We use the semantic regions for both the original and edited face to form a union of their respective masks, obtaining a boundary region around the relevant face region, as illustrated in Figure 5.5. We run a small number of optimization steps, optimizing for the boundary region of the edited image to appear as close as possible to the boundary region of the input frame while retaining the appearance of the edit. Finally, we use an affine transformation to re-insert the cropped face region.
Figure 5.5: **Border Composition.** We calculate the composition border based on face segmentations of the target image (a) and the edited inset (c). We unite the masks and dilate the resulting joint mask to obtain a boundary around the face regions (d) that should be optimized, which allows us to create the final composition (e).

back into the original frame and we alpha blend along the optimized boundary region to seamlessly composite the edit with the source frame.

**Flow-based View Adjustment.** During the process of re-inserting the edited face $F_{edit}$ back into the source frame $S$, a major challenge arises when the camera pose has been modified and the face is rendered from a novel view. This is because the face is oriented within the bounding box based on the facial keypoints as described in Section 5.3.1 while upon a camera pose change, the face pivots around the keypoints. When, for instance, attempting to replace a face viewed at an angle with a frontalized face while retaining the original crop boundary of the inset, the keypoints are still roughly in the same location, yet the mass of the head, and crucially, the neck is shifted according to the face rotation, as seen in Figure 5.6, which results in the inset being disconnected from the rest of the body even when using a boundary stitching technique.

To alleviate this problem, we introduce a simple yet effective technique to reposition the reference face region within the source frame. We discover the optimal position of the updated inset with respect to the source frame by estimating the optical flow between the face segmentation in the source frame $S$ and the face segmentation in the inset region $F_{edit}$ after camera rotation. We convert the images to grayscale and use Farnebäck optical flow [38] to evaluate a dense flow field of the displacement...
Figure 5.6: **View adjustment.** After cropping (a) and inverting (b) a face, we perform face editing (c) and change the camera viewpoint to an unseen angle (d). When replacing the face in the original frame with this edit, it yields poor quality (bottom center) even for small angular changes, because the rotated face is in the wrong location with respect to the body. We address this by estimating the optical flow (e) between the face crop and the edit and use the flow direction to correct the location of the reference face based on the prospective inset (f). This allows us to composite the edited face into the frame in a natural-looking fashion (bottom right).

between the edited and target faces. The optical flow is defined as a 2D displacement vector field \( \mathbf{d} \) with the displacement vector at image position \((x, y)\) given by
\[
\mathbf{d}(x, y) = (u(x, y), v(x, y))
\]
where the correspondence between the two images \( F_f \) and \( F_{edit} \) is:
\[
F_{edit}(x + u(x, y), y + v(x, y)) = F_f(x, y).
\]

We then compute vector magnitudes \( ||\mathbf{d}|| = \sqrt{u^2 + v^2} \) and directions \( \phi = \text{atan2}(v, u) \), respectively. After eliminating all vectors with a magnitude smaller than a threshold \( \epsilon \), we create a histogram of all remaining directions. We define a dominant displacement vector \( \mathbf{d}_{dom} \) from the median direction of the histogram bin with the largest count and the maximum vector length within that histogram bin. This ensures that erroneous flow directions from features that are present within one of the two images but not the other are not contributing to the final output.

The displacement vector \( \mathbf{d}_{dom} \) is reprojected from inset space into frame space.
and is used to correct the location of the reference face crop. To ensure a smooth transition between adjacent frames, we perform temporal Gaussian smoothing of the recovered displacement vectors. We then apply our inset optimization using the updated reference areas and obtain a significantly more faithful result, allowing for large camera changes with natural-looking results.

5.4 Experiments and Evaluation

We conduct a wide range of quantitative and qualitative comparisons to demonstrate the key contributions of our work along with ablation studies against baselines and simplified variants where proposed modules are removed. We showcase that VIVE3D is on par with StiiT in terms of reconstruction quality while it also greatly outperforms prior work in editing quality and identity preservation. However, a key novelty afforded by VIVE3D is the ability to render the edited face from novel viewpoints within the existing frame, a task for which comparisons are hard due to the absence of ground truth. We showcase this capability with qualitative results and videos.

We establish comparisons with StiiT [138], the key competitor to our work in the field of GAN video editing, and with VEG [148] for which many components (e.g. their stitching technique) are identical to StiiT, so we only compare facial similarity metrics. First, we discuss some of the differences between our proposed method and the related work to establish the parameters of our comparison.

- StiiT and VEG, which is closely related to StiiT, both use a StyleGAN2 backbone which outputs high quality images at 1024×1024px resolution, whereas our EG3D backbone's output resolution is 512×512px (obtained by super-resolution given 128×128px inputs), providing 3D-awareness at the expense of slightly inferior image quality to classic StyleGAN2. In order to compare quantitatively, we down-sample all results generated by StiiT and VEG to 512×512px, unless we compare
at the full video resolution, in which case the output of the respective generator is already resampled to fit the resolution of the original face crop in the video frame.

- StiiT and VEG rely on prior work for a reliable encoding framework, e4e [115], to yield good and coherent inversions, whereas we implemented an optimization strategy to obtain per-frame inversions. Implementing an encoding strategy for EG3D was outside the scope of our project, but would be an interesting topic for future endeavors. We expect that using an encoder would lower the embedding quality, but improve computation speeds.

- StiiT and VEG fine-tune their generator on all video frames simultaneously, thus achieving very good coherence to the input. In contrast, we fine-tune on a select few target faces, yielding a more generalizable generator, which, in consequence, is not optimized to replicate the video frame-by-frame.

- Like our approach, StiiT relies on InterfaceGAN [125] for discovering and applying latent space editing directions for many of their results. VEG shows their results using edits based on StyleClip [109]. To compare with their method, we adapted their code to also allow edits with InterfaceGAN – analogous to StiiT– before applying their temporal consistency strategy. Note that the discovered directions in StyleGAN2 and EG3D latent space do not yield identical results for the same attribute type and the strength needed to apply the direction vectors is different. We empirically chose weights to approximate the same edit strength when comparing results.

### 5.4.1 Inversion Quality

We provide a quantitative comparison of our inversion quality by measuring the reconstruction quality with respect to the input video in Table 5.1. We evaluate PSNR and SSIM metrics for our method and for StiiT on a set of 16 videos from the VoxCeleb [103]
Table 5.1: **Video Reconstruction Quality Metrics.** We compare the quality of our inversion with the StyleGAN2 inversion used by StiiT using reconstruction metrics on a subset of the VoxCeleb dataset.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>StiiT</td>
<td>38.0134</td>
<td>0.9798</td>
</tr>
<tr>
<td>VIVE3D</td>
<td><strong>38.1259</strong></td>
<td>0.9704</td>
</tr>
</tbody>
</table>

dataset, inverting the face region and recompositing it with the source video without edits. Both methods perform well on the reconstruction of the input signal and the final reconstruction quality of our technique is on par with StiiT.

To qualitatively evaluate the performance of our personalized generator inversion, we show Figure 5.7, where we compare a result of our multi-target inversion strategy (row (a)) to another single-image 3D GAN inversion technique [81] (row (b)). Please zoom in to observe the degradation of head shape and loss of identity when the head rotation diverges from the source image.

**Figure 5.7:** **Comparison to another 3D inversion technique.** To demonstrate the effectiveness of our Personalized Generator inversion, we compare the quality of our inversion with a recent technique of 3D GAN inversion [81]. Note how the head shape and identity correspondence deteriorate when rotating the head away from the original pose.
Table 5.2: **Image Quality Metrics.** We evaluate the Fréchet Inception Distance (FID) of the initial inversion step and two variants of edits with respect to the source video for our method and StiiT.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>INVERSION AGE</th>
<th>ANGLE EDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>StiiT</td>
<td>6.8329</td>
<td>15.8021</td>
</tr>
<tr>
<td>VIVE3D</td>
<td><strong>4.9852</strong></td>
<td><strong>9.3410</strong></td>
</tr>
</tbody>
</table>

### 5.4.2 Image Quality

To evaluate the image quality produced by the respective techniques, we compute the Fréchet Inception Distance (FID), a commonly used quality metric for GANs. To obtain the FID for a video, we compare the set of all frames of the inverted video, as well as selected edits, with all frames of the source video. We average the score over 16 videos selected from the VoxCeleb [103] dataset. Our method is able to score very good FID scores overall (see Table 5.2), confirming the quality of our results.

In Table 5.3, we analyze the reconstruction quality and editing image quality in more depth for three individual videos shown throughout this chapter, showing that our reconstruction and editing capabilities are on par with our main competitor technique StiiT for video inversion and editing tasks.

Table 5.3: **Detailed Reconstruction Metrics.** We provide an in-depth analysis of the quality of our inversion with StiiT using reconstruction metrics on a subset of videos. We also evaluate the Fréchet Inception Distance (FID) of inversion and edits with respect to the source video.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reconstruction Quality</th>
<th>Editing Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR ↑</td>
<td>SSIM ↑</td>
</tr>
<tr>
<td>Marques</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StiiT</td>
<td><strong>36.477</strong></td>
<td>0.965</td>
</tr>
<tr>
<td>VIVE3D</td>
<td>33.791</td>
<td><strong>0.987</strong></td>
</tr>
<tr>
<td>Obama</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StiiT</td>
<td>34.969</td>
<td><strong>0.976</strong></td>
</tr>
<tr>
<td>VIVE3D</td>
<td><strong>36.282</strong></td>
<td>0.969</td>
</tr>
<tr>
<td>Dennis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StiiT</td>
<td>40.708</td>
<td><strong>0.993</strong></td>
</tr>
<tr>
<td>VIVE3D</td>
<td><strong>40.804</strong></td>
<td>0.990</td>
</tr>
</tbody>
</table>
Table 5.4: **Face Similarity Metrics.** We evaluate the identity preservation of inversion and edits based on the cosine similarity of features extracted from generated face crops using the ArcFace [29] metric with respect to the face crops of the source video. To evaluate coherence over time, we measure the dissimilarity between consecutive frames.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Similarity to Source ↑</th>
<th>Temporal Difference ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inversion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4e</td>
<td>0.6923</td>
<td>6.2851</td>
</tr>
<tr>
<td>StiiT</td>
<td>0.9261</td>
<td>1.2361</td>
</tr>
<tr>
<td>VIVE3D</td>
<td>0.9203</td>
<td>1.0444</td>
</tr>
<tr>
<td><strong>Age Editing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StiiT</td>
<td>0.7891</td>
<td>1.4126</td>
</tr>
<tr>
<td>VEG</td>
<td>0.6004</td>
<td>1.7159</td>
</tr>
<tr>
<td>VIVE3D</td>
<td>0.8381</td>
<td>1.2257</td>
</tr>
<tr>
<td><strong>Angle Editing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StiiT</td>
<td>0.6695</td>
<td>1.3102</td>
</tr>
<tr>
<td>VEG</td>
<td>0.4955</td>
<td>1.4502</td>
</tr>
<tr>
<td>VIVE3D</td>
<td>0.8694</td>
<td>1.2761</td>
</tr>
</tbody>
</table>

### 5.4.3 Face Fidelity

We calculate the fidelity of our inversion and edits based on a facial similarity metric, ArcFace [29], which extracts a 512-D feature vector capturing facial characteristics from a face region. We compute the metrics based on the respective face crops of the final inset region, scaled to 512×512px, for our method, e4e encoding, StiiT, and VEG in Table 5.4 and average across a set of 16 videos from the VoxCeleb [103] dataset.

The facial similarity is evaluated both with respect to the input video (frame-by-frame) as well as temporally by calculating the dissimilarity of adjacent frames in order to survey the temporal coherence of facial characteristics. In all cases, we use the cosine similarity between the extracted ArcFace deep features. We observe that the quality of the inversion is good for both StiiT and VIVE3D, both significantly improving upon e4e encoding. VEG uses the same PTI-based inversion as StiiT and is therefore not listed.

For latent space editing, our proposed VIVE3D leads the competition. VEG exhibits good temporal coherence, however, the edits contain artifacts, resulting in a deterioration in the similarity to the source video. Additionally, our technique reconstructs
the facial identity faithfully even for angle edits, a task in which StiiT and VEG fail to produce plausible results due to their methods’ inability to accommodate changes in the head rotation.

We also analyze the inversion and editing performance of VIVE3D and related methods for two distinct videos in more detail in Table 5.5. We use the ArcFace features to calculate the minimum, maximum, and average similarity to the source video as well as the temporal difference by evaluating the metric on adjacent video frames. The quantitative scores show that our method is superior for both attribute and angular edits, the latter being a task at which the previous 2D-GAN-based methods fail.

Table 5.5: Additional Face Similarity Metrics. For a more in-depth analysis of identity preservation, we evaluate the quality of inversion and edits for two individual videos. To evaluate coherence over time, we measure the dissimilarity between consecutive frames.
Table 5.6: **Timings and Memory Requirements.** We provide runtimes and Memory requirements for ours and competing methods.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Total</th>
<th>Precompute</th>
<th>Main</th>
<th>Postprocess</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>StiiT</td>
<td>58min 53sec</td>
<td>28min 21sec</td>
<td>30min 19sec</td>
<td>—</td>
<td>22GB</td>
</tr>
<tr>
<td>VEG</td>
<td>159min 54sec</td>
<td>107min 9sec</td>
<td>34min 41sec</td>
<td>18min 3sec</td>
<td>19GB</td>
</tr>
<tr>
<td>VIVE3D</td>
<td>35min 43sec</td>
<td>6min 58sec</td>
<td>14min 54sec</td>
<td>13min 51sec</td>
<td>21GB</td>
</tr>
</tbody>
</table>

### 5.4.4 Resource Usage

We compare runtimes and memory requirements for the default pipeline of related methods and our method, respectively, in Table 5.6. We split each method, wherever applicable, into precomputation, main method, and postprocessing steps and used the hyperparameter settings according to the authors’ suggestions. Note that the precomputation step in our method has to only be run once per identity and can then be applied to multiple videos. All experiments are performed on an example video consisting of 200 frames at a resolution of 1920×1080px, using a single NVIDIA A100 GPU with 40GB memory.

### 5.4.5 Ablation Study

We quantitatively verify the effectiveness of different architecture choices we made, as shown in Table 5.7. We compare several metrics with respect to the source video for our default implementation, and the ablation experiments, respectively. We run four experiments on a set of 5 videos: (1) VIVE3D without generator fine-tuning, (2) VIVE3D without flow-based adjustment, (3) VIVE3D without the joint \( w_{ID} \) latent and offset regularization, (4) using only a single input face for the generator inversion and fine-tuning, which in practice is almost identical to only frame-by-frame inversion in EG3D without any personalized generator. We demonstrate that all experiments deteriorate the facial fidelity as well as the reconstruction quality.
Table 5.7: **Ablation Study.** We demonstrate the effect of removing various components of our pipeline on several quality and reconstruction metrics. We measure the face similarity using the cosine similarity of ArcFace features of the generated face crop, and the reconstruction metrics at the target video resolution.

<table>
<thead>
<tr>
<th>Ablation Type</th>
<th>Face Similarity</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIVE3D, full</td>
<td><strong>0.9101</strong></td>
<td><strong>35.5367</strong></td>
<td><strong>0.9763</strong></td>
</tr>
<tr>
<td>no generator fine-tuning</td>
<td>0.7191</td>
<td>33.1202</td>
<td>0.9616</td>
</tr>
<tr>
<td>no flow correction</td>
<td>0.8198</td>
<td>24.8845</td>
<td>0.9350</td>
</tr>
<tr>
<td>no regularization</td>
<td>0.8007</td>
<td>25.6537</td>
<td>0.9162</td>
</tr>
<tr>
<td>single input for inversion</td>
<td>0.7382</td>
<td>25.1950</td>
<td>0.9137</td>
</tr>
</tbody>
</table>

![Figure 5.8: VIVE3D Ablation Study](image)

Figure 5.8: **VIVE3D Ablation Study.** Our proposed approach VIVE3D *(2nd column)* with all the proposed components demonstrates better identity preservation, fixes the spatial misalignment between the face and neck when rendered from a novel view and better captures the fine-level details of the face and results in high-fidelity faithful renders of the person from new views.

To evaluate the impact of each module we also show qualitative results in Figure 5.8. Given a video of a person talking *(1st column)*, we demonstrate our complete approach when rendering the output video from a new viewpoint *(2nd column)*. In the next 4 columns of results, we strip one component at a time and observe different performance quality drops. For example, if we do not fine-tune the generator it is clear that the identity of the individual is not properly preserved *(3rd column)*. If we remove the flow correction module which is a key contribution of our approach, we observe that the face and the neck are not well aligned which makes the results seem unnatural *(4th column)*. The impact of the flow correction module is demonstrated also in Figure 5.6. If we strip the regularization *(5th column)*, we remove the joint latent
$w_{ID}$ and treat each target face in the initial inversion separately. This means we don’t constrain the individual latents to stay close to the common latent. We can see that this leads to a deterioration in the inversion quality of the video frames and produced artifacts, as the inverted latents no longer share information, and there is no constraint on the projected location in latent space. Finally, if we were to only perform single-frame inversion rather than multi-frame, we also observe a significant drop in fidelity (6th column), which indicates that the proposed approach of performing multi-frame fidelity is beneficial as it better captures the identity and the fine details of the face.

5.4.6 Evaluation of Semantic Edits

First, we demonstrate that the quality of semantic edits using VIVE3D, adapting well-established latent space editing techniques for 3D GANs, is on par with the editing quality of StiiT, as shown in Figure 5.9. Note that while both approaches discover latent

![Figure 5.9: Comparisons with StiiT on attribute editing.](image)

We show an example of our method and StiiT editing the subject’s age in the video frame (center column). Both methods yield plausible but distinctly different results. Our results (columns (a), age=-1.4 and (c), age=+2.3) vs StiiT (columns (b), age=-8 and (d), age=+12).
space directions using InterfaceGAN [125], the latent spaces and discovered directions are dissimilar, yet both plausible.

We additionally provide a qualitative comparison to related methods of GAN-based video editing, Stitch it in Time (StiiT) [138] and VideoEditGAN (VEG) [148].

We demonstrate in Figure 5.10 that all methods provide plausible results for classic semantic editing problems such as aging the target person.

Figure 5.10: **Comparison to related work on age editing.** Due to the distinctly different InterfaceGAN editing directions, we handpicked the edit strength for the respective latent space edits to showcase a similar effect in aging the target person. Our technique yields results that are at least on par with the previous StyleGAN2-based editing techniques (bottom two rows).
Figure 5.11: **Changing camera poses.** Our method can freely alter the camera pose and composite the result back with source frame by fixing the divergence between the source and target pose using our optical flow correction. The generated results look natural despite the static body pose.

### 5.4.7 Evaluation on Novel View Synthesis

We show that VIVE3D can accommodate changes in the camera view with natural-looking results for a wide range of views regardless of the input face orientation, as shown in Figure 5.11. This is nontrivial as we need to ensure that the person's identity is consistent from multiple viewpoints while the body in the source frame also defines a rigid constraint to which the head alignment must be adjusted to.

While StiiT produces good results for attribute edits, it cannot composite images with angular changes, despite the fact that a limited head pose change can be achieved by applying latent space manipulations. In Figure 5.12, we show a comparison where
StiiT is unable to generate a reasonable composition, whereas we can achieve natural-looking results for a similar head rotation.

Comparing with both StiiT and VEG, both related methods fail to yield plausible results for angle editing. In order to compare this task, we utilize a latent space direction discovered in StyleGAN2 that allows for slight angle changes. The comparison for these strategies is illustrated in Figure 5.13. The artifacts present in these qualitative results mirror the deterioration in quantitative scores indicated in Table 5.4.

5.4.8 Compositing with Challenging Boundaries

When parts of the head or hair are visible both inside and outside the face bounding box then compositing the edited frame back into the original input is a challenging task. In most cases, we address these scenarios by adjusting our camera parameters (e.g., choosing a wider field of view for our generator) but some configurations can still be challenging, especially when different textures need to be matched. We show in
Figure 5.13: **Comparison to related work on angle editing.** In this instance, a slight change in angle is achieved for the related methods by applying a yaw-changing latent space direction. However, we can observe that both methods fail at producing a reasonable composition given these edits (bottom two rows). We implement a similar angle change and demonstrate that our method creates a natural looking composition.

Figure 5.14 that our technique attempts to produce plausible inset optimizations even for such instances.

### 5.4.9 Experimental Edits

We showcase some additional experimental edits to illustrate the generalization abilities of our approach for general-purpose video editing. For example, we are able to use two completely disjoint videos of different subjects and achieve reasonable results at compositing them. We show in Figure 5.15 two instances of such applications: On
Figure 5.14: Spatial Consistency. VIVE3D composites images with challenging boundaries such as long hair, yielding faithful hair color change results. For hard boundary cases, such as matching with a static piece of hair outside the boundary crop, it plausibly connects the contents of the two images.

On the left (Figure 5.15 (a)), we use one personalized Generator with its "default" person latent $w_{ID}$, and a stack of video offsets encoding a sequence of face motions. We then compose these with a different body by running our inset optimization, using the target video frames and head angles from that particular video. On the right (Figure 5.15 (b)), we use the encoded face motion from the target video, projecting the motion onto a different person's face, thereby essentially replacing the head in the target video. Note that in order to achieve plausible results for these instances, we need to copy head and neck due to the slight differences in lighting between the source and target faces. Further, the segmentation masks need to be considered carefully and be big enough, e.g. in the right result, the hair sticking out from the source person's head needs to be covered by inpainting with background color during the inset optimization in order to achieve a reasonable result. Otherwise, the optimization will add extra hair and change the hairstyle. We observe that VIVE3D generates realistic results of placing the first person's head on the second person's body that are temporally and spatially consistent and follow the target frame motion. This is possible because of our proposed design.
Figure 5.15: **VIVE3D generalization to new identities.** The benefits from decomposing the inversion of the input into an identity latent and a set of offsets unlock applications of face/motion re-targeting with minimal effort. This is possible due to our novel personalized generator that can be trained on a specific person’s identity and then applied to edit an unseen video. We show two examples: In example (a), we use a person (top row) to invert and fine-tune the generator, and we determine the video offsets based on this video sequence. The bottom row determines the target frames as well as the face location and angles. For example (b), we use a personalized generator (top left), but the target frames, angles, as well as motion, stem from a distinct video, driving the motion of the target person.

(personalized generator, separate identity, and offset latents) which makes VIVE3D unique compared to prior work. It is worth noting that when the source and target videos have different light conditions then the results might have different lighting between the body and the face. This is because we do not explicitly tackle this problem in our architecture and hence lighting is baked in the final generated/edited head before it is placed on top of the new body. We identify the problem of better lighting transfer as an avenue for future work.
5.4.10 Limitations

Changing the camera parameters for the head in videos with fast motion or discontinuities results in artifacts because the flow estimation becomes unstable. Furthermore, we inherit the shortcomings of EG3D (see Figure 5.16): stronger entanglement of attribute edits compared to StyleGAN2, extreme camera poses are not captured in the training set, and texture sticking. Finally, since the video outside the face region is immovable, the range of possible changes is constricted to plausible compositions.

5.5 Optimization Details and Parameter Settings

Our approach is implemented in Python 3.8 and uses PyTorch. We build our approach on the pretrained models and the publicly available codebase of EG3D [37, 20]. During our pipeline, we propose several optimization steps. Each of them is relying on ADAM
as an optimizer. We run all experiments on a single NVIDIA A100 GPU and provide timings and hyperparameters for our various pipeline steps.

(1) **Generator Inversion**: For the initial inversion of the generator, we use a standard learning rate scheduler. We also ramp down the regularization weight of $o_f$ from $\lambda_{wdist}$ to $\lambda_{wdist\_target}$.

*Optimization hyperparameters*

\[
\lambda_{L1} = 0.05, \lambda_{face} = 1.0, \lambda_{LPIPS} = 0.75,
\lambda_{wdist} = 0.05, \lambda_{wdist\_target} = 0.005,
\text{initial}\_\text{learning}\_\text{rate} = 1e^{-2}, \text{num}\_\text{steps} = 600
\]

*Duration* 3 min 33 sec (5 target images)

(2) **Generator Fine-Tuning**: We fine-tune the weights of the StyleGAN2 backbone of EG3D as well as the neural renderer, leaving learned weights of the Upsampling module untouched.

*Optimization hyperparameters*

\[
\lambda_{L1} = 1.0, \lambda_{LPIPS} = 0.3, \text{learning}\_\text{rate} = 1e^{-3},
\text{num}\_\text{steps} = 300
\]

*Duration* 3 min 27 sec (5 target images)

(3) **Video Inversion**: During this optimization, we run a frame-per-frame inversion, starting from the average offset of all offsets $o_f$ discovered in step 1. Using this strategy, we invert the first frame for $\text{init}\_\text{num}\_\text{steps}$. Each consecutive frame is started from the previous offset and optimized for $\text{num}\_\text{steps}$. We provide an early stopping criterion $\text{loss}\_\text{threshold}$ and finish the current frame optimization in case the total loss falls below this threshold.

*Optimization hyperparameters*

\[
\lambda_{L1} = 0.25, \lambda_{face} = 1.2, \lambda_{LPIPS} = 1.0,
\lambda_{wdist} = 0.01, \text{learning}\_\text{rate} = 1e^{-2},
\]
loss_threshold = 0.25, init_num_steps = 200, num_steps = 50

*Duration* 19 sec (first frame), ~4 sec/frame (consecutive frames)

(4) **Optical Flow Evaluation**: During this step, we evaluate the flow between the source face crop and the (angle-edited) target face to estimate the correction of the source crop needed to achieve a plausible inset composition.

To estimate the flow, we use Farnebäck optical flow with the following parameters:

- pyr_scale = 0.5, levels = 8, winsize = 25,
- iterations = 7, poly_n = 5, poly_sigma = 1.2

*Duration* ~0.4 sec/frame

(5) **Inset Optimization**: In this optimization step, we can specify sizes `border_size` for the width of the segmentation boundary region that is optimized and `edge_size` to provide an offset distance for the border from the image boundary. We can again specify an early stopping criterion `border_loss_threshold` to stop when the border loss falls under this threshold, which in practice provides significant speedup.

*Optimization hyperparameters*

- weight_foreground = 1.0, weight_border = 2.0, edge_size = 50, border_size = 50,
- num_steps = 150, learning_rate = 1e-2, border_loss_threshold = 0.05

*Duration* ~4 sec/frame (~16 sec/frame w/o early stopping)

### 5.6 Social Impact

The ability to provide editability/customization in videos of humans has been an active area of research over the past few years. On one hand, it can have key applications in
providing people the ability to express themselves in different ways (e.g., the ability to change hair color, add glasses, etc) during video calls, or more broadly in how they interact in the digital world. At the same time, such techniques introduce use cases for potentially malicious use that are worth discussing. For example, the ability to replace someone's face in a video from another person resembles deepfakes and could be used by bad actors. While the results such as what is shown in Figure 5.15 are still not at the level that would be perceived as indistinguishable from an original video this is an important conversation to be had regardless. We encourage the interested reader to refer to concurrent work [32, 48, 104, 5] for deep fake detection to discover edited videos.

5.7 Conclusion

We introduced VIVE3D, a novel framework that uses prior information encoded in 3D GANs for video editing. Our edits are identity-preserving and temporally consistent. While we enable standard semantic edits, such as age, or expressions, a distinguishing feature of our work is that we facilitate edits that alter the view of the head. This capability is not available in any 2D GAN-based prior work. The key building blocks of our work are a new embedding algorithm that jointly embeds multiple frames and optimizes for camera pose as well as flow-guided video compositing. In future work, we aim to extend our framework to include a 3D GAN for head details and another 3D GAN for the body. We also plan to investigate performance speedups by replacing various optimizations with encoders.
Chapter 6

Conclusions

We present an overview of the individual contributions we made in the projects presented in this thesis, and we put them in the perspective of the general state of the art of generative modeling, drawing conclusions from our insights. We discuss the impact of our work on generative seamless editing and reflect on future avenues of research and how they can impact the individual domains we've focused on in our research.

**Large-Scale Synthesis** We described our work on leveraging the convolutional nature of GANs for tiling in Chapter 3. Recognizing the usefulness of input-size agnostic convolutions is an especially useful property in the domain of texture images, because adjacent texture tiles within a specific domain such as satellite images or artworks can be described by a single trained generator network. Merging texture tiles in latent space can then achieve realistic and natural blending between adjacent texture images, allowing for the creation of very large plausible heterogeneous texture images. Our work recognized the usefulness of the intermediate latent space of the trained Generator early on - even before StyleGAN2 was released - as a powerful space for editing, merging and control.

**Context and Detail** We showed how we can utilize the capabilities of specialized generators (e.g. generators trained to synthesize faces) in Chapter 4 to improve images regions within a larger context, such as generated human bodies by optimizing the
generator networks such that they achieve very good boundary coherence. This is useful because specialized generators are able to capture finer details realistically for important image regions. The high-level context for these details can be provided by a larger-scale canvas generator. We demonstrated that we can make multiple independently trained GANs work together to collaboratively create an image at superior quality that one single GAN is capable of generating. This is a powerful observation that generalizes beyond the domain of full-body synthesis as it proves that well-trained GANs are capable of incorporating local as well as global changes for seamless compositing.

**Temporally Consistent Editing** We utilized 3D-aware GANs in the context videos of human heads in Chapter 5. Our goal was to create a personalized Generator model for a specific target person in the latent space of a 3D GAN, which can be used to invert a sequence of video frames, allowing for semantic editing. The key advantage of using 3D GANs over previous 2D GAN-based approaches is that our method can create natural-looking changes to the head’s angle. In order to create a plausible composition with the surrounding video frame, we proposed an optical flow-based adjustment strategy that replaces the rotated target head in the correct location with respect to the original face and body. Our method unlocks novel application scenarios for GAN-based video editing such as face frontalization or arbitrary head rotations.

### 6.1 Impact of Our Work on Seamless Compositing

While our work was targeting several distinct domains of image synthesis and editing, a common theme spans all methods proposed in this thesis. We explored novel methods of utilizing the latent space of generative models in order to achieve a seamless composition of generated imagery with surrounding image content.

The idea of seamless editing can be applied to interconnecting adjacent latent spaces of the same generator in order to generate plausible transitions, to making
completely distinct generators play together or to fit generated images with static images or video sequences. Unlocking the usefulness of generative models in these scenarios is a key ingredient in real-world applications, where the ability to connect and interact with surrounding content is paramount for the usefulness and the success of generative image synthesis.

Generative AI is currently moving at a break-neck speed, with new works being published on a daily basis. While novel generative models may supplant the current state of the art in image synthesis, our insights on seamless compositing are generalizable to novel techniques. In order to make these new developments useful in the context of real images, we believe that our insights on how to create spatially and temporally coherent inpainting and interconnecting scenarios can remain a helpful ingredient even for novel generative techniques.

6.2 Future Work

We are excited to pursue several promising directions of research in our future work. One important avenue of research is the translation of the insights we made in the domain of GANs to Diffusion-based Models, which are a very promising emerging trend in the area of generative image synthesis, showcasing extremely strong results.

**Generative Texture Synthesis at Large Scale** We explored the synthesis of texture tiles using GANs. Our method was particularly suited to generating non-homogeneous textures such as natural materials, but struggled to generate repetitive tilings. Recent generative models offer much better and more precise levels of control over the image output, and we would like to explore an integration of both homogeneous and non-homogeneous as well as hybrid textures, exhibiting regular patterns but small deviations (e.g. knitting or fabric textures with regular patterns, but local deviations).

We would also like to extend the idea of tileable generative textures to explore how
to generate additional texture properties such as albedo or normals. This could be achieved through supervised training with appropriate data, however large collections of high-quality data with the respective channels would need to be available. Another avenue of research would be to investigate whether these properties can be implicitly generated by pretrained state-of-the-art models such as Stable Diffusion.

**Towards Realistic Full-Body Humans**  We presented an approach to synthesize reasonable full-body human images, which allows us to generate images of humans at state-of-the-art resolution with a good variety of identities, clothing and poses. However, more control of the output domain is still desirable. For example, we would like to be able to achieve control of partial regions of the body, such as being able to change the shirt of the person without touching the rest of the image, or switching the pants to a skirt without changing the upper garment or identity of the human subject. To that end, we are working on a novel technique to embed partial regions of the generated image for a mixing-and-matching of clothing and a faithful transfer of real items of clothing to a latent space of our human body GAN.

We are also working on refining our human model to achieve even better visual quality, with a better disentanglement of desirable features such as pose, clothing, skin color, hair style or identity. More specifically, we want to improve texture details in garments to apply this approach in practical use cases such as virtual try-on applications. Yet another ongoing project is to collect a larger and more generalizable training database of human images by distilling publicly available large-scale image collections.

Furthermore, translating human bodies to the domain of 3D generation is a very desirable, yet very challenging, task. Initial results in this domain are promising but far from photorealistic, since we have shown that the complexity of the domain is challenging to capture in 2D already, so good transference to 3D is far from obvious.
Seamless Editing  We have shown that 3D GANs are a powerful tool for editing both images and video sequences. We would like to achieve better performance for our method, eventually enabling it to run in real-time to be able to process live video stream input such as the video feed during a video conference. This would permit several interesting applications: e.g. the removal of headsets for VR conferencing, the real-time editing of semantic attributes in the target person or the frontalization of the persons face regardless of where they are looking (a common problem in video conferencing, where the target person is looking at the screen rather than the camera).

The availability of a full-body generator would unlock many additional possibilities in this domain, as currently the static position of the body in the video frame limits the deviation the face can have while still allowing for a natural composition.

Finally, the seamless composition of a 3D body GAN with a face GAN in this context would unite several of our key ideas, where we could demonstrate that a high-quality face generator can improve the detail and realism on a more general body generator also in 3D, which might be applicable for static as well as temporally coherent application scenarios.
REFERENCES


Controllable person image synthesis with attribute-decomposed GAN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020).


