Diss. ETH No. 29256

Structured Generative Models for Controllable Scene and 3D Content Synthesis

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by

Dario Pavllo

MSc EPFL in Computer Science
born on 19.04.1994

accepted on the recommendation of

Prof. Dr. Thomas Hofmann (ETH Zurich)	examiner
Prof. Dr. Aurelien Lucchi (University of Basel)	co-examiner
P.D. Dr. Federico Tombari (Google & TU Munich)	co-examiner

2023

ABSTRACT

Deep learning has fundamentally transformed the field of image synthesis, facilitated by the emergence of generative models that demonstrate remarkable ability to generate photorealistic imagery and intricate graphics. These models have advanced a wide range of industries, including art, gaming, movies, augmented & virtual reality (AR/VR), and advertising. While realism is undoubtedly a major contributor to their success, the ability to control these models is equally important in ensuring their practical viability and making them more useful for downstream applications. For instance, it is natural to describe an image through natural language, sketches, or attributes controlling the style of specific objects. Therefore, it is convenient to devise generative frameworks that follow a workflow similar to that of an artist. Furthermore, for interactive applications, the generated content needs to be visualized from various viewpoints while making sure that the identity of the scene is preserved and is consistent across multiple views. Addressing this issue is interesting not only from an application-oriented standpoint, but also from an image understanding perspective. Our visual system perceives 2D projections of 3D scenes, but the convolutional architectures commonly used in generative models ignore the concept of image formation and attempt to learn this structure from the data. Generative models that explicitly reason about 3D representations can provide disentangled control over shape, pose, appearance, can better handle spatial phenomena such as occlusions, and can generalize with less data. These practical requirements motivate the need for generative models driven by *structured representations* that are efficient, easily interpretable, and more aligned with human perception.

In this dissertation, we initially focus on the research question of controlling generative adversarial networks (GANs) for complex scene synthesis. We observe that, while existing approaches exhibit some degree of control over simple domains such as faces or centered objects, they fall short when it comes to complex scenes consisting of multiple objects. We therefore propose a weakly-supervised approach where generated images are described by a *sparse* scene layout (i.e. a sketch), and in which the style of individual objects can be refined through textual descriptions or attributes. We then show that this paradigm can effectively be used to generate complex images without trading off realism for control.

Next, we address the aforementioned issue of view consistency. Following recent advances in differentiable rendering, we introduce a *convolutional mesh generation* paradigm that can be used to generate textured 3D meshes using GANs. This model can natively reason using 3D representations, and can therefore be used to generate 3D content for computer graphics applications. We also demonstrate that our 3D generator can be controlled using standard techniques that can also be applied to 2D GANs, and successfully condition our model on class labels, attributes, and textual descriptions. We then observe that methods for 3D content generation typically require ground-truth poses, restricting their applicability to simple datasets where these are available. We therefore propose a follow-up approach to relax this requirement, demonstrating our method on a larger set of classes from ImageNet.

Finally, we draw inspiration from the literature on Neural Radiance Fields (NeRF) and incorporate this recently-proposed representation into our work on 3D generative modelling. We show how these models can be used to solve a series of downstream tasks such as single-view 3D reconstruction. To this end, we propose an approach that bridges NeRFs and GANs to reconstruct the 3D shape, appearance, and pose of an object from a single 2D image. Our approach adopts a *bootstrapped* GAN inversion strategy where an encoder produces a first guess of the solution, which is then refined through optimization by inverting a pre-trained 3D generator.

SOMMARIO (ITALIANO)

Il *deep learning* ha trasformato radicalmente il campo della generazione delle immagini, grazie all'introduzione di nuovi modelli generativi che dimostrano una notevole abilità nel produrre immagini fotorealistiche e grafiche complesse. Questi modelli hanno avuto un impatto in svariati settori, tra cui arte, videogiochi, industria cinematografica, realtà aumentata e virtuale (AR/VR), e pubblicità. Sebbene il realismo abbia indubbiamente contribuito al loro successo, la capacità di controllare questi modelli è altrettanto importante per permetterne l'uso in applicazioni pratiche. Ad esempio, è naturale descrivere un'immagine attraverso testo, bozze, o attributi che ne controllano lo stile. Allo stesso modo, per alcune applicazioni interattive, il contenuto generato deve poter essere visualizzato da diverse prospettive assicurando che l'identità della scena sia preservata e sia coerente tra le varie viste. Affrontare questo problema è interessante non solo da un punto di vista applicativo, ma anche da un punto di vista prettamente scientifico legato alla comprensione delle immagini. Il nostro sistema visivo percepisce proiezioni 2D di scene tridimensionali, ma le architetture convoluzionali comunemente usate nei modelli generativi ignorano il concetto di formazione dell'immagine e tentano di apprendere questa struttura dai dati. I modelli generativi che ragionano esplicitamente sulle rappresentazioni 3D possono fornire un controllo disgiunto su forma, posa e aspetto, possono gestire meglio fenomeni spaziali come le occlusioni, e possono generalizzare con meno dati. Questi requisiti pratici motivano la necessità di modelli generativi guidati da *rappresentazioni strutturate* che siano efficienti, facilmente interpretabili, e più in linea con la percezione umana.

In questa tesi, ci concentriamo inizialmente sulla questione del controllo delle *generative adversarial networks* (GAN) per la sintesi di scene complesse. Osserviamo che, sebbene gli approcci esistenti permettano un certo livello di controllo su casi di studio semplici come volti od oggetti centrati, essi presentano delle evidenti lacune quando si cerca di trattare scene complesse composte da più oggetti. Proponiamo quindi un approccio debolmente supervisionato (*weakly-supervised*) in cui le immagini generate sono descritte da un *layout sparso* (simile ad una bozza), in cui lo stile dei singoli oggetti può essere rifinito attraverso descrizioni testuali o attributi. Successivamente, dimostriamo che questo paradigma può essere efficacemente utilizzato per generare scene complesse senza necessariamente dover trovare un compromesso tra realismo e controllo.

Successivamente, affrontiamo il sopracitato problema della coerenza tra più viste. Costruendo sui recenti progressi nel campo del *rendering differenziabile*, introduciamo un paradigma di *generazione convoluzionale di mesh* che può essere utilizzato per generare mesh 3D e texture tramite GAN. Questo modello può ragionare nativamente attraverso rappresentazioni 3D e può quindi essere utilizzato per generare contenuti 3D per applicazioni di grafica computerizzata. Dimostriamo anche che il nostro generatore 3D può essere controllato utilizzando le stesse tecniche che vengono comunemente utilizzate su GAN 2D, e presentiamo esempi di condizionamento su classi, attributi, e descrizioni testuali. Nonostante i risultati promettenti ottenuti da questa tecnica, osserviamo che i metodi per la generazione di contenuti 3D richiedono tipicamente pose annotate, limitando la loro applicabilità a dataset semplici dove queste sono disponibili. Pertanto, proponiamo un'evoluzione dell'approccio sopra proposto in cui eliminiamo questa assunzione, e dimostriamo il nuovo metodo su un insieme di classi più ampio da ImageNet.

Infine, prendiamo ispirazione dalla letteratura sui Neural Radiance Fields (NeRF), e incorporiamo questa recente rappresentazione nel nostro lavoro di modellazione generativa 3D. Dimostriamo come questi modelli possano essere utilizzati per risolvere una serie di compiti tra cui la ricostruzione 3D da una singola immagine. A tal fine, proponiamo un approccio che connette NeRF e GAN, permettendo di ricostruire la geometria 3D, l'aspetto, e la posa di un oggetto da una singola immagine 2D. Il nostro approccio adotta una strategia di inversione GAN ibrida in cui un *encoder* produce una prima ipotesi della soluzione, che viene poi raffinata tramite ottimizzazione invertendo un generatore 3D pre-addestrato.

ACKNOWLEDGMENTS

First of all, I would like to extend my gratitude to my advisors Thomas Hofmann and Aurelien Lucchi for providing me with guidance throughout my doctoral studies, and for granting me access to a supportive environment where I was given the freedom to pursue my own research interests.

I am also deeply grateful to all of my lab mates with whom I shared this journey: Antonio, Leonard, Gregor, Luca, Lorenzo, Sotiris, Giulia, Sidak, as well as past members Jonas, Yannic, Paulina, Florian, Kevin, Octavian, Hadi, Gb, Gary, and Janis. The completion of this thesis would not have been possible without standing on the shoulders of giants.

I would also like to thank David, Marie-Julie, Federico (whom I thank twice for being on the thesis committee), and the other colleagues from Google for supporting me throughout my internship, and for our collaboration on interesting problems which are also covered in this dissertation.

Lastly, I acknowledge financial support from the Swiss National Science Foundation (SNF), grant #176004, and would like to thank our partners in KU Leuven with whom we shared a longstanding collaboration: Marie-Francine Moens, Graham Spinks, and Wolf Nuyts.

PUBLICATIONS

This dissertation is based on the following publications, which are at the core of my PhD research:

- Dario Pavllo, Aurelien Lucchi, and Thomas Hofmann. "Controlling Style and Semantics in Weakly-Supervised Image Generation." In *European Conference on Computer Vision (ECCV)*, 2020 [PLH20]. **Selected for spotlight presentation (top 5% of submissions).**
- Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. "Convolutional Generation of Textured 3D Meshes." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020 [Pav+20b]. **Selected for oral presentation (top 1% of submissions).**
- Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi. "Learning Generative Models of Textured 3D Meshes from Real-World Images." In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 [Pav+21].
- Dario Pavllo, David Joseph Tan, Marie-Julie Rakotosaona, and Federico Tombari. "Shape, Pose, and Appearance from a Single Image via Bootstrapped Radiance Field Inversion." In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023 [Pav+23].

Furthermore, the following publications originated during my time as a PhD student, but are not covered in this dissertation:

- Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. "3D human pose estimation in video with temporal convolutions and semi-supervised training." In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 [Pav+19].

- Dario Pavllo, Christoph Feichtenhofer, Michael Auli, David Grangier. “Modeling Human Motion with Quaternion-based Neural Networks.” In *International Journal of Computer Vision (IJCV)*, 2020 [Pav+20a].

- Ankit Dhall, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause. “Hierarchical image classification using entailment cone embeddings.” In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020 [Dha+20].

- Antonio Orvieto, Jonas Kohler, Dario Pavllo, Thomas Hofmann, and Aurelien Lucchi. “Vanishing Curvature in Randomly Initialized Deep ReLU Networks.” In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR, 2022 [Orv+22].

Code and pretrained models for the works covered in this dissertation are publicly available at `https://github.com/dariopavllo`.

CONTENTS

1 INTRODUCTION

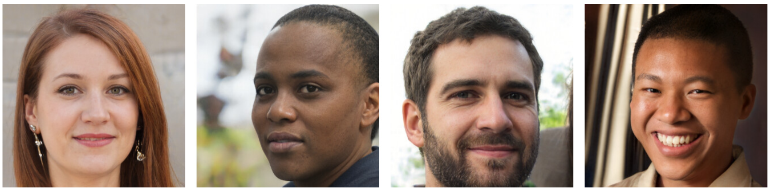

Figure 1.1: These people do not exist.

1.1 GENERATIVE MODELS

Generative models have gained considerable attention in recent years due to their ability to quickly generate new content that – in many cases – is almost indistinguishable from that produced by humans or nature (Figure 1.1). This has opened up new avenues for a wide range of applications, including speech synthesis [Che+21a; Oor+16], image synthesis [Kar+20b; Ram+22; Rom+22; Sah+22], generation of 3D objects [Cha+22; Pav+20b; Poo+22], realistic simulations of physical systems [Rav+21], and the creation of intelligent dialogue systems [Bro+20; Ouy+22].

When provided with sufficiently large datasets, these models are capable of capturing the complex patterns existing in the data, and can subsequently be used to generate new data that resembles the training examples. One of the key advantages of recently-proposed generation techniques is that they require minimal human annotations, allowing them to be trained on massive datasets scraped off the Internet with little effort [Gao+20; Sch+21].

In the realm of generative modeling, various subfields have emerged, each focused on distinct domains such as text, audio, and vision. In this dissertation, we concentrate specifically on the field of vision, which has seen a surge in research interest in recent times, thanks to models capable of generating almost-photorealistic images. To name a few, neural architectures such as StyleGAN [KLA19; Kar+20b], Stable Diffusion [Rom+22], DALL-E [Ram+22; Ram+21], and Imagen [Sah+22] can be used to create photorealistic renderings of people and products, graphics for articles, and artistic content. The applications of generative models in computer vision, however, extend beyond image synthesis. Other examples include image super-resolution (creating high-resolution images from low-resolution ones), inpainting (repairing or substituting missing parts in an image), and style transfer.

From a perspective of image understanding, generative models are very powerful tools, since they must capture the constraints occurring in natural images to produce realistic results. These constraints may be spatial (coherent pose and placement of objects), semantic (a boat does not sail on grass), and stylistic (a naturally-occurring dog cannot be purple). For practical content creation applications, however, it might be necessary to diverge from these constraints. For example, artists might want to specify what they want to see in the image, where the objects should be located spatially, and what "style" (e.g. color, texture) they should have. Furthermore, for particular illustrations, they might want to specify configurations that do not occur in the distribution of natural images (e.g. an astronaut riding a horse). While recent generation techniques have advanced tremendously with regard to the realism of the synthesized images, the level of structural, semantic, and stylistic control provided by these models has lagged behind. Many approaches that provide some degree of control focus on single-domain datasets that depict centered objects, and they fail to accurately model complex scenes consisting of multiple objects.

Motivated by these observations, the first research question which we address in this dissertation is how we can make generative models more controllable, with a focus on complex scenes. A viable compromise found in the literature is to condition the model on a semantic segmentation map [Iso+17; Par+19b; Wan+18], which is unfortunately impractical to manipulate or create from scratch, and still provides no interface for style control. It is clear that there is a trade-off between *weak conditioning* (e.g. describing an image through a label or a single textual prompt), which makes the task easier for a human but more difficult for a machine – potentially resulting in incoherent scenes or undesired results – and *rich conditioning* (e.g. full segmentation maps), which constrains the output and facilitates the learning task, but requires a prohibitive effort from a user who wants to generate or manipulate an image. One major goal of this dissertation is to explore ways to *condition* (i.e. control) these generative models through *structured representations* such as textual descriptions and sketches that describe the placement of the objects in the scene, and devise a viable framework that retains both a high generative quality and a proper level of control (Chapter 2).

The encouraging results achieved by these models lead us to another research question: in addition to realism and controllability, what are other important aspects that make a generative model useful? While the aforementioned techniques are excellent at generating static content, some applications such as animation, video games, and advertising require representations which are more structured. For example, in computer graphics it is often desired to work with multiple views of the same object while preserving its identity across different views, that is, enforce a disentanglement between pose and appearance. This issue can be addressed at a fundamental level by devising generative models that can natively synthesize 3D representations. We dedicate the second part of this dissertation to this topic, and propose an approach to generate textured 3D meshes using supervision from 2D datasets (Chapter 3, Chapter 4). We show that, as in the 2D case, these models can successfully be trained using generative adversarial networks (GANs) and can be controlled via textual prompts or other conditioning techniques commonly adopted in the 2D GAN literature. Finally, we draw inspiration from the recent literature on state-of-the-art representations based on Neural Radiance Fields (NeRF), and demonstrate that 3D generators can benefit additional downstream tasks in 3D computer vision, such as single-view 3D reconstruction (Chapter 5).

1.2 BACKGROUND

Throughout this dissertation, we will make use of some concepts that are well-established in the machine learning literature, and which constitute the building blocks of many of our contributions. In this section, we provide a quick overview of these concepts, starting with an introduction on frameworks for generative models and common patterns for conditioning these models. We then conclude with a section on 3D representations and their use in deep learning applications.

1.2.1 *A primer on generative models*

Intuitively speaking, the goal of a generative model is to produce new examples that mimic a training dataset. It is assumed that the real data is sampled from a "true data distribution" $p_{\text{data}}(\mathbf{x})$, which is potentially unknown, but for which we have samples (i.e. data points). The generative model is then trained to approximate this distribution. Generative models can be categorized into fundamentally different frameworks:

LIKELIHOOD-BASED MODELS. These are also referred to as *explicit models* because they model the probability density function explicitly, using maximum likelihood estimation or approximations. Popular methods include variational autoencoders (VAEs) [KW14] and autoregressive methods. The latter are particularly successful in sequential domains that are discrete or can easily be quantized, such as text and audio.

IMPLICIT MODELS. The probability density function is not modelled explicitly. Instead, these models only offer an interface to sample from the learned distribution. Generative adversarial networks (GANs), which are at the core of our work, are examples of implicit models. For this reason, we describe them comprehensively in the next section (Section 1.2.2).

SCORE-BASED MODELS. Instead of modelling the probability density function, these methods model its gradient (i.e. *score function*). In such a setting, the requirement for a (potentially intractable) normalizing constant is dropped, expanding the set of models that can be used in a tractable way. Sampling is then performed iteratively through Langevin dynamics, at the expense of inference speed. Such models can be regarded as a middle ground between implicit and explicit models. Examples of score-based models include [SE19; SE20] as well as recently-proposed *diffusion models* [HJA20; SME20], which have outperformed GANs in terms of image generation quality [DN21]. However, these models are still considered slower and somewhat less flexible than GANs, but this may change in the future as this research field is relatively recent.

There is no individual framework that outperforms alternative methods in every situation. The choice of the framework often depends on the particular domain (text, audio, time series, or images), on the kind of data (whether the data is discrete or continuous, whether it is partially or fully observable), and on domain requirements (whether quality is preferred over accurate statistical modelling). Maximum-likelihood methods are preferred in situations where statistical interpretation is important (e.g. financial time series), but are less favored for image generation tasks, where they tend to generate blurry results and do not scale well to high resolutions. GANs generate images that are more visually pleasing (with the side effect of potentially dropping some modes of the data distribution), and are therefore widely adopted in the computer vision literature (including our work). Diffusion models represent a very recent development and, given their extensive growth, it is likely that they will represent the leading paradigm in the near future. However, for reasons we explain in the next section, GANs remain the preferred approach in areas such as 3D vision.

1.2.2 *Generative adversarial networks*

A generative adversarial network (GAN) consists of two networks: a generator G and a discriminator D. The goal of the generator is to map a random vector $\mathbf{z}$ (distributed according to some simple prior distribution $p_{\mathbf{z}}$, and often referred to as *latent code*) to a generated ("fake") sample $G(\mathbf{z})$. A standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is usually chosen as the prior distribution. The discriminator is a binary classifier that takes a sample as input and predicts whether it comes from the dataset ("real") or is fake. The two models are optimized in an alternating fashion, where the discriminator is trained to distinguish between real and fake samples, and the generator is trained to fool the discriminator by making its samples more real according to the information (gradient) provided by the discriminator. The result is that the generator learns to map the prior distribution $p_{\mathbf{z}}$ to the data distribution p_{data}. However, GANs are categorized as *implicit models* because they do not model the data distribution explicitly (the output of G is an individual sample, not a probability distribution), but merely provide an interface for sampling from it. This loss in expressivity is rewarded by a gain in generative quality (but also a series of potential issues such as mode dropping). More formally, in their original formulation, GANs are optimized for the following objective:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \quad (1.1)$$

The authors [Goo+14] show that, under some optimality assumptions, the above objective minimizes the Jensen-Shannon divergence between the inferred distribution and the data distribution p_{data}. However, the above objective is notoriously unstable and other variants are used in practice, such as the non-saturating objective (also proposed in [Goo+14]), least-squares objective (LS-GAN) [Mao+17], or hinge loss [LY17]. The known instability of the GAN objective has also led to the development of several other tricks to improve its training dynamics, such as gradient penalties [ACB17; MGN18], spectral normalization [Miy+18], progressive growing [Kar+18], and projection discrimination [MK18].

GANs have been very successful in the realm of image generation, because they mainly target continuous data, allow for efficient inference, are flexible (allowing for a wide range of conditioning schemes and data assumptions), scale easily to high resolutions, and generate high-quality samples. They are however less effective in other domains such as text (which is discrete) or audio (where its harmonic structure is notoriously difficult to model accurately). In these sequential settings, autoregressive models are typically preferred.

As anticipated earlier, diffusion models are also increasingly gaining importance, outperforming GANs on image generation [DN21] and producing results on par with the best autoregressive techniques for speech generation [Che+21a]. However, it is still unclear how to apply these models to common synthesis tasks in 3D vision (generation and reconstruction), as they do not exhibit the same flexibility as GANs. For example, in GANs it is possible to couple different architectures for the generator and discriminator, e.g. a 3D generator combined with a 2D discriminator that takes 2D renderings as input (enabling 3D-aware approaches that can be supervised on 2D images); a generator of audio signals in the time domain paired with a discriminator in the (time-)frequency domain; and it is even possible to devise image-to-image translation frameworks between unpaired sets of images [Zhu+17a]. Another limitation of diffusion models is that they are slow in terms of inference speed, which makes them impractical for many real-time computer vision applications such as augmented & virtual reality (AR/VR), and robotics.

1.2.3 *Conditioning and disentanglement*

In the previous discussion, we have covered what is commonly referred to as *unconditional generation*, i.e. the generation of samples that resemble the training dataset without any kind of control or constraints. In practice, however, it is more useful to guide the generation process (*conditioning*) according to some desired specifications. For instance, in image synthesis it is common to condition the model on a category (*class label*) [BDS19; MK18; OOS17], semantic map [Iso+17; Par+19b; Wan+18], or textual description of a scene [Ram+21; Rom+22; Sah+22; Zha+17]. In other domains such as audio, the model is usually conditioned on a particular instrument & pitch (in the case of music) [Eng+19], or text & speaker identity (for text-to-speech systems) [Oor+16].

A simple strategy for conditioning generative models is *concatenation*. The conditioning information is transformed into a fixed-length vector **c** (either through an embedding layer or through a neural network, depending on the application) and connected to the model. In the generator of a GAN, this is often achieved by concatenating **c** to the latent code **z**, whereas in the discriminator possible strategies include auxiliary classifiers [OOS17] and projection discrimination [MK18], the latter being more effective in practice. In generator architectures that adopt normalization layers, it has also proven very successful to inject conditional information directly into these modules, leading to techniques such as adaptive instance normalization (AdaIN) [HB17] and conditional batch normalization [De +17], which are now considered standard tricks of the trade.

In text-conditioned models, however, compressing a full sentence into a fixed-length vector may represent an excessive bottleneck. For this reason, recent conditional diffusion models such as Imagen [Sah+22] and Stable Diffusion [Rom+22] adopt a cross-attention mechanism, where different image regions attend to individual token representations. Furthermore, some of the works presented in this dissertation introduce other types of cross-attention, e.g. in Chapter 2 we propose a *sentence-semantic* attention mechanism in the context of image generation from scene layouts [PLH20], whereas in Chapter 3 we propose a *sentence-UV map* attention mechanism for generating textured 3D meshes from text [Pav+20b].

Another important aspect of generative models is *disentanglement*. An assumption is that the data can be explained by an underlying set of *factors of variation*. In the specific example of faces, these include gender, hair color, hair style, lighting, and posture. These factors are often entangled, i.e. there are some correlations that occur in the data, such as gender and hair style. Disentanglement refers to the ability of the model to separate these factors of variation (as opposed to representing them as a mixture), allowing the user to control them separately. This property is even more relevant for 3D vision applications, where the appearance of an object (that is, its identity) needs to be disentangled from its pose (camera viewpoint). This motivates the need for generative models that can natively generate 3D representations, which represents one of the major contributions of this dissertation.

1.2.4 *3D representations*

While 3D representations are well understood in the computer graphics literature, in this section we mainly focus on their application to deep learning approaches, with a discussion of the advantages and drawbacks of each. Common representations include voxel grids (*voxels*), point clouds, triangle meshes, and implicit representations. Not to be confused with the probabilistic *implicit models* described in the previous sections, implicit representations represent a class of parameterizations that describe a 3D object/scene as a continuous function that can be queried at specific coordinates, without explicitly defining its surface. Examples include signed distance functions (SDFs) and neural radiance fields (NeRF).

VOXEL GRIDS. Voxel grids are a 3D generalization of raster images. The 3D space is divided into a regular grid of voxels (i.e. 3D pixels), each containing information about occupancy and, optionally, color. This representation is popular in deep learning approaches as it can be used out-of-the-box with standard convolutional architectures (3D convolutional neural networks), but it does not scale well to high resolutions since its memory requirement is cubic with respect to the subdivision steps.

POINT CLOUDS. These representations describe the 3D scene as a set of unordered points. They have the advantage of being memory-efficient, as they take into account the sparse structure of the scene. However, their unordered nature and lack of connectivity makes them more difficult to process directly by neural networks. Their sparsity can also lead to visible gaps when they are visualized.

TRIANGLE MESHES. The object is represented as a set of interconnected triangles, described by a set of 3D *vertex positions* and triplets of *indices* that specify the connectivity between vertices. Colors can be specified through *vertex colors*, or through *UV mapping*, where each vertex is mapped to 2D coordinates on a 2D image (*texture*). Triangle meshes are very efficient in terms of both memory and computation, and UV mapping allows simple meshes to be coupled with detailed textures. For this reason, they are arguably the most successful representation in computer graphics, as they are ubiquitous in movies and video games. While it was initially not clear how to effectively utilize this representation in deep learning approaches (since the rasterization step in mesh renderers is non-differentiable), the field of differentiable rendering [Che+19; KUH18; Liu+19b; LB14] has enabled a range of deep learning approaches that can deal with triangle meshes. Our works in Chapter 3 [Pav+20b] and Chapter 4 [Pav+21] also adopt triangle meshes owing to their flexibility and their ability to efficiently represent fine details. A major limitation of triangle meshes in most deep learning approaches, however, is that the topology of the object (number of vertices and their connectivity) needs to be specified in advance and cannot be altered. Most approaches adopt a sphere template which is then deformed, but this makes it challenging to model arbitrary topologies, e.g. a donut.

SIGNED DISTANCE FUNCTIONS. In SDFs, the shape of the object is described by a continuous function that assigns a signed distance value $d(\mathbf{x})$ (a scalar) to each point $\mathbf{x}$ in space. The contour of the object is implicitly determined by its intersection with the *0-level set*, i.e. $d(\mathbf{x}) = 0$, whereas the sign of the distance describes whether the point lies inside or outside the shape. For example, a sphere of radius r can be implicitly represented as:

$$d(\mathbf{x}) = \|\mathbf{x}\| - r = 0. \tag{1.2}$$

Rendering can be performed through iterative *ray marching* [Har96], which can be slow compared to rasterization. However, being implicit, this representation can easily represent fine structural details as it does not adhere to a fixed-resolution grid, can model arbitrary topologies, and provides analytical information about surface normals.

DEEP IMPLICIT REPRESENTATIONS. When an implicit representation (e.g. the distance function in SDFs) is parameterized using a deep neural network, we obtain a deep implicit representation. Common architectures for this task include multi-layer perceptrons (MLP) and residual networks [He+16]. Furthermore, sinusoidal activations [Sit+20] and positional encoding [Tan+20] have been shown to improve the ability of these models to represent high-frequency details. A major strength of deep implicit representations over alternative approaches is that they can be regarded as a form of variable-rate compression. Most of the capacity of the network is concentrated in the fine (high-frequency) details, whereas sparse areas take up a lesser portion of the network's capacity. Implicit neural networks can therefore easily compress a signal (not limited to 3D scenes) using few parameters. The main disadvantage is the high computational cost to query the model, although there has been recent research aimed at overcoming this limitation [Che+22].

1.2.5 *Neural radiance fields*

Neural radiance fields (NeRF) were originally proposed in [Mil+20] to tackle the novel view synthesis task, but were later shown to represent a more general tool in the 3D vision community. The idea behind NeRFs is the combination of deep implicit representations with volumetric rendering (*neural rendering*), resulting in a scene representation that can be optimized in an end-to-end fashion using 2D images for supervision. In its original formulation, a NeRF is trained using a set of 2D views along with the corresponding ground-truth camera poses, and the result is a 3D reconstruction of the scene comprising both shape and color, which can be used to render novel views.

A NeRF is typically parameterized using a multi-layer perceptron (MLP) that takes a 3D position $\mathbf{x}$ and optionally a view direction $\hat{\mathbf{d}}$ (a unit vector) as input, the latter of which is used to model view-dependent effects such as reflections. The output is a density $\sigma(\mathbf{x})$ (which does not depend on the view direction) and an RGB color $\mathbf{c} = f(\mathbf{x}, \hat{\mathbf{d}})$. Rendering is performed by casting rays through the camera frustum, sampling the MLP across multiple depth values, and integrating along each ray as follows:

$$\mathbf{C}(t) = \int_{t_{\text{near}}}^{t_{\text{far}}} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \hat{\mathbf{d}})dt\,, \tag{1.3}$$

where $\mathbf{r}(t) = \mathbf{o} + t\hat{\mathbf{d}}$ is the camera ray with origin $\mathbf{o}$ and direction $\hat{\mathbf{d}}$, bounded by the near and far planes of depth t_{near} and t_{far} respectively; $T(t) = \exp\left(-\int_{t_{\text{near}}}^{t} \sigma(\mathbf{r}(s))ds\right)$ is the integrated density, which models occlusions; finally, $\mathbf{C}(t)$ is the rendered RGB color. In practice, the integrals are approximated using a quadrature rule as described in [Max95]. It is also common to apply a two-step sampling procedure where a first pass (*coarse* step) samples the depths t uniformly via stratified sampling, and a second pass (*fine* step) uses importance sampling to sample points closer to surfaces more frequently.

Owing to their success in the novel-view synthesis task, there has been follow-up work bridging NeRFs and alternative representations such as SDFs [OPG21; Yar+21] and voxels [Fri+22; SSC22]. There has also been work on applying NeRFs to pose estimation [Yen+21], unconditional generation [Cha+21; NG21; Sch+20], and reconstruction from few views [Yu+21]. Finally, in an attempt to improve efficiency, some approaches have proposed lightweight alternatives to MLPs [Cha+22; Che+22]. Although these parameterizations are still somewhat slower than triangle meshes, NeRFs can model arbitrary topologies, thus overcoming a significant limitation of the former. For this reason, NeRFs are central to many recent approaches in the computer vision literature, including the last approach presented in this dissertation (Chapter 5) [Pav+23], which tackles 3D reconstruction from a single view. Finally, we mention that triangle meshes and NeRFs are not mutually exclusive. A recent research direction aims to distill NeRFs into meshes after training [Gao+22], harnessing the advantages of NeRFs and the fast inference speed of triangle meshes.

1.3 THESIS ORGANIZATION

Excluding this introductory chapter, the thesis is structured into four main chapters, each of which revolves around a publication, and a final chapter of closing remarks where we summarize our contributions and discuss future perspectives.

CHAPTER 2. We open this dissertation with a chapter on improving the controllability of generative models. Although technical advances in image synthesis have made it possible to generate photorealistic images in the unconditional (i.e. unconstrainted) setting, an important question is how we can control the generation process so as to precisely specify what we would like to see in the final result. Following a GAN framework, we introduce a weakly-supervised approach for conditional image generation of complex scenes where the user has fine control over the objects appearing in the scene. Such control can be exercised either through textual descriptions or through high-level attributes that can be assigned to the individual objects appearing in the scene. The scene layout is described through a semantic representation which we call *sparse mask*, and can essentially be regarded a sketch of the scene.

"Controlling Style and Semantics in Weakly-Supervised Image Generation." [PLH20]
Dario Pavllo, Aurelien Lucchi, and Thomas Hofmann.
European Conference on Computer Vision (ECCV), 2020.
Selected for spotlight presentation (top 5% of submissions).

CHAPTER 3. While in the previous chapter we focus on the *input* representation of generative models, another interesting research question is what we can do with the *output* representation, that is, whether we can devise representations that are more structured than pixels. Since the 2D images that we perceive are projections of 3D scenes, a natural observation is that generative models that can explicitly reason about 3D geometry (*3D-aware* *generators*) should enable further downstream applications such as the synthesis of 3D objects for animation, movies, and video games. Following the paradigm of *differentiable rendering*, we introduce our approach named *convolutional mesh generation*, which can synthesize triangle meshes and associated high-resolution texture maps. A key contribution of our work is the encoding of the mesh and texture as 2D representations, which are semantically aligned and can be easily modeled by a 2D convolutional GAN. Furthermore, our approach is trained using only 2D supervision from single-view natural images, and does not require ground-truth 3D shapes or multiple views of the same object. We conclude this chapter by demonstrating that many of the conditioning techniques commonly applied to 2D GANs, such as textual prompts, can also be applied to this case.

"Convolutional Generation of Textured 3D Meshes." [Pav+20b]
Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi.
Neural Information Processing Systems (NeurIPS), 2020.
Selected for oral presentation (top 1% of submissions).

CHAPTER 4. A significant limitation of the approach presented in the previous chapter is the requirement for annotated camera poses, which are typically only available in small datasets, as they require laborious work from human annotators. In this chapter, we relax this assumption and apply our method to real-world datasets such as ImageNet. We demonstrate that our new method achieves performance on par with the method presented in the previous chapter, while not requiring such pose annotations.

"Learning Generative Models of Textured 3D Meshes from Real-World Images." [Pav+21]
Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi.
International Conference on Computer Vision (ICCV), 2021.

CHAPTER 5. An important downstream task in 3D deep learning is *single-view 3D reconstruction*, where the goal is to recover the 3D shape of an object along with its pose and appearance, from a single image. This task has applications in content creation, augmented reality (AR), robotics, and represents a fundamental research question in computer vision. While this task has been tackled in various ways throughout the literature, it turns out that 3D-aware generators (as the ones presented in the previous chapters) represent an excellent tool for undertaking this task. Following a paradigm called *GAN inversion* – where the goal is to invert a generator so as to recover the latent code that best describes an input image – we propose a reconstruction framework that achieves state-of-the-art results on this task. In this work, we also switch to a recently-proposed scene representation – Neural Radiance Fields (NeRF) – which combines differentiable rendering with implicit representations, allowing us to model arbitrary topologies with a greater detail compared to previous work based on triangle meshes. We further introduce a bootstrapping mechanism to mitigate some of the issues of *vanilla* GAN inversion, such as the requirement for a laborious optimization procedure.

"Shape, Pose, and Appearance from a Single Image via Bootstrapped Radiance Field Inversion." [Pav+23]
Dario Pavllo, David Joseph Tan, Marie-Julie Rakotosaona, and Federico Tombari.
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

2 CONTROLLABLE WEAKLY-SUPERVISED IMAGE GENERATION

CHAPTER ABSTRACT. We propose a weakly-supervised approach for conditional image generation of complex scenes where a user has fine control over objects appearing in the scene. We exploit sparse semantic maps to control object shapes and classes, as well as textual descriptions or attributes to control both local and global style. In order to condition our model on textual descriptions, we introduce a semantic attention module whose computational cost is independent of the image resolution. To further augment the controllability of the scene, we propose a two-step generation scheme that decomposes background and foreground. The label maps used to train our model are produced by a large-vocabulary object detector, which enables access to unlabeled data and provides structured instance information. In such a setting, we report better FID scores compared to fully-supervised settings where the model is trained on ground-truth semantic maps. We also showcase the ability of our model to manipulate a scene on complex datasets such as COCO and Visual Genome.

OPEN SOURCE. Code and pretrained models for this work are available at `https://github.com/dariopavllo/style-semantics`.

This chapter is based on our ECCV 2020 paper "Controlling Style and Semantics in Weakly-Supervised Image Generation" [PLH20].

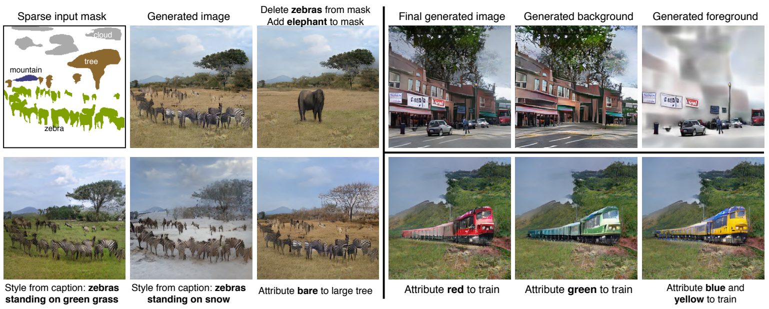

Figure 2.1: Our approach enables control over the style of a scene and its objects via high-level attributes or textual descriptions. It also allows for image manipulation through the mask, including moving, deleting, or adding object instances. The decomposition of the background and foreground (top-right corner) facilitates local changes in a scene.

2.1 INTRODUCTION

Deep generative models such as VAEs [KW14] and GANs [Goo+14] have made it possible to learn complex distributions over various types of data, including images and text. For images, the technical advances in [BDS19; Heu+17; Kar+18; Miy+18; MK18; Zha+19a] have enabled GANs to produce realistically-looking images for a large number of classes. However, these models often do not provide high-level control over image characteristics such as appearance, shape, texture, or color, and they fail to accurately model multiple (or compound) objects in a scene, thus limiting their practical applications. A related line of research aims at disentangling factors of variation [KLA19]. While these approaches can produce images with varied styles by injecting noise at different levels, the style factors are learned without any oversight, leaving the user with a loose handle on the generation process. Furthermore, their applicability has only been demonstrated for single-domain images (e.g. faces, cars, or birds). Some conditional approaches allow users to control the style of an image using either attributes [He+19; Yan+16a] or natural language [Xu+18; Zha+17; Zha+18a], but again, these methods only show compelling results on single-domain datasets.

One key aspect in generative modeling is the amount of required semantic information: i) *weak conditioning* (e.g. a sentence that describes a scene) makes the task underconstrained and harder to learn, potentially resulting in incoherent images on complex datasets. On the other hand, ii) *rich semantic information* (e.g. full segmentation masks) yields the best generative quality, but requires more effort from an artist or annotator. The applications of such richly-conditioned models are numerous, including art, animation, image manipulation, and realistic texturing of video games. Existing works in this category [CK17; Iso+17; Par+19b; Qi+18; Wan+18] typically require hand-labeled segmentation masks with per-pixel class annotations. Unfortunately, this is not flexible enough for downstream applications such as image manipulation, where the artist is faced with the burden of modifying the semantic mask coherently. Common transformations such as moving, deleting, or replacing an object require instance information (usually not available) and a strategy for infilling the background. Moreover, these models present little-to-no high-level control over the style of an image and its objects.

Our work combines the merits of both weak conditioning and strong semantic information, by relying on both mask-based generation – using a variant we call *sparse masks* – and text-based generation – which can be used to control the style of the objects contained in the scene as well as its global aspects. Figure 2.1 conceptualizes our idea. Our approach uses a large-vocabulary object detector to obtain annotations, which are then used to train a generative model in a weakly-supervised fashion. The input masks are sparse and retain instance information – making them easy to manipulate – and can be inferred from images or videos in-the-wild. We additionally contribute a conditioning scheme for controlling the style of the scene and its instances, either using high-level attributes or natural language with an attention mechanism. Unlike prior approaches, our attention model is applied directly to semantic maps (making it easily interpretable) and its computational cost does not depend on the image resolution, enabling its use in high-resolution settings. This conditioning module is general enough to be plugged into existing architectures. We also tackle another issue of existing generative models: local changes made to an object (such as moving or deleting) can affect the scene globally due to the learned correlations between classes. While these entangled representations improve scene coherence, they do not allow the user to modify a local part of a scene without affecting the rest. To this end, our approach relies on a multi-step generation process where we first generate a *background image* and then we generate foreground objects conditioned on the former. The background can be frozen while manipulating foreground objects.

Finally, we evaluate our approach on COCO [CUF18; Che+15; Lin+14] and Visual Genome [Kri+17], and show that our weakly-supervised setting can achieve better FID scores [Heu+17] than fully-supervised counterparts trained on ground-truth masks, and weakly-supervised counterparts where the model is trained on dense maps obtained from an off-the-shelf semantic segmentation model, while being more controllable and scalable to large unlabeled datasets. We show that this holds both in presence and in absence of style control.

2.2 RELATED WORK

Following the success of GANs in the unconditional setting, there have been a plethora of works whose goal is to condition the generator so as to make the synthesis process more controllable. In the context of image synthesis, there has been work on conditioning GANs on categorical labels [BDS19; Miy+18; MK18; Zha+19a], text [Ree+16a; Xu+18; Zha+17; Zha+18a], semantic maps [Iso+17; Par+19b; Wan+18], and conditioning images from other domains [Iso+17; Zhu+17a] (image-to-image translation). We review the literature on some of these paradigms, with a focus on the approaches that were directly relevant to our method at the time of publishing. For a discussion of more recent developments since this work was published, we refer the reader to Section 2.6, where we also cover synthesis techniques that evolved beyond GANs.

IMAGE GENERATION FROM SEMANTIC MAPS. In this setting, a semantic segmentation map is translated into a natural image.

Non-adversarial approaches are typically based on perceptual losses [CK17; Qi+18], whereas GAN architectures are based on patch-based discriminators [Iso+17], progressive growing [Kar+18; Wan+18], and conditional batch normalization where the semantic map is fed to the model at different resolutions [Par+19b]. Similarly to other state-of-the-art methods, our work is also based on this paradigm. Most approaches are trained on hand-labeled masks (limiting their application in the wild), but [Par+19b] shows one example where the model is weakly supervised on masks inferred using a semantic segmentation model [Che+18a]. Our model is also weakly supervised, but instead of a semantic segmentation model we use an object detector – which allows us to maintain instance information during manipulations, and results in *sparse masks*. While early work focused on class semantics, recent methods support some degree of style control. E.g. [Wan+18] trains an instance autoencoder and allows the user to choose a latent code from among a set of modes, whereas [Par+19b] trains a VAE to control the global style of a generated image by copying the style of a *guide* image. Both these methods, however, do not provide fine-grained style control (e.g. changing the color of an object to *red*). Another recent trend consists in generating images from structured layouts, which are transformed into semantic maps as an intermediate step to facilitate the task. In this regard, there is work on generation from bounding-box layouts [HHW19; Hon+18; SW19; Zha+19b] and scene graphs [JGF18]. Although these approaches tackle a harder task, they generate low-resolution images and are not directly relatable to our work, which tackles controllability among other aspects.

SEMANTIC CONTROL. Existing approaches do not allow for easy manipulation of the semantic map because they present no interface for encoding existing images. In principle, it is possible to train a weakly-supervised model on maps inferred from a semantic segmentation model, as [Par+19b] does for landscapes. However, as we show in Section 2.4.2, the results in this setting are notably worse than fully-supervised baselines. Furthermore, manipulations are still challenging because instance information is not available. Since the label masks are *dense*, even simple

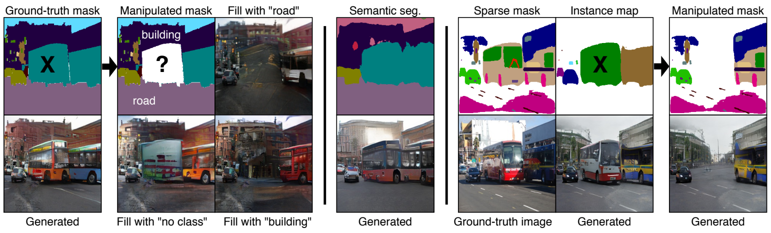

Figure 2.2: Left: when manipulating a ground-truth mask (e.g. deleting one bus), one is left with the problem of infilling the background which is prone to ambiguities (e.g. selecting a new class as either road or building). Furthermore, in existing models, local changes affect the scene globally due to learned correlations. **Middle:** in the wild, ground-truth masks are not available (neither are instance maps). One can infer maps using a semantic segmentation model, but these are often noisy and lack instance information (in the example above, we observe that the two buses are merged). **Right:** our weakly-supervised sparse mask setting, which combines fine-detailed masks with instance information. The two-step decomposition ensures that changes are localized.

transformations such as deleting or moving an object would create holes in the semantic map that need to be adjusted by the artist (Figure 2.2). *Dense* masks also make the task too constrained with respect to background aspects of the scene (e.g. sky, land, weather), which leaves less room for style control. Semantic control can also be framed as an unpaired image-to-image translation task [MCS19], but this requires ground-truth masks for both source and target instances, and can only translate between two classes.

TEXT-BASED GENERATION. Some recent models condition the generative process on text data. These are often based on autoregressive architectures [Ree+16b] and GANs [Ree+16a; Xu+18; Zha+17; Zha+18a]. Learning to generate images from text using GANs is known to be difficult due to the task being unconstrained. In order to ease the training process, [Zha+17; Zha+18a] propose a two-stage architecture named *StackGAN*. To avoid the instability associated with training a language model jointly with a GAN, they use a pretrained sentence encoder [Kir+15] that encodes a caption into a fixed-length vector which is then fed to the model. More advanced architectures such as *AttnGAN* [Xu+18] use an attention mechanism which we discuss in one of the next paragraphs. These approaches show interesting results on single-domain datasets (birds, flowers, etc.) but are less effective on complex datasets such as *COCO* [Lin+14] due to the intrinsic difficulty of generating coherent scenes from text alone. Some works [Jos+19; Yin+19] have demonstrated that generative models can benefit from taking as input multiple diverse textual descriptions per image. Finally, we are not aware of any prior work that conditions the generative process on *both* text and semantic maps (our setting).

MULTI-STEP GENERATION. Approaches such as [SOL19; Yan+17b] aim at disentangling background and foreground generation. While fully-unsupervised disentanglement is provably impossible [Loc+19], it is still achievable through some form of inductive bias – either in the model architecture or in the loss function. While [Yan+17b] uses spatial transformers to achieve separation, [SOL19] uses object bounding boxes. Both methods show compelling results on single-domain datasets that depict a centered object, but are not directly applicable to more challenging datasets. For composite scenes, [Tur+19] generates foreground objects sequentially to counteract merging effects. In our work, we are not interested in full disentanglement (i.e. we do not assume independence between background and foreground), but merely in separating the two steps while keeping them interpretable. Our model still exploits correlations among classes to maximize visual quality, and is applied to datasets with complex scenes. Finally, there has also been work on interactive generation using dialogue [Che+18b; El-+19; Sha+18].

ATTENTION MODELS IN GANS. For unconditional models (or models conditioned on simple class labels), self-attention GANs [BDS19; Zha+19a] use visual-visual attention to improve spatial coherence. For generation from text, [Xu+18] employ sentence-visual attention coupled with an LSTM encoder, but only in the generator. In the discriminator, the caption is enforced through a supervised loss based on features extracted from a pretrained

Inception [Sze+16] network. We introduce a new form of attention (*sentence-semantic*) which is applied to semantic maps instead of convolutional feature maps, and whose computational cost is independent of the image resolution. It is applied both to the generator and the discriminator, and on the sentence side it features a transformer-based [Vas+17] encoder.

2.3 APPROACH

2.3.1 *Framework*

Our main interest is conditional image generation of complex scenes where a user has fine control over the objects appearing in the scene. Prior work has focused on generating objects from ground-truth masks [Iso+17; Par+19b; Wan+18; Zhu+17a] or on generating outdoor scenes based on simple hand-drawn masks [Par+19b]. While the former approach requires a significant labeling effort, the latter is not directly suitable for complex datasets such as COCO-Stuff [CUF18], whose images consist of a large number of classes with complex (hard to draw) shapes. We address these problems by introducing a new model that is conditioned on sparse masks – to control object shapes and classes – and on text/attributes to control style and textures. This gives the ability to a user to produce scenes through a variety of image manipulations (such as moving, scaling or deleting an instance, adding an instance from another image or from a database of shapes) as well as style manipulations controlled using either high-level attributes on individual instances (e.g. *red*, *green*, *wet*, *shiny*) or using text that refers to objects as well as global context (e.g. "a red car at night"). In the latter case, visual-textual correlations are not explicitly defined but are learned in an unsupervised way.

SPARSE MASKS. Instead of training a model on precise segmentation masks as in [Iso+17; Par+19b; Wan+18], we use a mask generated automatically from a large-vocabulary object detector. Compared to a weakly-supervised setting based on semantic segmentation, this process introduces less artifacts (see Appendix

2.7.4) and has the benefit of providing information about each instance (which may not always be available otherwise), including parts of objects which would require significant manual effort to label in a new dataset. In general, our set of classes comprises countable objects (person, car, etc.), parts of objects (light, window, door, etc.), as well as uncountable classes (grass, water, snow), which are typically referred to as "stuff" in the COCO terminology [CUF18]. For the latter category, an object detector can still provide useful sparse information about the background, while keeping the model autonomous to fill-in the gaps. We describe the details of our object detection setup in Section 2.4.1.

TWO-STEP GENERATION. In the absence of constraints, conditional models learn class correlations observed in the training data. For instance, while dogs typically stand on green grass, zebras stand on yellow grass. While this feature is useful for maximizing scene coherence, it is undesirable when only a local change in the image is wanted. We observed similar global effects on other local transformations, such as moving an object or changing its attributes, and generally speaking, small perturbations of the input can result in large variations of the output. We show a few examples in the Appendix 2.7.4. To tackle this issue, we propose a variant of our architecture which we call *two-step* model and which consists of two concatenated generators (Figure 2.3, right). The first step (generator G_1) is responsible for generating a *background* image, whereas the second step (generator G_2) generates a *foreground* image conditioned on the background image. The definition of what constitutes background and foreground is arbitrary: our choice is to separate by class: static/uncountable objects (e.g. buildings, roads, grass, and other surfaces) are assigned to *background*, and moving/countable objects are assigned to *foreground*. Some classes can switch roles depending on the parent class, e.g. *window* is *background* by default, but it becomes *foreground* if it is a child of a foreground object such as a car.
When applying a local transformation to a foreground object, the background can conveniently be frozen to avoid global changes. As a side benefit, this also results in a lower computational cost to regenerate an image. Unlike work on disentanglement [SOL19;

Yan+17b] which enforces that the background is independent of the foreground without necessarily optimizing for visual quality, our goal is to enforce separation while maximizing qualitative results. In our setting, G_1 is exposed to both background and foreground objects, but its architecture is designed in a way that foreground information is not rendered, but only used to induce a bias in the background (see Section 2.3.2).

ATTRIBUTES. Our method allows the user to control the style of individual instances using high-level attributes. These attributes refer to appearance factors such as colors (e.g. white, black, red), materials (wood, glass), and even modifiers that are specific to classes (leafless, snowy), but not shape or size, since these two are determined by the mask. An object can also combine multiple attributes (e.g. black and white) or have none – in this case, the generator would pick a predefined mode. This setup gives the user a lot of flexibility to manipulate a scene, since the attributes need not be specified for every object.

CAPTIONS. Alternatively, one can consider conditioning style using natural language. This has the benefit of being more expressive, and allows the user to control global aspects of the scene (e.g. time of the day, weather, landscape) in addition to instance-specific aspects. While this kind of conditioning is harder to learn than plain attributes, in Section 2.3.2 we introduce a new attention model that shows compelling results without excessively increasing the model complexity.

2.3.2 *Architecture.*

We design our conditioning mechanisms to have sufficient generality to be attached to existing conditional generative models. In our experiments, we choose *SPADE* [Par+19b] as the backbone for our conditioning modules, which to our knowledge represents the state of the art. As in [Par+19b], we use a multi-scale discriminator [Wan+18], a perceptual loss in the generator using a pretrained VGG network [SZ14], and a feature matching loss in the discriminator [Wan+18].

ONE-STEP MODEL. Since this model (Figure 2.3, left) serves as a baseline, we keep its backbone as close as possible to the reference model of [Par+19b]. We propose to insert the required information about attributes/captions in this architecture by modifying the input layer and the conditional batch normalization layers of the generator, which is where semantic information is fed to the model. We name these S-blocks (short for *semantic-style* block).

SEMANTIC-STYLE BLOCK. For class semantics, the input sparse mark is fed to a pixel-wise embedding layer to convert categorical labels into 64D embeddings (including the empty space, which is a special class "no class"). To add style information, we optionally concatenate another 64D representation to the class embedding (pixel-wise); we explain how we derive this representation in the next two paragraphs. The resulting feature map is convolved with a 3×3 kernel, passed through a ReLU non-linearity and convolved again to produce two feature maps γ and $\boldsymbol{\beta}$, respectively, the conditional batch normalization gain and bias. The normalization is then computed as $\mathbf{y} = \mathrm{BN}(\mathbf{x}) \odot (1 + \gamma) + \boldsymbol{\beta}$, where $\mathrm{BN}(\mathbf{x})$ is the parameter-free batch normalization. The last step is related to [Par+19b] and other architectures based on conditional batch normalization. Unlike [Par+19b], however, we do not use 3×3 convolutions on one-hot representations in the input layer. This allows us to scale to a larger number of classes without signifi-

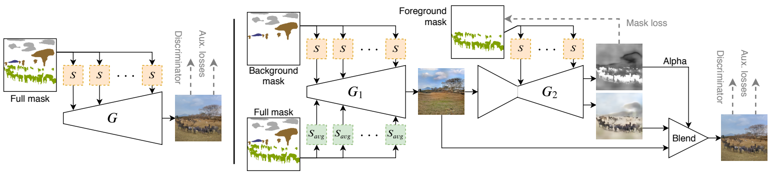

Figure 2.3: Left: One-step model. **Right:** two-step model. The background generator G_1 takes as input a *background mask* (processed by S-blocks) and the full mask (processed by S_{avg}-blocks, where positional information is removed). The foreground generator takes as input the output of G_1 and a *foreground mask*. Finally, the two outputs are alpha-blended. For convenience, we do not show attributes/text in this figure.

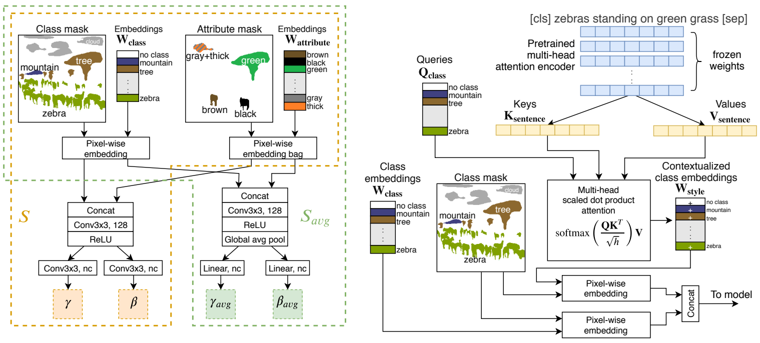

Figure 2.4: Left: Conditioning block with attributes. Class and attribute embeddings are concatenated and processed to generate the conditional batch normalization gain and bias. In the attribute mask, embeddings take the contour of the instance to which they refer. In G_1 of the two-step model, where S and S_{avg} are both used, the embedding weights are shared. **Right:** Attention mechanism for conditioning style via text. The sentence (of length $n = 7$ including delimiters) is fed to a pretrained attention encoder, and each token is transformed into a key and a value using two trainable linear layers. The queries are learned for each class, and the attention yields a set of contextualized class embeddings that are concatenated to the regular semantic embeddings.

cantly increasing the number of parameters. We apply the same principle to the discriminators.

CONDITIONING ON ATTRIBUTES. For attributes, we adopt a *bag-of-embeddings* approach where we learn a 64D embedding for each possible attribute, and all attribute embeddings assigned to an instance are broadcast to the contour of the instance, summed together, and concatenated to the class embedding. Figure 2.4 (left) (S-block) depicts this process. To implement this efficiently, we create a multi-hot *attribute mask* (1 in the locations corresponding to the attributes assigned to the instance, 0 elsewhere) and feed it through a 1×1 convolutional layer with N_{attr} input channels and 64 output channels. Attribute embeddings are shared among classes and are not class-specific. This helps the model generalize better (e.g. colors such as "white" apply both to vehicles and animals), and we empirically observe that implausible combinations (e.g. leafless person) are simply ignored by the generator without side effects.

CONDITIONING ON TEXT. While previous work has used fixed-length vector representations [Zha+17; Zha+18a] or one-layer attention models coupled with RNNs [Xu+18], the diversity of our scenes led us to use a more powerful encoder entirely based on self-attention [Vas+17]. We encode the image caption using a pretrained BERT$_{base}$ model [Dev+18] (110M parameters). It is unreasonable to attach such a model to a GAN and fine-tune it, both due to excessive memory requirements and due to potential instabilities. Instead, we freeze the pretrained model and encode the sentence, extract its hidden representation after the last or second-to-last layer (we compare these in Section 2.4.2), and train a custom multi-head attention layer for our task. This paradigm, which is also suggested by [Dev+18], has proven successful on a variety of NLP downstream tasks, especially when these involve small datasets or limited vocabularies. Furthermore, instead of storing the language model in memory, we simply pre-compute the sentence representations and cache them.

Next, we describe the design of our trainable attention layer (Figure 2.4, right). Our attention mechanism is different from the commonly-used sentence-visual attention [Xu+18], where attention is directly applied to convolutional feature maps inside the generator. Instead, we propose a form of sentence-semantic attention which is computationally efficient, interpretable, and modular. It can be concatenated to conditioning layers in the same way as we concatenate attributes. Compared to sentence-visual attention, whose cost is $\mathcal{O}(nd^2)$ (where n is the sentence length and $d \times d$ is the feature map resolution), our method has a cost of $\mathcal{O}(nc)$ (where c is the number of classes), i.e. it is independent of the image resolution. We construct a set of c *queries* (i.e. one for each class) of size $h = 64$ (where h is the attention head size). We feed the hidden representations of each token of the sentence to two linear layers, one for the *keys* and one for the *values*. Finally, we compute a scaled dot-product attention [Vas+17], which yields a set of c *values*. To allow the conditioning block to attend to multiple parts of the sentence, we use 6 or 12 attention heads (ablations in Section 2.4.2), whose output values are concatenated and further transformed through a linear layer. This process can be thought of as generating *contextualized class embeddings*, i.e. class embeddings customized according to the sentence. For instance, given a semantic map that depicts a car and the caption "a red car and a person", the query corresponding to the visual class *car* would most likely attend to "red car", and the corresponding value will induce a bias in the model to add redness to the position of the car. Finally, the *contextualized class embeddings* are applied to the semantic mask via pixel-wise matrix multiplication with one-hot vectors, and concatenated to the *class embeddings* in the same way as attributes. In the current formulation, this approach is unable to differentiate between instances of the same class. We propose a possible mitigation in Section 2.5.

TWO-STEP MODEL. It consists of two concatenated generators. G_1 generates the background, i.e. it models $p(x_{\text{bg}})$, whereas G_2 generates the foreground conditioned on the background, i.e. $p(x_{\text{fg}}|x_{\text{bg}})$. One notable difficulty in training such a model is that background images are never observed in the training set (we only observe the final image), therefore we cannot use an intermediate discriminator for G_1. Instead, we use a single, final discriminator and design the architecture in a way that the gradient of the discriminator (plus auxiliary losses) is redirected to the correct generator. The convolutional nature of G_1 would then ensure that the background image does not contain visible holes. A natural choice is *alpha blending*, which is also used in [SOL19; Yan+17b]. G_2 generates an RGB foreground image plus a transparency mask (*alpha* channel), and the final image is obtained by pasting the foreground onto the background via linear blending:

$$x_{\text{final}} = x_{\text{bg}} \cdot (1 - \alpha_{\text{fg}}) + x_{\text{fg}} \cdot \alpha_{\text{fg}} \tag{2.1}$$

where x_{final}, x_{bg}, and x_{fg} are RGB images, and α_{fg} is a 1-channel image bounded in $[0, 1]$ by a sigmoid. Readers familiar with highway networks [SGS15] might notice a similarity to this approach in terms of gradients dynamics. If $\alpha_{\text{fg}} = 1$, the gradient is completely redirected to x_{fg}, while if $\alpha_{\text{fg}} = 0$, the gradient is redirected to x_{bg}.

This scheme allows us to train both generators in an end-to-end fashion using a single discriminator, and we can also preserve auxiliary losses (e.g. VGG loss) which [Par+19b] has shown to be very important for convergence. To incentivize separation between classes as defined in Section 2.3.1, we supervise α_{fg} using a binary cross-entropy loss, and decay this term over time (see Section 2.4.1).

G_2 uses the same S-blocks as the ones in the one-step model, but here they take a *foreground mask* as input (Figure 2.3, right). G_1, on the other hand, must exploit foreground information without rendering it. We therefore devise a further variation of input conditioning that consists of two branches: (i) the first branch (S-block) takes a *background mask* as input and processes it as usual to produce the batch normalization gain γ and bias $\boldsymbol{\beta}$. (ii) the second branch (S_{avg}-block, Figure 2.4 left) takes the full mask as input (background plus foreground), processes it, and applies global average pooling to the feature map to remove information about localization. This way, foreground information is only used to bias G_1 and cannot be rendered at precise spatial locations. After pooling, it outputs γ_{avg} and $\boldsymbol{\beta}_{avg}$. (iii) The final conditional batch normalization is computed as:

$$\mathbf{y} = \mathrm{BN}(\mathbf{x}) \odot (1 + \gamma + \gamma_{avg}) + \boldsymbol{\beta} + \boldsymbol{\beta}_{avg} \tag{2.2}$$

Finally, the discriminator D takes the full mask as input (background plus foreground). Note that, if G_1 took the full mask as input without information reduction, it would render visible "holes" in the output image due to gradients never reaching the foreground zones of the mask, which is what we are trying to avoid. The Appendix 2.7.1 provides more details about our architectures, and 2.7.2 shows how G_2 can be used to generate one object at a time to fully disentangle foreground objects from each other (although this is unnecessary in practice).

2.4 EXPERIMENTS

For consistency with [Par+19b], we always evaluate our model on the COCO-Stuff validation set [CUF18], but we train on a variety of training sets:

COCO-Stuff (COCO2017) [CUF18; Lin+14] contains 118k training images with captions [Che+15]. We train with and without captions. COCO-Stuff extends COCO2017 with ground-truth semantic maps, but for our purposes the two datasets are equivalent since we do not exploit ground-truth masks.
Visual Genome (VG) [Kri+17] contains 108k images that partially overlap with COCO (≈50%). VG does not have a standard train/test split, therefore we leave out 10% of the dataset to use as a validation set (IDs ending with 9), and use the rest as a training set from which we remove images that overlap with the COCO-Stuff validation set. We extract the attributes from the scene graphs.
Visual Genome augmented (VG+) VG augmented with the 123k images from the COCO unlabeled set. The total size is 217k images after removing exact duplicates. The goal is to evaluate how well our method scales to large unlabeled datasets. We train without attributes and without captions.

For all experiments, we evaluate the *Fréchet Inception Distance* (FID) [Heu+17] (precise implementation details of the FID in the Appendix 2.7.3). Furthermore, we report our results in Section 2.4.2 and provide additional qualitative results in 2.7.4.

2.4.1 *Implementation details*

SEMANTIC MAPS. To construct the input semantic maps, we use the semi-supervised implementation of Mask R-CNN [He+17; Ren+15] proposed by [Hu+18]. It is trained on bounding boxes from Visual Genome (3000 classes) and segmentation masks from COCO (80 classes), and learns to segment classes for which there are no ground-truth masks. We discard the least frequent classes, and, since some VG concepts overlap (e.g. car, vehicle) leading to spurious detections, we merge these classes and end up with a total of $c = 280$ classes (plus a special class for "no class"). We set the threshold of the object detector to 0.2, and further refine the predictions by running a class-agnostic non-maximum-suppression (NMS) step on the detections whose mask intersection-over-union (IoU) is greater than 0.7. We also construct a transformation hierarchy to link children to their parents in the semantic map (e.g.

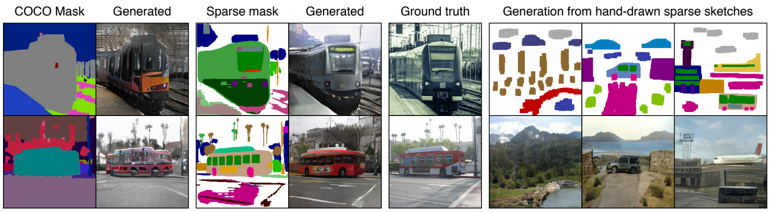

Figure 2.5: Left: the larger set of labels in our sparse masks improves fine details. These masks are easy to obtain with a semi-supervised object detector, and would otherwise be too hard to hand-label. **Right:** sparse masks are also easy to sketch by hand.

headlight of a car) so that they can be manipulated as a whole; further details in the Appendix 2.7.1. We select the 256 most frequent attributes, manually excluding those that refer to shapes (e.g. *short*, *square*).

TRAINING. We generate images at 256×256 and keep our experimental setting and hyperparameters as close as possible to [Par+19b] for a fair comparison. For the two-step model, we provide supervision on the alpha blending mask and decay this loss term over time, observing that the model does not re-entangle background and foreground. This gives G_2 some extra flexibility in drawing details that are not represented by the mask (reflections, shadows). Hyperparameters and additional training details are specified in the Appendix 2.7.1.

2.4.2 *Results*

QUANTITATIVE RESULTS. We show the FID scores for the main experiments in Table 2.1 (left). While improving FID scores is not the goal of our work, our weakly-supervised sparse mask baseline (#3) interestingly outperforms both the fully-supervised baseline on SPADE [Par+19b] (#1) and the weakly-supervised baseline (#2) trained on dense semantic maps. These experiments adopt an identical architecture and training set, no style input, and differ only in the type of input mask. For #2 we obtain the seman-

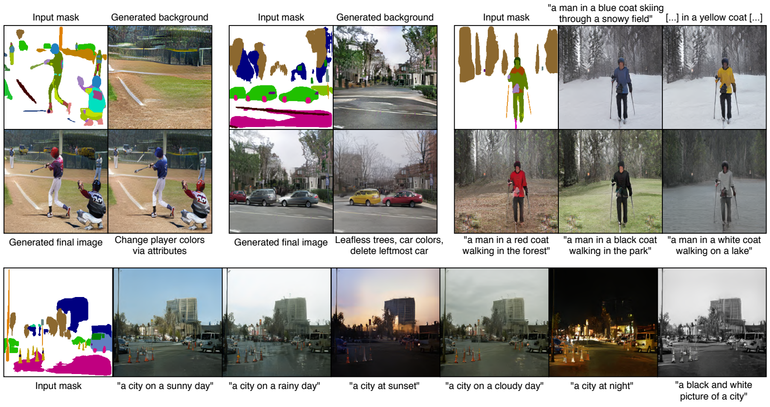
Input mask
Generated background
Generated final image
Change player colors via attributes
Input mask
Generated background
Generated final image
Leafless trees, car colors, delete leftmost car
Input mask
"a man in a blue coat skiing through a snowy field"
[...] in a yellow coat [...]
"a man in a red coat walking in the forest"
"a man in a black coat walking in the park"
"a man in a white coat walking on a lake"
Input mask
"a city on a sunny day"
"a city on a rainy day"
"a city at sunset"
"a city on a cloudy day"
"a city at night"
"a black and white picture of a city"

Figure 2.6: Qualitative results (256 × 256). **Top-left** and **top-middle**: two-step generation with manipulation of attributes and instances. **Top-right:** manipulating style (both context and instances) via text. **Bottom:** manipulating global style via text.

tic maps from *DeepLab-v2* [Che+18a], a state-of-the-art semantic segmentation model pretrained on COCO-Stuff. Our improvement is partly due to masks better representing fine details (such as windows, doors, lights, wheels) in compound objects, which are not part of the COCO class set. In Figure 2.5 (left) we show some examples. Moreover, the experiment on the *augmented* Visual Genome dataset highlights that our model benefits from extra unlabeled images (#4). Rows #5–9 are trained with style input. In particular, we observe that these outperform the baseline even when they use a two-step architecture (which is more constrained) or are trained on a different training set (VG instead of COCO). Row #6-7 draw their text embeddings from the last BERT layer and adopt 12 attention heads (the default), whereas #5 draws its embeddings from the 2nd-last layer, uses 6 heads, and performs slightly better.

QUALITATIVE RESULTS. In Figure 2.6 we show qualitative results as well as examples of manipulations, either through attributes or text. Additional examples can be seen in the Appendix 2.7.4, including latent space interpolation [KLH18]. In 2.7.5, we

#	Training set	Test set(s)	Type	Mask input	Style input	FID
1	COCO-train	COCO-val	1-step [Par+19b]	Ground truth	None	22.64
2	COCO-train	COCO-val	1-step †	Semantic seg.	None	23.97
3	COCO-train	COCO-val	1-step †	Sparse (ours)	None	20.02
4	VG+ (aug.)	COCO-val/VG-val	1-step †	Sparse (ours)	None	**18.93**/13.23
5	COCO-train	COCO-val	1-step †	Sparse (ours)	Text (6h, L_{n-1})	19.65
6	COCO-train	COCO-val	1-step †	Sparse (ours)	Text (12h, L_n)	20.63
7	COCO-train	COCO-val	2-step †	Sparse (ours)	Text (12h, L_n)	20.64
8	VG	COCO-val/VG-val	1-step †	Sparse (ours)	Attributes	21.13/15.12
9	VG	COCO-val/VG-val	2-step †	Sparse (ours)	Attributes	20.83/14.88

	Ref.	Experiment	FID (Δ)
I	#1	COCO "things" only	32.31 (+9.67)
II	#6	12h, L_n, attr. in D	20.44 (-0.19)
III	#6	12h, L_{n-1}	19.77 (-0.86)
IV	#6	6h, L_{n-1}	19.65 (-0.98)
V	#9	No f.g. info in S_{avg}	25.16 (+4.33)
VI	#9	Attr. randomization	20.64 (-0.19)

Table 2.1: Left: FID scores for the main experiments; lower is better. The first line represents the SPADE baseline [Par+19b]. For the models trained on VG, we also report FID scores on our VG validation set. (†) indicates that the model is weakly-supervised, (6h) denotes "6 attention heads", L_{n-1} indicates that the text embeddings are drawn from the second-to-last BERT layer. **Right:** ablation study with extra experiments.

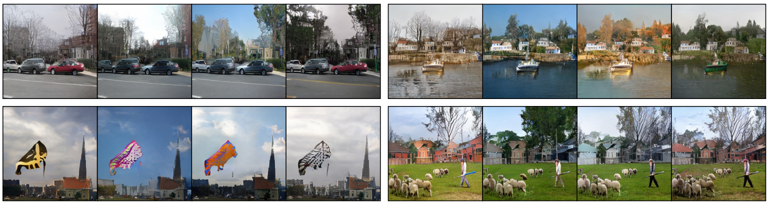

Figure 2.7: Random styles by sampling attributes from a per-class empirical distribution.

visualize the attention mechanism. Finally, we observe that sketching sparse masks by hand is very practical (Figure 2.5, right) and provides an easier interface than dense semantic maps (in which the class of every pixel must be manually specified).

STYLE RANDOMIZATION. Since we represent style explicitly, at inference we can randomize the style of an image by drawing attributes from a per-class empirical distribution. This is depicted in Figure 2.7, and has the additional advantage of being interpretable and editable (attributes can be refined manually after sampling). The two-step decomposition also allows users to specify different sampling strategies for the background and foreground; more details in the Appendix 2.7.2.

ABLATION STUDY. While Table 2.1 (left) already includes a partial ablation study where we vary input conditioning and some aspects of the attention module, in Table 2.1 (right) we make this more explicit and include additional experiments. First, we train a model on a sparsified COCO dataset by only keeping the "things" classes and discarding the "stuff" classes. This setting (I) performs significantly worse than #1 (which uses all classes), motivating the use of a large class vocabulary. Next, we ablate conditioning via text (baseline #6, which adopts the default hyperparameters of BERT). In (II), we augment the discriminator with ground-truth attributes to provide a stronger supervision signal for the generator (we take the attributes from Visual Genome for the images that overlap between the two datasets). The improvement is marginal, suggesting that our model can learn visual-textual correlations without explicit supervision. In (III), we draw the token representations from the second-to-last layer instead of the last, and in (IV) we further reduce the number of attention heads from 12 to 6. Both III and IV result in an improvement of the FID, which justifies the hyperparameters chosen in #5. Finally, we switch to attribute conditioning (baseline #9). In (IV), we remove foreground information at inference from the S_{avg} block of the first generator G_1 (we feed the background mask twice in S and S_{avg}). The FID degrades significantly, suggesting that G_1 effectively exploits foreground information to bias the result. In (V) we show that randomizing style at inference (previous paragraph) is not detrimental to the FID, but in fact seems to be slightly beneficial, probably due to the greater sample diversity.

ROBUSTNESS AND FAILURE CASES. Input masks can sometimes be noisy due to spurious object detections on certain classes. Since these are also present at train time, weakly-supervised training leads to some degree of noise robustness, but sometimes the artifacts are visible in the generated images. We show some positive/negative examples in the Appendix Fig. 2.15. In principle, mask noise can be reduced by using a better object detector. We also observe that our setup tends to work better on outdoor scenes and sometimes struggles with fine geometric details in indoor scenes or photographs shot from a close range.

2.5 SUMMARY

We introduced a weakly-supervised approach for the conditional generation of complex images. The generated scenes can be controlled through various manipulations on the sparse semantic maps, as well as through textual descriptions or attribute labels. Our method enables a high level of semantic/style control while benefiting from improved FID scores. From a qualitative point-of-view, we have demonstrated a wide variety of manipulations that can be applied to an image. Furthermore, our weakly supervised setup opens up opportunities for large-scale training on unlabeled datasets, as well as generation from hand-drawn sketches.

2.6 FUTURE DEVELOPMENTS

POTENTIAL IMPROVEMENTS. There are several ways one could pursue to further enrich the set of tools used to manipulate the generation process. For instance, the current version of our attention mechanism cannot differentiate between instances belonging to the same class and does not have direct access to positional information. While incorporating such information is beyond the scope of this work, we suggest that this can be achieved by appending a *positional embedding* to the attention queries. In the NLP literature, the latter is often learned according to the position of the word in the sentence [Dev+18; Vas+17], but images are 2D and therefore do not possess such a natural order. Additionally, this would require captions that are more descriptive than the ones in COCO, which typically focus on actions instead of style. Finally, in order to augment the quality of sparse maps, the object detector could be trained on a higher-quality, large-vocabulary dataset [GDG19].

RECENT LITERATURE. After this work was published, there has been a range of new approaches dealing with layout-based and text-based image generation, many of which steered away from GANs. Following the success of transformers [Vas+17] in image classification [Dos+21], transformers have also been employed in autoregressive techniques for image synthesis. Successful examples include ImageGPT [Che+20] (unconditional), DALL-E [Ram+21] (text-conditioned), and NÜVA [Wu+22] (text- and sketch-conditioned). More recently, diffusion models [HJA20; SME20] have achieved state-of-the-art results on these tasks, outperforming both GANs and autoregressive techniques. Architectures such as Imagen [Sah+22], DALL-E 2 [Ram+22], and Stable Diffusion [Rom+22] can generate complex scenes from text, and the latter has also been demonstrated on generation from layouts. Owing to these results, it is likely that diffusion models will represent the leading paradigm in the near future.

FOLLOW-UP WORK. In this chapter, we have mostly explored ways to condition the image synthesis process so as to make it more controllable. This is typically achieved by devising effective *input* representations which are injected into the model. Another interesting research question is whether it is possible to come up with useful *output* representations. The approaches analyzed and proposed so far generate RGB images, i.e. the synthesis process takes place in *pixel space*. While this is adequate for many applications (e.g. the creation of static content), there might be other downstream applications where richer representations are required. An interesting direction in this respect is represented by 3D-aware generators, which output 3D representations and have the advantage of disentangling viewpoint from appearance. This is useful in applications where the identity of the generated object/scene needs to be preserved when observed from different viewpoints, such as in the generation of videos or content for movies/video games. It is also more interesting from an image understanding perspective, as 3D representations are rooted in human perception (we perceive 2D projections of 3D scenes but can still reason about 3D geometry). Therefore, we dedicate the next chapter of this dissertation (Chapter 3) to this research direction.

2.7 APPENDIX

2.7.1 *Detailed architecture*

In this section, we provide additional implementation details about our architecture in order to consolidate the already-presented Figure 2.3 (overview of the generators) and Figure 2.4 (conditioning blocks).

ONE-STEP GENERATOR. In Section 2.3.2 we mention that we use [Par+19b] as the backbone for the one-step model, and that we insert conditioning information in the normalization blocks as well as in the very first layer of the generator. In Figure 2.8 we show the detailed architecture of this model. The implementation of an individual "SPADE ResBlock" is specified in [Par+19b], but for reference we mention that each residual block consists of two normalization blocks wrapped by a skip-connection. If the number of input and output channels does not match, the skip-connection is learned, i.e. a third normalization block is learned. In the models conditioned on captions, we never attach attention inputs to skip-connections (to avoid potential instabilities). Each normalization block learns its own set of weights, and in our case they correspond to the S or S_{avg} blocks specified in Figure 2.4.

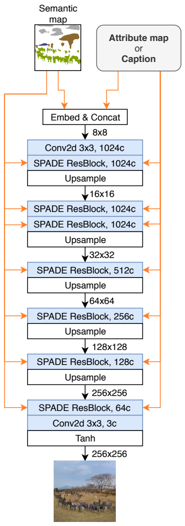

Figure 2.8: One-step generator using the SPADE backbone. "1024c" stands for "1024 output channels". The number on the right of an arrow specifies the feature map resolution at that level. Orange arrows indicate that the input information is fed to S blocks.

TWO-STEP GENERATOR. The architecture of the two-step generator is depicted in Figure 2.9, and differs significantly from the aforementioned implementation. The background generator G_1 is a simplified version of the one-step generator with fewer residual blocks. The foreground generator G_2 implements a bottleneck architecture that takes as input the generated background image and compresses it through a series of *unconditional* residual blocks. The low-resolution feature-map is then expanded again through a series of *conditional* blocks. Interestingly, for foreground manipulations it is possible to preprocess the feature maps up to the last *unconditional* downsampling block in G_2 (8×8 resolution) and greatly speed up regeneration.

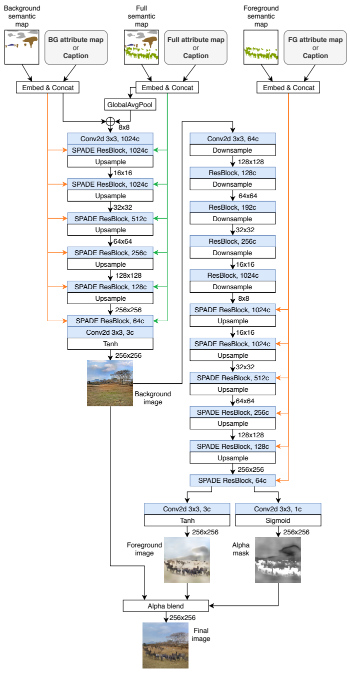

Figure 2.9: Two-step generator. The left side of the figure depicts G_1 (background generator), while the right side depicts G_2 (foreground generator). Orange arrows indicate that the input information is fed to S blocks, whereas green arrows denote inputs to S_{avg} blocks.

DISCRIMINATOR. We use the multi-scale discriminator from [Par+19b; Wan+18] and change its input layer to add information about attributes or captions. The architecture is shown in Figure 2.10. As usual with multi-scale discriminators, we train two instances: one which takes as input an image at full resolution, and one which takes as input a downsampled version (by a factor of two). They learn different sets of embeddings and different sets of attention heads if the style is conditioned on a sentence.

MODEL COMPLEXITY. Table 2.2 presents the number of parameters for all variants of our approach. The SPADE baseline trained on the 182 COCO-Stuff classes requires 97.5M parameters. Our 1-step baseline trained without style information (neither attributes nor captions) on our set of 280 classes requires a slightly lower number of parameters (94.2M) thanks to the pixel-wise class embeddings, even though the number of classes is larger. In the version with attributes, the added cost (+2.3M parameters) is only due to the learned attribute embeddings (256 64d embeddings per normalization block). In the version with captions, the custom attention modules add 12.5M parameters (for 6 heads) or 23.3M parameters (for 12 heads). The number of parameters can be easily tuned by varying the number of attention heads. We conduct a similar analysis on the two-step model. In this case, the background generator is slightly more powerful than the foreground generator.

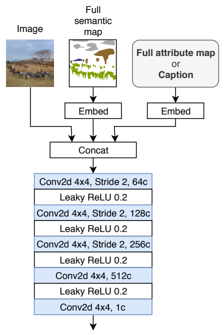

Figure 2.10: Discriminator backbone (used in all architectures).

Approach	Style input	# params
Baseline [Par+19b]	None	97.5M
1-step	None	94.2M
1-step	Attributes	96.5M
1-step	Text (6h)	106.7M
1-step	Text (12h)	117.5M
2-step	None	74.5M + 50.6M
2-step	Attributes	78.3M + 51.9M
2-step	Text (12h)	90.7M + 65.8M

Table 2.2: Number of parameters for different variations of our approach. For the two-step models we specify the numbers for both generators (respectively G_1 and G_2). "6h" denotes "6 attention heads".

SPARSE MAP GENERATION AND MANIPULATION. In this paragraph we provide further details in addition to those presented in Section 2.4.1. Specifically, we describe how we construct and maintain the data structure that enables instance manipulation and rasterization into a sparse semantic map. Since a scene may consist of objects that partially overlap, the order in which they are drawn on the semantic map matters, e.g. given a *car* and its *headlight*, we want to render the headlight semantic mask on top of the car and not the opposite. Therefore, we sort all instances by mask area and draw them from the largest to the smallest. Additionally, we construct a scene graph to facilitate manipulation: if 70% of the area of an instance is contained within another instance, it becomes a child of the latter. With regard to the previous example, moving the car would also move the headlights attached to it. Finally, in our experiments on Visual Genome, we link attributes to an instance if the IoU between the ground-truth region and the detected bounding box is greater than 0.5.

TRAINING DETAILS AND HYPERPARAMETERS. In all experiments, we train on 8 Pascal GPUs for 100 epochs using Adam (learning rate: 1e-4 for G, 4e-4 for D, one G update per D update), and start decaying the learning rate to 0 after the 50th epoch in a linear fashion. We use a batch size of 32 for the *one-step* model and 24 for the *two-step* model (the largest we can fit into memory), with synchronized batch normalization. Training takes one week for the one-step model and two weeks for the two-step model. For the alpha blending loss term, we start from a factor of 10, and decay it exponentially with $\alpha = 0.9997$ per weight update, down to 0.01. For the experiments with captions, since COCO comprises five captions per image, we randomly select one caption at training time. In the evaluation phase, we concatenate the representations of all captions since our attention model can easily decide which ones to attend to.

2.7.2 *Additional inference details*

RANDOMIZING STYLE. In Section 2.4.2 we mention that we can randomize the style of an image by sampling attributes from a per-class empirical distribution. More precisely, we estimate a discrete probability distribution of the attributes assigned to each class of the dataset. This includes the empty set (no attribute for a given instance) as well as compound attributes (e.g. *blue and red* is different than *blue* or *red*). At inference, for each instance, we sample an element from the distribution of the class to which the instance belongs. The two-step decomposition also allows us to specify different strategies for the background and foreground. In the examples in Figure 2.7, all background instances of a given class take the same attributes as input (e.g. all trees are leafless), which results in scenes with coherent styles. Conversely, foreground instances are still fully randomized (it would not be realistic to see cars all of the same color, for example). Within an individual instance, the style of its children is uniform, e.g. the same attributes are assigned to all wheels of a car, but of course wheel styles can be different across different cars.

INTERPOLATING STYLE. Our approach allows for smooth interpolation of attributes and text. While attention models usually preclude interpolation (whereas models based on fixed-length sentence embeddings such as [Zha+17] easily allow it), our *sentence-semantic* attention mechanism enables interpolation over the *contextualized class embeddings*, i.e. over the pooled attention values. For all cases (masks, attributes, text), we respectively interpolate between class embeddings, attribute embeddings, and contextualized class embeddings using spherical interpolation (*slerp*), which traverses regions with a higher probability mass [KLH18]. Unlike [Zha+17], we found it unnecessary to enforce a prior on the embeddings via a KL divergence term in the loss. We show some examples of interpolation in Figure 2.11.

GENERATING ONE OBJECT AT A TIME. To ensure that foreground objects do not affect each other in the two-step model, it may be interesting to generate them one-by-one. In our exper-

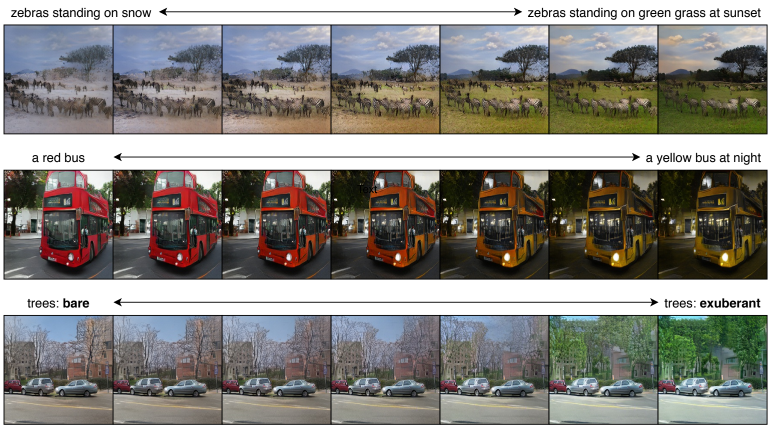

Figure 2.11: Interpolating style between two sentences (top two rows) and two attributes (bottom row). The smooth transitions across multiple factors of variation (e.g. color and time of the day) suggest that our latent space is structured and does not require regularization. For instance, in the middle row, the bus color traverses the region of *orange* while interpolating between red and yellow, even though it is not explicitly instructed to do so. Additionally, the headlights of the bus become increasingly brighter.

iments we generate all foreground objects at once by running a single instance of G_2, motivated by the much lower computational cost and the observation that foreground objects are usually well-separated. Nonetheless, our framework is flexible enough to support one-by-one generation of objects. In this regard, G_2 can be run independently for each object, and the output images and masks can be combined into a single, final image. Denoting the background image as $\mathbf{x}_{\text{bg}}$, the foreground images as $\mathbf{x}_{\text{fg}}^{[i]}$ ($i \in \{1 \dots N\}$), and the corresponding *unscaled* (i.e. before the activation function) transparency masks as $\boldsymbol{\alpha}'^{[i]}_{\text{fg}}$, we can generalize Equation 2.1 as follows:

$$\mathbf{w}_{\text{fg}}^{[i]} = \text{softmax}_i \left(\boldsymbol{\alpha}'^{[i]}_{\text{fg}} \right) \tag{2.3}$$

$$\mathbf{x}_{\text{fg}} = \sum_i \mathbf{x}_{\text{fg}}^{[i]} \odot \mathbf{w}_{\text{fg}}^{[i]} \tag{2.4}$$

$$\boldsymbol{\alpha}_{\text{fg}} = \sum_i \text{sigmoid} \left(\boldsymbol{\alpha}'^{[i]}_{\text{fg}} \right) \odot \mathbf{w}_{\text{fg}}^{[i]} \tag{2.5}$$

$$\mathbf{x}_{\text{final}} = \mathbf{x}_{\text{bg}} \cdot (1 - \boldsymbol{\alpha}_{\text{fg}}) + \mathbf{x}_{\text{fg}} \cdot \boldsymbol{\alpha}_{\text{fg}} \tag{2.6}$$

The second line combines foreground images into a single image through an object-wise weighted average. The same is repeated for the transparency channel (third line). Finally, the alpha blending is performed as in Equation 2.1. This formulation is differentiable and can be used for training the model, although the memory requirement may be excessive in high-resolution settings.

2.7.3 *FID evaluation*

The FID metric is very sensitive to aspects such as image resolution, number of images (where a low number results in underestimated FID scores), and the weights of the pretrained Inception network. To be consistent with [Par+19b], we try to follow their methodology as closely as possible. We resize the ground-truth images to the same resolution as the generated ones (256×256), and we keep the two sets aligned, i.e. one generated image per test image. We use the weights of the pretrained InceptionV3 network provided by PyTorch. To make the results in Table 2.1 comparable, we retrained the baseline from [Par+19b] and evaluated the results using our methodology.

2.7.4 *Additional results*

SEMANTIC AND STYLE MANIPULATION. Figure 2.12 and Figure 2.13 show examples of semantic manipulation and style manipulation (either using attributes or text). The last row of Figure 2.13 suggests that our attention mechanism can correctly exploit the contextualized token embeddings produced by BERT. For instance,

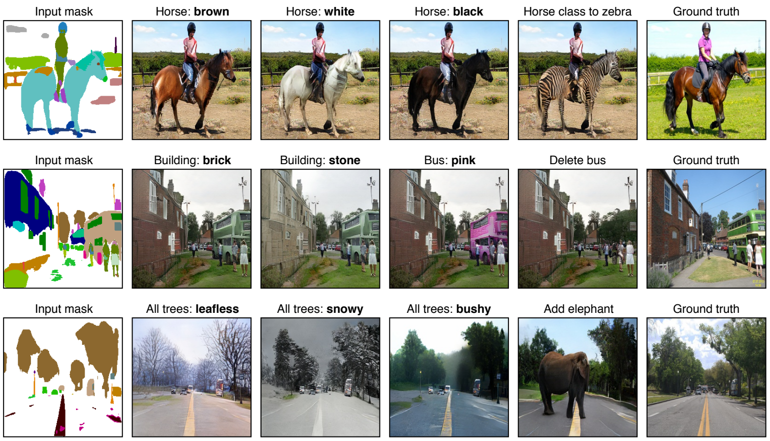

Figure 2.12: Examples of semantic and attribute manipulations (Visual Genome dataset). The images are generated by our two-step model. In the first row, the background is frozen to encourage locality.

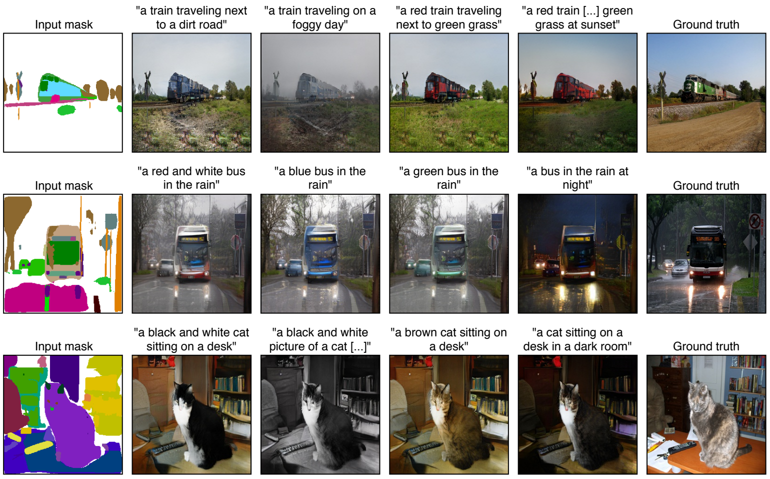
Input mask
"a train traveling next to a dirt road"
"a train traveling on a foggy day"
"a red train traveling next to green grass"
"a red train [...] green grass at sunset"
Ground truth
Input mask
"a red and white bus in the rain"
"a blue bus in the rain"
"a green bus in the rain"
"a bus in the rain at night"
Ground truth
Input mask
"a black and white cat sitting on a desk"
"a black and white picture of a cat [...]"
"a brown cat sitting on a desk"
"a cat sitting on a desk in a dark room"
Ground truth

Figure 2.13: Further examples of style manipulation using text (COCO validation set). It is possible to control the style of individual instances (albeit in a less targeted fashion than attributes) as well as the global style of the image.

the caption "a black and white cat" affects only the cat, while "a black and white picture of a cat" affects the entire scene by generating a black-and-white image.

TWO-STEP MODEL. Figure 2.18 shows additional demos generated by our two-step model on the Visual Genome validation set. In particular, we highlight the decomposition of the background and foreground, and the inputs taken by G_1 and G_2. Since G_2 outputs a soft transparency channel for the alpha blending, it can slightly violate the constraints imposed by the *foreground mask*. This allows it to draw reflections and shadows underneath foreground objects. Furthermore, as we mention in Section 2.3.1, the motivation behind the two-step generator is that it facilitates local changes. In Figure 2.16 we qualitatively compare one-step and two-step generation when manipulations are carried out on the input conditioning information (mask and style). We show that, in the two-step model, local manipulations do not result in global changes of the output. To further enhance locality, the background can be frozen when manipulating the foreground.

Approach	Input	Training set	Test set	FID
Sg2im [JGF18]	GT BBox layout	COCO-train	COCO-val	67.96
Layout2im [Zha+19b]	GT BBox layout	COCO-train	COCO-val	38.14
LostGAN [SW19]	GT BBox layout	COCO-train	COCO-val	34.31
Ours (#3)	Sparse mask	COCO-train	COCO-val	18.57
Ours (#5)	Sparse mask	VG+ (aug.)	COCO-val	**17.98**

Table 2.3: Comparison to layout-based methods. The metric is the FID score [Heu+17]; lower is better. "GT BBox" stands for "ground-truth bounding-box", whereas our approach uses the sparse masks inferred from an object detector as usual.

COMPARISON WITH LAYOUT-BASED METHODS. While in Section 2.4.2 we compare our approach to [Par+19b] under uniform settings, it is also interesting to see how our sparse mask setting compares to approaches that generate images from bounding-box layouts (which are also sparse by nature) [Hon+18; SW19; Zha+19b]. While these methods address a harder task (bounding boxes provide less information than segmentation masks), their applicability has only been demonstrated in low-resolution settings (typically 64×64), which makes them not directly comparable to our higher-resolution setting. To our knowledge, no bounding-box approach can currently generate high-resolution images that have the same visual quality and geometric coherence as mask-based approaches. Nonetheless, for completeness, in Table 2.3 we compare our sparse mask approach to these layout-based methods. We use the models trained on COCO or VG+ with no style input (rows #3 and #4 in Table 2.1, left), and downscale our images to 64×64 before computing the FID score.

QUALITATIVE COMPARISON OF INPUT MASKS. In Figure 2.17, we show qualitative results for different input masks, both in fully supervised and weakly supervised settings. Additionally, in the figure we show qualitative results for the *sparsified* COCO model (ablation I in Table 2.1, right), where we keep only the "thing" classes of COCO. While the outputs produced by the semantic segmentation maps are satisfactory, it is not clear how to manipulate them as they present banding artifacts and jagged edges.

2.7.5 *Attention visualization*

The behavior underlying our attention model can be easily visualized. Our formulation (*sentence-semantic* attention) is particularly suited for visualization tasks because it is tied to the semantic map, and not to feature maps in inner convolutional layers. Therefore, for each class in the semantic map (e.g. *person*, *tree*, empty space), we can observe how the sentence conditions that particular class. Considering that the attention modules have multiple entry points in the generator (one for each normalization block), it is easier to carry out this analysis in the discriminator, where there are only two entry points (in the input layer of each discriminator, since we adopt a multi-scale discriminator). We select the first discriminator for illustration purposes, and show the resulting attention maps in Figure 2.14. The figure shows what parts of the sentence the discriminator is attending to in order to discriminate whether the caption is suitable for the input image.

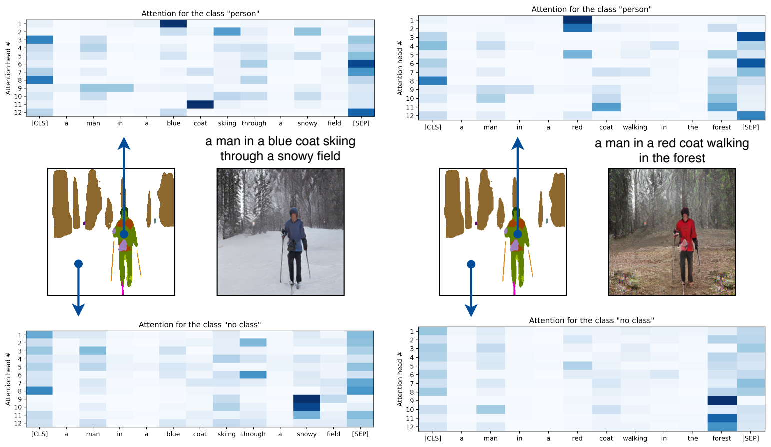

Figure 2.14: Visualization of the attention mechanism in the discriminator for two images generated from the same semantic map, but different captions. An attention map is produced for each class in the semantic map, and each of these consists of 6 or 12 independent attention heads (12 here). In this illustration we only show those corresponding to *person* and *no class* (i.e. blank space) for clarity. `[CLS]` and `[SEP]` are special delimiters indicating respectively the start and end of a sentence. A head paying attention to these can be interpreted as not being triggered by the sentence. In the attention maps, a darker color indicates a higher weight.

2.7.6 *Negative results*

In this section, we discuss some of the unsuccessful ideas that we explored before reaching our current formulation.

TWO-STEP MODEL. Before successfully achieving two-step generation with sparse masks, we tried to implement the same idea using dense COCO segmentation maps. In the areas corresponding to foreground objects, G_1 (the background generator) would always render visible gaps. We tried to regularize the model using *partial convolutions* (a recently-proposed approach for infilling), but this did not have the desired effect. We also experimented with an attention mechanism where foreground areas were masked in G_1. While this was partly successful in filling the gaps, the

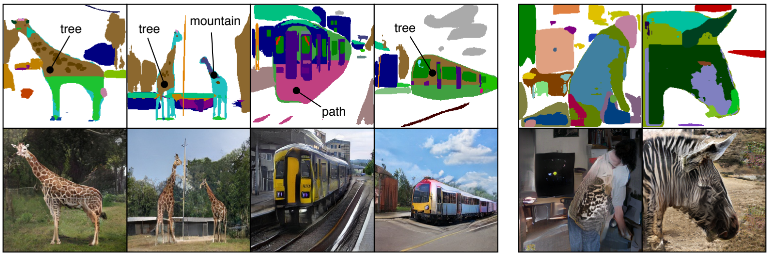

Figure 2.15: Left: in many cases, weakly-supervised training leads to input noise robustness, i.e. artifacts in the input mask are not visible in the generated images. **Right:** some failure cases where the artifacts are visible in the output images.

model was very difficult to train and the final visual quality was considerably lower.

DISCRIMINATOR ARCHITECTURE. We explored various ways of injecting conditional information in the discriminator. While SPADE uses input concatenation, recent GANs conditioned on classes [BDS19; Zha+19a] use *projection discrimination* [MK18]. This idea led to marginally better FID scores, but we observed that the contour of generated objects would stick too close to the input mask, essentially resulting in a "polygonal" appearance. On the other hand, input concatenation allows the model to slightly deviate from the input mask, possibly resulting in a greater robustness to mask noise.

HYPERPARAMETERS. We tried to vary the design of SPADE blocks, e.g. by stacking more layers or using dilated convolutions. These ideas had a detrimental effect on the final result and we decided not to pursue them further.

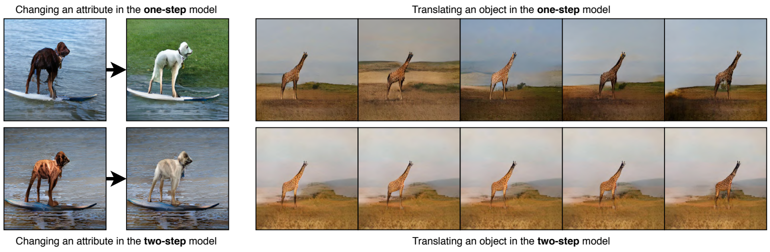
Changing an attribute in the **one-step** model
Translating an object in the **one-step** model
Changing an attribute in the **two-step** model
Translating an object in the **two-step** model

Figure 2.16: In a single-generator model, local changes (e.g. changing the color of the dog to white) affect the scene globally due to learned correlations. The same can be observed when moving an object (e.g. left to right), as the representation space is discontinuous. In the two-step model, we can locally manipulate the background and foreground.

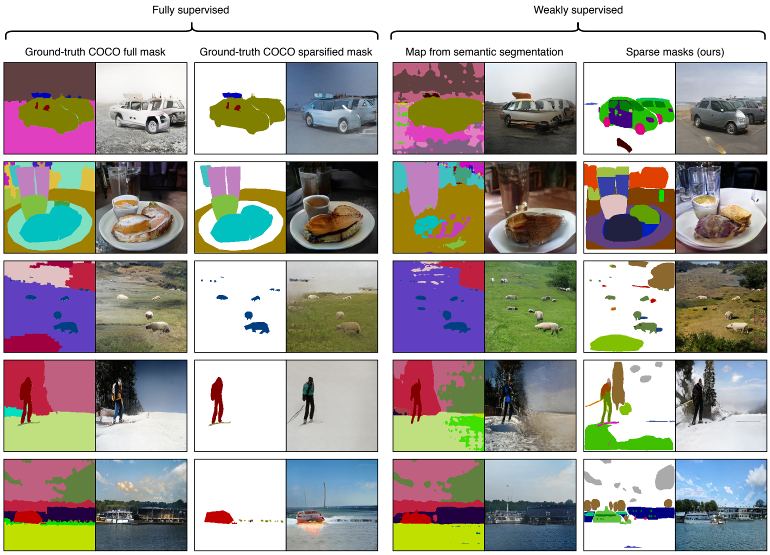
Fully supervised
Weakly supervised
Ground-truth COCO full mask
Ground-truth COCO sparsified mask
Map from semantic segmentation
Sparse masks (ours)

Figure 2.17: Input masks for different approaches, and corresponding generated images. Our sparse masks do not present the typical artifacts of semantic segmentation outputs and are much easier to sketch or manipulate than dense maps.

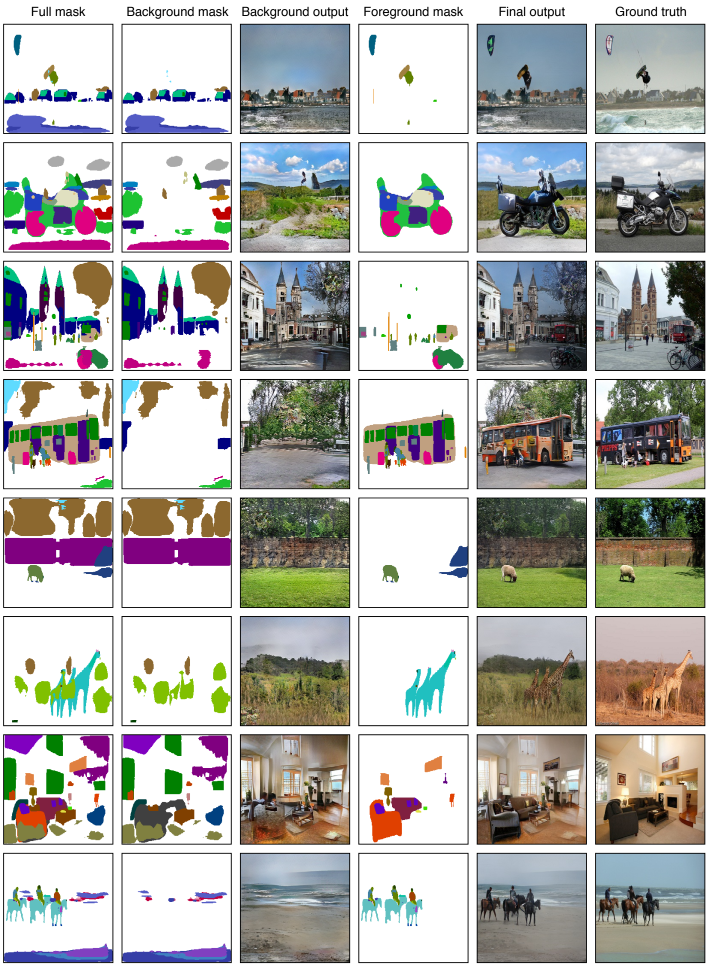

Figure 2.18: Demos generated by our two-step model. In addition to the full input mask, we show its decomposition into *background mask* and *foreground mask* (taken as input in S blocks respectively by G_1 and G_2). Note that G_1 also takes as input the full mask in S_{avg} blocks.

3 CONVOLUTIONAL GENERATION OF TEXTURED 3D MESHES

CHAPTER ABSTRACT. While recent generative models for 2D images achieve impressive visual results, they clearly lack the ability to perform 3D reasoning. This heavily restricts the degree of control over generated objects as well as the possible applications of such models. In this work, we bridge this gap by leveraging recent advances in differentiable rendering. We design a framework that can generate *triangle meshes* and associated high-resolution texture maps, using only 2D supervision from single-view natural images. A key contribution of our work is the encoding of the mesh and texture as 2D representations, which are semantically aligned and can be easily modeled by a 2D convolutional GAN. We demonstrate the efficacy of our method on Pascal3D+ Cars and CUB, both in an unconditional setting and in settings where the model is conditioned on class labels, attributes, and text. Finally, we propose an evaluation methodology that assesses the mesh and texture quality separately.

OPEN SOURCE. Code and pretrained models for this work are available at `https://github.com/dariopavllo/convmesh`.

This chapter is based on our NeurIPS 2020 paper “Convolutional Generation of Textured 3D Meshes” [Pav+20b].

3.1 INTRODUCTION

Thanks to a series of key technical contributions [Heu+17; Kar+18; Miy+18; MK18; Zha+19a], image synthesis models based on the GAN framework [Goo+14] can nowadays achieve photorealistic results. Furthermore, as we highlighted in the previous chapter, an important trend in this field has been to make generative models more controllable and of better use for downstream applications. This includes works that condition generative models on class labels [BDS19; MK18; Zha+19a], text [Li+19; Xu+18; Zha+17; Zha+18a], input images [Iso+17; Zhu+17a], as well as structured scene layouts such as semantic maps [MCS19; Par+19b; PLH20; Wan+18], bounding boxes [Hon+18; SW19; Zha+19b], and scene graphs [JGF18]. In the conclusion to the previous chapter, we also anticipated that, while these approaches achieve impressive visual results, they are all based on architectures that fundamentally ignore the concept of image formation. Real-world images depict 2D projections of 3D objects, and explicitly considering this aspect would lead to better generative models that can provide disentangled control over shape, color, pose, lighting, and can better handle spatial phenomena such as occlusions. A recent trend to account for such effects has been to disentangle factors of variation during the generation process in the hope of making it more interpretable [Kar+18; KLA19; SOL19; Yan+17b]. These approaches potentially learn a hierarchical decomposition of objects, and in some settings (e.g. faces) they can provide some degree of control over pose. However, the pose disentanglement assumptions made by these approaches have been shown to be unrealistic without some form of supervision [Loc+19], and they have not reached the degree of controllability that a native 3D representation would be capable of. More recent efforts have focused on incorporating 3D information into the model architecture, using either rigid transformations in feature space [Ngu+19] or analysis-by-synthesis [Mus+20]. These approaches represent an interesting middle ground between 2D and 3D generators, although their objective remains 2D image synthesis.

In this work, we propose a GAN framework for generating *triangle meshes* and associated textures, using only 2D supervision from single-view natural images. In terms of applications, our approach could greatly facilitate content creation for art, movies, video games, virtual reality, as well as augment the possible downstream applications of generative models. We leverage recent advances in differentiable rendering [Che+19; KUH18; Liu+19b; LB14] to incorporate 3D reasoning into our approach. In particular, we initially adopt a reconstruction framework to estimate meshes through a representation we name *convolutional mesh* which consists of a displacement map that deforms a mesh template in its tangent space. This representation is particularly well-suited for 2D convolutional architectures as both the mesh and its texture share the same topology, and the mesh benefits from the spatial smoothness property of convolutions. We then project natural images onto the UV map (mapping between texture coordinates and mesh vertices) and reduce the problem to a 2D modeling task where the representation is independent of the pose of the object. Finally, we train a 2D convolutional GAN in UV space where inputs to the discriminator are masked in order to deal with occlusions.

Our model generates realistic meshes and can easily scale to high-resolution textures (512×512 and possibly more) owing to the precise semantic alignment between maps in UV space, without requiring progressive growing [Kar+18]. Most importantly, since our model is based exclusively on 2D convolutions, we can easily adapt ideas from state-of-the-art GAN methods for 2D images, and showcase our approach under a wide range of settings: conditional generation from class labels, attributes, text (with and without attention), as well as unconditional generation. We evaluate our approach on Pascal3D+ Cars [XMS14] and CUB Birds [Wah+11], and propose metrics for evaluating FID scores [Heu+17] on meshes and textures separately as well as collectively. In summary, we make the following contributions:

- A novel convolutional mesh representation that is smooth by definition, and alongside the texture, is easy to model using standard 2D convolutional GAN architectures.

- A GAN framework for producing textured 3D meshes from a pose-independent 2D representation. In particular, in a GAN setting, we are the first to demonstrate full generation of textured triangle meshes using 2D supervision from *natural images*, whereas prior attempts have focused on limited settings supervised on synthetic data without a principled texture learning strategy.

- We demonstrate *conditional* generation of 3D meshes from text (with and without an attention mechanism) and show that our model provides disentangled control over shape and appearance.

3.2 RELATED WORK

Deep generative approaches that deal with 3D data typically target either *reconstruction*, where the goal is to predict a 3D representation from one or more images, or *generation*, where the goal is to synthesize 3D objects/scenes from scratch. We review the literature of both tasks as they are relevant to our work.

3D REPRESENTATIONS. Early approaches have focused on reconstructing 3D shapes using 3D supervision. These are typically based on voxel grids [Cho+16; Gir+16; HTM17; TDB17; Wu+17; Yan+17a; Zhu+17b], point clouds [FSG17], or signed distance functions [Par+19a]. However, 3D supervision requires ground-truth 3D shapes, which are usually available in synthetic datasets but not for real-world images. Therefore, a related line of research aims at reconstructing meshes using exclusively 2D supervision from images. Similarly, there has been work on voxel representations [Gwa+17; TEM18; Tul+17; WZ17; Yan+16b; Yan+18] as well as on point clouds [ID18], but these methods require supervision from multiple views which still limits their applicability. More recent approaches lift the requirement of multiple views in order to learn to reconstruct 3D shapes from a single view using a voxel representation [HMR19]. However, these representations tend to be computationally inefficient and do not explicitly support texture maps.

DIFFERENTIABLE RENDERING. Triangle meshes are an established representation in computer graphics, owing to their efficiency as well as flexibility in terms of vertex transformations and texturing. For this reason, they are used in almost every graphics pipeline, ranging from video games to animation. This has motivated a newer line of research where the goal is to predict triangle meshes and texture maps from single images, achieving high-quality visual results [Che+19; Kan+18; KUH18]. The basic building block of these approaches is a *differentiable renderer* (DR), i.e. a renderer that can compute gradients w.r.t. the scene parameters. While early DRs approximate gradients with respect to mesh vertices [KUH18; LB14], newer methods propose fully-differentiable formulations [Che+19; Liu+19b]. Our work is also based on this framework, and specifically we adopt DIB-R [Che+19] because it supports UV mapping.

3D MESH GENERATION. Analogous to reconstruction methods, 3D object generation has also been demonstrated using voxels [Bal+18; Gir+16; SM17; Wu+16; Xie+18; Zhu+18] and point clouds [Ach+18; GWM18], but again, these approaches require some form of 3D supervision which precludes training from natural images, in addition to the texturing limitations highlighted above. As for triangle meshes, [Che+19] propose a GAN framework where 2D images are discriminated after differentiable rendering, but they rely on multiple views of synthetic objects and cannot directly learn textures from images. Instead, they supervise the generator on textures predicted by a separate model previously trained for reconstruction. This intermediate step results in a noticeable loss of quality, and is absent in our approach, which can learn from natural images directly. A parallel work to ours [HTL20] also leverages 2D data to generate 3D meshes, but they adopt a VAE framework [KW14] and only predict face colors instead of UV-mapped textures (i.e. *texture maps*), which limits the visual detail of generated objects. An early work [Rez+16] generates untextured meshes in a variational framework using reinforcement learning to estimate gradients. Our work is based on GANs and can explicitly generate high-resolution texture maps which are then mapped to the mesh via UV mapping, enabling an arbitrary level of detail.

Unlike [Che+19], we learn textures directly from natural images, and introduce a pose-independent representation that reduces the problem to a 2D modeling task. Finally, we are not aware of any prior work that can generate 3D meshes from text.

3.3 METHOD

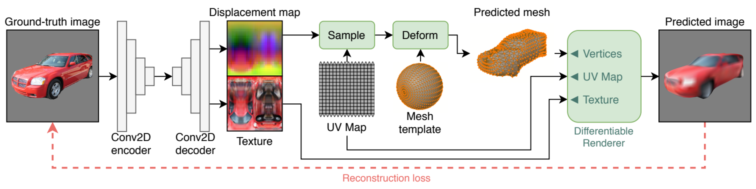

Figure 3.1: Initial mesh reconstruction using our convolutional mesh representation. This step follows a typical autoencoder setup where the goal is to reconstruct the input image after forcing it through a 3D representation and rendering it. RGB colors in the displacement map correspond to XYZ coordinates.

REQUIREMENTS. Our approach has data requirements similar to recent *reconstruction* methods [Che+19; Kan+18]. We require a dataset of single-view natural images, with annotated segmentation masks and pose, or alternatively, keypoints from which the pose can approximately be estimated. If ground-truth masks are not available (as in the ImageNet subset of Pascal3D+), we obtain them from an off-the-shelf segmentation model (we use Mask R-CNN [He+17]), whereas the pose is inferred from the keypoints using structure-from-motion, as was done in [Kan+18]. Our approach does not require ground-truth 3D meshes (i.e. 3D supervision) or multiple views of the same object.

MESH REPRESENTATION. As mentioned, we focus on triangle meshes due to their wide adoption in computer graphics, their flexibility in terms of vertex transformations, and their support for texture mapping. Following [Che+19; Kan+18; KUH18], we use a deformable sphere template with a fixed topology and a static UV map which maps vertices to texture coordinates. Previous work has used fully-connected networks to predict vertex positions, which ignores the topology of the mesh and the spatial correlation between neighboring vertices, essentially treating each vertex as independent. This issue is typically mitigated through regularization, e.g. by combining smoothness [KUH18] and Laplacian [Sor+04] loss penalties. Instead, we propose to regress the mesh through the same deconvolutional network that we use to regress the texture. The output is therefore a *displacement map* (Figure 3.1), which describes how the mesh should be deformed in its tangent space. Importantly, the displacement map and the texture share the same UV map, which ensures that the maps are topologically aligned (e.g. the vertices corresponding to the beak of a bird are co-located with the color of the beak). This detail is crucial for designing a discriminator that can jointly discriminate mesh and texture, that is, not just the mesh and texture separately, but also how well the texture fits the mesh. Furthermore, our mesh representation is smooth by nature since it benefits from the intrinsic spatial correlation of convolutional layers. A second major difference in terms of representation is that our mesh template is a *UV sphere* (2-pole sphere as shown in Figure 3.1), whereas prior work has used ico-spheres. While the latter exhibits a more regular mesh, it cannot be UV-mapped without gaps or arbitrary distortions that make the representation space discontinuous. On the other hand, except for the singularities at the two poles, a UV sphere presents a bijective mapping between vertices and texture coordinates, has a well-defined tangent map, and the circular boundary conditions along the x axis can be neatly incorporated in the model architecture using circular convolutions. Denoting the mesh template as $\mathbf{V}$ (an $N \times 3$ matrix with N vertices described by their xyz coordinates, where each vertex is indexed by i), the final position of the i-th vertex is computed as $\mathbf{v_i} + \mathbf{R_i} \boldsymbol{\Delta}_{\mathbf{v_i}}$ where $\boldsymbol{\Delta}_{\mathbf{v_i}}$ is the output of the model after sampling the displacement map, and $\mathbf{R_i}$ is a precomputed rotation matrix that describes the local normal, tangent, and bitangent of the vertex.

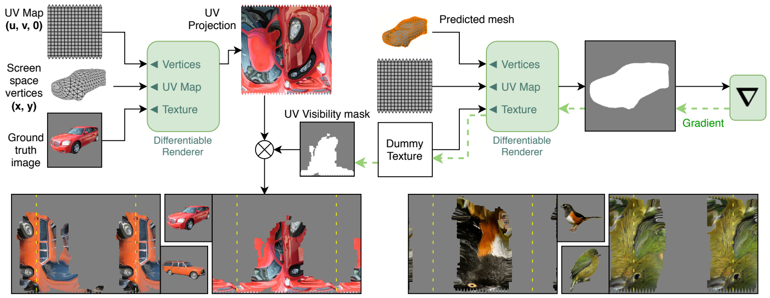

Figure 3.2: Projection of ground-truth images onto the UV map, producing *pseudo*-ground-truth textures. The bottom row shows additional examples (Pascal3D+ Cars on the left and CUB Birds on the right). The yellow dashed lines represent the boundaries of the textures, which have been extended to highlight the circular boundary conditions along the x axis.

3.3.1 *Pose-independent dataset*

Our approach initially augments the dataset by estimating a mesh for each training image (Figure 3.1). The images are then converted into a pose-independent representation (Figure 3.2), which can be finally modeled by a 2D GAN (Figure 3.3).

MESH ESTIMATION. This is a typical reconstruction task where the goal is to reconstruct the mesh from an input image. Our approach is loosely based on [Kan+18], but simplified since we are not interested in performing inference on unseen images. Our formulation can be regarded as a fitting process where we only keep the predicted meshes and discard the model weights/predicted textures. As depicted in Figure 3.1, the input image is fed to a convolutional encoder, compressed into a vector representation, and decoded through a convolutional decoder which jointly outputs a texture and a displacement map. The predicted texture is only used to facilitate the learning process and produce more semantically-aligned meshes, and is discarded afterwards. The mesh template is deformed as described by the displacement map, and the final result is rendered using a differentiable renderer. The model is trained to minimize the mean squared error (MSE) between the rendered image and the input image. While this generally leads to blurry textures, it does not represent an issue in our case as these textures are discarded. Since we are not interested in performing inference, we do not predict pose or keypoints, nor do we use texture flows or perceptual losses to improve predicted textures. For the camera model, we adopt a weak-perspective model where the pose of an image is described by a rotation $\mathbf{q} \in \mathbb{R}^4$ (a unit quaternion), a scale $s \in \mathbb{R}$, and a screen-space translation $\mathbf{t} \in \mathbb{R}^2$. For Pascal3D+, we augment the projection model with a perspective correction term z_0 (further details in the Appendix 3.7.2). While these are initially estimated using structure-from-motion on keypoints [Kan+18], we allow the optimizer to fine-tune s, $\mathbf{t}$, and z_0 (if used), i.e. we additionally optimize with respect to the dataset parameters[1]. This leads to a better alignment between rendered masks and ground-truth masks, facilitating the next step. As a side note, we mention that inaccurate camera assumptions (e.g. using an orthographic model on photographs that exhibit significant perspective distortion) would most likely not affect the mask alignment or convergence of the model, but might lead to distorted meshes. Nonetheless, our method can work with *any* projection model as long as the camera parameters are known or can be estimated.

2D DISCRIMINATION. The most obvious way to adapt the aforementioned reconstruction framework is to train a GAN where the generator **G** produces a 3D mesh and the discriminator **D** discriminates its 2D projection after differentiable rendering, as in [Che+19]. However, we found this strategy to lead to training instabilities due to the discrepancies of the representation being used by **G** and **D** (which are respectively pose-independent and pose-dependent). A further complication we observed is an aliasing effect in the gradient from the differentiable renderer. Successful 2D GAN models typically use complementary architectures for **G** and **D** (e.g. both convolutional), which motivates our next idea.

1 In an inference model this would be detrimental to generalization, but our goal is mesh fitting.

POSE-INDEPENDENT REPRESENTATION. We instead propose to project ground-truth images onto the UV map of the mesh template, thus reducing the generative model to a 2D GAN that can be trained with existing convolutional techniques. The construction of this representation is depicted in Figure 3.2, and can be regarded as a form of *inverse rendering*. We treat our previous mesh estimates as if they were texture coordinates, i.e. $(x, y) \rightarrow (u, v)$ (z is dropped), the UV map becomes the mesh to render (a flat surface with $z = 0$), and the texture is the ground-truth image. The result is the projection of the natural image onto the UV map. However, as can be seen in the figure, this process erroneously projects occluded vertices (the back of the car in the example), which should ideally be masked out as visual information associated with them is not available in the 2D image. We therefore mask the projection using a binary *visibility mask*, which describes what parts of the mesh are visible in UV space. The mask is obtained by rendering the mesh using a dummy texture (e.g. all white) and computing its gradient with respect to the texture (we provide implementation details in the Appendix 3.7.2). Only *texels* (pixels of the texture) that contribute to the final image (i.e. visible ones) will have non-zero gradients, therefore we obtain the visibility mask by thresholding these gradients. The final result is a pose-independent dataset of *pseudo*-ground-truth textures (because they are partially occluded). A useful consequence of this representation is that samples become semantically aligned, i.e. the positions of parts such as wheels or eyes are aligned across all images.

3.3.2 *GAN framework*

We directly use the estimated displacement maps and the *pseudo*-ground-truth textures to train a convolutional GAN, with the only obstacle that "real" textures are masked, while generated textures should ideally be complete. This can be easily dealt with by masking "fake" images before they reach **D**: as shown in Figure 3.3, we multiply the batch of generated textures with a random sample of visibility masks from the training set. This strategy avoids a distribution mismatch between fake and real

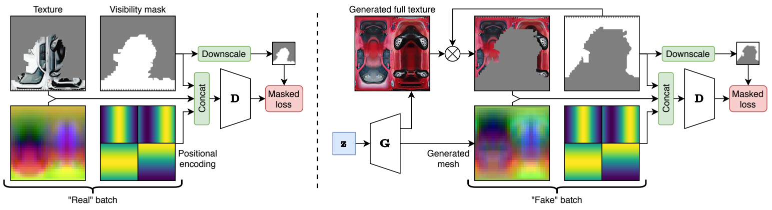

Figure 3.3: GAN training strategy. **Left:** discrimination of a "real" batch with *pseudo*-ground-truth textures. **Right:** discrimination of a "fake" batch after masking to reflect the "real" batch distribution.

textures in **D**, while acting as a gradient gate such that only gradients from the visible areas will reach the generator. Being convolutional and agnostic to the visibility mask, **G** will always generate the full texture.

In terms of architecture, the generator is a convolutional model that outputs both mesh (displacement map) and texture. Mesh and texture can have different resolutions – in our experiments we use 32×32 for the mesh and up to 512×512 for the texture. To support this, the generator branches out at some point and outputs mesh and texture through two different heads (this is also done in the *mesh estimation* model). The discriminator adopts a multi-scale architecture [Wan+18] (i.e. multiple discriminators trained jointly) and a patch-based loss [Iso+17] which is masked using the visibility mask scaled to the same resolution as the last feature map. The smallest discriminator discriminates both the mesh and the texture, which is downscaled to the same resolution as the mesh (32×32). It focuses on global aspects of the texture, while discriminating the mesh and how well it fits the texture. The higher-resolution discriminators are only texture discriminators (one for experiments at 256×256, up to two for experiments at 512×512 in which the intermediate one discriminates at 128×128).

D takes as input the displacement map (mesh), the masked texture, the visibility mask, as well as a *soft* positional encoding of the UV map: inspired by attention-based NLP methods that propose a similar idea [Vas+17] (this is unrelated to our attention method for text conditioning), we add a sinusoidal encoding to the input that gives convolutions a sense of where they are within the image. For a coordinate space $u, v \in [-1, 1]$, we add four channels $\cos(\pi u)$, $\sin(\pi u)$, $\cos(\pi(v/2 + 0.5))$, $\sin(\pi(v/2 + 0.5))$ such that the encoding smoothly wraps around the u (horizontal) axis and is discontinuous along the v (vertical) axis. Giving an absolute sense of position to the model is important as the semantics of the *texels* depend on their absolute position within the UV map, and we show this quantitatively in the ablation study (Section 3.4.3). Finally, the GAN framework allows us to condition the generator on a wide range of inputs: class labels, classes combined with attributes, and text. For the latter, we investigate both an attention mechanism and a method based on a simple fixed-length sentence embedding. We explain how these are implemented in Section 3.4.2.

3.4 EXPERIMENTS

3.4.1 *Evaluation and datasets*

Perceptual metrics such as the Fréchet Inception Distance (FID) [Heu+17] are widely employed for evaluating 2D GANs, as they have been shown to correlate well with human judgment [Zha+18b]. Although we focus on a different task, the FID still appears as a natural choice as it can easily be adapted to our task. Therefore, we suggest to evaluate FID scores on rendered 2D projections of generated meshes. To this end, we sample random poses (i.e. viewpoints) from the training set as we do not want the evaluation metric to be affected by our choice of poses. Moreover, this strategy allows us to evaluate mesh and texture separately: in addition to the *Full FID*, we report the *Texture FID*, where we use meshes estimated using the differentiable renderer instead of generated ones, and the *Mesh FID*, where we replace generated textures with *pseudo*-ground-truth ones. In the latter, using real poses ensures that we render the visible part of the *pseudo*-ground-truth texture, and occlusions are minimized. While we mostly rely on the *Full FID* to discuss our results, the individual ones represent a useful tool for analyzing how the model responds to variations of the architecture. Generated samples are rendered at 299×299 (the native resolution of Inception), and ground-truth images are also scaled to this resolution. In the Appendix 3.7.2, we provide some visualizations that give more insight into the conceptual differences between these metrics.

We evaluate our method on two datasets with annotated keypoints, and use the implementation of [Kan+18] to estimate the pose from keypoints using structure-from-motion.

CUB-200-2011 [Wah+11]. We use the train/test split of [Kan+18], which consists of ≈6k training images and ≈5.7k test images. Each image has an annotated class label (out of 200 classes) and 10 captions which we use for text conditioning. Using poses and labels (where applicable) from the training set, we evaluate the FID on test images, although we observe that the FID is almost identical between the two sets.

PASCAL3D+ (P3D) [XMS14]. We use the *cars* subset, which is the most abundant class in this dataset. Images are part of a low-resolution set (Pascal set) and a newer, high-resolution set from ImageNet [Den+09]. While we use the same split as [Kan+18] to train our mesh estimation model, the GAN is trained only on the ImageNet subset (≈ 4.7k usable images) since we noticed that the images in the Pascal set are too small for practical purposes. We infer segmentation masks using Mask R-CNN [He+17] since they are not available. The test split of [Kan+18] does not contain any ImageNet images, therefore we evaluate FID scores on training images [2], motivated by our previous observation on CUB. Finally, to demonstrate conditional generation on this dataset, we collected new annotations for the class (11 shape categories) and color (11 attributes) of each car (details and statistics in the Appendix 3.7.2).

3.4.2 *Implementation details*

MESH ESTIMATION. The model (Figure 3.1) is trained for 1000 epochs using Adam [KB14], with an initial learning rate of 10^{-4} halved every 250 epochs. We train with a batch size of 50 on a single Pascal GPU, which requires ≈12 hours. We use DIB-R

2 Given the already small size of the dataset, we decided not to split it further.

[Che+19] for differentiable rendering due to its support for texture mapping and its relatively low overhead. To stabilize training we adopt a warm-up phase, described in the Appendix 3.7.2. In the same section we also describe how we augment the camera model for Pascal3D+. Finally, the detailed architecture of the network can be found in the Appendix 3.7.1.

GAN ARCHITECTURE. Since our method is reduced to a 2D generation task, we adopt recent ideas from the 2D convolutional GAN literature. Our generator follows a ResNet architecture where the latent code **z** (64D, normally distributed) is injected in the input layer as well as after every convolutional layer through *conditional batch normalization*. Following [BDS19; Zha+19a], we use spectral normalization [Miy+18] in both **G** and **D**, but **D** does not employ further normalization, e.g. we tried instance normalization but found it detrimental. We adopt a hinge loss objective (patch-based and masked as described in Section 3.3), and train for 600 epochs with a constant learning rate of 0.0001 for **G** and 0.0004 for **D** (two time-scale update rule [Heu+17]). We update **D** twice per **G** update, and evaluate the model on a running average of **G**'s weights ($\beta = 0.999$) as proposed by [BDS19; Kar+18; KLA19; Yaz+19]. Detailed aspects about the architecture of our GAN can be found in the Appendix 3.7.1. Training the 512×512 models requires ≈ 20 hours on 4 Pascal GPUs, while the 256×256 models require roughly the same time on a single GPU. For all experiments, we use a total batch size of 32 and we employ synchronized batch normalization across multiple GPUs.

CONDITIONAL GENERATION. In settings conditioned on class labels, we simply concatenate a learnable 64D embedding to **z**, and use *projection discrimination* [MK18] in the last feature map of **D**. In the P3D experiment with attributes (i.e. colors), we split the embedding into a 32D shape embedding and a 32D color embedding. For text conditioning, we first encode the sentence using the pretrained RNN encoder from [Xu+18] (a bidirectional LSTM), and compare *(i)* a simple method where we concatenate the sentence embedding to **z** as before, *(ii)* an attention mechanism operating on all hidden states of the RNN. For the latter we add a single attention layer in **G** right before the mesh/texture branching, operating at 16×16 resolution. Likewise, we modify projection discrimination in **D** to apply attention on the last feature map. Detailed schemes can be found in the Appendix 3.7.1.

REPRESENTATION. Since the UV map of a UV sphere has circular boundary conditions along the horizontal axis, convolutional layers in the discriminator use circular padding horizontally and regular zero-padding vertically. Furthermore, in both the mesh estimation model and the GAN generator, we enforce reflectional symmetry across the x axis as done in [Kan+18], which has the dual benefit of improving quality and halving the computational cost to output a mesh/texture. In this case, convolutions use reflection padding horizontally instead of circular padding. Finally, to deal with the singularities of the UV sphere, the vertex displacements of the north and south pole are respectively taken to be the average of the top and bottom rows of the displacement map.

3.4.3 *Results*

QUANTITATIVE RESULTS. We report our main results in Table 3.1 (left). For CUB, we compare settings where the model is conditioned on class labels, captions (using the attention model), and no conditioning at all. For P3D, we compare unconditional

			FID (truncated σ)				FID (untruncated)		
Dataset	Tex. res.	Conditioning	σ	Full	Tex.	Mesh	Full	Tex.	Mesh
CUB	512x512	None	1	41.56	45.26	18.36	56.27	50.12	25.85
		Class	0.25	33.63	28.68	19.49	**41.33**	**30.60**	23.28
		Text	0.5	**18.45**	**22.91**	**12.05**	42.66	38.95	**21.18**
	256x256	Class	0.25	33.55	30.92	19.39	42.61	33.31	23.37
P3D	512x512	None	1	43.09	32.70	28.62	74.74	47.99	43.23
		Class	0.75	**27.73**	22.17	**23.76**	**49.56**	**29.98**	**34.10**
		Class+Color	0.5	31.30	**21.70**	27.75	52.55	30.29	36.32
	256x256	Class+Color	0.5	39.09	26.52	36.73	63.63	36.56	46.37

	FID	Δ
Baseline (class)	**33.63**	0
No pos. encoding	43.71	+10.08
Same G/D updates	41.55	+7.92
InstanceNorm	36.38	+2.75
Text with attention	**18.45**	0
No attention	22.14	+3.69

Table 3.1: Left: FID scores grouped by dataset, texture resolution, and conditioning, both in truncated and untruncated settings. Lower is better; bold = best. **Right:** Ablation study on CUB with a 512×512 texture resolution. We report truncated FID scores in the truncated setting.

generation and conditional generation on class labels (i.e. car shapes) as well as classes *plus* colors. We evaluate the FID every 20 epochs and report the model with the best *Full FID* in the table. Since there is no prior work to which we can compare under our setting, we set baselines on these two datasets. As proposed by [BDS19], we found it useful to sample latent codes $\mathbf{z}$ from a truncated Gaussian (only at inference time), which trades off sample diversity for quality and considerably improves FID scores. For each setting we specify the optimal truncation σ, but we also report scores in an untruncated setting as these are more directly comparable. As expected, conditional GANs result in better scores than their unconditional counterparts (with the text model being the best), but we generally observe that our approach is stable under all settings.

ABLATION STUDY. We conduct an ablation study in Table 3.1 (right). The results in the top section are relative to the 512×512 CUB model conditioned on classes (truncated FID). Removing the positional encoding from the discriminator leads to a significant FID degradation (+10.08), suggesting that giving convolutions a sense of absolute position in UV space is an important aspect of our approach. Likewise, updating **D** as often as **G** has a significant negative impact (+7.92) compared to two **D** updates per **G** update. Using instance normalization in **D** also leads to a slight degradation (+2.75), but beyond that we observe that, while training appears to converge faster initially, it rapidly becomes unstable. In the bottom section of the table, we compare the text attention model (baseline) to a model where a fixed-length sentence vector is simply concatenated to $\mathbf{z}$ (as in the other conditional models). The results show that the model effectively exploits the attention mechanism with the added benefit of being more interpretable.

QUALITATIVE RESULTS. Figure 3.4 shows a few generated meshes rendered from multiple views, as well as the corresponding textures. While results on CUB are generally of high visual quality, we observe that the back of the cars in P3D present some artifacts. After further investigation, we found that the dataset is very imbalanced, with only 10–20% of the images depicting

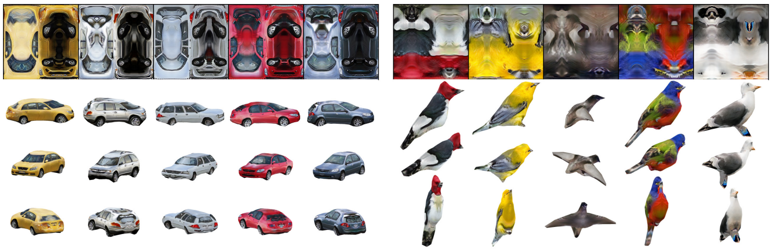

Figure 3.4: Qualitative results on P3D (left, conditioned on class *and* color) and CUB (right, conditioned on class). Each object is rendered from 3 views, and the top row depicts the unwrapped texture.

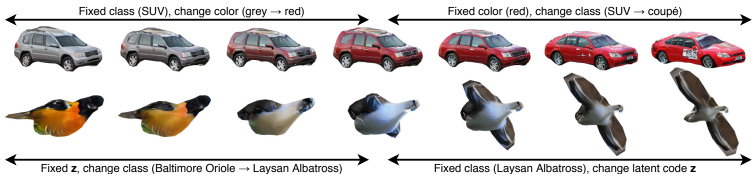
Fixed class (SUV), change color (grey → red)
Fixed color (red), change class (SUV → coupé)
Fixed **z**, change class (Baltimore Oriole → Laysan Albatross)
Fixed class (Laysan Albatross), change latent code **z**

Figure 3.5: Interpolation over conditioning inputs, which highlights that our model learns a structured latent space where factors of variation of both shape and texture are relatively disentangled.

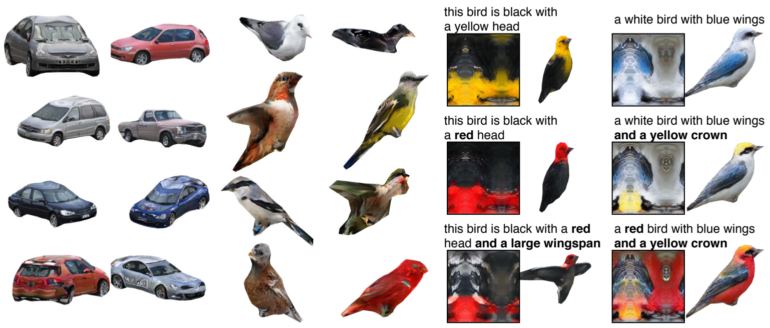
this bird is black with a yellow head
a white bird with blue wings
this bird is black with a **red** head
a white bird with blue wings **and a yellow crown**
this bird is black with a **red** head **and a large wingspan**
a **red** bird with blue wings **and a yellow crown**

Figure 3.6: Left: generated meshes rendered from random views on P3D and CUB, both conditioned on a random set of classes. **Right:** generation from text on CUB, which allows for fine control over both texture and mesh. We modify captions incrementally, where changes are highlighted in bold.

the back of the car and the majority depicting the front. Therefore, this issue could in principle be mitigated with more training data. In Figure 3.5 we show that the latent space of our models is structured. We interpolate over different factors of variation using spherical interpolation and observe that they are relatively disentangled, enabling isolated control over shape, color, and style in addition to the pose disentanglement guaranteed by the 3D representation itself. Figure 3.6 shows results rendered from random viewpoints (the scenario on which we evaluate FID scores) as well as generation conditioned on text, which enables precise control over both shape and appearance. Finally, in the Appendix 3.7.3 we show a wider range of qualitative results.

2D GAN BASELINE. An interesting baseline for generating 3D meshes is to first train a 2D GAN using a state-of-the-art architecture (e.g. StyleGAN [KLA19]), and then run a 3D mesh reconstruction model on top of the generated 2D images. First, we note that such a baseline would not exhibit the properties of a true 3D representation, such as pose disentangled from shape. Additionally, the reconstruction model would have difficulties dealing with occlusions, since it can only reliably infer information visible in the 2D image. To substantiate our observation, we investigate this baseline empirically: we train the 3D reconstruction model of [Kan+18] for 1000 epochs on CUB training images with an empty background (our setting). Evaluating this model on *training images* achieves an FID of 85.8 on reconstructions rendered from ground-truth viewpoints, which is already worse than all of our baselines and establishes a lower bound. If we run the model on CUB images produced by StyleGAN [KLA19] and evaluate the FID on renderings from sampled viewpoints, the FID further degrades to 101.9.

ATTENTION MECHANISM. Similar to other attention-based GANs conditioned on text [Xu+18], our attention mechanism can be easily visualized. Interestingly, since the attention is applied to our pose-independent representation in UV space and not on flat 2D images, our attention maps can be visualized both in UV space and on 2D renderings, as we show in Figure 3.7. Furthermore, our process is more interpretable and semantically meaningful. For instance, prompts that refer to a specific part of the object (e.g. "yellow crown", "red cheeks") activate the same area within the UV map. Most importantly, these correspondences are learned in an unsupervised fashion and are aligned among different images owing to our pose-independent representation.

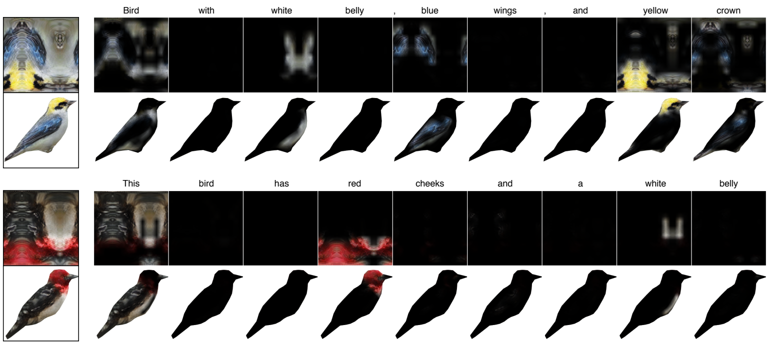

Figure 3.7: Visualization of the attention mechanism on our CUB model conditioned on text. The attention maps are visualized in UV space (first row) as well as on the rendered mesh (second row). In this particular bidirectional LSTM model, active tokens typically correspond to the adjectives that precede body parts. The first and last tokens are also active because the sentence representation does not comprise explicit sentence delimiters.

3.5 SUMMARY

We proposed a GAN-based framework for generating 3D meshes. Our approach can generate triangle meshes and corresponding texture maps at high resolution (512×512 and possibly more), requiring only 2D supervision from single-view natural images. We evaluate our method on Pascal3D+ Cars and CUB Birds, and showcase it under a wide range of conditional settings to demonstrate its high level of adaptation. Nonetheless, we have only scratched the surface of what can be done with this framework. In the next section, we identify potential directions to make this framework adoptable in broader settings.

3.6 FUTURE DEVELOPMENTS

POTENTIAL IMPROVEMENTS. Our approach can be enriched by employing different forms of supervision (e.g. semi-supervision by combining 3D supervision from synthetic datasets with 2D supervision from natural images) as well as incorporating more conditional information that would allow the model to disentangle further aspects of variation (e.g. lighting). As the datasets used in this work are considered small (around 5k images), it would also be interesting to experiment with larger datasets and scale the approach to full-scene generation. We believe a viable option is to decompose background and foreground generation as in [PLH20], and use a 3D mesh generator for foreground objects.

RECENT LITERATURE. Our proposed framework was used as a building block for recent approaches in the 3D reconstruction literature. For instance, [Bha+21] tackles the single-view 3D reconstruction task using cycle consistency, whereas [Dun+22] focuses on texture learning. [Zha+22] proposes a GAN inversion approach where a reference 2D image is transformed into a textured 3D mesh via optimization. All of these approaches adopt our proposed *convolutional mesh* representation. As for the generation of 3D objects from text, we mention CLIP-Mesh [Kha+22] (based on triangle meshes, similarly to our work), DreamFields [Jai+22] (based on NeRF), and DreamFusion [Poo+22] (based on NeRF and diffusion models). The latter approaches share the commonality of utilizing information derived from a pretrained contrastive visual-language model, CLIP [Rad+21]. Furthermore, in the case of unconditional generation, recent techniques have shifted their focus from triangle meshes to NeRFs. For a reference of these future developments, we refer the reader to Section 4.6.

FOLLOW-UP WORK. The main limitation of the presented approach is the requirement for annotated camera poses, which are typically only available in small datasets such as P3D and CUB. A natural future research direction is to investigate whether it is possible to relax this requirement and apply our method to real-world (*"in-the-wild"*) datasets such as ImageNet. We cover this follow-up direction in our next chapter (Chapter 4).

3.7 APPENDIX

3.7.1 *Detailed architecture*

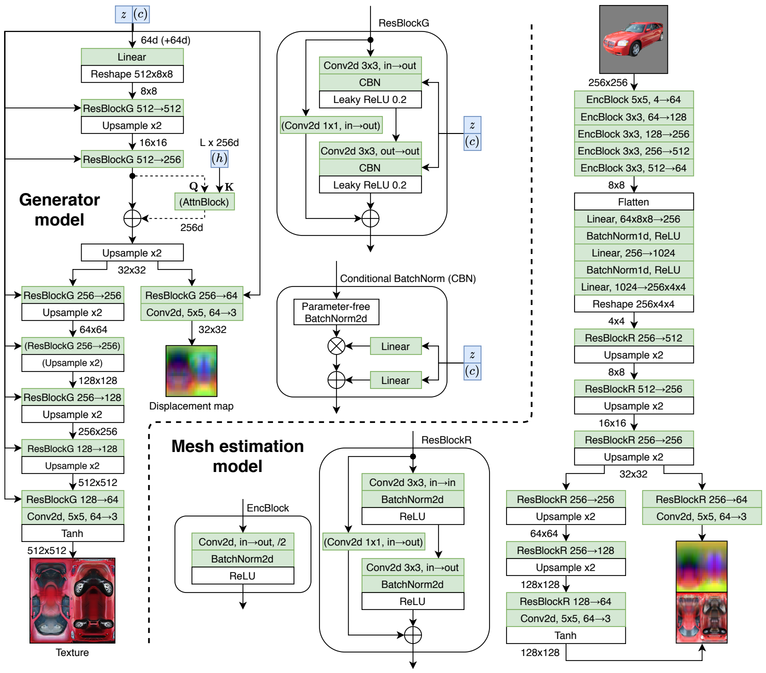

Figure 3.8: Generator architecture (left) and mesh estimation model (right). Green blocks comprise learnable parameters, whereas white ones are parameter-free. Dashed lines and blocks in parentheses represent optional connections which depend on the specific setting. We indicate the feature map resolution in a given position next to arrows (e.g. 128×128). 512 → 256 denotes "512 input channels, 256 output channels". "/2" in convolutional layers denotes "stride 2"; it is one if not indicated.

GENERATOR. Figure 3.8 (left) shows the detailed architecture of the generator of our GAN. As mentioned in Section 3.4.2, the random vector **z** is fed to every conditional batch normalization (CBN) layer as well as to the input layer, which matches the strategy adopted by many state-of-the-art GANs for 2D image generation [BDS19; MK18; Zha+19a]. A CBN layer consists of a parameter-free batch normalization (i.e. without a learned affine transformation) followed by a gain γ and bias $\boldsymbol{\beta}$ conditioned upon $\mathbf{z}$ via a learned linear layer. In settings conditioned on class labels, we concatenate a learned embedding $\mathbf{c}$ to $\mathbf{z}$, which is shared among all layers. The network follows a ResNet architecture where feature maps are progressively upsampled using nearest-neighbor interpolation after each residual block `ResBlockG`. This block consists of two convolutional layers wrapped by a skip-connection. If the number of input channels differs from the number of output channels, the skip-connection is learned. To accommodate for the varying output resolutions for mesh and texture, the generator branches out at 32×32 resolution. While the figure shows the architecture for 512×512 textures, to generate textures at 256×256 we simply remove one 256→256 `ResBlockG` block. For presentation purposes, we report square resolutions (e.g. 512×512), but in practice we only need to generate half of the feature map (e.g. 256×512) since we enforce symmetry across the x axis as mentioned in Section 3.4.2. The output textures and displacement maps are then simply padded with their reflection. On the other hand, the discriminator always observes full textures as *pseudo*-ground-truth textures are asymmetric.

ATTENTION MECHANISM. If the model is conditioned on text using an attention mechanism, we add an attention block right before the texture/mesh branch, so that the module influences both mesh and texture. We adopt a dot-product formulation similar to [Xu+18], in which the attention weights are computed as $\text{softmax}(\mathbf{Q}\mathbf{K}^T)$. The queries $\mathbf{Q}$ correspond to the flattened convolutional feature map from the generator, and the keys $\mathbf{K}$ are obtained by passing each RNN hidden state $\mathbf{h}_l$ ($l \in \{1..L\}$, where L is the sentence length) through a learned linear layer. The RNN is the pretrained bidirectional LSTM encoder from [Xu+18], and hidden states are 256-dimensional.

MESH ESTIMATION MODEL. Figure 3.8 (right) shows the detailed architecture of the mesh estimation model that we use for differentiable rendering (the first step of our algorithm as described in Figure 3.1). The natural image is concatenated to the segmentation mask (3+1 channels) and fed to a convolutional encoder. The representation is then flattened to a dense representation and passed through a series of linear layers. Finally, it is passed through a ResNet decoder whose architecture resembles that of the GAN generator. Since we are not interested in producing high-quality textures in this step as they are discarded, the texture resolution in this model is only 128×128, which results in faster training.

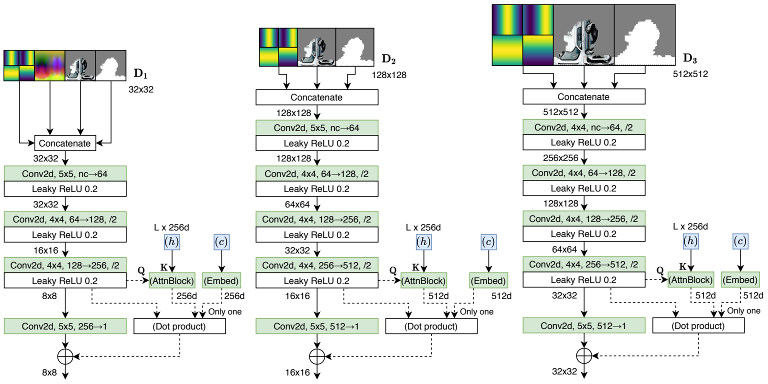

Figure 3.9: Multi-scale discriminator architecture for our biggest model (512×512 texture resolution). Only $\mathbf{D_1}$ discriminates the mesh, while $\mathbf{D_2}$ and $\mathbf{D_3}$ are texture discriminators. Dashed lines describe the optional connections for *projection discrimination* [MK18] in conditional settings, where feature maps are combined *either* with a learned embedding (for settings conditioned on classes or attributes) or with the values of an attention block (for settings conditioned on text), not both.

DISCRIMINATOR. The architecture of our multi-scale discriminator is depicted in Figure 3.9. In the most complex setting (used by the CUB model at 512×512), textures are discriminated at three scales: 32×32 ($\mathbf{D_1}$), 128×128 ($\mathbf{D_2}$), and 512×512 ($\mathbf{D_3}$). The smallest discriminator $\mathbf{D_1}$ is also a mesh discriminator. Following the general strategy of patch-based discriminators [Iso+17], our discriminators are relatively simple as they only consist of a series of spectrally-normalized convolutional layers. GANs have been shown to produce checkerboard artifacts [ODO16] depending on the choice of kernel sizes and strides in the discriminator. While humans do not perceive these to be particularly severe in images, checkerboard artifacts in the *displacement map* must be avoided as they might lead to noticeable mesh distortions. The generator already uses upsampling instead of transposed convolutions (which mitigates this issue), but we also carefully design the discriminator such that the kernel size of convolutions is divisible by the stride, ensuring that the gradient norms are uniform across pixels (see [ODO16] for further details). To this end, we use 5×5 convolutions in layers with stride 1, and 4×4 convolutions in layers with stride 2. $\mathbf{D_1}$ consists of 4 layers and has a relatively small receptive field. We explored a varying number of layers and different strides, but they always led to worse results. $\mathbf{D_2}$ and $\mathbf{D_3}$ consist of 5 layers and are identical except for the stride of the first layer, which is 1 for $\mathbf{D_2}$ and 2 for $\mathbf{D_3}$. In the experiments with a texture resolution of 256×256, we only use $\mathbf{D_1}$ and $\mathbf{D_2}$, where the latter directly discriminates at 256×256. For Pascal3D+ at 512×512, we found a small empirical advantage in dropping $\mathbf{D_2}$ and doubling the weight of $\mathbf{D_3}$'s loss. Finally, to incorporate conditional information, we use *projection discrimination* [MK18], in which we compute a pixel-wise dot product between the last feature map and a learned class embedding (a vector), and add it to the output. If the model is conditioned on text, we replace the class embedding with the output of an attention block. Each discriminator learns its own set of weights for the embeddings or the attention block.

MODEL COMPLEXITY. Table 3.2 summarizes the complexity of our models in different settings, expressed as the number of learnable parameters. Since the semantic alignment of our pose-independent representation facilitates the modeling task, our approach can successfully work with relatively small models (≈10M generator parameters). We also found it beneficial to adopt simple discriminators compared to the generator (which is more powerful).

Model	Resolution	**G**	**D$_1$**	**D$_2$**	**D$_3$**	**D** Total	RNN
Unconditional	512x512	11.58M	0.68M	2.77M	2.77M	6.22M	-
	256x256	10.33M			-	3.45M	-
CUB conditional	512x512	13.06M	0.73M	2.88M	2.88M	6.49M	-
	256x256	11.75M			-	3.61M	-
CUB text	512x512	11.64M	0.75M	2.91M	2.91M	6.57M	2.08M
P3D conditional	512x512	13.05M	0.69M	2.79M	2.79M	6.27M	-
	256x256	11.74M			-	3.48M	-

Table 3.2: Number of learnable parameters for different variants of our model.

3.7.2 *Additional implementation details*

MESH ESTIMATION. Since our *convolutional mesh* representation already encourages meshes to be smooth, our reconstruction model requires less regularization than similar frameworks based on fully-connected networks. We only found it beneficial to regularize the model with a smoothness loss $\mathcal{L}_{\text{flat}}$ [KUH18] at a very low strength $\alpha = 0.00005$, and no Laplacian regularization [Sor+04] (unlike [Che+19; Kan+18; KUH18] which all use this form of regularization). $\mathcal{L}_{\text{flat}}$ encourages the normals of neighboring faces to have similar directions, and is defined as follows:

$$\mathcal{L}_{\text{flat}} = \alpha \frac{1}{|E|} \sum_{i,j \in E} (1 - \cos \theta_{ij})^2,$$

where E is the set of all edges and $\cos \theta_{ij}$ is the cosine similarity between the normals of the faces i and j. In practice this is implemented by computing the dot product between the two normals.

We additionally observe that the initialization strategy of this model as well as early training iterations have a significant impact on the final result. Bad configurations such as self-intersecting meshes or vertices outside the camera frustum can cause the model to get stuck in bad local minima from which it cannot easily recover. This is especially the case for typical Gaussian initialization schemes in neural networks, which cause the mesh to start in an already self-intersected state for a spherical mesh template with radius 1 (our case). To ensure convergence and generate smooth meshes without self-intersections, we found it helpful to *(i)* zero-initialize the final layer of the mesh branch, which ensures that the first iteration starts with a smooth sphere, and *(ii)* adopt a warm-up phase where $\mathcal{L}_{\text{flat}}$ starts at a moderate strength $\alpha = 0.0005$ and linearly decays for 100 iterations, settling at the low-strength value mentioned above. In the GAN generator we also zero-initialize the final layer of the mesh head, but $\mathcal{L}_{\text{flat}}$ only uses a fixed $\alpha = 0.0001$ and no warm-up.

In Section 3.3, we mention that our camera projection model is a weak-perspective model. This model is a good approximation for photographs shot with high levels of zoom or that depict small objects, which is the case for birds (CUB dataset). However, we observed that the weak-perspective assumption is not a good fit for Pascal3D+, since most images are shot from a close range and present a significant degree of perspective distortion due to cars having elongated shapes. Therefore, for Pascal3D+ we augment the camera model with a learnable perspective correction term z_0, without however advancing to a full perspective model as we do not have enough information. z_0 is a scalar that describes the distance from the camera to the center of the object, and assumes that the object is centered. The x, y coordinates of each vertex in camera space are then multiplied with a factor $(z_0 + z/2)/(z_0 - z/2)$, where z is the depth of the vertex. Note that, as z_0 approaches infinity, the factor approaches 1 and the camera model reverts back to a weak-perspective model. This term is learned for every image in the dataset and is parameterized as $z_0 = 1 + e^w$ (w is a learnable parameter), which ensures *(i)* positivity, and *(ii)* that the transformed vertices lie inside the camera frame. While this aspect is not central to our approach, we found it helpful as it can slightly improve qualitative results even with approximate estimates.

CONSTRUCTION OF THE POSE-INDEPENDENT REPRESENTATION. In Section 3.3 we mention that we use the gradient from the differentiable renderer to produce the UV visibility mask which is used for masking projected textures. In practice, deep learning frameworks do not compute full Jacobians but only gradients of scalars (i.e. Jacobian-vector multiplication). However,

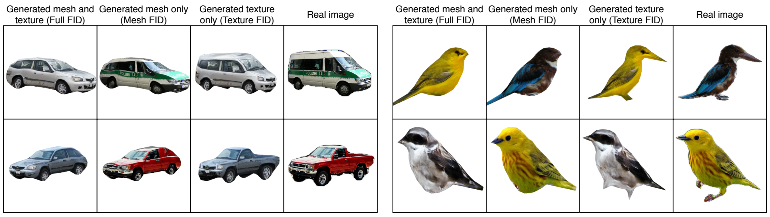
Generated mesh and texture (Full FID)
Generated mesh only (Mesh FID)
Generated texture only (Texture FID)
Real image
Generated mesh and texture (Full FID)
Generated mesh only (Mesh FID)
Generated texture only (Texture FID)
Real image

Figure 3.10: Examples of images on which we compute FID scores. Images are rendered from the viewpoint corresponding to *Real image* (a randomly-selected image from the training set). In the *Mesh FID* scenario, we render the generated mesh using the *pseudo*-ground-truth texture from the real image. In the *Texture FID*, the "real" mesh is textured using the generated texture. In the *Full FID* and *Mesh FID* of the top-left van we can observe that the silhouette of the mesh looks fine but straight lines and stripes present a "wobbling" effect caused by the underlying mesh, while in the *Texture FID* (which does not use generated meshes) the lines appear more straight.

the Jacobian of the rendering operation w.r.t. the texture has a structure such that it is zero for all texels that are not visible in the rendered image (i.e. are occluded) and non-zero elsewhere[3]. Based on this observation, it suffices to compute the average or sum of rendered pixels to reduce the image to a scalar which can then be differentiated with respect to a dummy texture. The same result can also be achieved by computing a Jacobian-vector multiplication with a vector of ones, which is what we do in our implementation.

FID EVALUATION. To give more context to Section 3.4.1, where we introduce our evaluation methodology, in Figure 3.10 we show some actual examples of rendered images on which we compute FID scores. The *Full FID* (our main metric) is computed on generated meshes coupled with generated textures, and evaluates the generation quality as a whole. However, it is also interesting to propose variations of this metric that can evaluate mesh and texture quality separately. Therefore, in the *Mesh FID* we use the

3 This property holds for DIB-R [Che+19] but may not hold for all differentiable renderers.

pseudo-ground-truth texture from the image corresponding to the random viewpoint we choose for rendering, which makes the evaluation independent of generated textures. Likewise, in the *Texture FID* we use meshes estimated using the differentiable renderer instead of the ones generated by our GAN. In all experiments, we generate as many images as there are in the set we compare to, since the FID is sensitive to the number of generated images. Finally, to evaluate text conditioning on CUB, we sample a random caption (out of 10 captions) for each image we generate.

PASCAL3D+ ANNOTATIONS. To demonstrate conditional generation on P3D, we collected shape and color annotations for the ImageNet subset of this dataset (i.e. the one we use to train our GAN). Although ImageNet images are already identified by their *synsets*, we found these to be unreliable and opted instead for collecting our own annotations. The set of labels and corresponding frequencies are summarized in Table 3.3. For consistency, all labels were collected by one annotator. Some categories (e.g. *F1*, *convertible*, and *oldtimer*) comprise a very low number of samples, which leads to unsatisfactory results on these classes in conditional settings. Nonetheless, this issue can be mitigated by collecting more data. Finally, although the ImageNet subset consists of ≈5.5k images, only 4.7k are usable as some are filtered out by the structure-from-motion routine of [Kan+18] due to unreliable pose estimates.

Class	Sedan	Hatchback	SUV	Station wagon	Van	Pickup	Coupé	City	F1	Convertible	Oldtimer	Total
# images	1137	851	814	691	674	649	295	193	119	39	13	5475
Color	**Gray**	**Black**	**Red**	**White**	**Blue**	**Green**	**Yellow**	**Orange**	**Brown**	**Purple**	**Pink**	**Total**
# images	1534	863	833	832	697	252	231	126	52	30	25	5475

Table 3.3: Relevant statistics for the P3D annotations we collected.

3.7.3 *Additional results*

DISENTANGLEMENT. Compared to a generative model for 2D images, a 3D generative model naturally disentangles pose and appearance. Furthermore, the use of *triangle meshes* with UV-

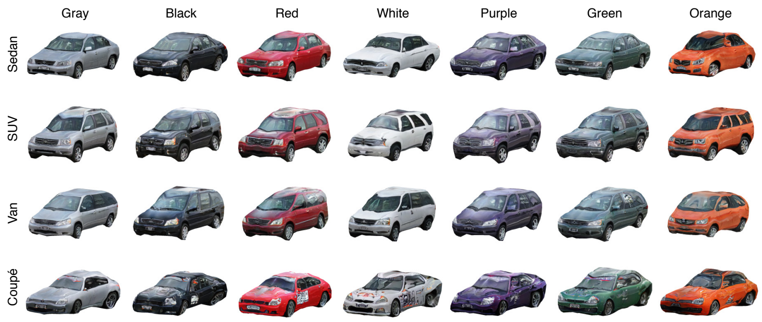

Figure 3.11: Generation on P3D with one varying factor at a time (color and shape) and a fixed random vector **z**. As can be seen, representations are relatively disentangled. In the bottom row, the class *coupé* is often associated with race cars, which causes stickers to appear on the body.

mapped textures ensures that shape is disentangled from color, essentially allowing for texture transfer between meshes (we use this property for our *Texture FID* and *Mesh FID* evaluation, as shown in Figure 3.10). Another interesting observation is that conditional models enable further disentanglement of aspects of variation. For instance, on P3D we can control shape and color separately as shown in Figure 3.11, and the latent space is structured enough to allow for interpolation of these aspects (Figure 3.5). In this setting, the random vector **z** can be used to control the style of the object.

ADDITIONAL QUALITATIVE RESULTS. In Figure 3.12, we show additional qualitative results grouped by type of conditioning. Our approach successfully generates meshes in both conditional and unconditional settings. In the figure, we additionally show untextured (i.e. *wireframe*) meshes, which highlights the smoothness of our convolutional mesh representation.

DEMO VIDEO. The supplementary material in [Pav+20b] includes a video where we show more results, including latent space interpolation, disentangled generation, generation from text, and attention maps on text-conditioned models.

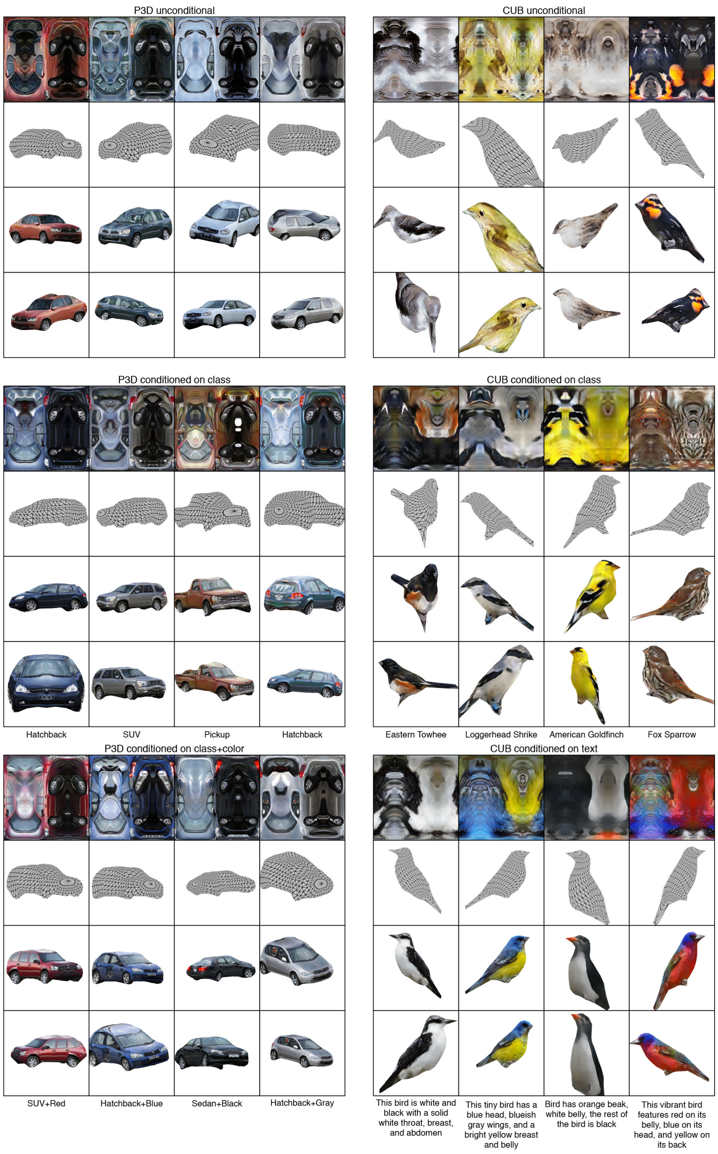
P3D unconditional
CUB unconditional
P3D conditioned on class
CUB conditioned on class
Hatchback
SUV
Pickup
Hatchback
Eastern Towhee
Loggerhead Shrike
American Goldfinch
Fox Sparrow
P3D conditioned on class+color
CUB conditioned on text
SUV+Red
Hatchback+Blue
Sedan+Black
Hatchback+Gray
This bird is white and black with a solid white throat, breast, and abdomen
This tiny bird has a blue head, blueish gray wings, and a bright yellow breast and belly
Bird has orange beak, white belly, the rest of the bird is black
This vibrant bird features red on its belly, blue on its head, and yellow on its back

Figure 3.12: Qualitative results for all settings, on both P3D (left) and CUB (right). First row = texture; second row = wireframe mesh; third and fourth rows = textured object from two random views.

4 LEARNING 3D MESH GENERATORS FROM REAL-WORLD IMAGES

CHAPTER ABSTRACT. Recent advances in differentiable rendering have sparked an interest in learning generative models of textured 3D meshes from image collections. These models natively disentangle pose and appearance, enable downstream applications in computer graphics, and improve the ability of generative models to understand the concept of image formation. Although there has been prior work on learning such models from collections of 2D images, these approaches require a delicate pose estimation step that exploits annotated keypoints, thereby restricting their applicability to a few specific datasets. In this work, we propose a GAN framework for generating textured triangle meshes without relying on such annotations. We show that the performance of our approach is on par with prior work that relies on ground-truth keypoints, and more importantly, we demonstrate the generality of our method by setting new baselines on a larger set of categories from ImageNet – for which keypoints are not available – without any class-specific hyperparameter tuning.

OPEN SOURCE. Code and models for this work are available at https://github.com/dariopavllo/textured-3d-gan.

This chapter is based on our ICCV 2021 paper "Learning Generative Models of Textured 3D Meshes from Real-World Images" [Pav+21].

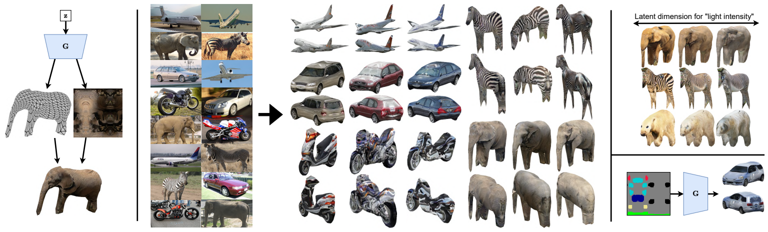

Figure 4.1: Left: as in the previous chapter, we focus on GANs, where our generator outputs a *triangle mesh* and a UV-mapped texture. **Middle:** our method learns to synthesize textured 3D meshes given a real-world collection of 2D images. **Top-right:** we showcase a setting where we train a single model to generate all classes. This model successfully disentangles some factors of the 3D environment (e.g. lighting/shadows) without explicit supervision. **Bottom-right:** we also demonstrate a conditional model that generates meshes from 3D semantic layouts.

4.1 INTRODUCTION

In the previous chapter, we hinted at the fact that most of the recent literature in the field of generative models focuses on 2D image generation [BDS19; KLA19; Kar+20b; MK18; Zha+19a], which largely ignores the fact that real-world images depict 2D projections of 3D objects. We also highlighted the advantages of generative models that can explicitly reason about 3D representations, namely a fully disentangled control over shape, appearance, pose, as well as an explicit representation of spatial phenomena such as occlusions. As a result, we mentioned that a growing line of research investigates textured 3D mesh generators, and we proposed our *convolutional mesh generation* approach [Pav+20b] in Chapter 3. These approaches (including ours) are trained with 2D supervision from a collection of 2D images, but require camera poses to be known in advance as learning a joint distribution over shapes, textures, and cameras is particularly difficult. Usually, the required camera poses are estimated from keypoint annotations using a factorization algorithm such as *structure-from-motion* (SfM) [MC09]. These keypoint annotations are, however, very expensive to obtain and are usually only available on a few datasets.

In this work, we propose a new approach for learning generative models of textured triangle meshes with minimal data assumptions. Most notably, we do not require keypoint annotations, which are often not available in real-world datasets. Instead, we solely rely on: *(i)* a single mesh template (optionally, a set of templates) for each image category, which is used to bootstrap the pose estimation process, and *(ii)* a pretrained semi-supervised object detector, which we modify to infer semantic part segmentations on 2D images. These, in turn, are used to augment the initial mesh templates with a 3D semantic layout that allows us to refine pose estimates and resolve potential ambiguities.

First, we evaluate our approach on benchmark datasets for this task (Pascal3D+ [Liu+19a] and CUB [Wah+11]), for which keypoints are available, and show that our approach is quantitatively on par with the state-of-the-art [Pav+20b] as demonstrated by FID metrics [Heu+17], even though we do not use keypoints. Secondly, we train a 3D generative model on a larger set of categories from ImageNet [Den+09], where we set new baselines without any class-specific hyperparameter tuning. To our knowledge, no prior works have so far succeeded in training textured mesh generators on real-world datasets, as they focus either on synthetic data or on simple datasets where poses/keypoints are available. We also show that we can learn a *single* generator for all classes (as opposed to different models for each class, as done in previous work [Che+19; HTL20; Pav+20b]) and notice the emergence of interesting disentanglement properties (e.g. color, lighting, style), similar to what is observed on large-scale 2D image generators [BDS19].

Finally, we quantitatively evaluate the pose estimation performance of our method under varying assumptions (one or more mesh templates; with or without semantic information), and showcase a proof-of-concept where 3D meshes are generated from sketches of semantic maps (*semantic mesh generation*), following the paradigm of image-to-image translation. In summary, our main contributions are as follows:

- We introduce a new approach to 3D mesh generation that does not require keypoint annotations, enabling its use on a wider range of datasets as well as new image categories.
- We showcase 3D generative models in novel settings, including learning a *single* 3D generator for all categories, and *conditional generation* from semantic mesh layouts. In addition, we provide a preliminary analysis of the disentanglement properties learned by these models.
- We propose a comprehensive 3D pose estimation framework that combines the merits of template-based approaches and semantic-based approaches. We further extend this framework by explicitly resolving pose ambiguities and by adding support to multiple templates.

4.2 RELATED WORK

DIFFERENTIABLE 3D REPRESENTATIONS. For a reference of how various 3D representations have been used in both reconstruction and synthesis techniques, we refer the reader to the related work section in the previous chapter (Section 3.2). Since we tackle a similar task in this chapter, the referenced discussion is also applicable to this work. Furthermore, as in [Pav+20b], this work also adopts triangle meshes due to their convenient properties: *(i)* their widespread use in computer graphics, movies, video games; *(ii)* their support for UV texture mapping, which decouples shape and color; *(iii)* the ability of efficiently manipulating and transforming vertices via linear algebra. We use *DIB-R* [Che+19] as our differentiable renderer of choice throughout this work, motivated by its support for UV maps.

KEYPOINT-FREE POSE ESTIMATION. The use of keypoints for pose estimation is limiting due to the lack of publicly available data and an expensive annotation process. Thus, a growing line of research focuses on inferring poses via semi-supervised objectives. To our knowledge, no approach has so far focused on *generation*, but there have been some successful attempts in the *reconstruction* literature. The initial pose estimation step of our framework is most closely related to [GKM20; Li+20], which both propose approaches for 3D mesh *reconstruction* without keypoints. In terms of assumptions, [GKM20] require a canonical mesh template for each category. Object poses are estimated by fitting the mesh template to the silhouette of the object and by concurrently optimizing multiple camera hypotheses (which helps to deal with the large amount of bad local minima). [Li+20] do not require a mesh template, but instead use object part segmentations from a self-supervised model (*SCOPS* [Hun+19]) to infer a 3D semantic template that is matched to the reference segmented image. Based on early experiments, we were unable to individually generalize these methods to *generation* (our goal), which we found to have a lower tolerance to errors due to the intrinsic difficulty in training GANs. Instead, we here successfully combine both ideas (mesh templates *and* semantics) and extend the overall framework with *(i)* the optional support for multiple mesh templates, *(ii)* a principled ambiguity resolution step that leverages part semantics to resolve conflicts among camera hypotheses with similar reprojection errors. We additionally adopt a more general object-part segmentation framework. Namely, we use a pre-trained semi-supervised object detector [Hu+18] modified to produce fine-grained semantic templates (Figure 4.2), as opposed to *SCOPS* (used in [Li+20]), which we found to require class-specific hyperparameter tuning.

MESH GENERATION. Again, we refer the reader to the related work section in the previous chapter (Section 3.2) for a discussion of various techniques for mesh synthesis. Throughout this work, we adopt our previously-proposed *convolutional mesh generation* approach [Pav+20b], since it represents a comprehensive framework that can model both meshes and UV-mapped textures, and can be applied to natural images (albeit with keypoint annotations). More precisely, we borrow the GAN architecture but substantially rework the supervision strategy to relax the keypoint requirement.

4.3 METHOD

DATA REQUIREMENTS. As usual in both the *reconstruction* [GKM20; Kan+18; KUH18; Li+20] and *generation* [Che+19; HF19;

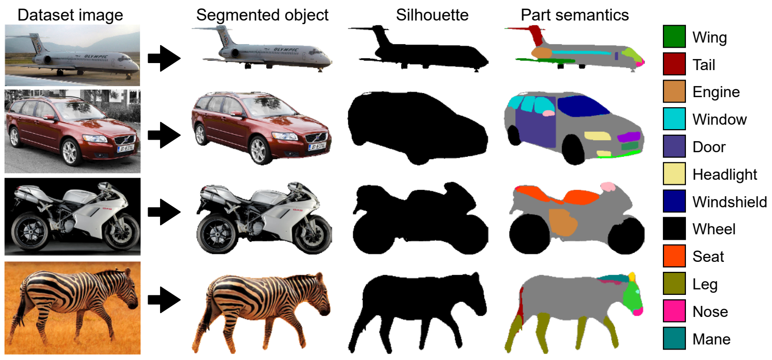

Figure 4.2: The dataset is initially processed into a clean collection of images with associated object masks and semantic part segmentations. This is done via off-the-shelf models and does not involve any additional data collection. Semantic classes have a precise meaning and are shared between different categories (e.g. wheels appear in both cars and motorbikes).

Pav+20b] literature, we require a dataset of segmented images. Segmentation masks (a.k.a. *silhouettes*) can easily be obtained through an off-the-shelf model (we use PointRend [Kir+20] pretrained on COCO [Lin+14]; details in Appendix 4.7.1). Whereas prior approaches require keypoint annotations for every image, we only require an untextured mesh template for each image category, which can be downloaded freely from the web. Optionally, our framework supports multiple mesh templates per category, a choice we explicitly evaluate in Section 4.4.2. We note that pose estimation from silhouettes alone can in some cases be ambiguous, and therefore we rely on object part semantics to resolve these ambiguities wherever possible. To this end, we use the semi-supervised, large-vocabulary object detector from [Hu+18; PLH20] to infer part segmentations on all images. We adopt their pretrained model as-is, without further training or fine-tuning, but post-process its output as described in Appendix 4.7.1.

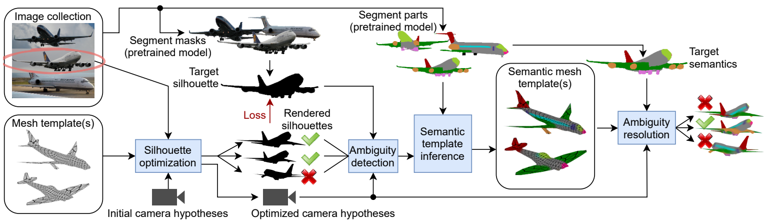
Image collection
Segment masks (pretrained model)
Segment parts (pretrained model)
Target silhouette
Target semantics
Loss
Rendered silhouettes
Semantic mesh template(s)
Mesh template(s)
Silhouette optimization
Ambiguity detection
Semantic template inference
Ambiguity resolution
Initial camera hypotheses
Optimized camera hypotheses

Figure 4.3: Schematic overview of the proposed pose estimation pipeline. The left side shows our data requirements (a collection of 2D images and one or more untextured mesh templates). For clarity, we only show the optimization process for the circled airplane, although the *semantic template inference* step involves multiple instances.

DATASET PREPARATION. Since our goal is to apply our method to real-world data that has not been manually cleaned or annotated – unlike the commonly-used datasets CUB [Wah+11] and Pascal3D+ [Liu+19a] – we attempt to automatically detect and remove images that do not satisfy some quality criteria. In particular, objects should not be *(i)* too small, *(ii)* truncated, or *(iii)* occluded by other objects (implementation details in Appendix 4.7.1). This filtering step is tuned for high precision and low recall, as we empirically found that it is beneficial to give more importance to the former. All our experiments and evaluations (Section 4.4) are performed on the dataset that results from this step. Finally, sample images and corresponding silhouettes/part segmentations can be seen in Figure 4.2, which also highlights how some semantic parts are shared across image categories.

4.3.1 *Pose estimation framework*

OVERVIEW. Most *reconstruction* and *generation* approaches require some form of pose estimation to initialize the learning process. Jointly learning a distribution over camera poses and shapes/textures is extremely challenging and might return a trivial solution that does not entail any 3D reasoning. Therefore, our approach also requires a pose estimation step in order to allow

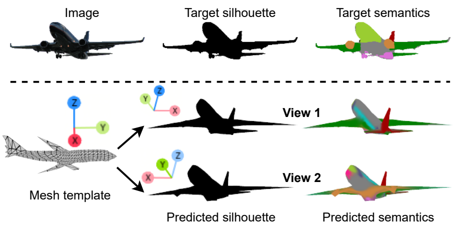

Figure 4.4: Ambiguity arising from opposite poses. The two camera hypotheses produce almost-identical silhouettes which closely approximate the target, but describe opposite viewpoints. This particular example would initially be rejected by our ambiguity detection test, but it would then be resolved once semantics are available.

the learning process to converge to meaningful solutions. Our proposed pose estimation pipeline is summarized in Figure 4.3: starting from a set of randomly-initialized camera hypotheses for each object instance, we render the mesh template(s) using a differentiable renderer and optimize the camera parameters so that the rendered silhouette matches the target silhouette of the object. At this point, no semantics, colors, or textures are involved, so the approach can lead to naturally ambiguous poses (see Figure 4.4, for an example). We then introduce a novel ambiguity detection step to select only images whose inferred pose is unambiguous, and use the most confident ones to infer a 3D *semantic template*, effectively augmenting the initial mesh templates with semantic information (more examples of such templates can be seen in Figure 4.6). Afterwards, the process is repeated – this time leveraging semantic information – to resolve ambiguities and possibly reinstate images that were previously discarded. The final output is a camera pose for each object as well as a confidence score that can be used to trade off recall (number of available images) for precision (similarity to ground-truth poses). In the following, we describe each step in detail.

SILHOUETTE OPTIMIZATION. The first step is a fitting procedure applied separately to each image. Following [GKM20], who observe that optimizing multiple camera hypotheses with differing initializations is necessary to avoid local minima, we initialize a set of N_c camera hypotheses for each image as described in Appendix 4.7.1. Our camera projection model is the *augmented* weak-perspective model of [Pav+20b], which comprises a rotation $\mathbf{q} \in \mathbb{R}^4$ (a unit quaternion), a scale $s \in \mathbb{R}$, a screen-space translation $\mathbf{t} \in \mathbb{R}^2$, and a perspective correction term $z_0 \in \mathbb{R}$ which is used to approximate perspective distortion for close objects. We minimize the mean squared error (MSE) in pixel space between the rendered silhouette $\mathcal{R}(\cdot)$ and the target silhouette $\mathbf{x}$:

$$\min_{\mathbf{q},\mathbf{t},s,z0} \left\| \mathcal{R}(\mathbf{V}_{\text{tpl}}, \mathbf{F}_{\text{tpl}};\ \mathbf{q}, \mathbf{t}, s, z_0) - \mathbf{x} \right\|^2, \tag{4.1}$$

where $\mathcal{R}$ is the differentiable rendering operation, $\mathbf{V}_{\text{tpl}}$ represents the (fixed) mesh template vertices, and $\mathbf{F}_{\text{tpl}}$ represents the mesh faces. Each camera hypothesis is optimized using a variant of Adam [KB14] that implements full-matrix preconditioning as opposed to a diagonal one. Given the small number of learnable parameters (8 for each hypothesis), the $\mathcal{O}(n^3)$ cost of inverting the preconditioning matrix is negligible compared to the convergence speed-up. We provide hyperparameters and more details about this choice in the Appendix 4.7.1. In the settings where we use multiple mesh templates N_t, we simply replicate each initial camera hypothesis N_t times so that the total number of hypotheses to optimize is $N_c \cdot N_t$. In this case, we compensate for the increase in optimization time by periodically pruning the worst camera hypotheses during optimization. Additionally, in all settings, we start by rendering at a low image resolution and progressively increase the resolution over time, which further speeds up the process. We describe how both strategies are implemented in the Appendix 4.7.1.

SCORING AND AMBIGUITY DETECTION. All symmetric objects (i.e. many natural and man-made objects) present *ambiguous poses*: opposite viewpoints that produce the same silhouette after 2D projection (Figure 4.4). Similar ambiguities can also arise as a result of noisy segmentation masks, inappropriate mesh templates, or camera hypotheses that converge to bad local minima. Since wrong pose estimates have a significant negative impact on the rest of the pipeline, this motivates the design of an *ambiguity detection* step. Ideally, we would like to accept pose estimates that are both *confident* – using the *intersection-over-union* (IoU) between the rendered/target silhouettes as a proxy measure – and *unambiguous*, i.e. no two camera hypotheses with high IoU should describe significantly different poses. We formalize this as follows: we first score each hypothesis k as $(\mathbf{v}_{\text{conf}})_k = (\text{softmax}(\mathbf{v}_{\text{IoU}} \,/\, \tau))_k$, where $\tau = 0.01$ is a temperature coefficient that gives similar weights to IoU values that are close to the maximum, and low weights to IoU values that are significantly lower than the maximum. Next, we require that highly-confident poses (as measured by $\mathbf{v}_{\text{conf}}$) should describe similar rotations. We therefore construct a pairwise distance matrix $\mathbf{D}$ of shape $N_c \times N_c$, where each entry d_{ij} describes the geodesic distance between the rotation of the i-th hypothesis and the rotation of the j-th hypothesis. Entries are then weighted by $\mathbf{v}_{\text{conf}}$ across both rows and columns, and are finally summed up, yielding a scalar agreement score v_{agr} for each image:

$$\mathbf{D} = 1 - (\mathbf{Q}^T\mathbf{Q})^{\circ 2}, \quad v_{\text{agr}} = \left\| \mathbf{D} \odot (\mathbf{v}_{\text{conf}}\, \mathbf{v}_{\text{conf}}^T) \right\|_1 \tag{4.2}$$

where $\mathbf{Q}$ is a $4 \times N_c$ matrix of unit quaternions (one per hypothesis), $\mathbf{M}^{\circ 2}$ denotes the element-wise square, and $\odot$ denotes the element-wise product.

The agreement score v_{agr} can be roughly interpreted as follows: a score of 0 (best) implies that all *confident* camera hypotheses describe the same rotation (they agree with each other). A score of 0.5 describes two poses that are rotated by 180 degrees from one another[1]. Empirically, we established that images with $v_{\text{agr}} > 0.3$ should be rejected.

SEMANTIC TEMPLATE INFERENCE. Simply discarding ambiguous images might significantly reduce the size and diversity of the training set. Instead, we propose to resolve the ambiguous

1 For example, consider a $\mathbf{D}$ matrix of size 2×2, where entries along the main diagonal are 0, and 1 elsewhere.

cases. While this is hardly possible when we only have access to silhouettes, it becomes almost trivial once *semantics* are available (Figure 4.4). A similar idea was proposed in [Li+20], who infer a 3D semantic template by averaging instances that are close to a predetermined exemplar (usually an object observed from the left or right side). Yet, our formulation does not require an exemplar but directly leverages samples that have passed the ambiguity detection test. Since our data requirements assume that mesh templates are untextured, our first step in this regard aims at augmenting each mesh template with part semantics. Among images that have passed the ambiguity test ($v_{\text{agr}} < 0.3$), we select the camera hypothesis with the highest IoU. For each mesh template, the semantic template is computed using the top $N_{\text{top}} = 100$ images assigned to that template, as measured by the IoU. Then, we frame this step as an optimization problem where the goal is to learn vertex colors while keeping the camera poses fixed, minimizing the MSE between the rendered (colored) mesh template and the 2D image semantics, averaged among the top samples:

$$\min_{\mathbf{C}_{\text{tpl}}} \frac{1}{N_{\text{top}}} \sum_i \left\| \mathcal{R}(\mathbf{V}_{\text{tpl}}, \mathbf{F}_{\text{tpl}}, \mathbf{C}_{\text{tpl}};\ \mathbf{q_i}, \mathbf{t_i}, s_i, z_{0i}) - \mathbf{C_i} \right\|^2, \tag{4.3}$$

where $\mathbf{C}_{\text{tpl}}$ represents the vertex colors of the template and $\mathbf{C_i}$ denotes the 2D semantic image. For convenience, we represent $\mathbf{C}_{\text{tpl}}$ as a $K \times N_v$ matrix, where N_v is the number of vertices and K is the number of semantic classes (color channels, not necessarily limited to 3), and $\mathbf{C_i}$ is a $K \times N_{pix}$ matrix, where N_{pix} is the number of image pixels. In the Appendix 4.7.1, we derive an efficient closed-form solution that requires only a single pass through the dataset. Examples of the resulting semantic templates are shown in Figure 4.6.

AMBIGUITY RESOLUTION. In the last step of our pose estimation pipeline, we repeat the scoring process described in "*Scoring and ambiguity detection*" with the purpose of resolving ambiguities. Instead of evaluating the scores on the IoU, however, we use the mean intersection-over-union (mIoU) averaged across semantic classes. Since our inferred semantic templates are continuous, we

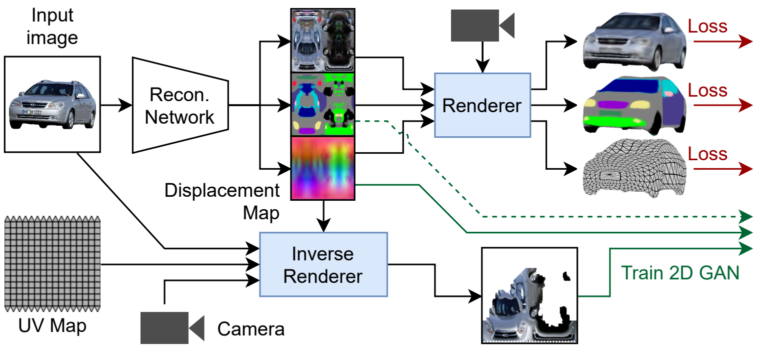

Figure 4.5: Generation framework using the *convolutional mesh* representation. Images are fed into a network trained to reconstruct meshes (parameterized as 2D displacement maps), given camera poses. The meshes are then used to project natural images onto the UV map. Finally, the resulting partial textures, displacement maps, and (optionally) predicted semantics are used to train a 2D convolutional GAN in UV space.

adopt a smooth generalization of the mIoU (weighted Jaccard similarity) in place of the discrete version:

$$\text{mIoU} = \frac{1}{K}\sum_k \frac{\|\min(\hat{\mathbf{C}}_\mathbf{k}, \mathbf{C}_\mathbf{k})\|_1}{\|\max(\hat{\mathbf{C}}_\mathbf{k}, \mathbf{C}_\mathbf{k})\|_1}, \tag{4.4}$$

where $\hat{\mathbf{C}}_\mathbf{k}$ is the rendered semantic class k and min, max (performed element-wise) represent the weighted intersection and union, respectively. We then recompute the confidence scores and agreement scores as before (using the mIoU as a target metric), discard the worst 10% images in terms of mIoU as well as those whose $v_{\text{agr}} > 0.3$, and select the best hypothesis for each image as measured by the mIoU. We found no practical advantage in repeating the semantic template inference another time, nor in re-optimizing/fine-tuning the camera poses using semantics. We show this quantitatively in Section 4.4.2 and discuss further details on various exploratory attempts in Appendix 4.7.4.

4.3.2 *Generation framework*

The camera poses obtained using the approach described in Section 4.3.1 can be used to train a generative model as shown in Figure 4.5. For this component, we build upon [Pav+20b], from which we borrow the *convolutional mesh* representation and the GAN architecture. Our generation approach mainly consists of three steps. *(i)* Given a collection of images, segmentation masks, and their poses[2], we train a reconstruction model to predict mesh, texture, and semantics given only the 2D image as input. Although predicted textures are not used in subsequent steps (the GAN learns directly from image pixels), [Pav+20b] observe that predicting textures during training has a beneficial regularizing effect on the mesh, and therefore we also keep this reconstruction term. Unlike [Pav+20b] (where semantics were not available), however, we also predict a 3D semantic part segmentation in UV space, which provides further regularization and enables interesting conditional generation settings (we showcase this in Section 4.4.2). As in [Pav+20b], we parameterize the mesh as a 2D displacement map that deforms a sphere template in its tangent space. *(ii)* Through an inverse rendering approach, image pixels are projected onto the UV map of the mesh, yielding partially-occluded textures. Occlusions are represented as a binary mask in UV space. *(iii)* Finally, displacement maps and textures are modeled in UV space using a standard 2D convolutional GAN, whose training strategy compensates for occlusions by masking inputs to the discriminator.

ARCHITECTURE. Our experiments (Section 4.4) analyze two different settings: **A** where we train a separate model for each category, and **B** where we train a single model for all categories. In setting **A**, we reuse similar reconstruction and GAN architectures to [Pav+20b] in order to establish a fair comparison with their approach. We only modify the output head of the reconstruction model, where we add K extra output channels for the semantic class prediction (K depends on the category). In setting **B**, we

2 In [Pav+20b], poses are estimated via structure-from-motion on ground-truth keypoints. In this work, we use our proposed approach (Section 4.3.1).

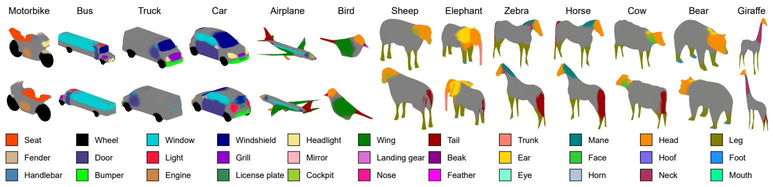

Figure 4.6: Learned 3D semantic templates. We show one template per category from two views (front/back). Colors are exaggerated for presentation purposes, but in practice the probability maps are smoother. We also highlight how semantic parts are shared among categories.

condition the model on the object category by modifying all BatchNorm layers and learning different gain and bias parameters for each category. Additionally, in the output head we share semantic classes among categories (for instance there is a unique output channel for *wheel* that is shared for buses, trucks, *etc.*; see Figure 4.6). We do not make any other change that would affect the model's capacity. As for the GAN, in both **A** and **B**, we use the same architecture as [Pav+20b]. Further details regarding hyperparameters, implementation and optimizations to improve rendering speed can be found in Appendix 4.7.1.

LOSS. The reconstruction model is trained to jointly minimize the MSE between *(i)* rendered and target silhouettes, *(ii)* predicted RGB texture and target 2D image, *(iii)* predicted semantic texture (with K channels) and target 2D semantic image. As in [Pav+20b], we add a smoothness loss to encourage neighboring faces to have similar normals. Finally, the availability of mesh templates allows us to incorporate a strong shape prior into the model via a loss term that can be regarded as an extreme form of semi-supervision: on images with very confident poses (high IoU), we provide supervision directly on the predicted 3D vertices by adding a MSE loss between the latter and the vertices of the mesh template (i.e. our surrogate ground-truth), only on the top 10% of images as measured by the IoU. This speeds up convergence and helps with modeling fine details such as wings of airplanes, where silhouettes alone provide a weak learning signal from certain views. This step requires *remeshing* the templates to align them to a common topology, which we describe in Appendix 4.7.1.

4.4 EXPERIMENTS

We quantitatively evaluate the aspects that are most central to our approach: pose estimation and generation quality.

POSE ESTIMATION. On datasets where annotated keypoints are available, we compare the poses estimated by our approach to poses estimated from *structure-from-motion* (SfM) on ground-truth keypoints. Since the robustness of SfM depends on the number of visible keypoints, we never refer to SfM poses as "ground-truth poses", as these are not available in the real-world datasets we use. Nonetheless, we believe that SfM poses serve as a good approximation of ground-truth poses on most images. Our evaluation metrics comprise *(i)* the *geodesic distance* [3] (GD) between the rotation $\mathbf{q}$ predicted by our approach and the SfM rotation $\mathbf{p}$, defined as $\text{GD} = 1 - (\mathbf{p} \cdot \mathbf{q})^2$ for quaternions, where $\text{GD} \in [0, 1]$; and *(ii)* the *recall*, which measures the fraction of usable images that have passed the ambiguity detection test. We evaluate pose estimation at different stages: after silhouette optimization (where no semantics are involved), and after the semantic template inference. Additionally, we compare settings where only one mesh template per category is available, and where multiple mesh templates are employed (we use 2–4 templates per category).

GENERATIVE MODELING. Following prior work on textured 3D mesh generation with GANs [Pav+20b], we evaluate the Fréchet Inception Distance (FID) [Heu+17] on meshes rendered from random viewpoints. For consistency, our implementation of this metric follows that of [Pav+20b]. Since our pose estimation framework discards ambiguous images and the FID is sensitive to the number of evaluated images, we *always* use the full dataset for computing reference statistics. As such, there is an incentive for

3 More commonly known as *cosine distance* when quaternions are used to describe orientations, as in our case.

optimizing both *GD* and *recall* metrics as opposed to trading one off for the other. Finally, consistently with [Pav+20b], we generate displacement maps at 32×32 resolution, textures at 512×512, and sample from the generator using a truncated Gaussian at $\sigma = 1.0$.

4.4.1 *Datasets*

We evaluate our approach on three datasets: CUB-200-2011 (CUB) [Wah+11], Pascal3D+ (P3D) [Liu+19a], and a variety of classes from ImageNet [Den+09]. The first two provide keypoint annotations and serve as a comparison to previous work, whereas on the latter we set new baselines. Combining all datasets, we evaluate our approach on 13 categories.

CUB (BIRDS). For consistency with prior work, we adopt the split of [Kan+18; Pav+20b] (≈6k training images). As we work in the unconditional setting, we do not use class labels.

PASCAL3D+ (P3D). Again, we adopt the split of [Kan+18; Pav+20b], and test our approach on both *car* and *airplane* categories. Since [Pav+20b] has only tested on cars, we train the model of [Pav+20b] on airplanes and provide a comparison here. P3D comprises a subset of images from ImageNet and [Pav+20b] evaluates only on this subset; for consistency, we adopt the same strategy.

IMAGENET. Our final selection of classes comprises the vehicles and animals that can be seen in Figure 4.6/4.7. The list of *synsets* used in each class as well as summary statistics are provided in the Appendix 4.7.3. The set of ImageNet classes includes *car* and *airplane*, which partially overlap with P3D. Therefore, when we mention these two classes, we always specify the subset we refer to (ImageNet or P3D). We also note that the dataset is heavily imbalanced, ranging from ≈300 usable images for *giraffe* to thousands of images for *car*. For this reason, in setting **B** we take measures to balance the dataset during training (Appendix 4.7.1).

Setting	Step	Bird GD(1)	Bird GD (Recall)	Car GD(1)	Car GD (Recall)	Airplane GD(1)	Airplane GD (Recall)
Single template	Silhouette	0.47	0.35 (52%)	0.12	0.05 (75%)	0.31	0.28 (85%)
	Semantics	**0.29**	**0.24** (74%)	0.11	0.06 (84%)	0.25	0.18 (78%)
	Repeat x2	**0.29**	**0.24** (76%)	0.15	0.11 (85%)	0.24	0.17 (75%)
Multiple templates	Silhouette	0.47	0.33 (44%)	0.10	0.05 (78%)	0.28	0.22 (81%)
	Semantics	0.32	0.27 (76%)	**0.06**	**0.04** (88%)	0.22	**0.15** (79%)
	Repeat x2	0.32	0.27 (78%)	0.07	0.05 (89%)	**0.21**	0.16 (80%)

Table 4.1: Pose estimation results under different settings. Best in **bold**; second best underlined. We report *geodesic distance* (GD; lower = better) after each step and associated recall (higher = better) arising from ambiguity detection. For comparison, we also report GD w/o ambiguity detection, GD(1), assuming 100% recall.

Setting	MBike	Bus	Truck	Car	Airplane	Bird	Sheep	Elephant	Zebra	Horse	Cow	Bear	Giraffe	All
Single TPL (**A**)	107.4	219.3	**164.1**	**30.73**	77.84	**55.75**	173.7	**114.5**	28.19	113.3	137.0	187.1	157.7	–
Multi TPL (**A**)	107.0	**160.7**	206.1	32.19	102.2	56.54	**155.1**	135.9	**22.10**	107.1	133.0	195.5	**126.0**	–
Single TPL (**B**)	94.74	204.98	179.3	39.68	**46.46**	88.47	169.9	127.6	24.47	**106.9**	139.4	**156.4**	176.8	**60.82**
Multi TPL (**B**)	**94.03**	187.75	204.7	46.11	77.27	77.23	163.8	146.2	31.70	113.4	**117.5**	189.9	158.0	63.00

Table 4.2: FID of our approach on ImageNet (except *bird*, which refers to CUB). We report results for models trained separately on different classes (setting **A**) and a single model that generates all classes (setting **B**). Legend: TPL = mesh template(s); lower = better, best in **bold**, second best underlined.

4.4.2 *Results*

POSE ESTIMATION. We evaluate our pose estimation framework on *bird*, *car*, and *airplane*, for which we have keypoint annotations. Reference poses are obtained using the SfM implementation of [Kan+18]. For birds (CUB), the scores are computed on all images, whereas for cars/airplanes they are computed on the overlapping images between P3D and our ImageNet subset. Results are summarized in Table 4.1. Interestingly, using multiple mesh templates does not seem to yield substantially different results, suggesting that our approach can work effectively with as

Method	Bird (CUB)	Car (P3D)	Airplane (P3D)
Keypoints+SfM [Pav+20b]	**41.56**	43.09	147.8*
Silhouette (single TPL)	73.67	38.16	100.5
Silhouette (multi TPL)	88.39	**36.17**	96.28
Semantics (single TPL)	<u>55.75</u>	<u>36.52</u>	**81.28**
Semantics (multi TPL)	56.54	37.56	<u>88.85</u>

Table 4.3: Comparison of our FID w.r.t. prior work, using either silhouettes alone or our full pipeline. Legend: * = trained by us; TPL = mesh template(s); lower = better, best in **bold**, second best <u>underlined</u>.

little as one template per class. Moreover, incorporating semantic information improves both GD and recall. Finally, we repeat the ambiguity detection and semantic template inference steps a second time, but observe no improvement. Therefore, in our following experiments we only perform these steps once. We further discuss these results in Appendix 4.7.2, where we aim to understand the most common failure modes by analyzing the full distribution of rotation errors. Qualitatively, the inferred 3D semantic templates can be found in Figure 4.6.

GENERATIVE MODEL. We report the FID on ImageNet in Table 4.2 (*bird* refers to CUB), where we set new baselines. As before, we compare settings where we adopt a single mesh template *vs* multiple templates. We also showcase a conditional model that learns to synthesize all categories using a single generator (setting **B**). Although this model has the same capacity as the individual models (but was trained to generate all classes at once), we note that its scores are in line with those of setting **A**, and in some classes (e.g. *airplane*) they are significantly better, most likely due to a beneficial regularizing effect. However, we also note that there is no clear winner on all categories. To our knowledge, no prior work has trained a single 3D generator on multiple categories without some form of supervision from synthetic data. Therefore, in one of the following paragraphs we analyze this model from a disentanglement perspective. Next, in Table 4.3, we compare our results to the state-of-the-art [Pav+20b] on the *bird*, *car*, and

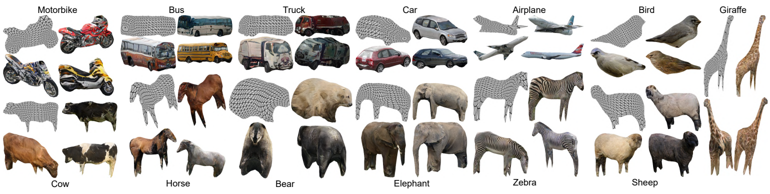

Figure 4.7: Qualitative results for all 13 classes used in our work. For each class, we show one wireframe mesh on the left, the corresponding textured mesh on the right, and two additional textured meshes on the second row. Meshes are rendered from random viewpoints.

airplane categories from CUB/P3D. We find that our approach outperforms [Pav+20b] on *car* and *airplane* (P3D) – even though we do not exploit ground-truth keypoints – and performs slightly worse on *bird* (CUB). We speculate this is mainly due to the fact that, on CUB, all keypoints are annotated (including occluded ones), whereas P3D only comprises annotations for visible keypoints, potentially reducing the effectiveness of SfM as a pose estimation method. Finally, we point out that although there is a large variability among the scores across classes, comparing FIDs only makes sense within the same class, since the metric is affected by the number of images.

QUALITATIVE RESULTS. In addition to those presented in Figure 4.1, we show further qualitative results in Figure 4.7. For animals, we observe that generated textures are generally accurate (e.g. the high-frequency details of zebra stripes are modeled correctly), with occasional failures to model facial details. With regards to shape, legs are sporadically merged but also appear correct on many examples. We believe these issues are mostly due to a pose misalignment, as animals are deformable but our mesh templates are rigid. As part of future work, we would like to add support for articulated mesh templates [Kul+20] to our method. As for vehicles, the generated shapes are overall faithful to what one would expect, especially on airplanes where modeling wings is very challenging. We also note, however, that the textures of rare

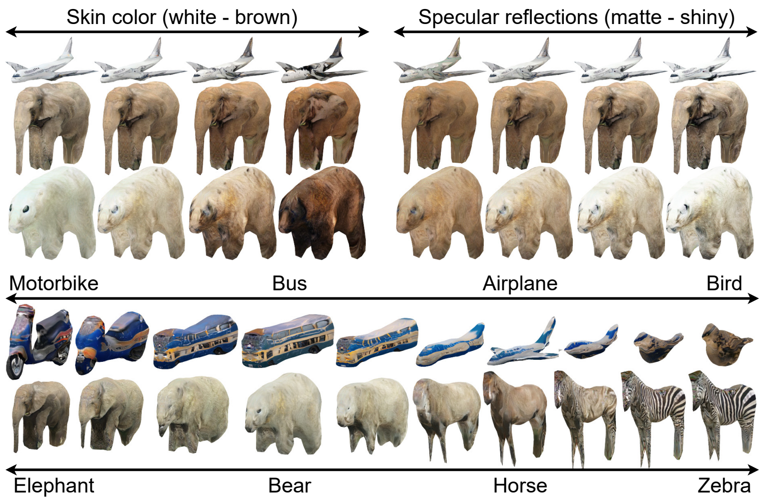
Skin color (white - brown)
Specular reflections (matte - shiny)
Motorbike
Bus
Airplane
Bird
Elephant
Bear
Horse
Zebra

Figure 4.8: Disentanglement and interpolation in the model trained to generate all classes (setting **B**). **Top:** directions in latent space that correlate with certain style factors, such as skin color and lighting. The effect is consistent across different classes. **Bottom:** interpolation between different classes with a fixed latent code.

classes (*truck* above all) present some incoherent details. Since we generally observe that the categories with more data are also those with the best results, these issues could in principle be mitigated by adding more images. Finally, we show additional qualitative results in the Appendix 4.7.2.

DISENTANGLEMENT AND INTERPOLATION. We attempt to interpret the latent space of the model trained to synthesize all classes (setting **B**), following [Här+20]. We identify some directions in the latent space that correlate with characteristics of the 3D scene, including light intensity (Figure 4.1, top-right), specular reflections and color (Figure 4.8). Importantly, these factors seem to be shared across different classes and are learned without explicit supervision. Although our analysis is preliminary, our findings suggest that 3D GANs disentangle high-level features

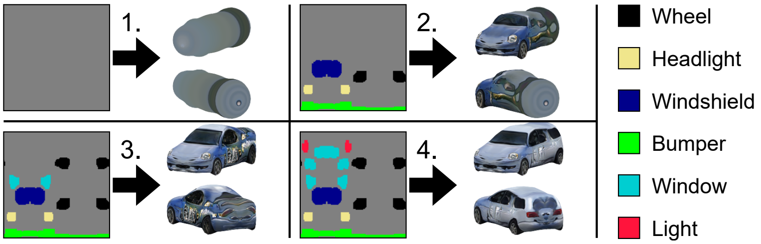

Figure 4.9: Conditional mesh generation from *semantic layouts*. In this demo, we progressively build a car by sketching its parts, proposing an interesting way of controlling the generation process.

in an interpretable fashion, similar to what is observed in 2D GANs to some extent (e.g. on pose and style). However, since 3D representations already disentangle appearance and pose, the focus of the disentangled features is on other aspects such as texture and lighting. Figure 4.8 (bottom) illustrates interpolation between different classes while keeping the latent code fixed. Style is preserved and there are no observable artifacts, suggesting that the latent space is structured.

SEMANTIC MESH GENERATION. Since our framework predicts a 3D semantic layout for each image, we can condition the generator on such a representation. In Figure 4.9, we propose a proof-of-concept where we train a conditional model on the *car* class that takes as input a semantic layout in UV space and produces a textured mesh. Such a setting can be used to manipulate fine details (e.g. the shape of the headlights) or the placement of semantic parts.

4.5 CONCLUSION

We proposed a framework for learning generative models of textured 3D meshes. In contrast to prior work, our approach does not require keypoint annotations, enabling its use on real-world datasets. We demonstrated that our method matches the results of prior works that use ground-truth keypoints, without having to rely on such information. Furthermore, we set new baselines on a subset of categories from ImageNet [Den+09], where keypoints are *not* available.

4.6 FUTURE DEVELOPMENTS

POTENTIAL IMPROVEMENTS. We believe there are still many directions of interest to pursue in order to improve this work. In addition to further analyzing disentanglement and exploring more intuitive semantic generation techniques, an interesting direction would be to experiment with articulated meshes, which should provide superior results on deformable classes such as animals.

RECENT LITERATURE. Following the success of Neural Radiance Fields (NeRF) in various areas of computer vision, NeRFs have also been explored for unconditional mesh generation. In such techniques, a 3D-aware generator is typically combined with a 2D discriminator, where the latter discriminates objects/scenes rendered from random viewpoints. Some examples are π-GAN [Cha+21] and EG3D [Cha+22], although we provide a more comprehensive overview of these methods in Section 5.2. As anticipated, the main advantage of NeRFs over triangle meshes is their ability to represent arbitrary topologies, instead of being restricted to a fixed template. For the same reason, NeRFs are much better at modelling fine shape details, but worse at modelling appearance due to their lack of UV mapping. Another drawback is the higher computational cost compared to triangle meshes. Nonetheless, the use of NeRFs in 3D synthesis tasks is advancing quickly, and it is likely that the future literature will be dominated by approaches that combine NeRFs and triangle meshes to varying extents.

FOLLOW-UP WORK. A natural question that comes to mind when working with 3D-aware generative models is whether we could use them to perform additional downstream tasks (other than unconditional generation / content creation). For example, a useful application in augmented reality (AR) and robotics is *single-view 3D reconstruction*, where the goal is to reconstruct the

3D representation corresponding to an input image. While there is a variety of work on this topic, a recent trend is represented by *GAN inversion* [Xia+22], where the objective is to leverage an *unconditional* GAN to recover the latent code that best fits a target image. As we will discuss in the next chapter (Chapter 5), this framework presents several advantages over its conditional counterpart.

4.7 APPENDIX

4.7.1 *Implementation details*

DATASET PREPARATION. We infer object silhouettes using PointRend [Kir+20] with an X101-FPN backbone, using their pre-trained model on COCO [Lin+14]. We set the object detection threshold to 0.9 to select only confident objects. As mentioned in Section 4.3, we discard object instances that are either *(i)* too small (mask area $< 96^2$ pixels), *(ii)* touch the borders of the image (indicator of possible truncation), or *(iii)* collide with other detected objects (indicator of potential occlusion). For the object part segmentations, we use the semi-supervised object detector from [Hu+18], which can segment all 3000 classes available in Visual Genome (VG) [Kri+17] while being supervised only on mask annotations from COCO. Although this model was not conceived for object part segmentation, we find that it can be used as a cost-effective way of obtaining meaningful part segmentations without collecting extra data or using co-part segmentation models that require class-specific hyperparameter tuning, such as *SCOPS* [Hun+19]. Specifically, since VG presents a long tail of rare classes, as in [PLH20] we found it beneficial to first pre-select a small number of representative classes that are widespread across categories (e.g. all land vehicles have wheels, all animals have legs). We set the detection threshold of this model to 0.2 and, for each image category, we only keep semantic classes that appear in at least 25% of the images, which helps eliminate spurious detections. On our data, this leads to a number of semantic classes $K \approx 10$ per image category (33 across all categories). The full list of semantic classes can be seen in Figure 4.6. To deal with potentially overlapping part detections (e.g. the segmentation mask of the door of a car might overlap with a window), the output semantic maps represent probability distributions over classes, where we weight each semantic class proportionally to the object detection score. Additionally, we add an extra class for "no class" (depicted in gray in our figures).

MESH TEMPLATES AND REMESHING. We borrow a selection of mesh templates from [Kul+20] as well as meshes freely available on the web. In the experiments where we adopt multiple mesh templates, we only use 2–4 meshes per category. An important preliminary step of our approach, which is performed even before the pose estimation step, consists in *remeshing* these templates to align them to a common topology. This has the goal of reducing their complexity (which translates into a speed-up during optimization), removing potential invisible interiors, and enabling efficient batching by making sure that every mesh has the same number of vertices/faces. Additionally, as mentioned in Section 4.3.2, remeshing is required for the semi-supervision loss term in the reconstruction model. We frame this task as an optimization problem where we deform a 32×32 UV sphere to match the mesh template. More specifically, we render each template from 64 random viewpoints at 256×256 resolution, and minimize the MSE loss between the rendered deformed sphere and the target template in pixel space ($\mathcal{L}_{\text{MSE}}$). Moreover, we regularize the mesh by adding *(i)* a smoothness loss $\mathcal{L}_{\text{flat}}$, which encourages neighboring faces to have similar normals, *(ii)* a Laplacian smoothing loss $\mathcal{L}_{\text{lap}}$ with quad connectivity (i.e. using the topology of the UV map as opposed to that of the triangle mesh), and *(iii)* an edge length loss $\mathcal{L}_{\text{len}}$ with quad connectivity, which encourages edges to have similar lengths. $\mathcal{L}_{\text{flat}}$ and $\mathcal{L}_{\text{len}}$ are defined as follows:

$$\mathcal{L}_{\text{flat}} = \frac{1}{|E|} \sum_{i,j \in E} (1 - \cos\theta_{ij})^2, \tag{4.5}$$

$$\mathcal{L}_{\text{len}} = \frac{1}{|UV|} \sum_{i \in U} \sum_{j \in V} \frac{\|\mathbf{v}_{i+1,j} - \mathbf{v}_{i,j}\|_1 + \|\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j}\|_1}{6}, \tag{4.6}$$

where E is the set of edges, $\cos\theta_{ij}$ is the cosine similarity between the normals of faces i and j, and $\mathbf{v}_{i,j}$ represents the 3D vertex at the coordinates i, j of the UV map.
Finally, we weight each term as follows:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + 0.00001\,\mathcal{L}_{\text{flat}} + 0.003\,\mathcal{L}_{\text{lap}} + 0.01\,\mathcal{L}_{\text{len}}. \tag{4.7}$$

Additionally, in the experiments with multiple mesh templates, we add a pairwise similarity loss $\mathcal{L}_{\text{align}}$ which penalizes large variations of the vertex positions between different mesh templates (only within the same category):

$$\mathcal{L}_{\text{align}} = \frac{1}{N_t^2} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \|\mathbf{V_i} - \mathbf{V_j}\|_2, \tag{4.8}$$

where $\mathbf{V_i}$ is a matrix that contains the vertex positions of the i-th mesh template (of shape $3 \times N_v$), and N_t is the number of mesh templates. This loss term is added to the total loss with weight 0.001. Note that we use a non-squared L2 penalty for this term, which encourages a sparse set of vertices to change between mesh templates.
We optimize the final loss using SGD with momentum (initial learning rate $\alpha = 0.0001$ and momentum $\beta = 0.9$). We linearly increase α to 0.0005 over the course of 500 iterations (warm-up) and then exponentially decay α with rate 0.9999. We stop when the learning rate falls below 0.0001. Additionally, we normalize the gradient before each update. Figure 4.10 shows two qualitative examples of remeshing.

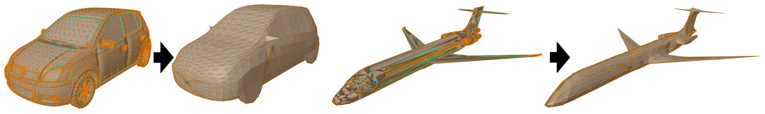

Figure 4.10: Remeshing of the mesh templates. In this figure we show two demos (one template for *car* and one for *airplane*).

POSE ESTIMATION. For the silhouette optimization step, we initialize $N_c = 40$ camera hypotheses per image by uniformly quantizing azimuth and elevation (8 quantization levels along azimuth and 5 levels along elevation). We optimize each camera hypothesis using Adam [KB14] with full-matrix preconditioning, where we set $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The implementation of our variant of Adam as well its theoretical justification are described in the next paragraph. We optimize each hypothesis for 100 iterations, with an initial learning rate $\alpha = 0.1$ which is decayed to 0.01 after the 80th iteration. After each iteration, we reproject quaternions onto the unit ball. As a performance optimization, silhouettes are initially rendered at 128×128 resolution, which is increased to 192×192 after the 30th iteration and 256×256 after the 60th iteration. Finally, in the settings where we prune camera hypotheses, we discard the worst 50% hypotheses as measured by the intersection-over-union (IoU) between projected and target silhouettes. This is performed twice: after the 30th and 60th iteration.

Algorithm 1 Adam with full-matrix preconditioning.
Changes w.r.t. the original algorithm are highlighted.

```
1:  require $\alpha$ (step size), $\beta_1, \beta_2, \epsilon$
2:  initialize time step $t \leftarrow 0$
3:  initialize parameters $\boldsymbol{\theta}_0$ ($d$-dimensional col. vector)
4:  initialize first moment $\boldsymbol{m}_0 \leftarrow \boldsymbol{o}$ ($d$-dimensional col. vector)
5:  initialize second moment $\boldsymbol{V}_0 \leftarrow \boldsymbol{o}$ ($d \times d$ matrix)
6:  repeat
7:      $t \leftarrow t + 1$
8:      $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$    ▷ gradient
9:      $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$    ▷ first moment
10:     $\boldsymbol{V}_t \leftarrow \beta_2 \boldsymbol{V}_{t-1} + (1 - \beta_2)\, \boldsymbol{g}_t \boldsymbol{g}_t^T$    ▷ second moment
11:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$    ▷ bias correction
12:     $\hat{\boldsymbol{V}}_t \leftarrow \boldsymbol{V}_t / (1 - \beta_2^t)$    ▷ bias correction
13:     $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha\, (\hat{\boldsymbol{V}}_t + \epsilon \mathbf{I}_d)^{-\frac{1}{2}} \hat{\boldsymbol{m}}_t$    ▷ update
14: until stopping criterion
15: return $\boldsymbol{\theta}_t$
```

FULL-MATRIX PRECONDITIONING. Adam [KB14] is an established optimizer for training neural networks. Its use of diagonal preconditioning is an effective trick to avoid storing an $\mathcal{O}(d^2)$ matrix for the second moments (where d is the number of learnable parameters), for which a matrix square root and inverse need to be subsequently computed (an extra $\mathcal{O}(d^3)$ cost for each of the two operations). However, since our goal is to optimize camera parameters, we observe that:

1. Optimizers with diagonal preconditioning are not rotation invariant, i.e. they have some preferential directions that might bias the pose estimation result.

2. Since each camera hypothesis comprises only 8 parameters, inverting an 8×8 matrix has a negligible cost.

Using a rotation invariant optimizer such as SGD (with or without momentum) is a more principled choice as it addresses the first observation. However, based on our second observation, we take the best of both worlds and modify Adam to implement full-matrix preconditioning. This only requires a trivial modification to the original implementation, which we show in Algorithm 1 (changes w.r.t. the original algorithm are highlighted in green).

SEMANTIC TEMPLATE INFERENCE. As mentioned in Section 4.3.1, the goal of this step is to infer a 3D semantic template for each mesh template, given an initial (untextured) mesh template, the output of the silhouette optimization step, and a collection of 2D semantic maps. Recapitulating from Section 4.3.1, we solve the following optimization problem:

$$\mathcal{L}_i = \left\| \mathcal{R}(\mathbf{V}_{\text{tpl}}, \mathbf{F}_{\text{tpl}}, \mathbf{C}_{\text{tpl}};\ \mathbf{q_i}, \mathbf{t_i}, s_i, z_{0i}) - \mathbf{C_i} \right\|^2 \tag{4.9}$$

$$\mathbf{C}^*_{\text{tpl}} = \min_{\mathbf{C}_{\text{tpl}}} \frac{1}{N_{\text{top}}} \sum_i \mathcal{L}_i. \tag{4.10}$$

Conceptually, our goal is to learn a shared semantic template (parameterized using vertex colors) that averages all 2D semantic maps in vertex space. We propose the following closed-form solution which uses the gradients from the differentiable renderer and requires only a single pass through the dataset:

$$\mathbf{A} = \sum_i \nabla_{\mathbf{C}_{\text{tpl}}}(\mathcal{L}_i) \tag{4.11}$$

$$(\mathbf{C}^*_{\text{tpl}})_k = \frac{\epsilon + \mathbf{a_k}}{K\epsilon + \sum_j \mathbf{a_j}}, \tag{4.12}$$

where $\mathbf{A}$ is an accumulator matrix that has the same shape as the $\mathbf{C}_{\text{tpl}}$ (the vertex colors), and ϵ is a small *additive smoothing* constant that leads to a uniform distribution on vertices that are never rendered (and thus have no gradient). This operation can be regarded as projecting the 2D object-part semantics onto the mesh vertices and computing a color histogram on each vertex. We show a sample illustration in Figure 4.11.

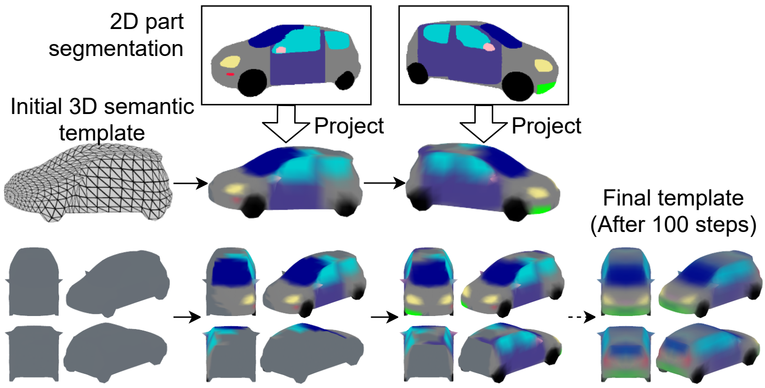
2D part segmentation
Initial 3D semantic template
Project
Project
Final template (After 100 steps)

Figure 4.11: Semantic template inference, starting from an untextured 3D mesh template (left-to-right progression). In this figure we show a demo with two sample images, and the final result using the top 100 images as measured by the IoU.

In section Section 4.3.1 we explained that we compute the semantic template using the top $N_{\text{top}} = 100$ images as measured by the IoU, among those that passed the ambiguity detection test ($v_{\text{agr}} < 0.3$). To further improve the quality of the inferred semantic templates, we found it beneficial to add an additional filter where we only select poses whose cosine distance is within 0.5 (i.e. 45 degrees) of the left/right side. Objects observed from the left/right side are intrinsically unambiguous, since there is no complementary pose that results in the same silhouette. Therefore, we favor views that are close to the left/right as opposed to the front/back or top/bottom, which are the most ambiguous views. Note that this filter is only used for the semantic template inference step.

GENERATIVE MODEL. We train the single-category reconstruction networks (setting **A**) for 130k iterations, with a batch size of 32, and on a single GPU. The multi-category model (setting **B**) is trained for 1000 epochs, with a total batch size of 128 across 4 GPUs, using synchronized batch normalization. In both settings, we use Adam [KB14] (the original one, not our variant with full-matrix preconditioning) with an initial learning rate of 0.0001 which is halved at 1/4, 1/2, 3/4 of the training schedule. For the GAN, we use the same hyperparameters as [Pav+20b], except in the multi-category model (setting **B**), which is trained with a batch size of 64 instead of the default 32. Furthermore, in setting **B**, and for both models (reconstruction and GAN), we equalize classes during mini-batch sampling. This is motivated by the large variability in the amount of training images, as explained in Section 4.4.1, and as can also be seen in Table 4.4. Finally, as in [GKM20; Kan+18; Li+20; Pav+20b], we force generated meshes to be left/right symmetric.

SEMANTIC MESH GENERATION. In the setting where we generate a 3D mesh from a semantic layout in UV space, we modify the generator architecture of [Pav+20b]. Specifically, we replace the input linear layer (the one that projects the latent code $\mathbf{z}$ onto the first 8×8 convolutional feature map) with four convolutional layers. These progressively downsample the semantic layout from 128×128 down to 8×8 (i.e. each layer has stride 2). The first layer takes as input a *one-hot* semantic map (with K semantic channels) and yields 64 output channels (128, 256, 512 in the following layers). In these 4 layers, we use Leaky ReLU activations (slope 0.2), spectral normalization, but no batch normalization. We leave the rest of the network unchanged. In this model, we also found it necessary to fine-tune the batch normalization statistics prior to evaluation, which we do by running a forward pass over the entire dataset on the *running average* model. As for the discriminator, we simply resize the semantic map as required and concatenate it to the input.

4.7.2 *Additional results*

POSE ESTIMATION. In Figure 4.12, we provide more insight into the *geodesic distance* metric, which measures the cosine distance between the rotations predicted by our approach (Section 4.3.1) and SfM rotations. In particular, as opposed to the results presented in Table 4.1 (which shows only the average), here we show the full distribution of errors. A distance of 0 means that the two rotations match exactly, whereas a distance of 1 (maximum value) means that the rotations are rotated by 180 degrees from one another. On the analyzed classes (*car*, *airplane*, and *bird*, for which we have SfM poses), we can generally observe a bimodal distribution: a majority of images where pose estimation is correct, i.e. the GD is close to zero, and a small cluster of images where the GD is close to one. This is often the case for ambiguities: for instance, in cars we sometimes observe a front/back confusion. As expected, exploiting semantics (step 2) mitigates this issue and increases the amount of available images (this is particularly visible on *bird*). We also note that, for rigid objects such as *car* and *airplane*, the distribution is more peaky, whereas for *bird* the tail of errors is longer, most likely because pose estimation is more ill-defined for articulated objects.

QUALITATIVE RESULTS. We show extra qualitative results in Figure 4.13. In particular, we render each generated mesh from two random viewpoints and showcase the associated texture and wireframe mesh. Additionally, in Figure 4.14 we show the most common failure cases across categories. We can identify some general patterns: for instance, in vehicles we sometimes observe incoherent textures (this is particularly visible in *truck* due to the small size of this dataset). On animals, as mentioned, we observe occasional failures to model facial details, merged/distorted legs, and more rarely, mesh distortions. To some extent, these issues can be mitigated by sampling from the generator using a lower truncation threshold (we use $\sigma = 1.0$ in our experiments), at the expense of sample diversity.

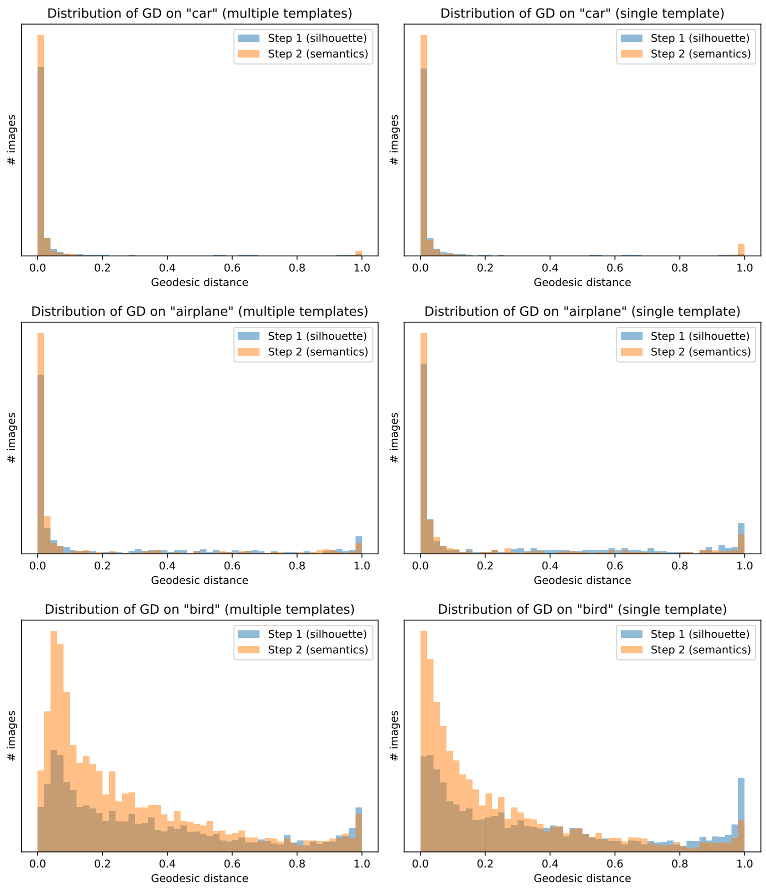

Figure 4.12: Distribution of pose estimation errors on *car*, *airplane*, and *bird*. We compare settings where we use multiple mesh templates (left) and a single template (right).

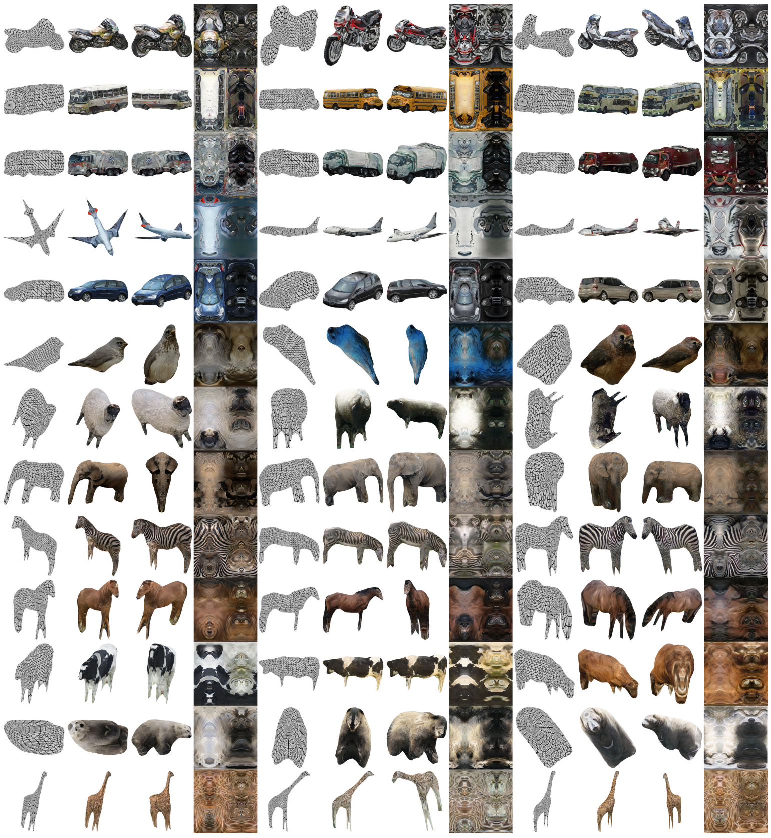

Figure 4.13: Additional qualitative results. We show three examples per category. Each example is rendered from two random views, and the corresponding texture/wireframe mesh is also shown.

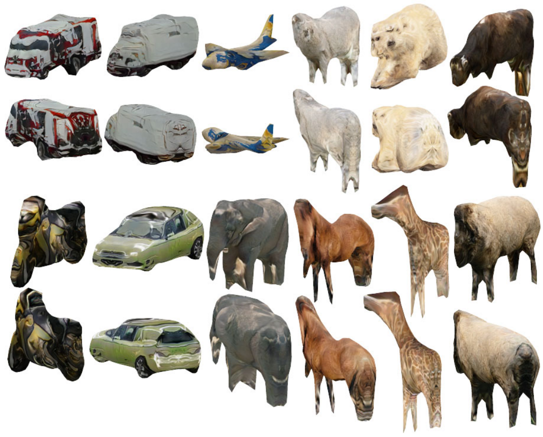

Figure 4.14: Failure cases for a variety of categories.

SEMANTIC TEMPLATES. Figure 4.15 shows the full set of learned semantic templates for every category. Most results are coherent, although we observe a small number of failure cases, e.g. in *truck* one or two templates are mostly empty and are thus ineffective for properly resolving ambiguities. This generally happens when the templates have too few images assigned to them and explains why the multi-template setting does not consistently outperform the single-template setting.

DEMO VIDEO. The supplementary material in [Pav+21] includes a video where we show additional qualitative results. First, we showcase samples generated by our models in setting **A** and explore the latent space of the generator. Second, we analyze the latent space of the model trained to generate multiple classes (setting **B**), and discover interpretable directions in the latent space, which can be used to control shared aspects between classes (e.g. lighting, shadows). We also interpolate between different classes while keeping the latent space fixed, and highlight that style is preserved during interpolation. Finally, we showcase a setting where we generate a mesh from a hand-drawn semantic layout in UV space, similar to Figure 4.9.

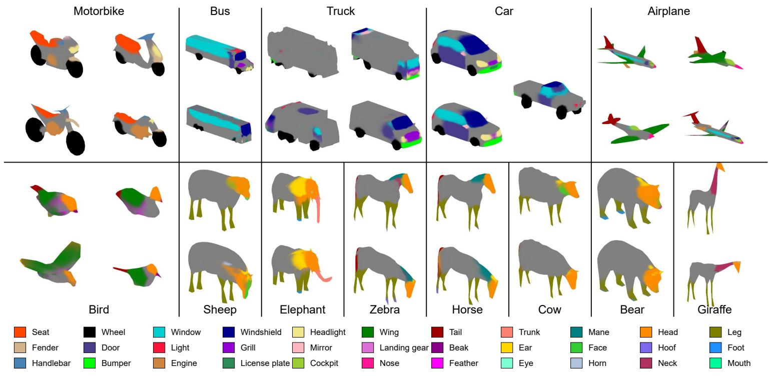

Figure 4.15: Visualization of *all* the learned 3D semantic templates (2–4 per category). While most results are as expected, the figure highlights some failure cases, e.g. in *truck* some templates have very few images assigned to them, which leads to incoherent semantics.

4.7.3 *Dataset information*

For our experiments on ImageNet, we adopt the *synsets* specified in Table 4.4. Since some of our required synsets are not available in the more popular ImageNet1k, we draw all of our data from the larger ImageNet22k set.

4.7.4 *Negative results*

To guide potential future work in this area, we provide a list of ideas that we explored but did not work out.

SILHOUETTE OPTIMIZATION. For the silhouette optimization step with multiple templates, before reaching our current formulation, we explored a range of alternatives. In particular, we tried to smoothly interpolate between multiple meshes by optimizing a set of interpolation weights along with the camera parameters. This yielded inconsistent results across categories, which convinced us to work with a "discrete" approach as opposed to a smooth one. We then tried a reinforcement learning approach inspired

Class	Synsets	Raw images	Valid instances
Motorbike	n03790512, n03791053, n04466871	4037	1351
Bus	n04146614, n02924116	2641	1190
Truck	n03345487, n03417042, n03796401	3187	1245
Car	n02814533, n02958343, n03498781, n03770085, n03770679, n03930630, n04037443, n04166281, n04285965	12819	4992
Airplane	n02690373, n02691156, n03335030, n04012084	5208	2540
Sheep	n10588074, n02411705, n02413050, n02412210	4682	864
Elephant	n02504013, n02504458	3927	1434
Zebra	n02391049, n02391234, n02391373, n02391508	5536	1753
Horse	n02381460, n02374451	2589	664
Cow	n01887787, n02402425	2949	861
Bear	n02132136, n02133161, n02131653, n02134084	6745	2688
Giraffe	n02439033	1256	349

Table 4.4: Synsets and summary statistics for our ImageNet data. For each category, we report the number of raw images in the dataset, and the number of extracted object instances that have passed our quality checks (size, truncation, occlusion).

by multi-armed bandits: we initialized each camera hypothesis with a random mesh template, and used a UCB (upper confidence bound) selection algorithm to select the optimal mesh template during optimization. This led to slightly worse results than interpolation. Finally, we reached our current formulation, where we simply replicate each camera hypothesis and optimize the different mesh templates separately. We adopt pruning to make up for the increase in computation time.

RE-OPTIMIZING POSES MULTIPLE TIMES. In our current formulation, after the semantic template inference step, we use the semantic templates to resolve ambiguities, but there is no further optimization involved. Naturally, we explored the idea of repeating the silhouette optimization step using semantic information.

However, we were unable to get this step to work reliably, even after attempting with multiple renderers (we tried both with DIB-R [Che+19] and SoftRas [Liu+19b]). We generally observed that the color gradients are too uninformative for optimizing camera poses, even after trying to balance the different components of the gradient (silhouette and color). We believe this is a fundamental issue related to the non-convexity of the loss landspace, which future work needs to address. We also tried to smooth out the rendered images prior to computing the MSE loss, without success.

REMESHING. Since target 3D vertices are known in this step, we initially tried to use a 3D chamfer loss to match the mesh template. This, however, led to artifacts and merged legs in animals, and was too sensitive to initialization. We found it more reliable to use a differentiable render with silhouette-based optimization.

5 BOOTSTRAPPED INVERSION FOR SINGLE-VIEW 3D RECONSTRUCTION

CHAPTER ABSTRACT. Neural Radiance Fields (NeRF) coupled with GANs represent a promising direction in the area of 3D reconstruction from a single view, owing to their ability to efficiently model arbitrary topologies. Recent work in this area, however, has mostly focused on synthetic datasets where exact ground-truth poses are known, and has overlooked pose estimation, which is important for certain downstream applications such as augmented reality (AR) and robotics. We introduce a principled end-to-end reconstruction framework for natural images, where accurate ground-truth poses are not available. Our approach recovers an SDF-parameterized 3D shape, pose, and appearance from a single image of an object, without exploiting multiple views during training. More specifically, we leverage an unconditional 3D-aware generator, to which we apply a hybrid inversion scheme where a model produces a first guess of the solution which is then refined via optimization. Our framework can de-render an image in as few as 10 steps, enabling its use in practical scenarios. We demonstrate state-of-the-art results on a variety of real and synthetic benchmarks.

OPEN SOURCE. Code and models for this work are available at https://github.com/google-research/nerf-from-image.

This chapter is based on our CVPR 2023 paper "Shape, Pose, and Appearance from a Single Image via Bootstrapped Radiance Field Inversion" [Pav+23].

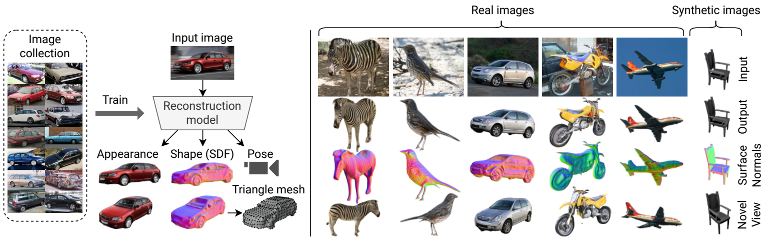

Figure 5.1: Given a collection of 2D images representing a specific category (e.g. cars), we learn a model that can fully recover shape, pose, and appearance from a single image, without leveraging multiple views during training. The 3D shape is parameterized as a signed distance function (SDF), which facilitates its transformation to a triangle mesh for further downstream applications.

5.1 INTRODUCTION

In this chapter, we move from the unconditional generation task and focus on the task of single-view 3D reconstruction, where the goal is to reconstruct shape, appearance, and camera pose from a single image of an object (Figure 5.1). Although – at first – the latter may seem only loosely connected to the 3D-aware generation task described in the previous chapters, in this chapter we will highlight how these tasks are closely interconnected, and how advances in one task benefit the other. Single-view 3D reconstruction has applications in content creation, augmented & virtual reality (AR/VR), robotics, and is also interesting from a scientific perspective, as most neural architectures cannot reason about 3D scenes. As humans, we learn *object priors*, abstract representations that allow us to imagine what a partially-observed object would look like from other viewpoints. Incorporating such knowledge into a model would enable higher forms of 3D reasoning. While early work on 3D reconstruction has focused on exploiting annotated data [Gir+16; HTM17; Wu+17; Yan+17a; Zhu+17b], e.g. ground-truth 3D shapes or multiple 2D views, more recent work has relaxed the assumptions required by the task. In particular, there has been effort in learning this task from single-view collections of images depicting a specific category [GKM20; Kan+18; Li+20] (e.g. a dataset of cars), and we also follow this line of work.

Most established 3D representations in the single-view reconstruction literature are based on deformable triangle meshes [GKM20; Kan+18; Li+20], although Neural Radiance Fields (NeRF) [Bar+21; Mil+20] have recently become more prominent in the broader 3D vision community owing to their ability to efficiently model arbitrary topologies. These have been combined with GANs [Goo+14] for unconditional 3D generation tasks [Cha+21; Cha+22; NG21; Xue+22], as they produce more perceptually pleasing results. There has also been work on combining the two in the single-view reconstruction task, e.g. Pix2NeRF [Cai+22], which is however demonstrated on simple settings of faces or synthetic datasets where perfect ground-truth poses are available. Furthermore, there has been less focus overall on producing an end-to-end reconstruction system that additionally tackles *pose estimation* (beyond simple settings), which is particularly important for AR applications. In our work, we bridge this gap by proposing a more general NeRF-based end-to-end reconstruction pipeline that tackles both reconstruction and pose estimation, and demonstrate its broader applicability to natural images where poses cannot be accurately estimated. We further characterize the problem by comparing encoder-based approaches (the majority of methods in the single-view reconstruction literature) to inversion-based approaches (which invert a generator via optimization), and show that the latter are more suited to real datasets without accurate ground-truth poses.

Motivated by this, we propose a *hybrid GAN inversion* technique for NeRFs that can be regarded as a compromise between the two: an encoder produces a first guess of the solution (*bootstrapping*), which is then refined via optimization. We further propose a series of technical contributions, including: *(i)* the adoption of an SDF representation [Yar+21] to improve the reconstructed surfaces and facilitate their conversion to triangle meshes, *(ii)* regularizers to accelerate inversion, and *(iii)* the addition of certain equivariances in the model architecture to improve generalization. We show that we can invert an image in as few as 10 optimization steps, making our approach usable even in constrained scenarios. Furthermore, we incorporate a principled pose estimation framework [Wan+19] that frames the problem as a regression of a canonical representation followed by Perspective-*n*-Point (PnP), and show that it boosts pose estimation accuracy without additional data assumptions. We summarize our main contributions as follows:

- We introduce an end-to-end single-view 3D reconstruction pipeline based on NeRFs. In this setting, we successfully demonstrate 360° object reconstruction from natural images under the CMR [GKM20] benchmark.
- We propose a hybrid inversion scheme for NeRFs to accelerate the reversal of pre-trained 3D-aware generators.
- Inspired by the literature on pose estimation, we propose a principled PnP-based pose estimator that leverages our framework and does not require extra data assumptions.

To validate our contributions, we obtain state-of-the-art results on both real/synthetic benchmarks. Furthermore, to our knowledge, we are the first to demonstrate NeRF-based reconstruction on *in-the-wild* datasets such as ImageNet.

5.2 RELATED WORK

INVERSE RENDERING AND SCENE REPRESENTATIONS. Although 3D reconstruction is an established task, the representations and supervision methods used to tackle this problem have evolved throughout the literature. Early approaches have focused on reconstructing shapes using 3D supervision, adopting voxel grids [Gir+16; HTM17; Wu+17; Yan+17a; Zhu+17b], point clouds [FSG17], or SDFs [Par+19a], and require synthetic datasets where ground-truth 3D shapes are available. The introduction of differentiable rendering [Che+19; Che+21b; KUH18; Liu+19b; LB14] has enabled a new line of work that attempts to reconstruct shape and texture from single-view datasets, leveraging triangle mesh representations [Bha+21; Che+19; GKM20; HF18; Kan+18; Li+20; Zha+22]. Each 3D representation, however, comes with its own set of trade-offs. For instance, voxels do not scale efficiently with resolution, while triangle meshes are efficient but struggle with arbitrary topologies (most works deform a sphere template). In recent developments, implicit representations encode a 3D scene as the weights of an MLP that can be queried at specific coordinates, which allows them to model arbitrary topologies using lightweight networks. In such a setting, there has been work on 3D reconstruction using implicit SDFs [DP22; LWL20] as well as neural radiance fields (NeRF) [Bar+21; Mil+20]. Finally, some works incorporate additional structural information into 3D representations, e.g. [Yao+22] reconstructs articulated shapes using skeleton priors, [Che+21b; WWR22] disentangle albedo from reflectance, and [Xu+22] uses depth cues. These techniques are orthogonal to ours and may positively benefit each other.

NERF-BASED RECONSTRUCTION. The standard use-case of a NeRF is to encode a single scene given multiple 2D views and associated camera poses, which does not necessarily lead to learned shared representations. There have however been attempts at learning an object prior by training such models on a category-specific dataset (e.g. a collection of cars). For instance, [JA21; Reb+22] train a shared NeRF backbone conditioned on a learned latent code for each object instance. [Yu+21] tackles reconstruction conditioned on an image encoder, although it requires multiple ground-truth views for supervision and does not adopt an adversarial setting, thereby relying on accurate poses from synthetic datasets and leading to blurry results. [Cai+22; Mi+22] adopt an adversarial setting and only require a single view during training, but they focus on settings with simple pose distributions. Finally, there has been work on using diffusion models [HJA20; Wat+22] and distillation [RTT21] for novel-view synthesis, though such methods do not explicitly recover a 3D surface.

ENCODER- *vs* INVERSION-BASED METHODS. Most aforementioned methods can be categorized as *encoder-based*, where a 2D ConvNet encodes the input image into a latent representation, then decoded into a 3D scene. This paradigm is analogous to an autoencoder, and therefore requires some form of pixel-level loss between predicted and input images. While this is appropriate for synthetic datasets with exact poses, it leads to blurry or distorted results when such poses are inaccurate (i.e. the case in natural images). Following the 2D GAN inversion literature [Xia+22], there has been work on applying inversion methods to 3D reconstruction, where the goal is to leverage a pretrained unconditional GAN and find the latent code that best fits the input image via optimization. Since unconditional GANs tend to be more robust to inaccurate poses (as they mostly rely on the overall pose distribution as opposed to the pose of each image), we argue that inversion-based approaches are better suited to natural images. As part of our work, we characterize this phenomenon experimentally. 3D GAN inversion has been applied to untextured shapes [Dug+22; Zha+21a], textured triangle meshes [Zha+22], and its use with NeRF-based approaches is suggested in [Cai+22; Cha+21; Cha+22], although it is not their focus.

OUR WORK. We propose a *hybrid* inversion paradigm, where an encoder produces a first guess of the latent representation and pose (*bootstrapping*), and these are then refined for a few iterations via optimization. Although [Dug+22] introduce a similar idea, they focus on shape completion from LiDAR data, whereas we focus on shape, pose, and appearance prediction from an image. Under our setting, Pix2NeRF [Cai+22] provides a proof-of-concept of refinement using such a method, but it is still trained using an encoder-based paradigm and is thus affected by the aforementioned issues. By contrast, we propose a principled end-to-end hybrid reconstruction approach that takes full advantage of an unconditional generator and can also optimize with respect to pose (unlike [Cai+22; Cha+21; Cha+22]), a task that requires a suitable pose parameterization. We also mention that [Zha+21b] propose a similar idea to bootstrapping (without inversion), but they adopt a 2D image generator as opposed to a 3D-aware one, which does not fully disentangle pose from appearance.

UNCONDITIONAL GENERATION. Since inversion-based approaches rely on a pretrained generator, we briefly discuss recent architectures for this task. Our works presented in the previous chapters [HTL20; Pav+21; Pav+20b] learn to generate triangle meshes and textures using 2D supervision from single-view collections of natural images. [Cha+21] learns this task using NeRFs, although it suffers from the high computational cost of MLP-based NeRFs. [Cha+22; Or-+22; Gu+21; NG21; Sch+20; SZW19; Xue+22] incorporate both 2D and 3D components as a trade-off between 3D consistency and efficiency. Finally, [Gao+22] proposes an approach to train a NeRF-based generator whose outputs can be distilled into triangle meshes. The generator used in our work leverages an EG3D-like backbone [Cha+22].

5.3 METHOD

We now present our single-view reconstruction approach. We break down our method into three main steps. *(i)* Initially, we train an unconditional generator following the literature on 3D-aware GANs [Cha+21; Cha+22], where a NeRF-based generator **G** is combined with a 2D image discriminator. This framework requires minimal assumptions, namely 2D images and the corresponding pose distribution. We further apply a series of technical improvements to the overall framework in order to positively impact the subsequent reconstruction step, as explained in Section 5.3.1. *(ii)* We freeze **G** and train an image encoder **E** that jointly estimates the pose of the object as well as an initial guess of its latent code (*bootstrapping*). For pose estimation, we adopt a principled approach that predicts a canonical map [Wan+19] in screen space followed by a Perspective-*n*-Point (PnP) algorithm. We explain these steps in Section 5.3.2. Finally, *(iii)* we refine the pose and latent code for a few steps via gradient-based optimization (*hybrid inversion*), as described in Section 5.3.3.

REQUIREMENTS. For training, our method requires a category-specific collection of images, along with segmentation masks for datasets with a background (we use an off-the-shelf segmentation model, PointRend [Kir+20]), which we use to pre-segment the images. An approximate pose distribution must also be known. For inference, only a single, unposed input image is required.

5.3.1 *Unconditional generator pre-training*

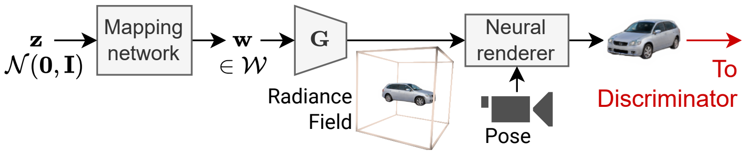

Figure 5.2: Unconditional generation framework.

We adopt EG3D [Cha+22] as a starting point for the backbone of our generator. It consists of a *mapping network* that maps a prior $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ to a latent code $\mathbf{w} \in \mathcal{W}$, the latter of which is plugged into a StyleGAN2 generator [Kar+20b]. The output feature map is then split into three orthogonal planes (xy, xz, yz), which are queried at specific coordinates via bilinear sampling. The resulting features are finally summed and plugged into a tiny MLP (*triplanar decoder*) to output the values of the radiance field (density and color). The generator $\mathbf{G}$ is trained using a GAN framework where the discriminator takes 2D renderings as input. We apply some adjustments to the triplanar decoder and training objective, including the ability to model view-dependent effects as well as improvements to the adaptive discriminator augmentation (ADA) technique [Kar+20a], which is used on small datasets (see Appendix 5.8.1). In the next paragraphs, we focus on the changes that are central to our reconstruction approach.

SDF REPRESENTATION. We found it beneficial to parameterize the object surface as a signed distance function (SDF), as opposed to the standard volume density parameterization adopted in EG3D [Cha+22]. In addition to an empirical advantage (Section 5.5), SDFs facilitate the extraction of the surface and its subsequent conversion to other representations (e.g. triangle meshes), since they provide analytical information about surface boundaries and normals. SDFs have already been explored in unconditional generators [Or-+22] and in the broader NeRF literature [OPG21; Wan+21; Yar+21; Yu+22], but less so in the single-view reconstruction setting. We follow VolSDF [Yar+21], in which the volume density $\sigma(\mathbf{x})$ is described as:

$$\sigma(\mathbf{x}) = (1/\alpha)\,\Psi_\beta(-d(\mathbf{x}))\,, \tag{5.1}$$

where $\mathbf{x}$ are the query coordinates, $d(\mathbf{x})$ is the SDF (i.e. the output of the generator), and Ψ_β is the cumulative density function (CDF) of the Laplace distribution with scale β and zero mean. $\alpha, \beta > 0$ are learnable parameters. We also incorporate an Eikonal loss to encourage the network to approximate a valid distance function:

$$\mathcal{L}_{\text{Eikonal}} = \mathbb{E}_{\mathbf{x}}[(\|\nabla_{\mathbf{x}} d(\mathbf{x})\| - 1)^2]. \tag{5.2}$$

We efficiently approximate the expectation using stratified sampling across the bounding volume of the scene, and employ a custom bilinear sampling implementation in the triplanar encoder which supports double differentiation w.r.t. the input query points. Furthermore, we initialize the SDF to a unit sphere via pretraining. Implementation details can be found in the Appendix 5.8.1.

REMOVING SUPER-RESOLUTION NETWORK. In [Cha+22], the rendered image is further processed through a super-resolution network, which increases its resolution and corrects for any distribution mismatch at the expense of 3D consistency. Since we aim to address fully 3D-consistent reconstruction instead of a more relaxed novel-view-synthesis task, we remove this component and feed the rendered image directly through the discriminator. This choice also makes it easier to fairly compare our approach to existing work.

ATTENTION-BASED COLOR MAPPING. A robust 3D reconstruction technique should be as much as possible equivariant to certain transformations in order to improve generalization on unseen data. These include geometric transformations (e.g. a 2D translation in the input image should be reflected in the 3D pose, which motivates our principled pose estimation technique in Section 5.3.2) as well as color transformations, e.g. changing the hue of an object (an image of a red car into that of a white car) should result in an

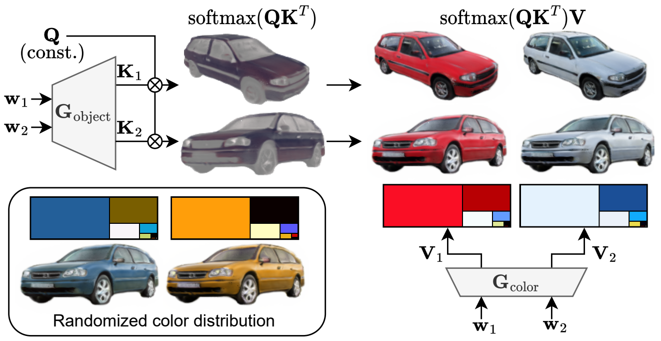

Figure 5.3: Illustration of our color mapping technique with two objects generated by two different latent codes $\mathbf{w}_1$ and $\mathbf{w}_2$. The object generator models a latent radiance field of keys $\mathbf{K}$ (each of which represents a semantic embedding at a specific spatial position), which are multiplied with a fixed set of queries $\mathbf{Q}$ (i.e. learned prototype embeddings for each "semantic channel") and processed through a softmax to produce a probability distribution across these semantic channels, whose meaning is learned. In the case of cars, the learned semantic channels include body, headlights, wheels, and reflections. In the image, we show a rendering of the result of this operation in false colors, where the weight of one of the classes (car body) is highlighted. Finally, the latter is multiplied with the values $\mathbf{V}$ (color distribution, i.e. a color for each semantic channel) produced by another module (color network), resulting in the final RGB colors. While during training the same latent code goes into both networks so as to learn the correct data distribution, at inference we can split it to swap the color distribution among different object identities (**top-right**) or randomize it entirely (**bottom-left**).

equivalent change in the radiance field. As an extreme example, without such an equivariance incorporated in the architecture, a model trained on a dataset of red cars will not generalize to one of white cars. This motivates us to disentangle the color distribution from the identity (pseudo-"semantics") of the generated objects, as shown in Figure 5.3.

Our formulation is a soft analogy to UV mapping, where the lookup is done through an attention mechanism instead of texture coordinates. This approach additionally provides simple manipulation capabilities (see Figure 5.3). A useful property of our formulation is that the color mapping operator is linear w.r.t. the colors. It can be applied either *before* (in the radiance field sampler) or *after* the rendering operation (in the rendered multi-channel "semantic image"), since the rendering operation is also linear w.r.t. the colors. In a reconstruction scenario, this allows the end user to efficiently reproduce the color distribution of the input image with a single rendering pass. In Section 5.5 we show that, in addition to the useful manipulation properties, this module leads to an empirical advantage in the reconstruction task.

PATH LENGTH REGULARIZATION REVISITED. Initially proposed in StyleGAN2 [Kar+20b], this regularizer encourages the mapping between the latent space $\mathcal{W}$ and the output space $\mathcal{Y}$ to be orthogonal, which facilitates inversion (recovering the latent code $\mathbf{w}$ corresponding to a certain image via optimization). This is achieved by applying a gradient penalty to the Jacobian $\partial g(\mathbf{w})/\partial \mathbf{w}$. The use of path length regularization on the full backbone, however, is prohibitively expensive as this term requires double differentiation, and this feature was dropped in EG3D [Cha+22]. We propose to reinstate a more efficient variant of this regularizer which computes the path length penalty up to the three orthogonal planes, leaving the triplanar decoder unregularized. We find that this compromise provides the desired benefits without a significant added computational cost, as the main bottleneck is represented by the triplanar decoder, and enables us to greatly increase the learning rate during the inversion process (and in turn reduce the number of iterations).

5.3.2 *Bootstrapping and pose estimation*

Given a pretrained generator, it is in principle possible to invert it using one of the many techniques described in the literature for 2D images [Roi+22], which usually involve minimizing some pixel-level loss (e.g. L1 or VGG) w.r.t. the input latent code. For the 3D case, the minimization needs to be carried out over both the latent code and camera pose. In practice, however, recovering the camera pose is a highly non-convex problem that can easily get stuck in local minima. It is also crucial that the initial pose is

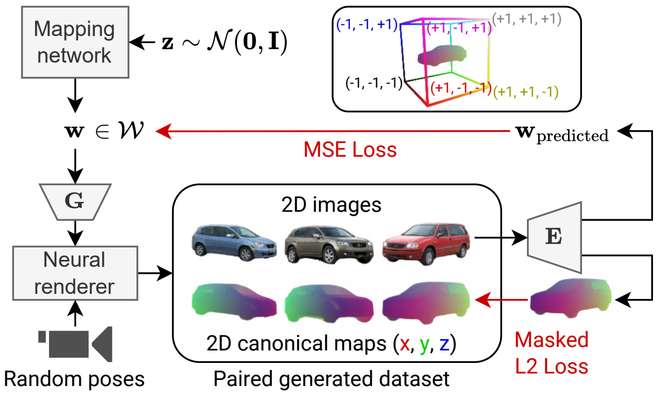

Figure 5.4: Data generation process for training the encoder (**E**). We randomly generate synthetic batches of images and associated 2D canonical maps. The encoder is then trained to predict the latent code and canonical map from the RGB image. We then use real images for inference. See also the bounding volume on top, which describes how colors should be interpreted.

"good enough", otherwise the latent code will converge to a degenerate solution. Therefore, most approaches [Cai+22; Cha+22] initialize the pose using an off-the-shelf pose estimator and only carry out the optimization w.r.t. the latent code. Moreover, existing approaches start from an average or random latent code [Cha+22; Zha+22], resulting in a slow convergence (often requiring hundreds of steps), which makes these methods less applicable to real-time scenarios. This motivates our hybrid inversion scheme, where an encoder produces a first guess of the latent code and pose, and these are *both* refined for a small number of iterations. Thanks to the ensuing acceleration, we can invert an image in as few as 10 optimization steps.

POSE ESTIMATION. In previous methods [Cai+22; GKM20; Kan+18; Li+20], poses are estimated by directly regressing the pose parameters (e.g. view matrix or quaternion/scale/translation). While this strategy can learn the task to some extent, it does not effectively incorporate the equivariances required by the problem (e.g. translation equivariance) and instead relies on learning them from the data, potentially generalizing poorly in settings other than simple synthetic datasets. More principled approaches can be found in the pose estimation literature, such as NOCS [Wan+19], which frames the problem as a regression of a *canonical map* (NOCS map) in image space, i.e. a 2D rendering of the (x, y, z) world-space coordinates of an object (Figure 5.4). The mapping is then inverted using a Perspective-n-Point (PnP) solver to recover the pose parameters. The main limitation of NOCS [Wan+19] is that it requires either ground-truth 3D meshes or hand-modeled synthetic meshes that are representative of the training dataset, since ground-truth canonical maps are not available on real datasets. By contrast, our availability of an object generator allows us to overcome this limitation, as we describe next.

TRAINING AND INFERENCE. The main idea underlying our approach – in contrast to NOCS [Wan+19] – is that we use data generated from our unconditional generator to train the encoder instead of handcrafted data. This allows us to obtain a mapping between latent codes and images, as well as pseudo-ground truth canonical maps that we can use for pose estimation. During training, we sample a minibatch of priors $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, feed them through the mapping network to obtain the latent codes $\mathbf{w} \in \mathcal{W}$, and generate the corresponding RGB images and canonical maps from randomly-sampled viewpoints[1]. Finally, we train the network (a SegFormer [Xie+21] segmentation network with a custom regression head) to predict the canonical map and the latent code $\mathbf{w}$ from the RGB image. Losses, detailed architecture, and hyperparameters are described in the Appendix 5.8.1. For inference, we feed a real image, convert the predicted canonical map to a point cloud, and run a PnP solver to recover all pose parameters (view matrix and focal length).

1 Rendering canonical maps requires only a trivial change to standard NeRF implementations, namely integrating the query coordinates (x, y, z) instead of the RGB channels.

5.3.3 *Reconstruction via hybrid GAN inversion*

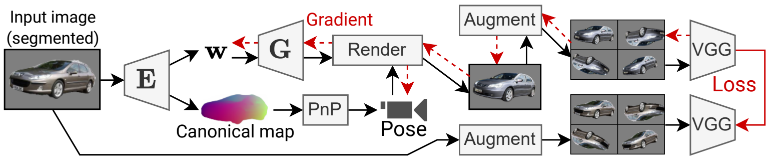

Figure 5.5: Hybrid inversion process. From the input image, the encoder **E** predicts an initial latent code **w** and a canonical map, the latter of which is used to recover the pose parameters through a PnP solver. Both **w** and the pose are then refined via optimization using a multi-crop VGG loss.

The final step of our pipeline is the refinement of the latent code and pose via gradient-based optimization (Figure 5.5). In this step, we found it beneficial to split the initial latent code **w** into a different vector for each layer, which we refer to as $\mathbf{w}^+ \in \mathcal{W}^+$. For a fixed number of steps N, we update $\mathbf{w}^+$ and the pose to minimize a reconstruction error between the rendered image and the input image. We experimented with various loss functions including MSE, L1 and a VGG perceptual loss [Zha+18b], finding that the former two lead to overly blurry results. Eventually, we settled on a VGG loss [Zha+18b] with random augmentations, where both the predicted and target images are randomly augmented with geometric image-space transformations (we use 16 augmentations and average their losses). This helps reduce the variance of the gradient, allowing us to further increase the learning rate. We also find that the pose parameterization is an important aspect to consider, and describe it in detail in the Appendix 5.8.1 (among additional details for this step).

5.4 EXPERIMENTAL SETTING

We compare against two main directions from the single-view 3D reconstruction literature: real images, following CMR [Kan+18], and synthetic images, following Pix2NeRF [Cai+22].

REAL IMAGES. Firstly, we adopt the evaluation methodology of CMR [Kan+18] and follow-up works [Bha+21; GKM20; Li+20; Zha+22], which focus on real datasets where ground-truth novel views are not available. These methods evaluate the mean IoU between the input images and the reconstructions rendered from the input view. While this metric describes how much the reconstruction matches the input image, it is limited since it does not evaluate how realistic the object looks from other viewpoints. Therefore, in the comparison to these works, we also include the FID [Heu+17] evaluated from random viewpoints, which correlates with the overall generative quality of the reconstructed objects. In this setting, we evaluate our approach on the standard datasets used in prior work – CUB Birds [Wah+11] and Pascal3D+ Cars [XMS14] – each of which comprise $\sim$5k training images and an official test split which we use for reporting. For the pose distribution used to train the unconditional generator, we rely on the poses estimated by CMR [Kan+18] using keypoints. It is worth noting that CMR uses a weak-perspective camera projection model. We found this appropriate for birds, which are often photographed from a large distance, but not for cars, which exhibit varying levels of perspective distortion. Therefore, we upgraded the camera model of P3D Cars to a full-perspective one as described in the Appendix 5.8.1.

EXTRA BASELINES. To further demonstrate the applicability of our method to real-world datasets, we establish new baselines on a variety of categories from ImageNet: cars, motorbikes, airplanes, as well as deformable categories such as zebras and elephants. For these classes, we use the splits from [Pav+21], which comprise 1-4k training images each (allowing us to assess how well our method fares on small datasets), and use the unsupervised pose estimation technique in [Pav+21] to obtain the pose distribution, which we also upgrade to a full-perspective camera model. Since no test split is available, we evaluate all metrics on the training split. Moreover, as we observe that the official test set of P3D Cars is too small ($\sim$200 images) to reliably estimate the FID, we construct another larger test set for P3D using non-overlapping images from the car class of ImageNet.

SYNTHETIC IMAGES. Secondly, we evaluate our approach on synthetic datasets: ShapeNet-SRN Cars & Chairs [Cha+15; SZW19], and CARLA [Dos+17]. We follow the experimental setting of Pix2NeRF [Cai+22], in which in addition to the FID from random views, pixel-level metrics (PSNR, SSIM) are also evaluated on ground-truth novel views from the test set. On these datasets, we also evaluate the pose estimation performance, as exact ground-truth poses are known. Following [Cai+22], we compute all metrics against a sample of 8k images from the test split, but use all training images. Although ground-truth novel views are available on ShapeNet, we only use such information for evaluation purposes and not for training.

IMPLEMENTATION DETAILS. We describe training hyperparameters as well as additional details in the Appendix 5.8.1.

5.5 RESULTS

INVERSION DYNAMICS AND SETTINGS. Before presenting our main results, we carry out a preliminary study on how to achieve the best speed on the hybrid inversion task. In Figure 5.6, we analyze the inversion dynamics under different gain factors for the learning rate of the latent code **w** (1x, 5x, 10x, 20x) along with a corresponding reduction in the number of optimization steps. When both path length regularization and color mapping are used, we find the dynamics to be almost linear up to a certain point. Both the FID (evaluated on random views) and PSNR (computed on the input view) improve monotonically, eventually reaching a "sweet spot" after which the FID starts degrading, indicating overfitting. When we remove these components, the inversion dynamics become less predictable and the overall performance is affected when higher gains are used. We also find that using a lower learning rate is generally better, but requires more iterations. As a result, we propose the following settings: a higher-quality but slower schedule, *Hybrid Slow*, with N=30 inversion steps at 5x gain, and *Hybrid Fast*, where we ramp up the gain to 20x and use only N=10 steps. We also experimented with higher gains (up to 50x), but could not get these to reliably converge. Furthermore, for

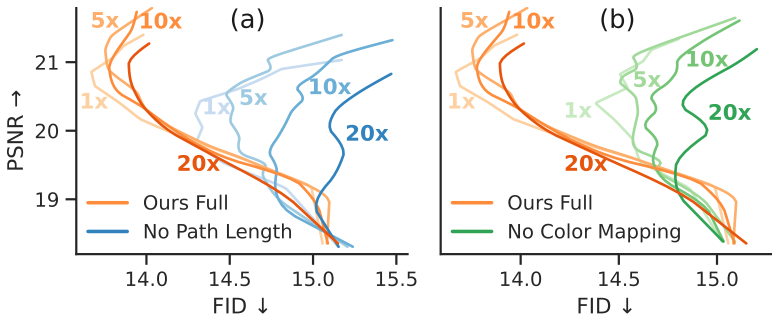

Figure 5.6: Inversion dynamics and ablations on P3D Cars on a larger test set from ImageNet, under different learning rate gains (1x, 5x, 10x, 20x) for the latent code **w**. All curves start from the bottom-right corner. When path length regularization is applied (**a**), the curves exhibit a higher linearity, which allows us to increase the learning rate while reducing the number of optimization steps. Conversely, when the regularizer is not adopted, the curves are more spaced apart and performance degrades quickly as the gain increases. Furthermore, our color mapping module (**b**) allows for a better reconstruction. We also identify an overfitting region, where the PSNR keeps increasing but the FID starts degrading, indicating that there is a trade-off between these metrics.

a fair comparison with works that are purely feed-forward-based, we also report a baseline with N=0, i.e. we evaluate the output of the encoder with no inversion.

QUANTITATIVE EVALUATION (REAL IMAGES). Table 5.1 (top) shows our main comparison on datasets of real images, following the CMR [Kan+18] protocol. On P3D Cars and CUB, our initial guess of the pose and latent code (N=0) already provides an advantage over existing approaches, with a 36% decrease in FID on CUB over the state-of-the-art, and a 9% increase in IoU on P3D Cars, despite our model not being trained to optimize the latter metric (unlike the other approaches, which are all encoder-based and include a supervised loss). We attribute this improvement to our more powerful NeRF-based representation (as opposed to sphere-topology triangle meshes used in prior works), as well as a better pose estimation performance. Following refinement via

	Pascal3D+ Cars		CUB Birds	
Method	IoU ↑	FID ↓	IoU ↑	FID ↓
CMR [Kan+18]	0.64	273.28	0.706	105.04
U-CMR [GKM20]	0.646	223.12	0.644	69.42
UMR [Li+20]	-	-	0.734	43.83
SDF-SRN [LWL20]	0.81	254.90	-	-
ViewGeneralization [Bha+21]	0.78	-	0.629	-
StyleGANRender [Zha+21b]	0.80	-	-	-
Ours Init. (N=0)	**0.883**	**75.90** (15.08)	**0.739**	**28.15**
MeshInv. (N=200) ($*$) (†) [Zha+22]	-	-	0.752	31.60
Ours Hybrid Slow (N=30) (†)	**0.920**	73.53 (14.36)	**0.844**	**24.70**
Ours Hybrid Fast (N=10) (†)	0.917	**73.12** (14.36)	0.835	25.65

	Car		Motorcycle		Airplane		Zebra		Elephant	
Method	IoU ↑	FID ↓	IoU ↑	FID ↓	IoU ↑	FID ↓	IoU ↑	FID ↓	IoU ↑	FID ↓
Init. N=0	0.933	9.88	0.804	40.65	0.749	**18.77**	0.724	**21.58**	0.781	107.34
Slow N=30 (†)	**0.953**	**8.77**	**0.851**	**38.6**	**0.813**	19.78	**0.802**	24.47	**0.848**	**99.77**
Fast N=10 (†)	0.952	8.91	0.85	39.72	0.805	21.33	0.793	26.41	0.845	104.12

Table 5.1: Evaluation on real datasets (CMR setting with predicted camera) on P3D/CUB (upper table) and ImageNet (bottom table). The first rows are purely feed-forward-based, while the remaining are inversion-based. All FIDs have been computed by us at 128×128 under uniform settings, wherever a public implementation was available. Note that the seemingly high FIDs on P3D are due to the small size of the test set (∼200 images), and therefore in parentheses we report an additional FID evaluated against a non-overlapping test set from ImageNet Cars. **Legend:** ($*$) Uses class-conditional model; (†) Uses optimization for N iterations.

hybrid inversion, performance is further boosted in as few as 10 steps. Finally, we also establish new baselines on categories from ImageNet (Table 5.1, bottom), demonstrating that our method is effective beyond benchmark datasets.

QUANTITATIVE EVALUATION (SYNTHETIC IMAGES). In Table 5.2, we further evaluate our approach against [Cai+22] on synthetic data. Again, even before applying hybrid inversion, we observe an improvement in the FID (-68% on chairs and -83% on CARLA) as well as in the novel-view evaluation (PSNR, SSIM). Applying hybrid inversion further widens this gap.

Method	SRN Cars			SRN Chairs			CARLA
	PSNR ↑	SSIM ↑	FID ↓	PSNR ↑	SSIM ↑	FID ↓	FID ↓
Pix2NeRF [Cai+22]	-	-	-	18.14	0.84	26.81	38.51
Ours Init. (N=0)	18.54	0.848	12.39	18.26	0.857	8.64	6.49
Ours Hybrid Slow (N=30)	**19.55**	**0.864**	**11.37**	**19.36**	**0.875**	**7.44**	**5.97**
Ours Hybrid Fast (N=10)	19.24	0.861	12.26	19.02	0.871	7.62	6.18

Table 5.2: Evaluation on synthetic datasets. All metrics are computed at 128×128 using *predicted* poses. PSNR and SSIM are evaluated on novel views (not available on CARLA), and the FID on random views. Since [Cai+22] is not evaluated on SRN Cars, we establish baselines on this category.

Method	SRN Cars ↓	SRN Chairs ↓	CARLA ↓
Pix2NeRF Encoder [Cai+22]	-	15.55°	4.23°
Direct pose regression	17.08°	19.51°	3.21°
Ours NOCS + PnP	**10.84°**	**7.29°**	**1.08°**

Table 5.3: Pose estimation accuracy (mean rotation error in degrees) on synthetic datasets, where ground-truth poses are available. All methods are feed-forward (no inversion). Results for [Cai+22] are computed after a rigid alignment to the ground-truth reference frame.

QUALITATIVE RESULTS. Figure 5.7 shows a side-by-side comparison to [GKM20; Kan+18; Li+20; Zha+22] on P3D/CUB, while Figure 5.8 shows a comparison to [Cai+22] on synthetic datasets. To further demonstrate the applicability of our approach to real-world images, in Figure 5.9 we display extra results on ImageNet. Furthermore, for our method, we also show the surface normals obtained by analytically differentiating the SDF. We refer the reader to the respective figures for a discussion of the advantages and shortcomings of our method. Finally, we include additional qualitative results in Appendix 5.8.2.2.

POSE ESTIMATION. We evaluate pose estimation in Table 5.3. For this experiment, we use synthetic datasets for which exact ground-truth poses are known. We compare our NOCS-inspired approach to two baselines: *(i)* direct regression of pose parameters (using a quaternion-based parameterization, see Appendix

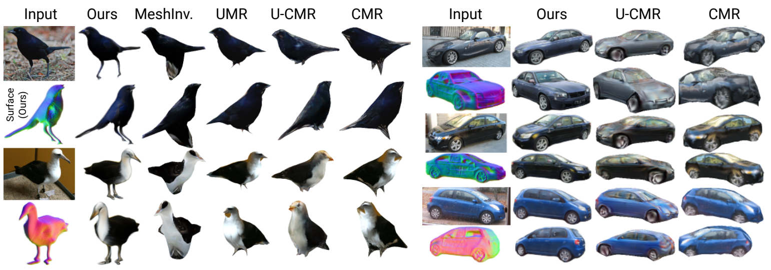

Figure 5.7: Qualitative results and side-by-side comparison on the test set of CUB (**left**) and Pascal3D+ Cars (**right**), at 128×128. The first row of each sample is rendered from the input viewpoint, whereas the second row illustrates a random view. Compared to the other works, which adopt a triangle mesh representation with a fixed topology, our SDF parameterization can model arbitrary topologies and can easily represent fine details such as the legs of the birds or the geometry of the cars, without enforcing any symmetry constraints. We observe occasional artifacts in the surface that are not visible from the RGB image, e.g. concave areas in the wings of birds or near the headlights of cars, which arise from the unconditional generator and can in principle improve with better supervision techniques.

5.8.1), where we keep the SegFormer backbone unchanged and only switch the output regression head for a fair comparison, and *(ii)* Pix2NeRF's encoder [Cai+22], which is trained to predict azimuth/elevation, a less expressive pose representation specific to the pose distribution of these datasets. We evaluate the mean rotation angle between predicted and ground-truth orientations, and observe that our NOCS-inspired approach achieves a significantly better error (53% and 74% reduction on chairs and CARLA, respectively) while being more general. Interestingly, our direct pose regression baseline achieves a similar performance to Pix2NeRF's encoder despite using a more expressive transformer architecture, suggesting that the main bottleneck lies in the pose representation itself and not in the architecture. As a side note, we also observe that the NOCS-based model converges much faster than the pose regression baseline, as the NOCS framework better incorporates equivariances to certain geometric transformations, while the baseline method has to learn them from the data.

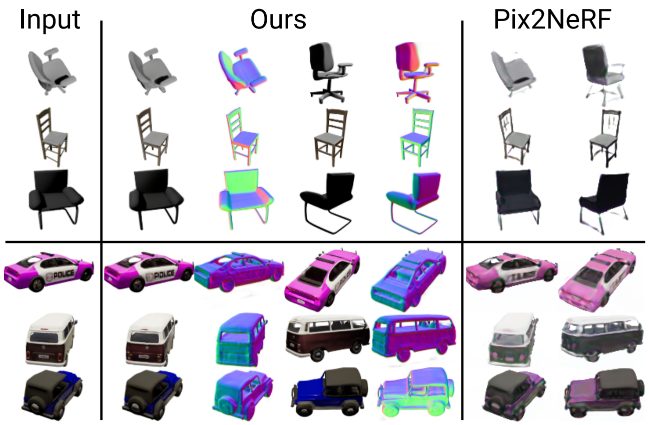

Figure 5.8: Qualitative results on synthetic datasets (test set of ShapeNet Chairs & CARLA) and side-by-side comparison to Pix2NeRF [Cai+22] on input and random views at 128×128. We observe that our method better predicts fine details such as the legs of the chairs, the text on cars, and color distributions.

ABLATIONS. In addition to those in Figure 5.6, we conduct further ablation experiments in Appendix 5.8.2.1. Among other things, we evaluate the impact of SDFs, compare our hybrid inversion method to an encoder baseline with a comparable architecture, and assess the impact of pose estimation.

CONVERSION TO TRIANGLE MESH. We can easily convert our reconstructions to triangle meshes in a principled way by extracting the 0-level set of the SDF and using marching cubes [LC87], as we show in the Appendix 5.8.2.2.

FAILURE CASES. We show and categorize these in Appendix 5.8.2.3. Furthermore, to guide future research, in Appendix 5.8.3 we discuss ideas that we explored but did not work out.

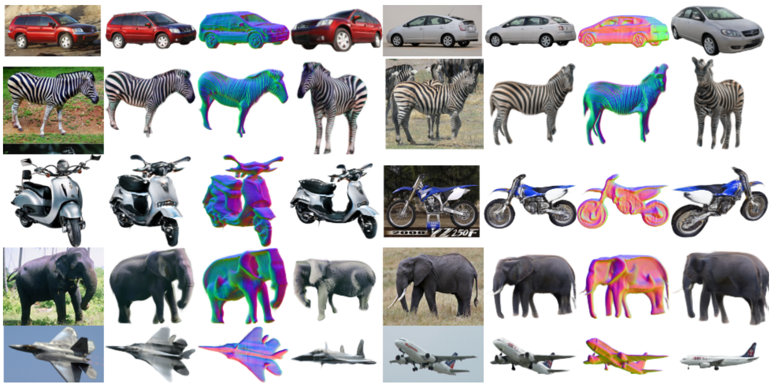

Figure 5.9: Additional qualitative results produced by our method on ImageNet. More results can be found in the appendix. Most classes learn a correct geometry despite being trained with only 1-4k images. We only observe some spurious concavities in the shape of the elephants, as well as a failure to correctly disentangle the zebra stripes from the surface.

5.6 SUMMARY

We introduced a framework for reconstructing shape, appearance, and pose from a single view of an object. Our approach leverages recent advances in NeRF representations and frames the problem as a 3D-aware GAN inversion task. In a hybrid fashion, we accelerate this process by learning an encoder that provides a first guess of the solution and incorporates a principled pose estimation technique. We achieve state-of-the-art performance on both synthetic and real benchmarks, and show that our approach is efficient (requiring as few as 10 inversion steps to reconstruct an image) and effective on small datasets.

5.7 FUTURE DEVELOPMENTS

In the future, we would like to scale to higher resolutions and improve the reconstructed surface quality, e.g. by leveraging semi-supervision on extra views or shape priors. We would also like to explore ways to automatically infer the pose distribution from the data.

5.8 APPENDIX

5.8.1 *Implementation details*

DATASET PREPARATION. For CUB [Wah+11], we use the segmentation masks and poses from CMR [Kan+18] estimated using structure-from-motion. These poses adopt a weak-perspective camera model, which we keep as-is (our neural renderer implementation supports multiple projection models). Similarly, for P3D Cars [XMS14], we use poses from CMR but obtain the segmentation masks using Mask R-CNN [He+17] as was done in previous work. Additionally, we upgrade its camera projection model to a full perspective model by freezing the rotations and re-estimating all the other parameters using the procedure described in the next paragraph. For ImageNet, we estimate poses from scratch using the same procedure and predict segmentation masks using PointRend [Kir+20]. Additionally, we augment all real datasets mentioned so far with horizontal flips. We do not augment synthetic datasets (CARLA [Dos+17] and ShapeNet [Cha+15]).

POSE ESTIMATION AND PARAMETERIZATION. While most neural renderers adopt *camera-to-world* view matrices (plus focal length), this representation is not necessarily the best for optimization purposes. Among various issues, we mention the necessity to enforce orthogonality constraints in the rotation matrix, a dependency between rotation and translation (which can be solved by switching to a *world-to-camera* representation), and an entanglement between translation and focal length (ideally, as depth increases, the learning rate for the translation needs to be amplified in order to keep a linear behavior in projective space). Therefore, for all our steps where pose optimization is involved (namely, the initial data preparation and the hybrid inversion step), we adopt a custom pose parameterization that tackles these issues and is easier to optimize, while being fully differentiable and convertible to view matrices for use in neural renderers. We also make sure that our neural renderer is fully differentiable w.r.t. the pose, even when coarse-fine importance sampling is used and view-dependent effects are enabled, as we found that existing implementations present gradient detachments in some nodes of the pipeline.

Our pose representation can be regarded as an augmentation of a weak-perspective camera model, and describes a *world-to-camera* transformation parameterized by a rotation $\mathbf{q} \in \mathbb{R}^4$ (a unit quaternion), a screen-space scale $s \in \mathbb{R}$, a screen-space translation $\mathbf{t}_2 \in \mathbb{R}^2$, and a perspective distortion factor $z_0 \in \mathbb{R}$. At runtime, we derive the focal length $f = 1 + \exp(z_0)$ and the 3D translation $\mathbf{t}_3 = [\mathbf{t}_2/s; f/s]$. Such a parameterization is equivalent to a full-perspective model, but results in more "linear" optimization dynamics.

For the datasets where we estimate the poses ourselves (ImageNet and, partially, P3D Cars), we use the template-based pose estimation technique described in [Pav+21] with our pose parameterization.

UNCONDITIONAL GENERATOR. We train the unconditional generator for 300k iterations, except for CUB and ImageNet elephants, where we use 200k. Similarly, we adopt R1 regularization on all datasets with $\gamma = 5$, except on elephants where we use $\gamma = 10$. Optimizer, learning rate, and batch size are the same as in [Cha+22]. We found it beneficial for stability to warm up the learning rate of both the generator and discriminator, starting from 1/10th of the specified value and linearly increasing it over 2000 iterations.

As in [Cha+22], we condition the discriminator on the pose, but we nonetheless observe that all our models converge even without such conditioning, although this sometimes leads to surface artifacts (such as objects appearing concave). On all datasets except ShapeNet, we enable adaptive discriminator augmentation (ADA) [Kar+20a], which reduces discriminator overfitting by enhancing its input images with differentiable augmentations. We only adopt geometric transformations (scale, translation, rotation). However, we observe that the implementation of ADA in [Cha+22] is not 3D-aware, as the augmentations are only carried out in image space, while the discriminator is conditioned on the original camera pose, leading to artifacts. We implemented a 3D-aware version of ADA where both the image and camera pose are augmented with the same transformation, and the discriminator is conditioned on the augmented pose.

SDF DETAILS. As for the SDF representation [Yar+21], the volume density is modulated by two learnable parameters $\alpha, \beta > 0$. Unlike [Yar+21], which ties $\alpha = \beta$ (according to Equation 5.1), we found it helpful for convergence to learn them separately. We initialize $\alpha = 1$ and $\beta = 0.1$, and clamp their lower bound to 10^{-3} for stability during training. Before training the unconditional model, we initialize the SDF to a unit sphere through optimization. We pre-train the model for 1000 iterations using the following loss:

$$\mathcal{L}_{\text{SDF}} = \mathbb{E}_{\mathbf{x}}\left[\left(d(\mathbf{x}) - (\|\mathbf{x}\| - 1)\right)^2\right]. \tag{5.3}$$

A visualization of the SDF can be seen in Figure 5.12 (right). For the Eikonal loss, we use a weight of 0.1 as recommended by [Yar+21].

VIEW-DEPENDENT EFFECTS. We also incorporate the ability to efficiently model view-dependent effects. The view direction of each pixel (which is constant across depth, and can therefore be computed only once per pixel) is processed through a small feed-forward network to produce a 32-dimensional vector, and summed with another 32-dimensional vector coming from the triplanar decoder. The result is then processed through a Leaky ReLU activation and another linear layer to produce the final output. This late fusion strategy ensures that memory consumption is minimal. We enable this feature only on CARLA, as ShapeNet does not have any specular reflections and the other datasets are too small to properly disentangle appearance and specular effects.

BOOTSTRAPPING AND POSE ESTIMATION. For the encoder architecture, we adopt SegFormer B5 [Xie+21], a recently-proposed transformer-based backbone for semantic segmentation. The output feature map from the backbone is connected to two heads: a fully-connected one that regresses the latent code $\mathbf{w}$ and a convolutional one that regresses the canonical map and the associated

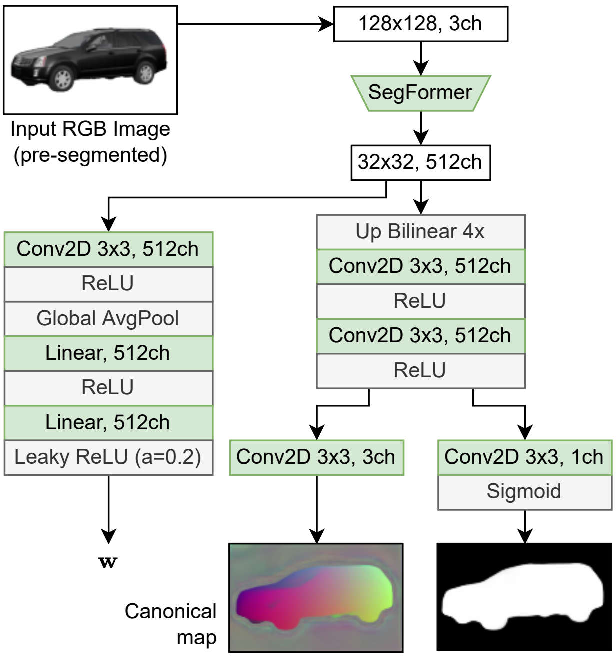

Figure 5.10: Architecture of the encoder used for bootstrapping the latent code and pose estimation. Note that we include a Leaky ReLU activation in the final layer of the latent code regressor, which mimics the behavior of the mapping network.

segmentation mask. The detailed architecture is shown in Figure 5.10. We initialize the backbone using ImageNet weights and train the model end-to-end for 120k iterations with a batch size of 32 samples, using Adam optimizer. We adopt an initial learning rate of 6e-5, which we decay to 6e-6 after 60k iterations. As for the losses, we use a simple mean squared error (MSE) loss for the latent code ($\mathcal{L}_{\text{latent}}$), an L1 loss for the segmentation mask ($\mathcal{L}_{\text{mask}}$), and a masked L2 loss (with square root) for the canonical map

($\mathcal{L}_{\text{map}}$), i.e. a rotation-invariant version of the L1 loss, for better robustness to artifacts coming from the generator:

$$\mathcal{L}_{\text{latent}} = \|\hat{\mathbf{w}} - \mathbf{w}\|^2, \tag{5.4}$$

$$\mathcal{L}_{\text{mask}} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} |\hat{m}_{i,j} - m_{i,j}|, \tag{5.5}$$

$$\mathcal{L}_{\text{map}} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} m_{i,j} \|\hat{\mathbf{p}}_{i,j} - \mathbf{p}_{i,j}\|, \tag{5.6}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{latent}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{map}} \tag{5.7}$$

where $\hat{\mathbf{w}}$ is the predicted latent code, $\hat{\mathbf{p}}_{i,j}$ is the predicted canonical map at the i, j image-space coordinates (a 3D vector for each position), $\mathbf{p}$ is the ground-truth one, $\hat{m}$ is the predicted mask, and m is the ground-truth mask. This contrasts NOCS [Wan+19], which frames the task as a classification problem using quantized coordinates. For inference, the regressed canonical map is thresholded using the predicted mask, converted to a dense point cloud, and used as the input for SQPnP [TL20], a fast PnP solver that retrieves a global optimum (we use the implementation in OpenCV [Bra00]). Since SQPnP – as most PnP methods – requires the focal length to be pre-determined, we select 10 representative focal lengths from the training set (one for each 10th percentile), run the algorithm for each, and select the solution with the lowest reprojection error.

GAN INVERSION. For the hybrid inversion step, we use Adam optimizer [KB14] with a base learning rate of 0.02, and optionally amplify the learning rate of the latent code $\mathbf{w}$ by a gain factor (as reported in the individual experiments). We additionally set $\beta_2 = 0.95$ for a faster reaction. We do not dynamically adjust the hyperparameters throughout the procedure. For the pose, we use our previously-described parameterization as this results in better optimization dynamics. We optimize the following objective:

$$\min_{\mathbf{w}, \mathbf{q}, s, \mathbf{t}_2, z_0} \frac{1}{K} \sum_{k=1}^{K} \text{LPIPS}(\mathbf{c}_{\text{pred}}^{[k]}, \mathbf{c}_{\text{gt}}^{[k]}), \tag{5.8}$$

where $\mathbf{c}_{\text{pred}}$ represents the predicted image, $\mathbf{c}_{\text{gt}}$ is the ground-truth one, k is the augmentation index (we use $K = 16$ augmentations), and the LPIPS operator [Zha+18b] describes the distance between the VGG embeddings of the two images. After each iteration, we reproject the pose parameters onto the valid set of constraints ($\mathbf{q}$ unit length).

EVALUATION DETAILS. For the evaluation on real datasets (P3D, CUB, ImageNet), we follow the protocol of [Pav+21; Pav+20b] and evaluate the FID on an empty background (value $= 0$, i.e. gray). All scores are evaluated at 128×128, using antialiased resampling (also referred to as *area* interpolation) if resizing is needed, as the FID is sensitive to this aspect. For the approaches we compare to, we compute their FID under the same settings by modifying their public implementations.

5.8.2 *Additional results*

5.8.2.1 *Ablation experiments*

ENCODER-BASED ARCHITECTURE. For our next experiment, we build an encoder-based variant of our architecture and switch to a conditional GAN setting. Instead of using a mapping network to map $\mathbf{z}$ to $\mathbf{w}$, we learn a convolutional encoder that takes a 2D image as input and directly predicts $\mathbf{w}$. We also experiment with various supervision strategies, including a dual discriminator (an unconditional one that discriminates random views plus a conditional one that discriminates the input view), and a single unconditional discriminator with an L1 or MSE loss to fit the input image, as in Pix2NeRF [Cai+22]. Based on early experiments, we found that the dual discriminator approach yields the best results (as it does not require balancing the losses), and we use this strategy throughout our ablations. Moreover, for a fair comparison, we use the same backbone for the conditional (encoder-based) and unconditional (inversion-based) experiments.

ENCODER- *VS* INVERSION-BASED BASELINES. In Figure 5.11 (left), we compare our hybrid inversion approach to the afore-

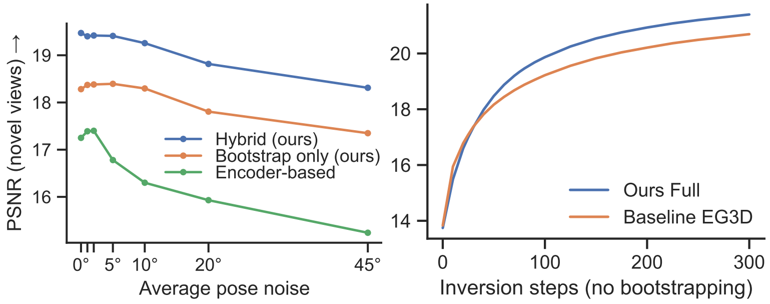

Figure 5.11: Additional ablations on ShapeNet Chairs, where we evaluate the PSNR on novel views from the test set. **Left:** comparison of our hybrid inversion approach (and initial bootstrapping without refinement) to an encoder-based baseline under simulated pose perturbations. **Right:** inversion on a vanilla EG3D backbone *vs* our proposed architecture. Since the goal of this experiment is to evaluate only the impact of the unconditional generator, we start from an average latent code (i.e. no bootstrapping) and use the ground-truth pose.

mentioned encoder-based baseline. We conduct this experiment on ShapeNet Chairs, where exact ground-truth poses are known. In this setting, we randomly perturb individual poses by injecting noise at different levels (from 0° to 45°) without altering the overall pose distribution, and study which approach is more robust to inaccurate poses as noise increases. Importantly, for a fair comparison to encoder methods (which are feed-forward), we also include a *bootstrap only* baseline where we do not refine the initial guess of our solution. We immediately observe that our *bootstrap only* baseline achieves a higher PSNR compared to the encoder-based approach. Furthermore, the performance of our approach decreases gracefully as noise increases, whereas with the encoder-based method we can observe a sharp degradation as early as 5°. Based on these findings, we conclude that inversion-based approaches are more appropriate for real datasets where poses are potentially inaccurate, as these methods rely on the overall pose distribution as opposed to the correctness of individual poses.

IMPACT OF THE BACKBONE. In Figure 5.11 (right), we assess the impact of the backbone on the final reconstruction result. We leave out some of our contributions (bootstrapping, pose estimation, and hybrid inversion) and purely focus on our proposed backbone components (SDF, color mapping, optimized path length regularization, as well as the minor improvements over [Cha+22]). To this end, we conduct a vanilla inversion experiments on ShapeNet Chairs, using ground-truth poses and starting from the "average" latent code in $\mathcal{W}$. We compare our full backbone to a vanilla EG3D, and observe an advantage when adopting our proposed changes.

SDF REPRESENTATION. Similar to Figure 5.6, we conduct an analysis of the inversion dynamics with and without our proposed SDF representation (Figure 5.12). We find that the optimization dynamics are similar, but the SDF baseline gets an FID boost owing to a better unconditional generator, in addition to the other practical benefits (e.g. ability to easily extract surface, normals, and mesh).

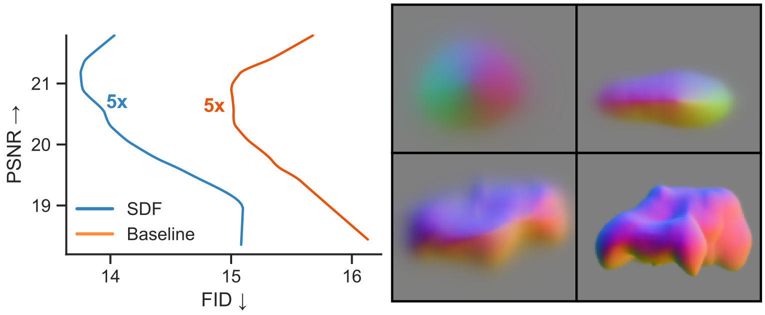

Figure 5.12: Left: impact of the SDF representation on the inversion dynamics (at gain 5x). As in Figure 5.6, the analysis is carried out on our larger P3D Cars test set. **Right:** spherical initialization of the SDF and its evolution as training progresses.

COLOR DISTRIBUTION DISENTANGLEMENT. To visually motivate our color mapping approach, we show examples of color disentanglement in Figure 5.13. When our color mapping network is used, the object identity is fully disentangled from its color distribution. By contrast, attempting the same on a vanilla EG3D architecture is unsuccessful, even when adopting techniques such as *style mixing*.

POSE ESTIMATION. Pose prediction is a useful feature for AR applications and real-world datasets, where ground-truth poses are imprecise or not available. As such, it is not meant to improve performance, as it actually makes the learning task harder. Nonetheless, it is interesting to evaluate its effect on quantitative metrics. In Table 5.4, we conduct an ablation experiment on a dataset of real images (P3D Cars) where we turn off pose prediction and use ground-truth poses from the dataset (which are imprecise). As expected, we find that qualitative metrics (FID) are mostly unaffected, whereas the IoU is degraded when switching to poses from the dataset, regardless of whether inversion is used. This confirms that, on real datasets such as P3D, individual poses are inaccurate, but our generative framework is robust to them as it relies on the overall pose distribution.

		Pascal3D+ Cars	
		IoU ↑	FID ↓
Ours (predicted poses)	(N=0)	**0.883**	75.90 (15.08)
Ours (dataset poses)	(N=0)	0.803	73.20 (16.39)
Ours (predicted poses)	(N=30)	**0.920**	73.53 (14.36)
Ours (dataset poses)	(N=30)	0.802	72.22 (15.10)

Table 5.4: Ablation experiment on pose prediction (P3D Cars dataset). The first two rows are purely feed-forward-based, while the remaining are inversion-based. In parentheses, we also report the FID on our larger test set from ImageNet.

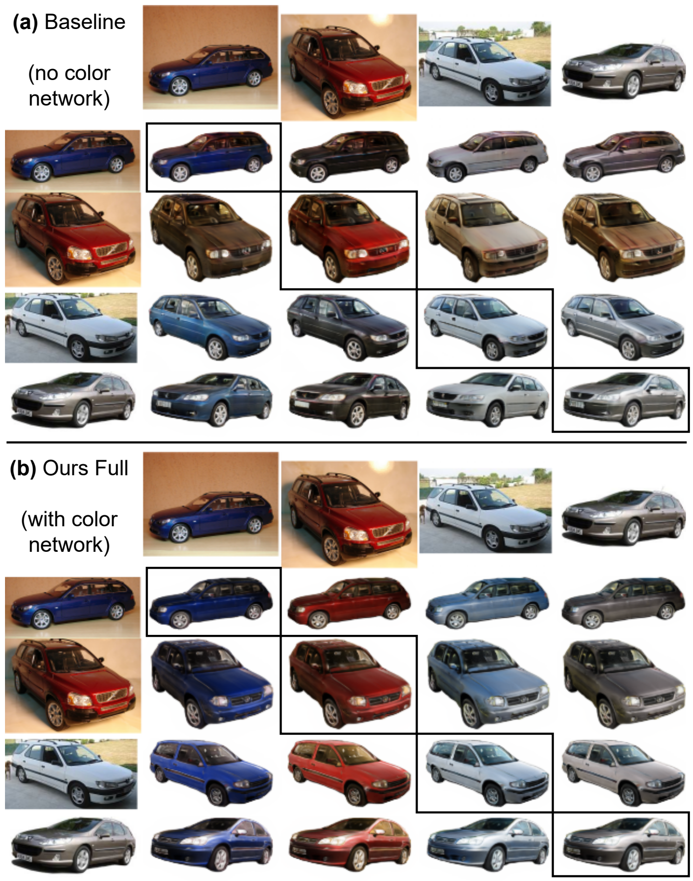

Figure 5.13: Disentanglement of the color distribution from the object identity. **(a)** In the baseline network without color mapping (EG3D), we attempt to achieve disentanglement via *style mixing*, i.e. we split the latent code w into two sections (before and after the 8th layer) and mix it between the two object instances. Although this leads to some variation in color, we find that disentanglement is not properly achieved. **(b)** With our color mapping technique, color and object identity are fully disentangled. When two different latent codes are combined, it is possible to "borrow" the color distribution from another image in a realistic way.

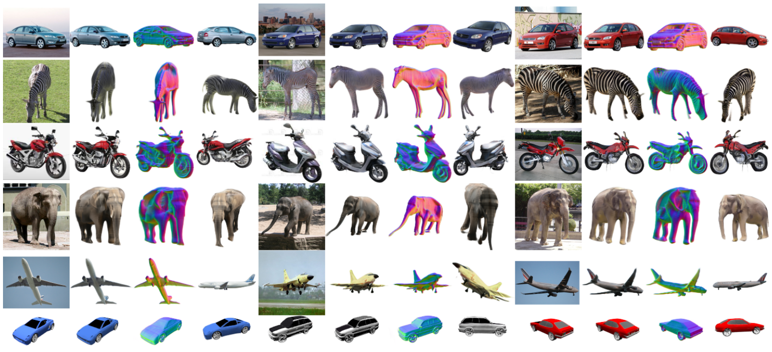

Figure 5.14: Additional qualitative results produced by our method on ImageNet (top rows) and ShapeNet Cars (last row).

5.8.2.2 *Qualitative results*

ADDITIONAL QUALITATIVE RESULTS. Following the format in the main text, we report extra qualitative results for all datasets in Figure 5.14 (novel results on ImageNet and ShapeNet Cars), Figure 5.15 (ShapeNet Chairs & CARLA, and comparison to Pix2NeRF [Cai+22]), and Figure 5.19 (CUB and P3D Cars, including comparison to prior work).

CONVERSION TO TRIANGLE MESH. Our adoption of an SDF representation allows us to easily extract a triangle mesh from a generated object (Figure 5.16). We first quantize the SDF to a fixed-size grid, and then extract its 0-level set (i.e. zero-crossings) via marching cubes [LC87], obtaining a set of vertices and triangles. Finally, we sample colors from the radiance field by querying the network at the locations specified by the vertex positions.

5.8.2.3 *Limitations and failure cases*

In this section, we highlight the most common failure modes of our method and attempt to categorize them in common patterns.

SHAPE ARTIFACTS. In some cases, we observe that reconstructed shapes present some artifacts, even though the appear-

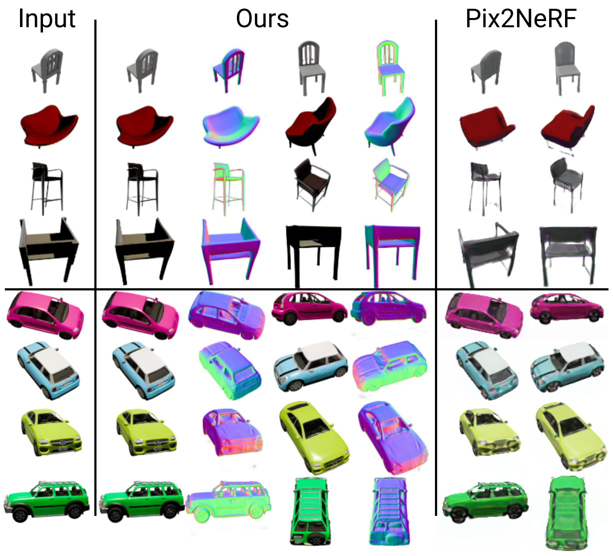

Figure 5.15: Additional qualitative results on synthetic datasets (test set of ShapeNet Chairs & CARLA) and side-by-side comparison to Pix2NeRF [Cai+22] on input and random views at 128×128.

ance of the object looks correct when rendered. For instance, in animals, features such as the beaks of the birds are sometimes bifurcated (Figure 5.17, top). We attribute this issue to a lack of density in some areas of the pose distribution of the dataset, e.g. most birds are observed from the side, but rarely from the front. On some small datasets such as zebras and elephants – which comprise only 1.7k and 1.4k images respectively – we also observe concavities (see Figure 5.9 in the main text), and in the specific case of zebras, a failure to disentangle the stripes from the shape. We expect these entanglement issues to improve with larger datasets.

POSE ESTIMATION ERRORS. More rarely, failures are caused by inaccurate pose estimation. If the initial estimated pose is too far from the true one, the optimizer can get stuck in a local minima and cause the reconstructed surface to become distorted (Figure 5.17, bottom). We also notice that pose estimation is more

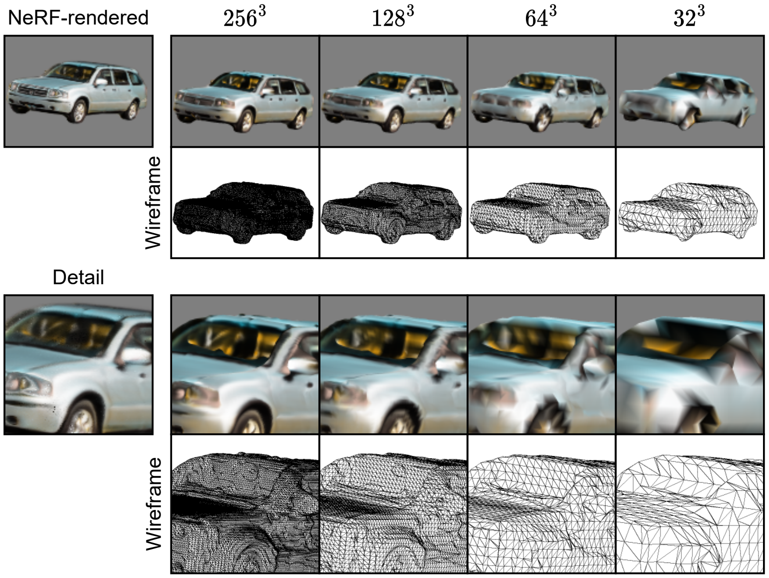

Figure 5.16: Extraction of a colored triangle mesh from an SDF. On the left, we show a car rendered using a neural renderer as well as zoomed viewpoint. On the right, we show the corresponding triangle mesh (including wireframe visualization) at different quantization steps.

challenging on examples with "extreme" postures, e.g. open wings in birds and head flexion in zebras, but also observe that many of these cases are handled correctly (Figure 5.18). Most likely, the failure cases are due to these poses being underrepresented in the unconditional generator, and are not due to a limitation of the pose estimation framework itself.

INCOMPLETE INVERSION. For some hard examples, the encoder might return an initial solution which is far from the optimum, which in turn needs to be optimized for longer to correctly match the input image. Since we adopt a fixed schedule (i.e. the same number of optimization steps for all images), some images may only be partially inverted. As part of future work, this issue could be mitigated by using an adaptive optimization schedule that varies for each sample.

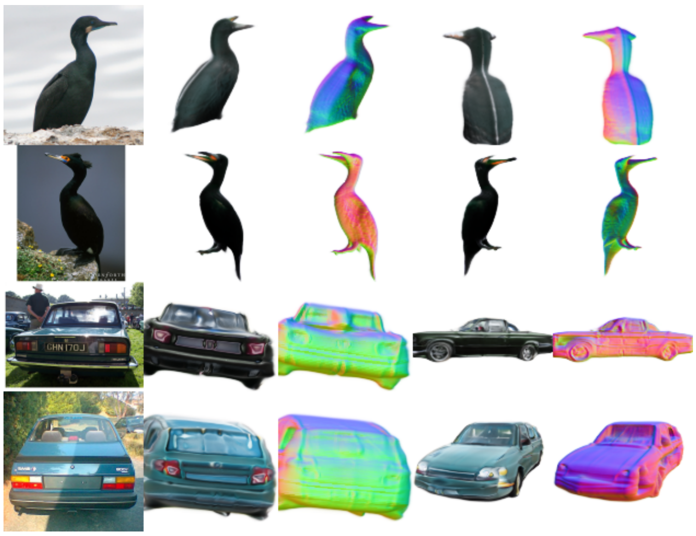

Figure 5.17: Most common modes of failure on CUB Birds and P3D Cars. On CUB (first two rows), we sometimes observe a "split beak" phenomenon. On P3D (last two rows), in the rare instances the pose is estimated imprecisely, the optimizer can get stuck in a local minima and lead to a distorted reconstruction.

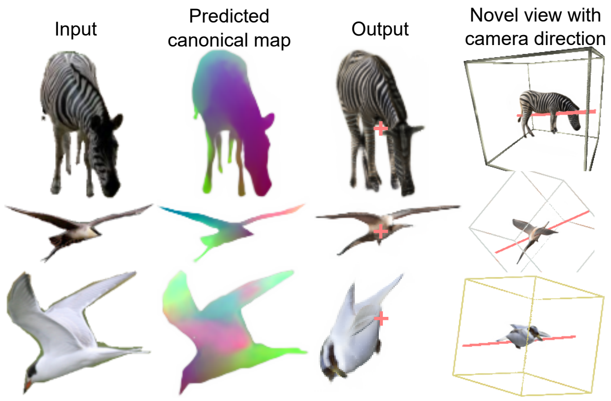

Figure 5.18: Our pose estimation approach can correctly handle "extreme" postures (top and middle), although some failure cases are still possible (bottom).

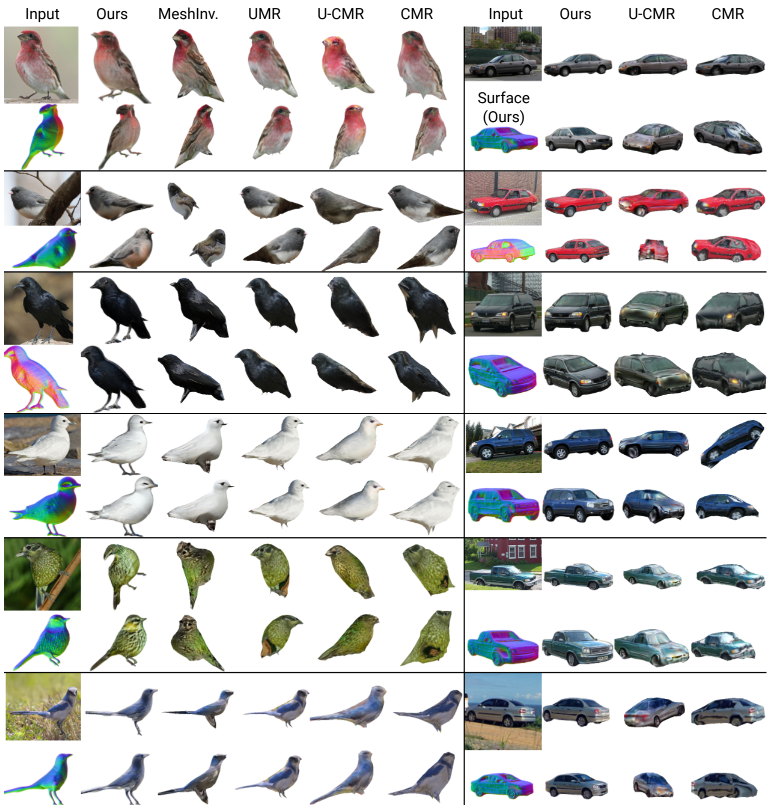

Figure 5.19: Additional qualitative results and side-by-side comparison on the test set of CUB (**left**) and Pascal3D+ Cars (**right**), at 128×128. The first row of each sample is rendered from the input viewpoint, whereas the second row illustrates a random view.

5.8.3 *Negative results*

Throughout the development of our method, we experimented with various techniques drawing inspiration from the literature on GANs and representation learning. To guide further research in this area, we provide a list of ideas we explored but did not work out as expected.

- We initially experimented with various NeRF representations, including MLP-based, voxel-based, and triplanar-based. We eventually settled with the triplanar representation of [Cha+22] since it was as expressive as the other ones but more efficient.

- Our initial attempts at solving the reconstruction task used an encoder-based approach with multiple discriminators. While this was appropriate for synthetic data, we quickly found out that it was not robust on real datasets with imprecise poses, which is also one of the main motivating factors for our approach. Based on the intuition that the issue might have been caused by the limited expressivity of the encoder, we tried to replace the encoder with an embedding layer that learned a different latent code for each instance (similar to [JA21; Reb+22], but using a GAN framework with multiple discriminators), essentially decoupling the impact of the encoder architecture from that of the learning framework. In this setting, we found that the issue was not resolved, which prompted us to explore other ideas.

- Before settling with our hybrid inversion framework, we also explored techniques from the representation learning literature, such as bidirectional GANs (BiGAN) [DKD17]. In this setting, the encoder is not connected to the generator and the learning signal comes from a joint discriminator. Our expectation was that this setting would make the approach less reliant on precise poses, but we found that the reconstructions did not mirror the input images closely enough.

6 CONCLUSION

Throughout this dissertation, we explored the field of generative models through the lens of structured representations. The goal of our research was to investigate ways to make image synthesis models more controllable, more interpretable, and more useful for downstream applications. To this end, we proposed techniques to generate complex scenes using structured input representations such as textual descriptions, high-level attributes, and sketches based on *sparse masks* (Chapter 2). Afterwards, we presented a series of approaches to generate textured 3D meshes using supervision from real-world 2D datasets (Chapter 3, Chapter 4), and suggested that these types of generators will become increasingly common in the future, as they are useful for creating interactive content in AR/VR applications. Finally, we demonstrated that these generators can be employed in applications beyond content creation. In particular, in Chapter 5, we leveraged 3D object generators and recently-proposed NeRF representations to tackle the task of single-view 3D reconstruction, and achieved state-of-the-art results across both qualitative and quantitative metrics.

The field of image synthesis is expanding rapidly, yet it still faces a series of unresolved issues. Despite recent technical advances in neural architectures – which have resulted in nearly photorealistic quality – state-of-the-art approaches still exhibit trivial failure cases that constrain their controllability. These are particularly evident with complex prompts that consist of many interacting objects. Furthermore, although the issue of biases is perhaps less explored in image synthesis than in natural language processing (NLP) [Bol+16], it is an important concern. Synthesis models often reflect or amplify societal biases and stereotypes present in the data [CZB22], posing risks for sensitive applications. Future research is expected to focus on mitigating some of these issues. As for the generation of 3D content, while recent approaches (including the ones we propose in this dissertation) show very promising results, the data assumptions required by these models can restrict their application in some cases. One such example is the knowledge of the pose distribution of the dataset, which is not easy to estimate on real-world datasets. Therefore, further research in the area of 3D vision is likely to concentrate on relaxing these assumptions. In this area, other promising research directions include the adoption of diffusion models for 3D synthesis tasks, as well as the ability to generate complex 3D scenes as opposed to individual objects.

Finally, we also note that these models are undergoing a rapid increase in complexity, often necessitating billions of parameters, and are trained on massive datasets that are not readily accessible. Consequently, the ability to train such models is concentrated among a limited number of parties with adequate resources. In light of these constraints, a further avenue of investigation involves improving the data-efficiency of these models as well as their performance in restricted hardware settings.

We conclude this dissertation with a note on the societal implications of this rapidly-growing field of research. Owing to their effectiveness, generative models have also been the subject of significant controversy in recent years, due to the ethical implications that they raise. Potential concerns include the misuse of these models to create misleading information and fake content (*DeepFakes*). Some artists and content creators also worry that generative models may be used to copy their work, without being able to properly protect their intellectual property rights. Nonetheless, this field is evolving rapidly and it is inevitable that generative models will become increasingly ubiquitous in the future, as they present a tremendous potential for advancing a wide range of industries. It is therefore important to guarantee that these models are deployed responsibly in the real world. For a more in-depth discussion of this subject, we refer the reader to [CC19].

BIBLIOGRAPHY

[Ach+18] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. "Learning Representations and Generative Models for 3D Point Clouds." In: *International Conference on Machine Learning*. 2018, pp. 40–49 (cit. on p. 57).

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 214–223 (cit. on p. 6).

[Bal+18] Elena Balashova, Vivek Singh, Jiangping Wang, Brian Teixeira, Terrence Chen, and Thomas Funkhouser. "Structure-aware shape synthesis." In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 140–149 (cit. on p. 57).

[Bar+21] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5855–5864 (cit. on pp. 123, 125).

[Bha+21] Anand Bhattad, Aysegul Dundar, Guilin Liu, Andrew Tao, and Bryan Catanzaro. "View generalization for single image textured 3d models." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6081–6090 (cit. on pp. 72, 124, 135, 138).

[Bol+16] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016) (cit. on p. 159).

[Bra00] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000) (cit. on p. 147).

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis." In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on pp. 7, 18, 20, 23, 50, 54, 66, 68, 74, 84, 85).

[Bro+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 1877–1901 (cit. on p. 1).

[CUF18] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "COCO-Stuff: Thing and stuff classes in context." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 20, 24, 25, 31, 32).

[Cai+22] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. "Pix2NeRF: Unsupervised Conditional pi-GAN for Single Image to Neural Radiance Fields Translation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3981–3990 (cit. on pp. 123, 125, 126, 132, 134, 136, 138–141, 148, 153, 154).

[Cha+21] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5799–5809 (cit. on pp. 12, 104, 123, 126, 127).

[Cha+22] Eric R. Chan et al. "Efficient Geometry-aware 3D Generative Adversarial Networks." In: *CVPR*. 2022 (cit. on pp. 1, 12, 104, 123, 126–129, 131, 132, 144, 150, 158).

[Cha+15] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015 (cit. on pp. 136, 143).

[Che+22] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. "TensoRF: Tensorial radiance fields." In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer. 2022, pp. 333–350 (cit. on pp. 11, 12).

[Che+18a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40.4 (2018), pp. 834–848 (cit. on pp. 21, 34).

[Che+20] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. "Generative pretraining from pixels." In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 1691–1703 (cit. on p. 38).

[Che+21a] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. "WaveGrad: Estimating Gradients for Waveform Generation." In: *International Conference on Learning Representations (ICLR)*. 2021 (cit. on pp. 1, 7).

[CK17] Qifeng Chen and Vladlen Koltun. "Photographic image synthesis with cascaded refinement networks." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1511–1520 (cit. on pp. 19, 21).

[Che+19] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. "Learning to predict 3d objects with an interpolation-based differentiable renderer." In: *Neural Information*

Processing Systems. 2019, pp. 9605–9616 (cit. on pp. 10, 55, 57, 58, 61, 66, 77, 79, 85–87, 119, 124).

[Che+21b] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. "DIB-R++: learning to predict lighting and material with a hybrid differentiable renderer." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22834–22848 (cit. on pp. 124, 125).

[Che+15] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. "Microsoft COCO captions: Data collection and evaluation server." In: *arXiv preprint arXiv:1504.00325* (2015) (cit. on pp. 20, 32).

[Che+18b] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. "Sequential attention GAN for interactive image editing via dialogue." In: *arXiv preprint arXiv:1812.08352* (2018) (cit. on p. 23).

[CC19] Bobby Chesney and Danielle Citron. "Deep fakes: A looming challenge for privacy, democracy, and national security." In: *Calif. L. Rev.* 107 (2019), p. 1753 (cit. on p. 160).

[CZB22] Jaemin Cho, Abhay Zala, and Mohit Bansal. "DALL-Eval: Probing the reasoning skills and social biases of text-to-image generative transformers." In: *arXiv preprint arXiv:2202.04053* (2022) (cit. on p. 159).

[Cho+16] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction." In: *European conference on computer vision*. Springer. 2016, pp. 628–644 (cit. on p. 56).

[De +17] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. "Modulating early visual processing by language." In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017) (cit. on p. 8).

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pp. 248–255 (cit. on pp. 65, 85, 98, 104).

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 29, 37).

[Dha+20] Ankit Dhall, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause. "Hierarchical image classification using entailment cone embeddings." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2020, pp. 836–837 (cit. on p. viii).

[DN21] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat GANs on image synthesis." In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), pp. 8780–8794 (cit. on pp. 5, 7).

[DKD17] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial Feature Learning." In: *International Conference on Learning Representations*. 2017 (cit. on p. 158).

[Dos+17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator." In: *Conference on robot learning*. PMLR. 2017, pp. 1–16 (cit. on pp. 136, 143).

[Dos+21] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *International Conference on Learning Representations (ICLR)*. 2021 (cit. on p. 37).

[DP22] Shivam Duggal and Deepak Pathak. "Topologically-aware deformation fields for single-view 3D reconstruction." In: *Proceedings of the IEEE/CVF Confer-*

ence on Computer Vision and Pattern Recognition. 2022, pp. 1536–1546 (cit. on p. 125).

[Dug+22] Shivam Duggal, Zihao Wang, Wei-Chiu Ma, Sivabalan Manivasagam, Justin Liang, Shenlong Wang, and Raquel Urtasun. "Mending neural implicit modeling for 3d vehicle reconstruction in the wild." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 1900–1909 (cit. on p. 126).

[Dun+22] Aysegul Dundar, Jun Gao, Andrew Tao, and Bryan Catanzaro. "Fine detailed texture learning for 3d meshes with generative models." In: *arXiv preprint arXiv:2203.09362* (2022) (cit. on p. 72).

[Or-+22] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. "Stylesdf: High-resolution 3d-consistent image and geometry generation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13503–13513 (cit. on pp. 127, 128).

[El-+19] Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kahou, Yoshua Bengio, and Graham W. Taylor. "Tell, Draw, and Repeat: Generating and Modifying Images Based on Continual Linguistic Instruction." In: *IEEE International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 23).

[Eng+19] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. "GANSynth: Adversarial Neural Audio Synthesis." In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 8).

[FSG17] Haoqiang Fan, Hao Su, and Leonidas J Guibas. "A point set generation network for 3d object reconstruction from a single image." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 605–613 (cit. on pp. 56, 124).

[Fri+22] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. "Plenoxels: Radiance fields without neural networks." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5501–5510 (cit. on p. 12).

[GWM18] Matheus Gadelha, Rui Wang, and Subhransu Maji. "Multiresolution tree networks for 3d point cloud processing." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 103–118 (cit. on p. 57).

[Gao+22] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images." In: *Advances In Neural Information Processing Systems*. 2022 (cit. on pp. 12, 127).

[Gao+20] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. "The pile: An 800GB dataset of diverse text for language modeling." In: *arXiv preprint arXiv:2101.00027* (2020) (cit. on p. 1).

[Gir+16] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. "Learning a predictable and generative vector representation for objects." In: *European Conference on Computer Vision*. Springer. 2016, pp. 484–499 (cit. on pp. 56, 57, 122, 124).

[GKM20] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. "Shape and viewpoint without keypoints." In: *European Conference on Computer Vision*. Springer. 2020, pp. 88–104 (cit. on pp. 87, 91, 112, 123, 124, 132, 135, 138, 139).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Neural Information Processing Systems*. 2014, pp. 2672–2680 (cit. on pp. 6, 18, 54, 123).

[Gu+21] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. "Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis." In: *arXiv preprint arXiv:2110.08985* (2021) (cit. on p. 127).

[GDG19] Agrim Gupta, Piotr Dollar, and Ross Girshick. "LVIS: A Dataset for Large Vocabulary Instance Segmentation." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5356–5364 (cit. on p. 37).

[Gwa+17] JunYoung Gwak, Christopher B Choy, Manmohan Chandraker, Animesh Garg, and Silvio Savarese. "Weakly supervised 3d reconstruction with adversarial constraint." In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 263–272 (cit. on p. 56).

[HTM17] Christian Häne, Shubham Tulsiani, and Jitendra Malik. "Hierarchical surface prediction for 3d object reconstruction." In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 412–420 (cit. on pp. 56, 122, 124).

[Här+20] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. "GANSpace: Discovering Interpretable GAN Controls." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. 2020, pp. 9841–9850 (cit. on p. 102).

[Har96] John C Hart. "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces." In: *The Visual Computer* 12.10 (1996), pp. 527–545 (cit. on p. 10).

[He+17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2961–2969 (cit. on pp. 32, 58, 65, 143).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on p. 11).

[He+19] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. "Attgan: Facial attribute editing by only changing what you want." In: *IEEE Transactions on Image Processing* (2019) (cit. on p. 18).

[HF18] Paul Henderson and Vittorio Ferrari. "Learning to Generate and Reconstruct 3D Meshes with only 2D Supervision." In: *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*. BMVA Press, 2018, p. 1 (cit. on p. 124).

[HF19] Paul Henderson and Vittorio Ferrari. "Learning single-image 3D reconstruction by generative modelling of shape, pose and shading." In: *International Journal of Computer Vision* (2019), pp. 1–20 (cit. on p. 87).

[HTL20] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. "Leveraging 2D Data to Learn Textured 3D Mesh Generation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on pp. 57, 85, 126).

[HMR19] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. "Escaping Plato's Cave: 3D Shape From Adversarial Rendering." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9984–9993 (cit. on p. 56).

[Heu+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs trained by a two time-scale update rule converge to a local Nash equilibrium." In: *Neural Information Processing Systems*. 2017, pp. 6626–6637 (cit. on pp. 18, 20, 32, 47, 54, 55, 64, 66, 85, 97, 135).

[HHW19] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. "Generating Multiple Objects at Spatially Distinct Locations." In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 21).

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851 (cit. on pp. 5, 38, 125).

[Hon+18] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. "Inferring semantic layout for hierarchical text-to-image synthesis." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7986–7994 (cit. on pp. 21, 47, 54).

[Hu+18] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. "Learning to segment every thing." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4233–4241 (cit. on pp. 32, 87, 88, 106).

[HB17] Xun Huang and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1501–1510 (cit. on p. 8).

[Hun+19] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. "Scops: Self-supervised co-part segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 869–878 (cit. on pp. 87, 106).

[ID18] Eldar Insafutdinov and Alexey Dosovitskiy. "Unsupervised learning of shape and pose with differentiable point clouds." In: *Advances in Neural Information Processing Systems*. 2018, pp. 2802–2812 (cit. on p. 56).

[Iso+17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. "Image-to-image translation with conditional adversarial networks." In: *IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR). 2017, pp. 1125–1134 (cit. on pp. 2, 7, 19–21, 24, 54, 63, 75).

[Jai+22] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. "Zero-shot text-guided object generation with dream fields." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 867–876 (cit. on p. 72).

[JA21] Wonbong Jang and Lourdes Agapito. "Codenerf: Disentangled neural radiance fields for object categories." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12949–12958 (cit. on pp. 125, 158).

[JGF18] Justin Johnson, Agrim Gupta, and Li Fei-Fei. "Image generation from scene graphs." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1219–1228 (cit. on pp. 21, 47, 54).

[Jos+19] KJ Joseph, Arghya Pal, Sailaja Rajanala, and Vineeth N Balasubramanian. "C4Synth: Cross-Caption Cycle-Consistent Text-to-Image Synthesis." In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 358–366 (cit. on p. 23).

[Kan+18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. "Learning Category-Specific Mesh Reconstruction from Image Collections." In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 57, 58, 60, 61, 65, 67, 70, 77, 80, 87, 98, 99, 112, 123, 124, 132, 134, 135, 137–139, 143).

[Kar+18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of GANs for improved quality, stability, and variation." In: *International Conference on Learning Representations (ICLR)*. 2018 (cit. on pp. 6, 18, 21, 54, 55, 66).

[Kar+20a] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. "Training generative adversarial networks with limited data." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12104–12114 (cit. on pp. 128, 144).

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4401–4410 (cit. on pp. 2, 18, 54, 66, 70, 84).

[Kar+20b] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Analyzing and improving the image quality of stylegan." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8110–8119 (cit. on pp. 1, 2, 84, 128, 131).

[KUH18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3D mesh renderer." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3907–3916 (cit. on pp. 10, 55, 57–59, 77, 87, 124).

[Kha+22] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. "CLIP-mesh: Generating textured meshes from text using pre-trained image-text models." In: *arXiv preprint arXiv:2203.13333* (2022) (cit. on p. 72).

[KLH18] Yannic Kilcher, Aurelien Lucchi, and Thomas Hofmann. "Semantic Interpolation in Implicit Models." In: *International Conference on Learning Representations (ICLR)*. 2018 (cit. on pp. 34, 43).

[KW14] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes." In: *International Conference on Learning Representations (ICLR)*. 2014 (cit. on pp. 4, 18, 57).

[KB14] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In: *International Conference on Learning Representions (ICLR)*. 2014 (cit. on pp. 65, 91, 108, 109, 112, 147).

[Kir+20] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. “Pointrend: Image segmentation as rendering.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9799–9808 (cit. on pp. 88, 106, 127, 143).

[Kir+15] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. “Skip-thought vectors.” In: *Neural Information Processing Systems*. 2015, pp. 3294–3302 (cit. on p. 22).

[Kri+17] Ranjay Krishna et al. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations.” In: *International Journal of Computer Vision (IJCV)* 123.1 (2017), pp. 32–73 (cit. on pp. 20, 32, 106).

[Kul+20] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. “Articulation-aware canonical surface mapping.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 452–461 (cit. on pp. 101, 107).

[Li+19] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. “Controllable Text-to-Image Generation.” In: *Neural Information Processing Systems*. 2019 (cit. on p. 54).

[Li+20] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. “Self-supervised single-view 3D reconstruction via semantic consistency.” In: *European Conference on Computer Vision*. Springer. 2020, pp. 677–693 (cit. on pp. 87, 93, 112, 123, 124, 132, 135, 138, 139).

[LY17] Jae Hyun Lim and Jong Chul Ye. “Geometric GAN.” In: *arXiv preprint arXiv:1705.02894* (2017) (cit. on p. 6).

[LWL20] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. “Sdf-srn: Learning signed distance 3d object reconstruction from static images.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11453–11464 (cit. on pp. 125, 138).

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft COCO: common objects in context.” In: *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 740–755 (cit. on pp. 20, 23, 32, 88, 106).

[Liu+19a] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. “Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer.” In: *International Conference on Learning Representations* (2019) (cit. on pp. 85, 89, 98).

[Liu+19b] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. “Soft rasterizer: A differentiable renderer for image-based 3d reasoning.” In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 7708–7717 (cit. on pp. 10, 55, 57, 119, 124).

[Loc+19] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. “Challenging common assumptions in the unsupervised learning of disentangled representations.” In: *International Conference on Machine Learning (ICML)*. 2019 (cit. on pp. 23, 54).

[LB14] Matthew M Loper and Michael J Black. “OpenDR: An approximate differentiable renderer.” In: *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 154–169 (cit. on pp. 10, 55, 57, 124).

[LC87] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm.” In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169 (cit. on pp. 141, 153).

[Mao+17] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. "Least squares generative adversarial networks." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2794–2802 (cit. on p. 6).

[MC09] Manuel Marques and João Costeira. "Estimating 3D shape from degenerate sequences with missing data." In: *Computer Vision and Image Understanding* 113.2 (2009), pp. 261–272 (cit. on p. 84).

[Max95] Nelson Max. "Optical models for direct volume rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108 (cit. on p. 12).

[MGN18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 3481–3490 (cit. on p. 6).

[Mi+22] Lu Mi, Abhijit Kundu, David Ross, Frank Dellaert, Noah Snavely, and Alireza Fathi. "im2nerf: Image to neural radiance field in the wild." In: *arXiv preprint arXiv:2209.04061* (2022) (cit. on p. 125).

[Mil+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In: *ECCV*. 2020 (cit. on pp. 11, 123, 125).

[Miy+18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral Normalization for Generative Adversarial Networks." In: *International Conference on Learning Representations (ICLR)*. 2018 (cit. on pp. 6, 18, 20, 54, 66).

[MK18] Takeru Miyato and Masanori Koyama. "cGANs with Projection Discriminator." In: *International Conference on Learning Representations (ICLR)*. 2018 (cit. on pp. 6–8, 18, 20, 50, 54, 66, 74–76, 84).

[MCS19] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. "Instance-aware Image-to-Image Translation." In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on pp. 22, 54).

[Mus+20] Siva Karthik Mustikovela, Varun Jampani, Shalini De Mello, Sifei Liu, Umar Iqbal, Carsten Rother, and Jan Kautz. "Self-Supervised Viewpoint Learning From Image Collections." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 3971–3981 (cit. on p. 54).

[Ngu+19] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. "Hologan: Unsupervised learning of 3d representations from natural images." In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 7588–7597 (cit. on p. 54).

[NG21] Michael Niemeyer and Andreas Geiger. "Giraffe: Representing scenes as compositional generative neural feature fields." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11453–11464 (cit. on pp. 12, 123, 127).

[ODO16] Augustus Odena, Vincent Dumoulin, and Chris Olah. "Deconvolution and Checkerboard Artifacts." In: *Distill* (2016) (cit. on p. 76).

[OOS17] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier GANs." In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 2642–2651 (cit. on pp. 7, 8).

[OPG21] Michael Oechsle, Songyou Peng, and Andreas Geiger. "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction." In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 5589–5599 (cit. on pp. 12, 128).

[Oor+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499* (2016) (cit. on pp. 1, 8).

[Orv+22] Antonio Orvieto, Jonas Kohler, Dario Pavllo, Thomas Hofmann, and Aurelien Lucchi. "Vanishing Curvature in Randomly Initialized Deep ReLU Networks." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2022, pp. 7942–7975 (cit. on p. viii).

[Ouy+22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. "Training language models to follow instructions with human feedback." In: *arXiv preprint arXiv:2203.02155* (2022) (cit. on p. 1).

[Par+19a] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "DeepSDF: Learning continuous signed distance functions for shape representation." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 165–174 (cit. on pp. 56, 124).

[Par+19b] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. "Semantic image synthesis with spatially-adaptive normalization." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2337–2346 (cit. on pp. 2, 7, 19–21, 24, 26, 27, 31, 33, 35, 39–41, 45, 47, 54).

[Pav+20a] Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. "Modeling Human Motion with Quaternion-based Neural Networks." In: *International Journal of Computer Vision (IJCV)* 128.4 (2020), pp. 855–872 (cit. on p. viii).

[Pav+19] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. “3d human pose estimation in video with temporal convolutions and semi-supervised training.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7753–7762 (cit. on p. vii).

[Pav+21] Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi. “Learning Generative Models of Textured 3D Meshes from Real-World Images.” In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021 (cit. on pp. vii, 10, 14, 83, 116, 126, 135, 144, 148).

[PLH20] Dario Pavllo, Aurelien Lucchi, and Thomas Hofmann. “Controlling Style and Semantics in Weakly-Supervised Image Generation.” In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. vii, 8, 13, 17, 54, 72, 88, 106).

[Pav+20b] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurélien Lucchi. “Convolutional Generation of Textured 3D Meshes.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on pp. vii, 1, 8, 10, 14, 53, 81, 84–88, 91, 95–98, 100, 101, 112, 126, 148).

[Pav+23] Dario Pavllo, David Joseph Tan, Marie-Julie Rakotosaona, and Federico Tombari. “Shape, Pose, and Appearance from a Single Image via Bootstrapped Radiance Field Inversion.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 (cit. on pp. vii, 12, 15, 121).

[Poo+22] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. “Dreamfusion: Text-to-3d using 2d diffusion.” In: *arXiv preprint arXiv:2209.14988* (2022) (cit. on pp. 1, 72).

[Qi+18] Xiaojuan Qi, Qifeng Chen, Jiaya Jia, and Vladlen Koltun. “Semi-parametric image synthesis.” In: *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR). 2018 (cit. on pp. 19, 21).

[Rad+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision." In: *International Conference on Machine Learning (ICML)*. PMLR. 2021, pp. 8748–8763 (cit. on p. 72).

[Ram+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical text-conditional image generation with CLIP latents." In: *arXiv preprint arXiv:2204.06125* (2022) (cit. on pp. 1, 2, 38).

[Ram+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. "Zero-shot text-to-image generation." In: *International Conference on Machine Learning (ICML)*. PMLR. 2021, pp. 8821–8831 (cit. on pp. 2, 7, 38).

[RTT21] Pierluigi Zama Ramirez, Alessio Tonioni, and Federico Tombari. "Unsupervised novel view synthesis from a single image." In: *arXiv preprint arXiv:2102.03285* (2021) (cit. on p. 125).

[Rav+21] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. "Skilful precipitation nowcasting using deep generative models of radar." In: *Nature* 597.7878 (2021), pp. 672–677 (cit. on p. 1).

[Reb+22] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. "LOL-NeRF: Learn from One Look." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1558–1567 (cit. on pp. 125, 158).

[Ree+16a] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. “Generative adversarial text to image synthesis.” In: *arXiv preprint arXiv:1605.05396* (2016) (cit. on pp. 20, 22).

[Ree+16b] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt Botvinick, and Nando de Freitas. “Generating interpretable images with controllable structure.” In: *OpenReview* (2016) (cit. on p. 22).

[Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards real-time object detection with region proposal networks.” In: *Neural Information Processing Systems*. 2015, pp. 91–99 (cit. on p. 32).

[Rez+16] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. “Unsupervised learning of 3d structure from images.” In: *Advances in Neural Information Processing Systems*. 2016, pp. 4996–5004 (cit. on p. 57).

[Roi+22] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. “Pivotal tuning for latent-based editing of real images.” In: *ACM Transactions on Graphics (TOG)* 42.1 (2022), pp. 1–13 (cit. on p. 131).

[Rom+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. “High-resolution image synthesis with latent diffusion models.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10684–10695 (cit. on pp. 1, 2, 7, 8, 38).

[Sah+22] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. “Photorealistic text-to-image diffusion models with deep language understanding.” In: *arXiv preprint arXiv:2205.11487* (2022) (cit. on pp. 1, 2, 7, 8, 38).

[Sch+21] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs." In: *arXiv preprint arXiv:2111.02114* (2021) (cit. on p. 1).

[Sch+20] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. "Graf: Generative radiance fields for 3d-aware image synthesis." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20154–20166 (cit. on pp. 12, 127).

[Sha+18] Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimi Kahou, and Yoshua Bengio. "Chatpainter: Improving text to image generation using dialogue." In: *arXiv preprint arXiv:1802.08216* (2018) (cit. on p. 23).

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 26).

[SOL19] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. "FineGAN: Unsupervised Hierarchical Disentanglement for Fine-Grained Object Generation and Discovery." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 23, 25, 30, 54).

[Sit+20] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions." In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 7462–7473 (cit. on p. 11).

[SZW19] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene representation networks: Continuous 3d-structure-aware neural scene representations." In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on pp. 127, 136).

[SM17] Edward J Smith and David Meger. "Improved Adversarial Systems for 3D Object Generation and Reconstruction." In: *Conference on Robot Learning*. 2017, pp. 87–96 (cit. on p. 57).

[SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." In: *arXiv preprint arXiv:2010.02502* (2020) (cit. on pp. 5, 38).

[SE19] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on p. 5).

[SE20] Yang Song and Stefano Ermon. "Improved techniques for training score-based generative models." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12438–12448 (cit. on p. 5).

[Sor+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. "Laplacian surface editing." In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 2004, pp. 175–184 (cit. on pp. 59, 77).

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." In: *ICML Deep Learning Workshop*. 2015 (cit. on p. 30).

[SSC22] Cheng Sun, Min Sun, and Hwann-Tzong Chen. "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5459–5469 (cit. on p. 12).

[SW19] Wei Sun and Tianfu Wu. "Image synthesis from reconfigurable layout and style." In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 10531–10540 (cit. on pp. 21, 47, 54).

[Sze+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826 (cit. on p. 24).

[Tan+20] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. "Fourier features let networks learn high frequency functions in low dimensional domains." In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 7537–7547 (cit. on p. 11).

[TDB17] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2088–2096 (cit. on p. 56).

[TL20] George Terzakis and Manolis Lourakis. "A consistently fast and globally optimal solution to the perspective-n-point problem." In: *European Conference on Computer Vision*. Springer. 2020, pp. 478–494 (cit. on p. 147).

[TEM18] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. "Multi-view consistency as supervisory signal for learning shape and pose prediction." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2897–2905 (cit. on p. 56).

[Tul+17] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. "Multi-view supervision for single-view reconstruction via differentiable ray consistency." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2626–2634 (cit. on p. 56).

[Tur+19] Mehmet Ozgur Turkoglu, William Thong, Luuk Spreeuwers, and Berkay Kicanaoglu. “A layer-based sequential framework for scene generation with GANs.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 8901–8908 (cit. on p. 23).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In: *Neural Information Processing Systems*. 2017, pp. 5998–6008 (cit. on pp. 24, 29, 37, 63).

[Wah+11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011 (cit. on pp. 55, 65, 85, 89, 98, 135, 143).

[Wan+19] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. “Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 124, 127, 133, 147).

[Wan+21] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. “NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021, pp. 27171–27183 (cit. on p. 128).

[Wan+18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. “High-resolution image synthesis and semantic manipulation with conditional GANs.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8798–8807 (cit. on pp. 2, 7, 19–21, 24, 26, 40, 54, 63).

[Wat+22] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. "Novel view synthesis with diffusion models." In: *arXiv preprint arXiv:2210.04628* (2022) (cit. on p. 125).

[WZ17] Olivia Wiles and Andrew Zisserman. "SilNet : Single- and Multi-View Reconstruction by Learning from Silhouettes." In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Gabriel Brostow Tae-Kyun Kim Stefanos Zafeiriou and Krystian Mikolajczyk. BMVA Press, 2017, pp. 99.1–99.13 (cit. on p. 56).

[WWR22] Felix Wimbauer, Shangzhe Wu, and Christian Rupprecht. "De-rendering 3D Objects in the Wild." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18490–18499 (cit. on p. 125).

[Wu+22] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. "Nüwa: Visual synthesis pre-training for neural visual world creation." In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*. Springer. 2022, pp. 720–736 (cit. on p. 38).

[Wu+17] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. "Marrnet: 3d shape reconstruction via 2.5D sketches." In: *Neural Information Processing Systems*. 2017, pp. 540–550 (cit. on pp. 56, 122, 124).

[Wu+16] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90 (cit. on p. 57).

[Xia+22] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. "Gan inversion: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on pp. 105, 126).

[XMS14] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild." In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2014 (cit. on pp. 55, 65, 135, 143).

[Xie+21] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. "SegFormer: Simple and efficient design for semantic segmentation with transformers." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12077–12090 (cit. on pp. 133, 145).

[Xie+18] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. "Learning descriptor networks for 3D shape synthesis and analysis." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8629–8638 (cit. on p. 57).

[Xu+22] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. "Sinnerf: Training neural radiance fields on complex scenes from a single image." In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer. 2022, pp. 736–753 (cit. on p. 125).

[Xu+18] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. "AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1316–1324 (cit. on pp. 18, 20, 22, 23, 29, 54, 66, 70, 74).

[Xue+22] Yang Xue, Yuheng Li, Krishna Kumar Singh, and Yong Jae Lee. “Giraffe hd: A high-resolution 3d-aware generative model.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18440–18449 (cit. on pp. 123, 127).

[Yan+16a] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. “Attribute2image: Conditional image generation from visual attributes.” In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 776–791 (cit. on p. 18).

[Yan+16b] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. “Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision.” In: *Advances in Neural Information Processing Systems*. 2016, pp. 1696–1704 (cit. on p. 56).

[Yan+17a] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. “3d object reconstruction from a single depth view with adversarial learning.” In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 679–688 (cit. on pp. 56, 122, 124).

[Yan+18] Guandao Yang, Yin Cui, Serge Belongie, and Bharath Hariharan. “Learning single-view 3d reconstruction with limited pose supervision.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 86–101 (cit. on p. 56).

[Yan+17b] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. “LR-GAN: Layered recursive generative adversarial networks for image generation.” In: *International Conference on Learning Representations (ICLR)*. 2017 (cit. on pp. 23, 26, 30, 54).

[Yao+22] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. “LASSIE: Learning Articulated Shapes from Sparse Image Ensemble via 3D Part Discovery.” In:

Advances in Neural Information Processing Systems. 2022 (cit. on p. 125).

[Yar+21] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. "Volume rendering of neural implicit surfaces." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4805–4815 (cit. on pp. 12, 123, 128, 129, 145).

[Yaz+19] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. "The Unusual Effectiveness of Averaging in GAN Training." In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 66).

[Yen+21] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. "iNeRF: Inverting neural radiance fields for pose estimation." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1323–1330 (cit. on p. 12).

[Yin+19] Guojun Yin, Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. "Semantics Disentangling for Text-to-Image Generation." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2327–2336 (cit. on p. 23).

[Yu+21] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. "pixelNeRF: Neural Radiance Fields from One or Few Images." In: *CVPR*. 2021 (cit. on pp. 12, 125).

[Yu+22] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. "Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction." In: *arXiv preprint arXiv:2206.00665* (2022) (cit. on p. 128).

[Zha+19a] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." In: *International Conference on Machine*

Learning (ICML). 2019 (cit. on pp. 18, 20, 23, 50, 54, 66, 74, 84).

[Zha+17] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5907–5915 (cit. on pp. 7, 18, 20, 22, 29, 43, 54).

[Zha+18a] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. "StackGAN++: Realistic image synthesis with stacked generative adversarial networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 41.8 (2018), pp. 1947–1962 (cit. on pp. 18, 20, 22, 29, 54).

[Zha+21a] Junzhe Zhang, Xinyi Chen, Zhongang Cai, Liang Pan, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. "Unsupervised 3d shape completion through GAN inversion." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1768–1777 (cit. on p. 126).

[Zha+22] Junzhe Zhang, Daxuan Ren, Zhongang Cai, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. "Monocular 3D Object Reconstruction with GAN Inversion." In: *European Conference on Computer Vision*. 2022 (cit. on pp. 72, 124, 126, 132, 135, 138, 139).

[Zha+18b] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The unreasonable effectiveness of deep features as a perceptual metric." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 586–595 (cit. on pp. 64, 134, 148).

[Zha+21b] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. "Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering." In: *International Conference on Learning Representations*. 2021 (cit. on pp. 126, 138).

[Zha+19b] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. "Image generation from layout." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8584–8593 (cit. on pp. 21, 47, 54).

[Zhu+17a] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2223–2232 (cit. on pp. 7, 20, 24, 54).

[Zhu+18] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. "Visual object networks: Image generation with disentangled 3D representations." In: *Neural Information Processing Systems*. 2018, pp. 118–129 (cit. on p. 57).

[Zhu+17b] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. "Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 57–65 (cit. on pp. 56, 122, 124).