SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Probabilistic 3D Shape Generation from an RGB Image

Yiheng Xiong

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Probabilistic 3D Shape Generation from an RGB Image

Probabilistische 3D-Formgenerierung aus einem RGB-Bild

Author:	Yiheng Xiong
Supervisor:	Prof. Dr. Angela Dai
Advisor:	Prof. Dr. Angela Dai
Submission Date:	15.12.2023

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

Munich, 15.12.2023 Yiheng Xiong

Acknowledgments

I extend my sincere gratitude to my supervisor and advisor, Angela Dai, for her unwavering guidance and insightful feedback throughout the entire duration of this thesis. I really appreciate her investment in terms of both time and resources for this project. I would like to express my heartfelt appreciation for the invaluable mental support provided by my family. Their constant encouragement has been a driving force throughout my studies. A special acknowledgment also goes to our IT-Systems-Administrator, Christoph Weiler. His technical support played a pivotal role in the development and execution of this project.

Abstract

Automatically generating 3D shapes from a single RGB image is a significant and practical challenge. In traditional content creation, modeling artists manually craft 3D models based on a guided image. Automating this process not only streamlines what was once deemed complex but also democratizes content creation, making it more accessible to a wider audience, particularly those with limited experience in 3D modeling. Additionally, in the realm of machine perception, cost-effective robots require the capability to comprehend three-dimensional information from their environment using a single camera. This involves generating/reconstructing the 3D structure and geometry of the subject based on the image captured via the single camera.

Typically, this generation process is probabilistic rather than deterministic due to the ambiguity inherent in a single image. To address this, we propose to leverage data-driven approaches rooted in deep learning to model the probabilistic distribution encapsulating the spectrum of plausible 3D structures conditioned on the input image. To achieve this, we initially create training pairs that are suitable for our task from existing datasets. Building upon AutoSDF [1], our method integrates a one-image-to-many-ground-truth-shape (**O2M**) training strategy, a **conditional module** and a **constraint module**. This integration aims to generate shapes with two criteria: (1) a high degree of diversity, and (2) a strong alignment with the visual content present in the input image. Comparative analyses with other networks demonstrate the superior performance of our method, showcasing excellence in both generation quality and diversity.

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**
1.1 Content Creation . . . 2
1.2 Machine Perception . . . 3
1.3 Probabilistic 3D Shape Generation . . . 4
1.4 Thesis Structure . . . 5

2 Background **7**
2.1 Shape Representations . . . 7
2.1.1 Voxel Grids . . . 7
2.1.2 Point Clouds . . . 8
2.1.3 Polygonal Meshes . . . 8
2.2 Datasets of 3D Shape . . . 9
2.3 Shape Generation from a Single Image . . . 9
2.3.1 Non-generative Neural Networks . . . 10
2.3.2 Generative Neural Networks . . . 10

3 Method Overview **13**
3.1 Discretized Latent Space for 3D Shapes . . . 13
3.2 Conditional Autoregressive Shape Generation . . . 14
3.2.1 Autoregressive Modeling . . . 14
3.2.2 Conditional Shape Generation . . . 14

4 Data Generation **18**
4.1 Rendering Images with Ambiguity . . . 19
4.1.1 Occlusion . . . 19
4.1.2 Limited Camera Range . . . 20
4.2 Creating Image-to-shape Training Pairs . . . 20
4.3 Generating Truncated Signed Distance Fields . . . 22

5 Network Architecture **25**
5.1 Image Encoder . . . 25
5.2 Conditional Module . . . 25
5.3 Autoregressive Modeling with Transformer . . . 27

5.4 Constraint Module 27
5.5 Training and Inference 28
5.5.1 Training 28
5.5.2 Inference 29

6 Results and Discussion **31**
6.1 Quantitative Evaluation 31
6.1.1 Synthetic Data 32
6.1.2 Real-world Data 32
6.2 Ablation Studies 34
6.3 Qualitative Samples 35
6.4 Limitations 35

7 Conclusion **42**

List of Figures **43**

List of Tables **44**

Bibliography **45**

1 Introduction

In recent years, the field of artificial intelligence (AI) has gained immense attention and recognition, both in academic research and various industries. AI refers to the simulation of human intelligence in machines, allowing them to perform tasks that typically require human intelligence. Within the convergence of AI and computer vision, there has been remarkable progress over the past few decades in the domain of understanding RGB images. This progress includes tasks like image classification [2, 3, 4, 5], semantic segmentation [6, 7, 8, 9, 10], and object detection [11, 12, 13, 14], particularly driven by the development of data-driven techniques rooted in deep learning. Nonetheless, our reality exists in a three-dimensional space, and solely interpreting these 2D images at the pixel level does not suffice when aspiring to endow AI agents with human-like intelligence.

Figure 1.1: A chair with high self-occlusion.

For a human being, when presented with an RGB image capturing an object from a specific viewpoint, we possess the capacity to not only identify the object but also mentally conjure its 3D geometric intricacies. These mental processes apply to visible components within the image, and extend to invisible parts due to occlusion, and/or limitations imposed by the camera's range. Typically, such mental conjuring process is not deterministic, but with a range of possibilities. For instance, consider an image of a chair with substantial self-occlusion, as illustrated in Figure 1.1. At a first glance, we can readily identify it as a chair. When it comes to occluded elements, such as the chair's arms or seat, our imaginative faculties come into play, offering various hypotheses about their 3D geometry. We may contemplate whether the arms are curved or straight, whether the seat is thick or thin, and so forth. Even

(a) Screenshot from the game *Grand Theft Auto*.

(b) Screenshot from Google Earth.

(c) Hotel room interior design.

Figure 1.2: Examples for content creation.

for non-occluded elements, such as the chair's legs, there exist a multitude of possibilities concerning their intricate 3D geometry.

Hence, the generation of multiple 3D shape hypotheses based on an RGB image assumes critical importance when striving to replicate human-like intelligence. Towards this goal, this thesis places its primary emphasis on harnessing the power of probabilistic deep learning techniques, to generate multiple plausible 3D shapes from a single RGB image.

1.1 Content Creation

In contemporary computer graphics, a significant focal point revolves around content creation, specifically the generation of authentic 3D assets. This emphasis has gained even more traction with the burgeoning development of virtual reality (VR). To illustrate, crafting a cityscape within the game *Grand Theft Auto* demands lifelike car models as shown in Figure 1.2a. Likewise, in Figure 1.2b, VR-driven navigation systems rely on realistic building models, while interior design endeavors necessitate a rich assortment of authentic furniture models as depicted in Figure 1.2c [1]. The creation of a 3D model of an environment not only facilitates unrestricted exploration from various viewpoints but also lays the groundwork for potential interactions within a game-like setting. In the absence of authentic and varied 3D objects, virtual worlds tend to appear empty, leading to reduced levels of engagement.

Despite this need, the task of manually crafting 3D assets is widely acknowledged as a formidable challenge, demanding a combination of creativity, proficiency in 3D design, and access to intricate software known for its steep learning curve. In the process of manual content creation, concept artists first draw a 2D image of the desired content and gradually enhance it guided by feedback from art direction and other artists. Then modeling artists create a low polygon model based on the image. This phase involves creating 3D meshes (see subsection 2.1.3) with a relatively small number of faces, focusing on conveying the essential

[1] https://freelancers3d.com/en/portfolio/7130/hotel-room-interior-design

details with minimal geometric complexity. Subsequently, high polygon modeling comes to play. This advanced stage centers on shaping 3D meshes with a substantial number of faces, allowing for the intricate representation of details to achieve a high level of realism. In contrast to low-polygon models, high-polygon counterparts prioritize detailed geometry, aiming to capture nuanced surface features like wrinkles and pores. The complexity of the whole pipeline makes the creation of 3D content inaccessible to less-experienced users.

Hence, it is not surprising that automatically generating 3D assets guided by an RGB image has become an active research area. Such automation not only simplifies the process of content creation that was traditionally considered complex and demanding, but it also opens up new horizons for a broader range of users, especially those lacking extensive experience in 3D modeling.

1.2 Machine Perception

Machine perception, particularly in the domain of 3D vision, refers to the ability of machines to perceive and understand three-dimensional information from the environment, focusing on tasks that involve depth perception, spatial understanding, and the interpretation of 3D structures. It is a highly challenging problem in the computer vision community and has various practical applications.

Enabling machines to perceive 3D world via a single RGB image is an even more challenging yet practical task. Formally, given an RGB image, the goal is to generate/reconstruct the 3D structure and geometry of the subject based on the single image. It is valuable in scenarios where obtaining multiple views or depth data is limited or not feasible. For instance, consider a scenario where a budget-friendly household robot equipped with a single RGB camera and lacking depth sensors, is capable of comprehending the spatial layout of its environment. This ability is crucial for enabling the robot to execute real-time tasks including obstacle avoidance, placing an object beside another one and so on, especially within a limited budget.

In fact, in such a single RGB image, objects are frequently only partially visible. This partial visibility can be attributed to several factors. Firstly, it may result from self-occlusion (e.g., if we observe a chair horizontally from the rear, we can hardly see its seat as shown in Figure 1.1). Additionally, depicted in central part of Figure 1.3, mutual-occlusion can also be a common reason, where objects overlap and obstruct each other within the scene. Besides, invisibility can also arise due to the limited range of the camera as shown in the right bottom corner of Figure 1.3, where only the back of the chair is visible. The presence of various forms of invisibility introduces a considerable degree of ambiguity during the reconstruction process. This inherent uncertainty means that the process of reconstruction in such complex scenarios becomes intrinsically probabilistic in nature. In other words, when faced with these invisible or partially visible elements in an image, multiple viable reconstructed 3D shapes can coexist, each corresponding to a plausible interpretation of the image's content.

To address this inherently ill-posed problem, researchers have invested substantial effort, particularly leveraging recent advancements in deep learning methods [15, 16, 17, 18, 19, 20, 21] and the availability of 3D shape datasets [22, 23, 24, 25, 21]. However, most of these

Figure 1.3: Multiple forms of invisibility. In the central portion, chairs and tables obstruct with each other. In the right bottom corner, only the back of the chair is visible.

approaches primarily yield a deterministic 3D output based on an RGB image. On one hand, these methods are deterministic in nature. On the other hand, data-driven reconstruction techniques rooted in deep learning necessitates abundant image-shape training pairs. These training pairs are often in a one-to-one or many-to-one fashion, lacking the one-to-many pattern. Thus, reconstructing multiple plausible 3D shapes from an RGB image still remains challenging.

1.3 Probabilistic 3D Shape Generation

While it may seem that tasks such as content creation and machine perception are inherently distinct in their applications, they converge at a common juncture: the shared necessity for generating a distribution that encapsulates the spectrum of plausible 3D structures conditioned on an image.

Furthermore, in light of the advancements in deep generative neural networks, these two tasks can be effectively integrated into a unified framework with a non-deterministic approach. We can aptly characterize this non-deterministic process as "probabilistic 3D shape generation". To formalize it, when presented with an RGB image, our objective is to produce a conditional probabilistic distribution of 3D shapes, such that the samples drawn from this distribution satisfy two essential criteria: (1) a high degree of diversity, and (2) a strong alignment with the visual content present in the input image.

1.4 Thesis Structure

In this thesis, we devise a data-driven approach that directly takes a single RGB image as input and yields the corresponding distribution of 3D shapes as output. The samplings from the distribution not only exhibit a notable level of diversity amongst themselves but also maintain a strong alignment with the visual content depicted in the input image.

To accomplish this, we draw inspiration from AutoSDF [1], wherein we train a transformer-like model in an autoregressive fashion using image-shape pairs. What sets our approach apart in terms of diversity is the adoption of a novel one-image-to-many-ground-truth-shape (**O2M**) training strategy with the help of new-generated **one-to-many training pairs** from existing datasets. In addition, we also utilize a **cross-attention conditional mechanism** and a **constraint module** to further improve the quality of the generated shapes. These unique design choices empower the network with the capacity to produce multiple distinct and credible 3D shapes from a single RGB image at test time.

The rest of the thesis is structured as follows:

Chapter 2 explores different 3D data representations and their strengths as well as weaknesses when being re-generated. It also gives an overview over recent advances in image-to-shape generation task, ranging from datasets and methods, emphasizing the difficulty of generating a probabilistic distribution of 3D shapes from an RGB image.

Chapter 3 introduces the core concepts of our approach and outlines the techniques employed to bring them to fruition. This chapter provides a condensed preview of the subsequent two chapters, offering a comprehensive overview of the key details to follow.

Chapter 4 elaborates on the process of crafting the data utilized by our method, derived from existing datasets. It delineates the requisite data representations, their characteristics, and the procedures employed in their generation.

Chapter 5 outlines the comprehensive components constituting the neural network architecture and the training and inference methodology.

Chapter 6 presents and analyzes the results of our method, compares its performance with different baselines, and conducts ablation studies on the key components of our approach.

Chapter 7 serves as the conclusion, summarizing the findings of this work and offering insights into potential enhancements and future avenues for exploration.

2 Background

3D shape generation presents numerous practical applications and continues to pose many challenges, as has already been discussed in chapter 1. In this work, we focus on processing a single RGB image to yield 3D shapes, more specifically on the question of generating multiple plausible hypotheses from the image.

In this chapter, we will give an overview of the field of 3D shape generation. Firstly, we will discuss the major types of 3D shape representations. Representing objects in our complex three-dimensional world properly is essential for re-generating them. Secondly, we will delve into 3D shape datasets which play a pivotal role in data-driven methods we are poised to adopt. Then we will give an overview over influential and recent neural networks for 3D shape generation from a single image, which we categorize into two groups: non-generative networks and generative networks.

2.1 Shape Representations

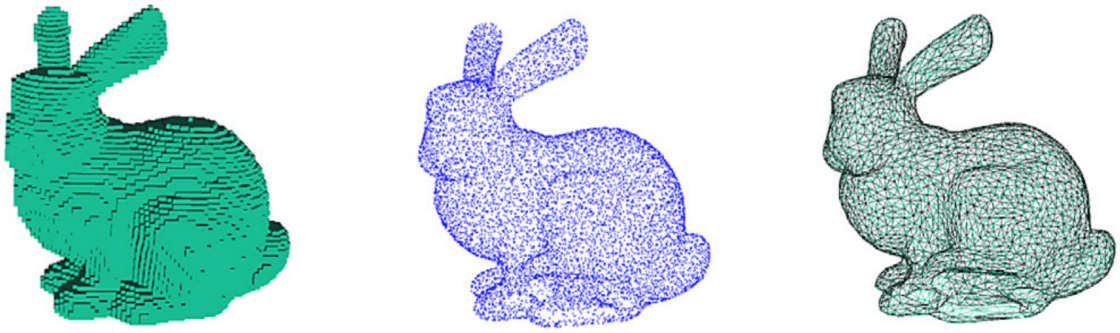
The computer vision and computer graphics domains have introduced a myriad of 3D shape representations. Common ones include voxel grids, point clouds and meshes. Each of these representations boasts its unique strengths and, in equal measure, has its inherent limitations when applied to the multifaceted task of 3D generation.

2.1.1 Voxel Grids

The constituent pixels of a 2D image provide a stable and uniform structure for the image. Analogously, voxels (see Figure 2.1a) with an orderly and regular structure are an intuitive extension of pixels in 3D space: voxel grids are the 3D counterpart of pixel representation. They serve as a representation for 3D shapes and can store various types of information, such as geometry occupancies [26, 22], volume densities [27, 28], or distance values [1].

For geometry occupancies, each voxel conveys whether it is part of the 3D structure contained within the grid or remains unoccupied. In the case of volume densities, each voxel in the grid stores a numerical value that represents the density of a particular attribute at that point in 3D space. As for distance values, implicit distance functions serve as a means to encode the surface of a 3D structure by designating the zero-label set. In this scenario, each voxel encapsulates the distance from its position in 3D space to the nearest surface, effectively mapping the spatial relationship to the structure's boundaries.

While employing voxel grids offers a straightforward means of 3D shape representation, their inherent regularity imposes a natural constraint on their scalability. There comes a point where the size of voxel grids can grow to a computationally prohibitive extent. Consequently,



(a) Voxel grid. (b) Point cloud. (c) Polygonal mesh.

Figure 2.1: Different 3D representations.

the resolution of 3D voxel grids is frequently curtailed as a necessary trade-off and this limitation can give rise to step artifacts, which, in turn, can compromise the quality of the generated shapes.

In particular, our method uses a volumetric Truncated-Signed Distance Field (T-SDF) for representing a 3D shape. It involves encoding the distance from each point in space to the surface of the object, with the sign indicating whether the point is inside or outside the surface. The "truncated" part means that the function is typically defined only within a certain range or region of interest.

2.1.2 Point Clouds

Point clouds (see Figure 2.1b) offer a more precise, light-weight representation of 3D shapes in comparison to voxel grids. The 3D coordinates (x, y, z) of point samples directly represent shape geometry. Point clouds are often the raw output format of 3D acquisition devices. With advancements in technology, such as the emergence of devices like Kinect [29], the obtaining of point cloud data has become increasingly efficient.

Nonetheless, it is important to note that point clouds lack the inherent attribute of spatial order. They are defined as unordered assortments of points scattered irregularly throughout 3D space. Hence, generating 3D shapes in the form of point clouds is a highly challenging problem. Normally, point clouds are sampled from generated meshes (see subsection 2.1.3) and used for evaluation metrics such as chamfer distance (see chapter 6).

2.1.3 Polygonal Meshes

Polygonal meshes (see Figure 2.1c), a type of non-Euclidean data, depict the surfaces of 3D shapes using a combination of vertices, edges, and faces. Unlike voxel-based approaches, meshes are dedicated to representing the surfaces of 3D scenes, resulting in more space-efficient representations. In contrast to point clouds, meshes offer explicit connectivity details between surface points, facilitating the accurate modeling of relationships among these

points. Thanks to these inherent advantages, polygonal meshes hold a prominent position in traditional computer graphics applications, including tasks like geometry processing, animation, and rendering.

However, the irregularity and intricate nature of such topological structures pose significant challenges in the direct generation of meshes. Thankfully, methods such as Marching Cubes [30] offer a solution. We can generate a regular grid as intermediate output, which is considerably more manageable in a deep learning context. Subsequently, we can extract a mesh from this regular grid by marching cubes. We adopt this process as part of the generation pipeline (see chapter 6) in this thesis.

2.2 Datasets of 3D Shape

Just like any data-driven method, the quality of the dataset employed for training plays a pivotal role in the success of deep learning techniques. Consequently, there has been a recent surge in publications introducing novel and meticulously tailored datasets, designed explicitly to support the field of deep learning. These datasets are poised to enhance the efficacy and applicability of deep learning methods.

For 3D shape generation, ModelNet [26] and ShapeNet [23] are among the most renowned. Both of these datasets encompass a substantial number of high-quality 3D CAD models spanning various categories. Furthermore, PartNet [24], an extension of ShapeNet, enriches the landscape by providing detailed, fine-grained, instance-level, and hierarchical 3D part information. Complementing these, Pix3D [25] and Things3D [21] offer diverse collections of image-shape pairs, with precise 2D-3D alignment.

Despite the abundance of available 3D shape datasets, none of them can be directly adopted in our specific use case. Either it does not provide image-shape pairs at all, or it does not provide image-shape pairs in a one-to-many pattern. Such one-to-many-pattern image-shape pairs are essential supervision signals in our training process as discussed in section 1.4.

Therefore, we make use of existing ShapeNet and PartNet datasets, combined with rendering techniques provided by Choy et al. [15], generating a new dataset that is tailored to our training pipeline (see chapter 4).

2.3 Shape Generation from a Single Image

Typically, neural networks for shape generation from an RGB image can be roughly divided into two groups: non-generative networks and generative networks. Non-generative networks are only capable of generating a shape given an input image. On the other hand, generative networks tend to learn the probabilistic distribution of a range of shapes conditioned on the input image.

2.3.1 Non-generative Neural Networks

One of the first approaches was presented by Choy et al. [15] using voxel representation for 3D reconstruction. The researchers proposed 3D-R2N2, a recurrent neural network to learn the mapping between images and their underlying 3D shapes in an approach that is capable of reconstructing the 3D shape from as few as a single image. To produce higher resolution reconstructions, Richter and Roth [16] proposed Matryoshka networks. The main idea of the work is to encode the 3D shape more compactly by representing a shape as an n-channel image, where n is the number of voxels along the z-direction. This novel approach allows for single-view reconstructions up to 256^3 voxels.

Mescheder et al.[17] proposed Occupancy Networks (OccNet), a network architecture that is capable of reconstructing 3D shapes from diverse input modalities in function space. The researchers formulated the reconstruction task as a binary classification problem where the shape's surface is implicitly represented as the continuous decision boundary (see subsection 2.1.1) of the classifier. To train the model for single-view reconstruction, the input images are fed into a pretrained ResNet [18] image encoder to produce image encodings that serve as conditional information for reconstructing the shape based on the provided input image. Xu et al. [19] proposed to directly regress continuous (signed) distances values (see subsection 2.1.1) and introduced the Deep Implicit Surface Network (DISN), a network architecture that has individual modules for camera pose estimation, global feature extraction, local feature extraction, and for point location to feature space mapping.

Xie et al. [20, 21] proposed parallel approaches to encode one or several image inputs and decode them into coarse volumes simultaneously. The coarse volumes are then fed into a context-aware fusion network that bases its fusion on a learned fusion score. Finally, the fused volume is further refined using the third component of the model, the refiner module.

2.3.2 Generative Neural Networks

Among a wide range of generative models, generative adversarial networks (GANs), denoising diffusion probabilistic models (DDPMs) and autoregressive models (ARs) are commonly used for 3D shape generation from images.

GANs. GANs were originally proposed by Goodfellow et al. [31]. A GAN consists of two networks: a discriminator, which estimates the probability that a sample comes from the real data distribution, and a generator, which approximates the real data distribution by trying to fool the discriminator with synthetic samples. The two networks are trained simultaneously via a max–min game.

Wu et al. proposed 3D-GAN [32]. It uses a generator that is fed a latent vector sampled from a probabilistic latent space and maps it to a voxel grid representing the 3D object. It also uses a discriminator that learns to distinguish between objects generated from the generator and real 3D objects. To equip 3D-GAN to perform shape generation from images, the researchers incorporated a VAE [33] as an image encoder that maps a 2D image into the latent space mentioned above with additional reconstruction loss and Kullback Leibler divergence loss. Following 3D-GAN, Smith et al. [34] introduced another approach to 3D GANs that enables

the generation of 3D full objects from 2D images. Knyaz et al. [35] proposed a pipeline to recover 3D shape from a single 2D image through an adopted z-GAN architecture [36] and frustum voxel model.

DDPMs. DDPMs [37] are a family of probabilistic generative models that progressively destruct data by injecting noise, then learn to reverse this process for sample generation. In particular, diffusion models have shown impressive quality, diversity, and expressiveness in various tasks such as image synthesis [37, 38, 39, 40], super-resolution [41], image editing [42], text-to-image synthesis [43, 44, 45, 46, 47] and so on.

In contrast to the flourishing research on diffusion models for 2D data, diffusion models have not yet been fully explored for 3D data. Zhou et al. [48] proposed to denoise in the 3D space of world coordinates to generate point clouds. With recent advances in latent diffusion models [49], Luo et al. [50] and Zeng et al. [51] made use of diffusion models in the latent space(s) of point clouds for point cloud generation. To adapt diffusion models to 3D generation from images, Cheng et al. [52] and Sbrolli et al. [53] combined latent diffusion models with the classifier-free conditional generation mechanism [54] to generate implicit 3D representations conditioned on the input image.

ARs. ARs are generative models that aim to model distributions of high dimensional data by factoring the joint probability distribution to a series of conditional distributions via the chain rule [55]. They can serve as powerful density estimators [56], are more stable during training [56, 57], and can generalize well on held-out data.

Sun et al. [58] proposed PointGrow, an AR for generating point cloud shapes with the possibility of generating shapes conditioned on 2D images. The Octree Transformer [59] combines the AR with an octree-based network. With the transformer [60] involved, the input is encoded into a sequenced octree. Then, compressed shorter sequence latent vectors can be trained with a classic transformer decoder. Yan et al. [61] proposed a transformer-based architecture to model conditional distribution by a sequence consisting of quantized features from the encoder. Furthermore, an implicit function called the vector quantized deep implicit function (VQDIF) was introduced, which can compress shape into a sequence of sparse local features. After autoregressive sampling, the VQDIF decoder converts this sequence back to the deep implicit function used for mesh extraction.

Besides, also employing the AR is AutoSDF [1]. While VQ-VAE [62] learns quantized and compact latent representations for images and Esser et al. [63] learned autoregressive generation from discrete VQ-VAE, AutoSDF extends the method of Esser et al. to the domain of 3D shapes but with a generic non-sequential autoregressive prior. This method can generate shapes from single-view images through a "naive" task-specific [1] conditional model. Thus, the image-to-shape relationship is not directly indicted in the autoregressive modeling.

For our method, we combine the shape generation idea of Mittal et al. [1] with our novel **O2M** training strategy, and employ the **cross-attention conditional mechanism** to model the image-to-shape relationship directly in the autoregressive approach. To enhance the fidelity of the generated shapes, we also incorporate a **constraint module** into our method.

3 Method Overview

Our approach seeks to model the distribution $p(\cdot|c)$ representing potential 3D shapes X conditioned on the input image c. We achieve this by training the generation pipeline in a fully supervised manner. The training process involves pairs of single-view images and 3D shape models. Notably, our method is designed to handle cases where one image may correspond to multiple ground-truth shapes.

We produce input and ground-truth data for training and testing by combining ShapeNet [23] and PartNet [24] datasets. The input images are rendered from CAD models within the ShapeNet dataset, employing customized settings to introduce essential ambiguity. The mappings from one image to multiple ground-truth shapes are established using per-pixel part labels in the rendered images and visible parts in 3D space, both extracted from annotations within the PartNet dataset. Further details on the implementation can be found in chapter 4.

Building upon the principles of AutoSDF [1], our method utilizes a volumetric Truncated Signed Distance Field (T-SDF) to represent 3D shapes and integrates a Transformer-based [60] neural autoregressive model. Acknowledging the quadratic increase in computational complexity with input dimensionality in the transformer, we initially map the high-dimensional 3D shape to a corresponding low-dimensional, discretized latent space, similar to AutoSDF. Subsequently, we learn the conditional distribution of shapes based on the input image over this compressed representation.

3.1 Discretized Latent Space for 3D Shapes

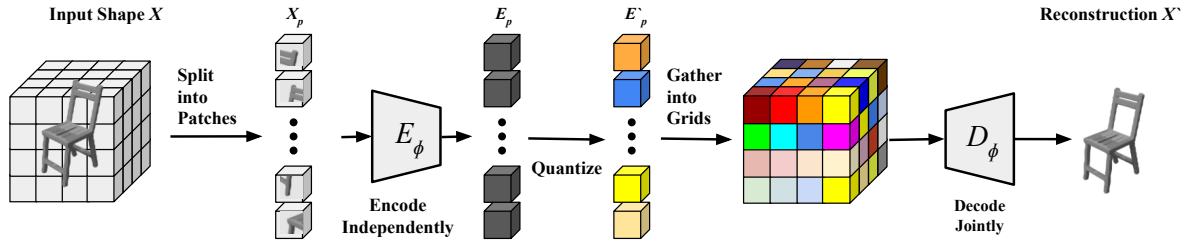


Figure 3.1: Patch-wised encoding VQ-VAE overview. The input shape X is first split into patches X_p, each independently encoded via E_ϕ. Then the patch encodings E_p are quantized and gathered into grids. Finally, D_ϕ is employed to jointly decode the quantized patch encodings E'_p, yielding the final reconstruction output X'.

For each shape X, a shape encoder $E_\phi(\cdot)$ encodes its Truncated Signed Distance Field (T-SDF) into a low-dimensional 3D grid feature $E \in g^3 \times D$, where g represents the resolution of the

3D grid, and D denotes the dimension of the 3D grid features. We establish a discrete latent space containing K embedding vectors, each with a dimension of D. For the shape feature e_i at grid position i, a vector quantization operation $VQ(\cdot)$ is employed to find its nearest neighbor e'_i from the K embedding vectors in the discrete latent space. The feature input into the shape decoder $D_\phi(\cdot)$ is the 3D discrete latent feature grid $E' \in g^3 \times D$, comprising discrete features e'_i at grid locations i. The formula is:

$$E = E_\phi(X), E' = VQ(E), X' = D_\phi(E'). \tag{3.1}$$

where X' is the reconstructed shape, and ϕ is the set of learnable parameters of the decoder. As a result, a shape X can be represented as a 3D latent feature index grid $Q \in g^3$, where q_i at each location corresponds to a discrete shape feature e'_i.

In particular, we adopt Patch-wised encoding VQ-VAE (P-VQ-VAE) [1] to learn this process since it allows for a better representation for local details. Given the 3D T-SDF X of the shape, we first split X into N patches of X_p. Then E_ϕ encodes each patch independently and D_ϕ decodes all patches jointly. Empirically, we set $K = 512$, $D = 256$, $g = 8$ and $N = 512$. Training the network uses a combination of three losses proposed by van den Oord et al. [64]: reconstruction loss, the vector quantization objective and the commitment loss. Figure 3.1 shows the overall architecture of P-VQ-VAE.

3.2 Conditional Autoregressive Shape Generation

Employing P-VQ-VAE is instrumental in transforming the high-dimensional 3D shapes into low-dimensional latent representations, aligning well with the requirements of our transformer-based model. The resulting latent representations are discrete grids, where each location i holds a 3D latent feature index q_i corresponding to the shape feature e'_i. The decoder D_ϕ then utilizes these shape features to produce the final reconstruction output. In our method, we utilize this representation for the generation pipeline.

3.2.1 Autoregressive Modeling

Low-dimensional discretized representations E' are well-suited for autoregressive 3D modeling. We can simplify the task of learning the conditional distribution over continuous 3D shapes $p(X|c)$ to learning $p(E'|c)$. A typical autoregressive model can approximate this distribution by sequentially factorizing it as a product of location-specific conditionals:

$$p(E'|c) = \prod_{i=[1,1,1]}^{[g,g,g]} p(e'_i|e'_{<i}, c). \tag{3.2}$$

3.2.2 Conditional Shape Generation

In AutoSDF [1], Mittal et al. model the joint distribution (Equation 3.2) as a product of the autoregressive prior, coupled with an independent "naive" conditional term that weakly

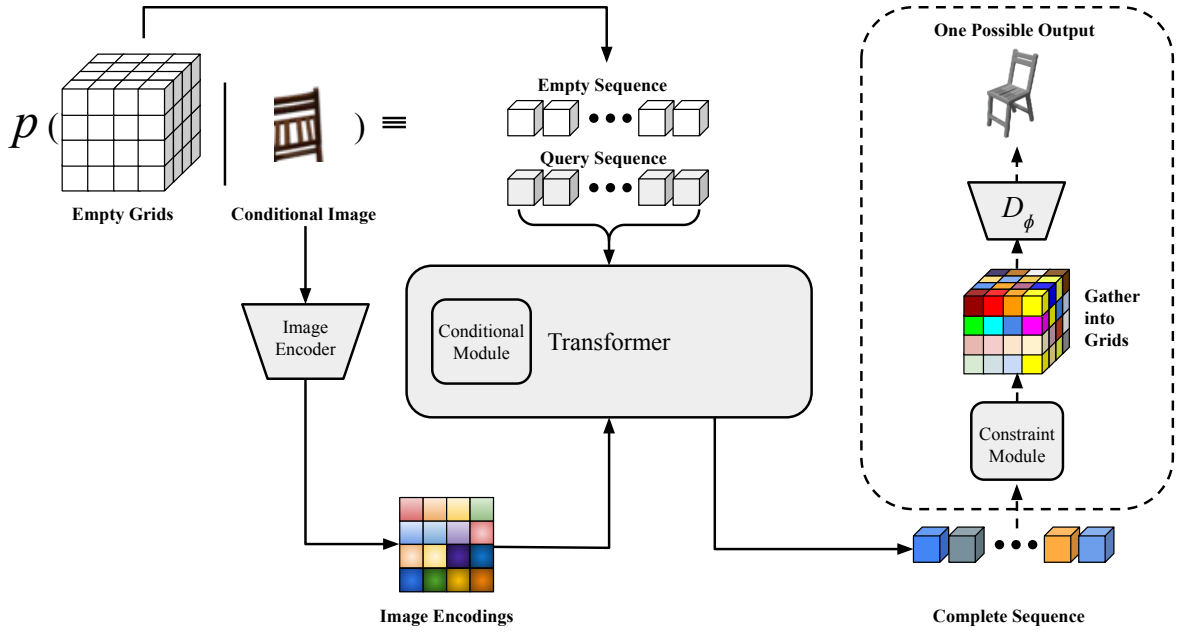


Figure 3.2: Conditional shape generation pipeline overview. The encodings of the conditional image are extracted via an image encoder. Accompanying with the image encodings, the empty sequence flattened from the empty grids and a query sequence are put into the transformer, where they interact mutually within the conditional module and finally output the complete sequence where each cell possesses the conditional probability distribution of latent features. The dotted lines indicate the sampling and decoding to one possible output shape assisted by a constraint module.

captures the dependence on the conditional image c:

$$p(E'|c) = \prod_{i=[1,1,1]}^{[g,g,g]} p(e'_i|e'_{<i}, c) \approx \prod_{i=[1,1,1]}^{[g,g,g]} p_\theta(e'_i|e'_{<i}) \cdot \prod_{i=[1,1,1]}^{[g,g,g]} p_\mu(e'_i|c). \tag{3.3}$$

where θ represents the learnable parameters of the autoregressive model and μ represents the learnable parameters of the naive conditional model. Applying such modeling directly to our use case implies that the mapping from an image to multiple ground-truth shapes is merely captured by the naive conditional model instead of the inherently probabilistic autoregressive model. We argue that the naive conditional model may not fully address the complexities in our use case: (1) images with significant ambiguity; (2) the mapping from one image to multiple ground-truth shapes. To this end, we propose to model the one-to-many mapping by training the autoregressive generation framework end-to-end, conditioning it on the input image.

Figure 3.2 visualizes the high-level architecture of our method. Given the conditional image c, we initiate the process by utilizing an image encoder to extract its encodings. Simultaneously, empty grids are introduced to the network. These empty grids are flattened into a sequence and incorporated into the transformer along with the image encodings and a query sequence. Within a conditional module, the interaction between the empty sequence, the query sequence and the image encodings takes place. Subsequently, the transformer outputs a complete sequence. Each cell within this sequence denotes the conditional probability distribution of latent features. To generate one plausible shape from the complete sequence, we first add constraints to the distribution via a constraint module and then sample from this constrained distribution. The constrained samplings are then gathered into grids. Finally, these grids are fed into D_ϕ to reconstruct one plausible 3D shape.

We assess the performance of our detailed architecture (chapter 5) on the synthetic training and test pairs (chapter 4), in comparison to the approach proposed by Mittal et al. [1] and several non-generative methods. Furthermore, we extend the evaluation to the generation of shapes from real-world images using our proposed method.

4 Data Generation

Our approach to 3D shape generation involves feeding a single RGB image of the object into a network architecture and generating multiple plausible 3D shapes in the form of Truncated Signed Distance Fields (T-SDFs). We opt for a fully supervised training approach, necessitating the creation of input images with a sufficient degree of ambiguity. Additionally, each image needs multiple ground-truth models that align with the visual content of it. In this chapter, we discuss the process of generating a dataset that contains all these elements.

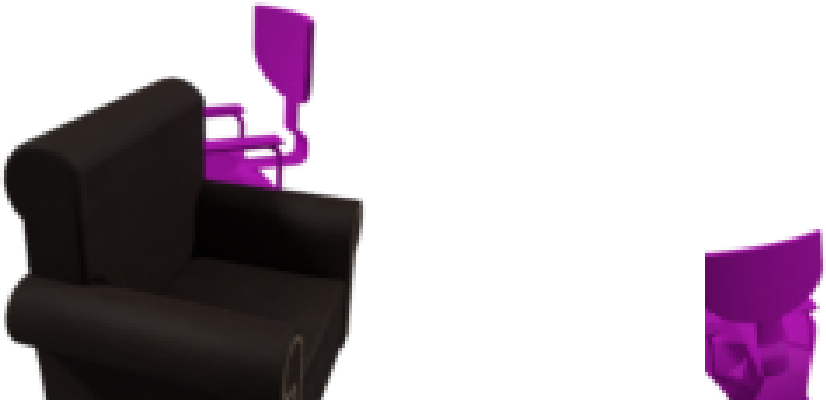
We build upon the foundation of widely adopted and publicly accessible 3D shape datasets, specifically ShapeNet [23] and PartNet [24], in our dataset generation process. ShapeNet contains 3D models from a multitude of semantic categories and organizes them under the WordNet taxonomy [65]. In particular, we utilize ShapeNetCore, a subset of the full ShapeNet dataset with single clean 3D models and manually verified category and alignment annotations. Figure 4.1 shows sample CAD models from ShapeNet. Besides, PartNet, an extension of ShapeNet, is a consistent, large-scale dataset of 3D objects annotated with fine-grained, instance-level, and hierarchical 3D part information (see Figure 4.2). We use them for both input and ground-truth data generation.

Essentially, our analysis centers on the *chair* and *table* categories, chosen for their inherent diversity and ubiquitous presence in our daily lives. We render CAD meshes from ShapeNetCore in tailored settings to get renderings of shapes with enough ambiguity. Then we establish ground-truth shape mappings for each rendering, leveraging part-level annotations from PartNet. This approach allows for the possibility that each image may have multiple distinct ground-truth shapes. Furthermore, we also generate T-SDFs for each model as required by our method.

(a) (b) (c) (d) (e)

Figure 4.1: ShapeNet CAD model samples.

Figure 4.2: Fine-grained instance-level segmentation visualization from PartNet [24]. Different colors indicate different parts.



(a) Ambiguity from occlusion.

(b) Ambiguity from limited camera range.

Figure 4.3: Two strategies to induce ambiguity.

4.1 Rendering Images with Ambiguity

To introduce the necessary ambiguity into our input images, we employ two strategies. Firstly, we manually induce occlusion, encompassing both self-occlusion and mutual-occlusion scenarios. Secondly, we simulate scenarios where certain parts of objects extend beyond the camera's range. Utilizing these strategies enhances the realism, diversity, and complexity of the input images within the dataset.

4.1.1 Occlusion

To manually introduce occlusion to the rendered models, we first fix the azimuth and yaw of the virtual camera as well as setting its X and Y coordinates to $(0, 0)$ in ZXY mode. Subsequently, we randomly set the camera's altitude, and the distance between the origin

$O(0,0,0)$ and itself along the Z-axis, with both parameters falling within predefined ranges. Placing the target object at the origin O, we then rotate it randomly along the Y-axis, ensuring that the renderings capture varying levels of self-occlusion. Additionally, we randomly select another object from the available set as the occluding object to induce mutual-occlusion. In this setting, we position the occluding object after random rotation between the camera and the target object, ensuring (1) no overlap between the occluding and target objects in 3D space; (2) reasonable occlusion of the target object by the occluding one in the 2D image. Figure 4.3a shows an example for occlusion.

4.1.2 Limited Camera Range

To simulate the scenarios where parts of the model are beyond the camera's range, we employ the previously described camera settings. Then we put the target object at the origin O and randomly move it along $\pm$X-axis and $\pm$Y-axis within predefined ranges and rotate it randomly along Y-axis. This deliberate translation and rotation ensure that certain parts of the object exceed the camera's range, making them invisible in the resulting 2D image. Figure 4.3b shows an example for limited camera range.

In each of the aforementioned scenarios, we generate renderings from 10 different views with the resolution of 137×137 using the techniques outlined by Choy et al. [15], resulting in a total of 20 views of images per model.

4.2 Creating Image-to-shape Training Pairs

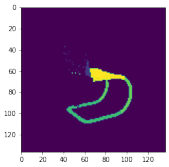
In the settings outlined above, it is evident that each image corresponds to only one ground-truth shape, which is not conducive to our training pipeline. In our network, we require the input image to be mapped to potentially multiple ground-truth shapes that align with the image.

To this end, we initially classify CAD models from the dataset into similar groups. When evaluating two models, we take into account factors such as the number of included parts, the semantics of those parts, and the overall geometric similarity of the entire models. To be more specific, if two models exhibit identical part counts and semantics, and their geometric similarity surpasses a predefined threshold, we classify them as similar. The part information is derived from PartNet annotations, and we employ the Bidirectional Hausdorff Distance as our measure of geometric similarity.

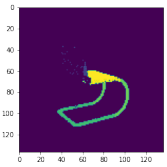
Once we generate groups of similar CAD models, we proceed to create ground-truth mappings for each rendering based on these groups. This involves considering two key factors: (1) per-pixel part labels in the images, and (2) the geometric similarity of visible parts in 3D space.

During the rendering process as described in section 4.1, not only do we get RGB colors for each pixel, but we also generate per-pixel part labels (see Figure 4.4b and Figure 4.4c). In particular, for each mesh, we begin by selecting the corresponding point cloud in PartNet.

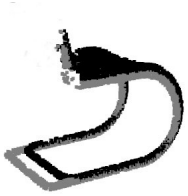
(a) Image for the occluded target object.



(b) Per-pixel part labels for the occluded target object. Different colors indicate different part labels.



(c) Per-pixel part labels for the occluded candidate object. Different colors indicate different part labels.



(d) Visible points for the occluded target and candidate objects. Black points indicate the target object and grey ones indicate the candidate object.

Figure 4.4: Intermediate outputs for generating ground-truth mappings. The target and the candidate objects are processed by exactly the same rendering settings. We compare the per-pixel part labels and the visible points between the target and the candidate objects to define whether the candidate one is included to the ground-truth shape mapping for the image.

We then align the point cloud with the mesh using certain transformations. Next, for each face of the mesh, we find the nearest point in the aligned point cloud to its center (average of coordinates of the vertices). The part label of the face is then set to the part label of the nearest point, achieved by modifying its material color based on a predefined part-label-to-color mapping. This process ensures that we obtain per-pixel part labels in the same projection setting as the corresponding image.

At the same time, we iterate through the points in the aligned point cloud to find the visible ones in 3D space (see Figure 4.4d). Specifically, for each point, if the part label of its projected pixel coordinates is not *none*, we consider it a visible point. In essence, this allows us to recover the visible parts of the model in 3D space.

For each rendered view of the target model, we possess per-pixel part labels and visible points in 3D space. We consider models from the similar group of the target model as mapping candidates. We iterate through these candidates, employing exactly the same rendering parameters as the target model, and obtain their per-pixel part labels and visible parts in 3D space (see Figure 4.4).

We then compare this information with that of the target model: if the overlap of per-pixel part labels and the geometric similarity of visible parts exceed predefined thresholds, we include the candidate in the ground-truth mapping for the image. This process acts like a refinement of the similar groups, ensuring that the visual content of the image aligns with the mapping model in that specific rendering settings. On average, each image maps to three corresponding ground-truth shapes. Figure 4.5 shows a sample mapping for a specific rendering.

(a) Rendering. (b) Targeting ground-truth shape. (c) Mapping ground-truth shape 1. (d) Mapping ground-truth shape 2.

Figure 4.5: Sample of image to many ground-truth shapes mapping.

4.3 Generating Truncated Signed Distance Fields

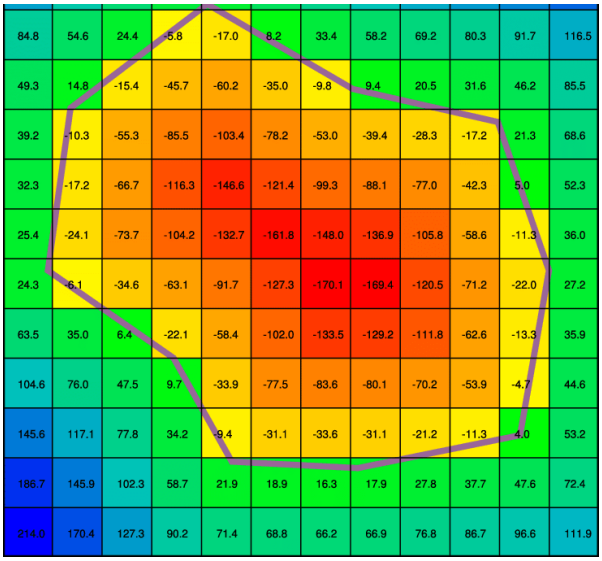


Figure 4.6: Signed distance field representation of a polygonal shape (from [66]). Space is discretized into cells. Cells storing a negative distance are inside the shape; cells with a positive distance are outside.

Since our training pipeline involves Truncated Signed Distance Fields (T-SDFs) as 3D representations, we also generate a T-SDF with 64^3 voxels from each model. This representation encodes the distance to the closest surface in each voxel. Figure 4.6 visualizes an example of SDF.

A signed distance field comprises voxels whose values represent the distance to the surface. Voxels farther away possess higher absolute values (depicted in red and blue in Figure 4.6), while closer ones exhibit lower absolute values. These values are also signed, with voxels outside the object having a negative sign and those inside having a positive one. The actual

surface is located at the zero-crossing between voxels. Extraction methods, such as Marching Cubes [30], leverage this property to generate surface meshes from these distance fields. To extract SDF, we follow the preprocessing steps in DISN [19]. The shapes are normalized to lie in a origin-centered cube in [-1, 1], and their signed distance function is evaluated at locations in a uniformly sampled 64^3 grid. We use 0.2 as the threshold to further obtain the Truncated-SDF representations.

5 Network Architecture

We introduce a network architecture designed to take a single image as input and autonomously generate the corresponding shapes in an autoregressive manner (see Figure 3.2). The framework comprises four key components: the image encoder, the conditional module, the autoregressive modeling with transformer and the constraint module.

In this chapter, we go into a comprehensive exploration of these components, providing intricate details on their functionalities. Additionally, we elaborate on the specifics of the training and inference procedures adopted in our framework.

5.1 Image Encoder

The role of the image encoder is pivotal in the context of image-based 3D shape generation. An effective image encoder possesses the capability to extract essential features from images, playing a crucial role in facilitating the subsequent 3D shape generation process.

We adopt a pretrained CLIP [67] as our image encoder, chosen for its robust capacity to capture rich image representations. The CLIP encoder is trained in a contrastive manner, where the model learns to bring representations of semantically similar image-text pairs closer together and push representations of dissimilar pairs apart. It typically uses a vision transformer architecture [68], which leverages multi-head self-attention mechanisms to capture complex relationships between different patches in an image.

We specifically choose the "ViT-B/32" version of CLIP, yielding image encodings with a shape of $N \times D$, where $N = 50$ represents the number of image tokens, and $D = 768$ indicates the feature dimension.

5.2 Conditional Module

The conditional module establishes a connection between the image encodings and the flattened grid sequences. This ensures that the output grid sequence incorporates the essential image features for subsequent operations. Therefore, selecting a suitable conditional mechanism within this module is crucial to align the generated shape effectively with the visual content of the input image.

In this module, we propose to adopt the cross-attention mechanism to interact the grid sequences with the image encodings. Figure 5.1 depicts the architecture of it. Initially, we concatenate the query and the empty sequences, with each cell of these sequences containing its own positional embeddings. The concatenated sequence is then multiplied with a weight matrix Q. This matrix multiplication results in a sequence of *queries* with the shape of $N' \times d$,

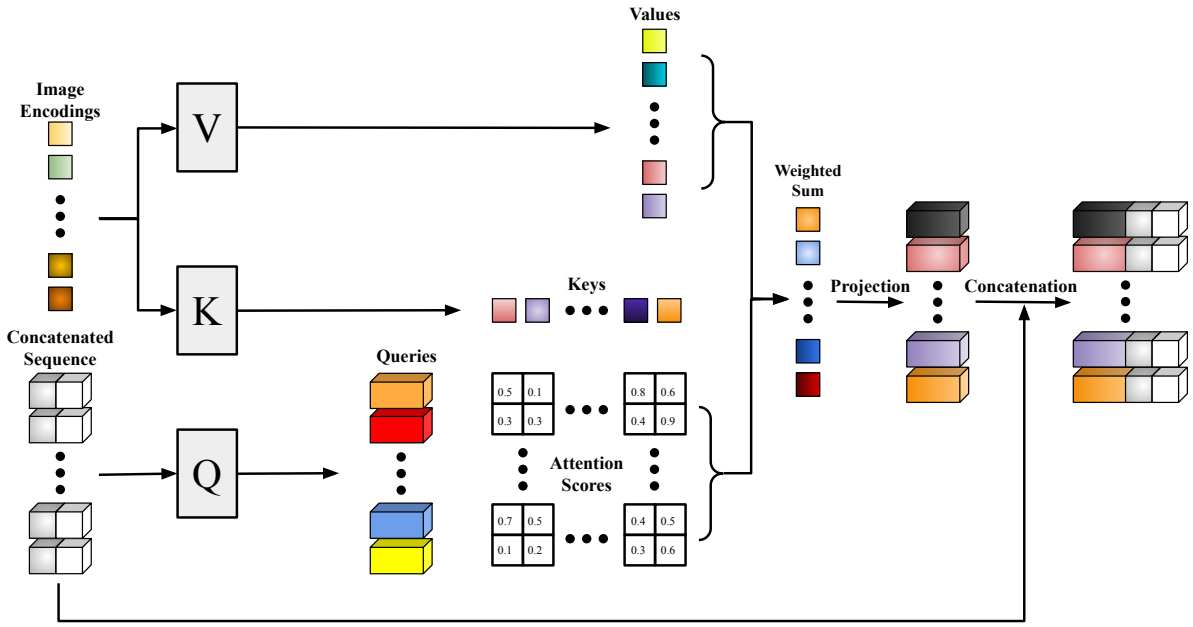


Figure 5.1: Cross-attention conditional module overview. We perform the cross-attention mechanism between the image encodings and the concatenated sequence such that the output sequence contains the necessary image features.

where N' is the length of the concatenated sequence and d is a predefined hidden dimension. Likewise, the image encodings are multiplied with weight matrices V and K independently, generating *values* and *keys* respectively. Both are in the shape of $N \times d$. Subsequently, each *query* performs dot product with each *key*, generating corresponding attention scores α:

$$\alpha_{mn} = \frac{softmax(q_m \cdot k_n)}{\sqrt{d}}. \tag{5.1}$$

where q_m is the m-th *query* and k_n is the n-th key. Then for each cell i of the concatenated sequence, its embedding is replaced by the weighted sum of *values*:

$$emb_i = \sum_{j=0}^{N-1} \alpha_{ij} \cdot v_j. \tag{5.2}$$

where v_j is j-th *value*. Finally, these embeddings are projected to the original dimension and concatenated with the original 3D embeddings for subsequent processing.

The input images inherently exhibit significant ambiguity, with the targeting object often positioned away from the image center, leading to some parts extending beyond the image plane, or being partially occluded by another object. This creates visually sparse content within the images and/or introduces substantial interference. Intuitively, we argue that the cross-attention mechanism between the image encodings and the sequence can effectively discern the region of interest in the image corresponding to a specific cell in the sequence.

The final concatenation step ensures the preservation of the original 3D features in the output sequence.

5.3 Autoregressive Modeling with Transformer

After obtaining the sequence that contains the necessary image features, we randomly permute it following the implementations from AutoSDF [1] before feeding it into the transformer. Thus, the conditional distribution $P(E'|c)$ is:

$$p(E'|c) = p_\theta([e'_{h_1}, c]) \cdot p_\theta([e'_{h_2}, c]|[e'_{h_1}, c]) \cdot p_\theta([e'_{h_3}, c]|[e'_{h_1}, c], [e'_{h_2}, c]) \cdots . \tag{5.3}$$

where θ represents learnable parameters of the conditional module and the transformer, e' represents the latent feature, and h represents an arbitrary known location. We learn this model by simply maximizing the log-likelihood of the latent representations conditioned on the input images using randomized orders for autoregressive generation.

In practice, the transformer comprises 12 encoder layers, each with 12 multi-head attention heads and a hidden dimension of 768. Notably, it does not contain a decoder, indicating that all attention layers are self-attention.

5.4 Constraint Module

While directly sampling from the conditional distribution produced by the transformer can yield 3D shapes with higher diversity, this diversity may come at the cost of compromising the overall quality of the generated shapes. This trade-off could result from the cumulative diversity introduced at each step of the autoregressive sampling process. Therefore, we introduce a constraint module to optimize the sampling process and enhance the quality of the generated shapes while still preserving the essential diversity.

Layer name	Weights/Parameters	Input size	Output size
ResNet-18 [18]	-	$B \times 3 \times 256 \times 256$	$B \times 512 \times 8 \times 8$
Linear_to_3D	-	$B \times 512 \times 64$	$B \times 512 \times 512$
3D ResNet Block	kernel 3×3, stride 1, padding 1	$B \times 512 \times 8 \times 8 \times 8$	$B \times 256 \times 8 \times 8 \times 8$
3D ResNet Block	kernel 3×3, stride 1, padding 1	$B \times 256 \times 8 \times 8 \times 8$	$B \times 128 \times 8 \times 8 \times 8$
3D ResNet Block	kernel 3×3, stride 1, padding 1	$B \times 128 \times 8 \times 8 \times 8$	$B \times 64 \times 8 \times 8 \times 8$
3D ResNet Block	kernel 3×3, stride 1, padding 1	$B \times 64 \times 8 \times 8 \times 8$	$B \times 64 \times 8 \times 8 \times 8$
Conv3D	kernel 3×3, stride 1, padding 1	$B \times 64 \times 8 \times 8 \times 8$	$B \times 512 \times 8 \times 8 \times 8$

Table 5.1: Architecture for the constraint module. B refers to the batch size.

The constraint module is trained independently, using the same image-shape pairs. It consists of two sub-modules. We integrate a pretrained ResNet18 [18] encoder as the first

sub-module. The second sub-module encompasses a set of layers designed to project 2D features onto 3D grids, aligning with the exact shape of the outputs from the transformer. Each cell within the grids encapsulates per-location constraints. Additional architectural details about the constraint module are provided in Table 5.1.

5.5 Training and Inference

In this section, we discuss some implementation details during training and inference about the main components outlined previously.

5.5.1 Training

During training, we randomly select one view from the rendered images as input. For the ground-truth shapes of autoregressive modeling, we uniformly sample one from the corresponding ground-truth mapping. While for the constraint module, we simply set the ground-truth shape to the target one without utilizing the mapping information. We also fine-tune the pretrained image encoders in both autoregressive modeling and the constraint module to make them more adaptable to our specific use case.

The input grid sequences to the **conditional module** are essentially the **complete sequence** containing the **ground-truth** elements **after random permutation**: $\left\{e'_{h_1}, e'_{h_2}, \cdots, e'_{h_{n-1}}\right\}$, and the query sequence containing the query locations for the **next** elements $\{l_{h_2}, l_{h_3}, \cdots, l_{h_n}\}$. Subsequently, to predict the h_j-th element in the sequence by the transformer, we have:

$$p_\theta(e'_{h_j}) = T_\theta(\left\{e'_{h_1}, e'_{h_2}, \cdots, e'_{h_{<j}}\right\}, l_{h_j}, c). \tag{5.4}$$

where T_θ indicates the transformer and the conditional module.

The training within the transformer is done in parallel and the targets are the elements starting from e'_{h_2}. We feed the attention mask with upper-triangular matrix of $-\infty$, and zeros on the diagonal to make sure the information do not leak from the future elements. We use fourier features for the positional embedding for all locations i following Tancik et al. [69]. The training objective for both autoregressive modeling and the constraint module is minimizing the expected negative log-likelihood:

$$L = \mathbf{E}_{X\sim p(X|c)} - [logp(E'|c)]. \tag{5.5}$$

Furthermore, to train the autoregressive framework, we use ADAM as the optimizer with an initial learning rate of 1e-4. Specifically, we utilize a learning rate of 1e-5 to fine-tune the CLIP image encoder. We decay both learning rates by multiplying 0.9 every 30 epochs. The batch size is fixed to 10 pairs of images and shapes per batch. We train it until convergence and it takes approximately 24 hours. As for the constraint module, we use the same settings as above except that we choose a batch size of 150. We follow Mittal et al. [1] to split training/test sets, where training set contains 4995 samples for the *chair* category and 5173 samples for the

table category, and test set contains 1244 samples for the *chair* category and 1304 samples for the *table* category. All networks are trained on a single NVIDIA GTX 1080ti.

5.5.2 Inference

During inference, we complete an empty sequence conditioned on the image in an autoregressive manner from the transformer, assisted by the constraint module. At step t, we have:

$$p^*(e'_{h_t}) = C\left[T_\theta(\left\{\hat{e'_{h_{t-1}}}, \hat{e'_{h_{t-2}}}, \hat{e'_{h_{t-3}}}, \cdots\right\}, l_{h_t}, c), c\right]. \tag{5.6}$$

where C represents the constraint module and $\hat{e'_{h_t}}$ is generated by sampling from the constrained likelihood distribution $p^*(e'_{h_t})$. In practice, to restrict the sampling process, we feed the same image into the constraint module and get the corresponding constraints. Hence, the final log-likelihood is essentially a weighted sum of raw log-likelihood and these constraints:

$$logp^*(e'_{h_t}) = w \cdot logp_c(e'_{h_t}) + (1-w) \cdot logp_\theta(e'_{h_t}). \tag{5.7}$$

where p_c indicates the constraints, p_θ indicates that the raw likelihood generated from the transformer, and w represents the constraint weight. Once we obtain the constrained complete sequence, we simply feed it into D_ϕ to generate one plausible 3D shape.

6 Results and Discussion

In this chapter, we present the results of our experiments and engage in a comprehensive discussion. Initially, we conduct experiments using our synthetic data, as outlined in chapter 4. Then we extend our experimentation to the Pix3D [25] dataset, which comprises real-world images rather than renderings from CAD models.

We call our novel one-image-to-many-ground-truth-shape strategy as "O2M" and other components retain their full names. We compare our method with several alternative methods encompassing non-generative networks such as Pix2Vox [20] and Pix2Vox++[21], as well as the generative network we build upon, AutoSDF[1].

Furthermore, we do several ablation studies in terms of O2M and the constraint module to validate the effectiveness of including these components.

In the end, we showcase qualitative samples and discuss some of the limitations of our approach.

6.1 Quantitative Evaluation

To assess the diversity in generated shapes, we employ the Total Mutual Difference (TMD), computed as the average chamfer distance among k generated shapes. We set $k = 6$ in our experiments, aligning with the average number of models within a similar group (refer to chapter 4). For evaluating the generation quality, we utilize bidirectional l_2 chamfer distance (CD) and F-score@1% [70] as metrics. In particular, CD calculations are based on 2048 sampled points from meshes, while F-score calculations use 8192 points. CD between the ground-truth points G and predicted points P is defined as:

$$CD(G,P) = \frac{1}{|P|}\sum_{p\in P}\min_{g\in G}||p-g||_2^2 + \frac{1}{|G|}\sum_{g\in G}\min_{p\in P}||p-g||_2^2. \tag{6.1}$$

And F-score is calculated as:

$$e_p = \min_{g\in G}||p-g||_2, \qquad Pr(d) = \frac{1}{||P||}\sum_{p\in P}[e_p < d], \tag{6.2}$$

$$e_g = \min_{p\in P}||g-p||_2, \qquad Re(d) = \frac{1}{||G||}\sum_{g\in G}[e_g < d]. \tag{6.3}$$

$$F(d) = \frac{2Pr(d)Re(d)}{Pr(d)+Re(d)}. \tag{6.4}$$

where d indicates a distance threshold (1%). We present the minimal CD and the highest F-score among the k generated shapes in our results.

6.1.1 Synthetic Data

Firstly, we compare our results with other methods from experiments conducted on the synthetic data from ShapeNet [23]. For our method, we utilize the O2M strategy and set the weight of constraint module to 0.2. The ablation on the O2M strategy and the reason for choosing the constraint weight are discussed in section 6.2. For each of other methods, we conduct experiments both with and without the O2M strategy. All methods are (re-)trained on the synthetic data and evaluated under the same criteria.

	O2M	Chair			Table			Average		
		TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑
Pix2Vox [20]	✘	0.022	7.74	0.1339	0.022	18.92	0.0717	0.022	13.46	0.1028
	✔	0.023	7.76	0.1338	0.023	18.05	0.0712	0.023	13.12	0.1017
Pix2Vox++ [21]	✘	0.023	7.50	0.1322	0.023	17.58	0.0718	0.023	12.65	0.1012
	✔	0.023	7.77	0.1311	0.023	17.46	0.0753	0.023	12.72	0.1025
AutoSDF [1]	✘	0.040	2.90	0.2535	0.039	3.59	0.2679	0.039	3.25	0.2609
	✔	0.042	2.97	0.2433	0.041	3.65	0.2648	0.041	3.32	0.2543
Ours	✔	**0.047**	**2.59**	**0.2681**	**0.053**	**3.44**	**0.2782**	**0.050**	**3.02**	**0.2710**

Table 6.1: Results comparisons on synthetic data. We use the Total Mutual Difference (TMD) to measure the diversity of the generated shapes and bidirectional l_2 Chamfer Distance (CD) ($\times$1000) and F-score@1% to evaluate the quality of generation. Best results are in bold.

From Table 6.1, it is clear that our method outperforms the two non-generative networks Pix2Vox and Pix2Vox++, and the generative network AutoSDF in all evaluation metrics, either they are trained with or without O2M. In particular, despite our method is trained with O2M, meaning it is required to handle a more complex one-to-many relationship, it is still better than AutoSDF without O2M in terms of generation quality. These results suggest that (1) the generated shapes align well with the visual content of the input images; and (2) the generated shapes exhibit high diversity.

6.1.2 Real-world Data

To further assess the domain transferability of our method, we conduct experiments using real-world data from the Pix3D dataset [25]. This dataset comprises real-world images along with corresponding CAD models, and for our experiments, we analyze the *chair* and the *table* categories. We specifically choose images that are labeled as "truncated" or "occluded", and randomly split them to training/test sets at an 80:20 ratio, where training set contains 690

samples for *chair* and 720 samples for *table*, and test set contains 174 samples for *chair* and 182 samples for *table*. Since the renderings from the synthetic data do not contain background, we also mask out the background of the images in the Pix3D using given binary masks. For our method, we fine-tune all components except the transformer from the pretrained model using real-world data for 100 epochs, and do not include the constraint module. The intuition is that we want to adapt our pretrained model to a new distribution but still preserve the ability to generate diverse shapes, and this ability is mainly derived from the autoregressive modeling via the transformer. We compare our method with AutoSDF, where we train it from scratch using the same real-world data.

	Chair			Table			Average		
	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑
AutoSDF [1]	0.019	3.03	**0.5706**	0.020	7.13	0.4925	0.019	5.09	**0.5312**
Ours	**0.025**	**2.60**	0.5523	**0.027**	**3.93**	**0.5046**	**0.026**	**3.27**	0.5282

Table 6.2: Results comparisons on real-world data. The evaluation criteria is the same as used in the synthetic data. Best results are in bold.

Results from Table 6.2 clearly demonstrates that our method outperforms AutoSDF in most evaluation metrics. Notably, despite AutoSDF is trained from scratch, indicating that it directly centers on the real-world data distribution, our fine-tuned model is still capable of producing higher-quality shapes from real-world images. These facts suggest that our pretrained model on the synthetic data is able to be transferred to the real-world domain properly, preserving both generation diversity and quality.

# epochs	Chair			Table			Average		
	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑
50	**0.032**	2.85	0.4905	**0.032**	**3.82**	0.4872	**0.032**	3.33	0.4888
100	0.025	**2.60**	0.5523	0.027	3.93	0.5046	0.026	**3.27**	0.5282
150	0.021	2.72	**0.5702**	0.024	3.95	0.5123	0.022	3.34	0.5410
200	0.020	2.68	0.5699	0.023	3.90	**0.5179**	0.021	3.29	**0.5441**

Table 6.3: Analysis of performance changes on real-world data in relation to the number of epochs during fine-tuning our pretrained model. Best results are in bold.

Additionally, we conduct an analysis of the pretrained model's performance in relation to the number of epochs during fine-tuning. The results, detailed in Table 6.3, indicate that the model fine-tuned for 100 epochs achieves the highest level of generation quality in terms of average CD. Interestingly, both generation diversity and quality show a decrease beyond the 100-epoch mark. This observation underscores the inherent capability of the pretrained model to produce high-quality and diverse shapes even within a relatively short fine-tuning period.

6.2 Ablation Studies

In our ablation studies, our primary focus is on evaluating the impact of the O2M strategy and the constraint module within our method. We ablate these components on our synthetic dataset. To assess the effectiveness of the O2M strategy, we conduct two additional experiments: (1) utilizing a less strict image-to-shape mapping, where we set all candidates from the similar groups as the ground-truth shapes, and (2) forgoing the use of any mapping information. Further details about mapping can be found in chapter 4. Regarding the constraint module, we conduct ablations on the constraint weights, ranging from 0.0 to 0.6, where 0.0 implies no constraints are added during the sampling process.

O2M	Constraint Weight	Chair			Table			Average		
		TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑	TMD ↑	CD ↓	F-score ↑
✘	0.2	0.045	2.59	0.2751	0.052	3.39	0.2868	0.048	2.99	0.2810
✔		0.047	2.59	0.2681	0.053	3.44	0.2782	0.050	3.02	0.2710
non-strict		**0.056**	2.83	0.2401	**0.061**	3.63	0.2530	**0.058**	3.23	0.2467
✔	0.0	0.051	2.78	0.2539	0.060	3.71	0.2643	0.055	3.25	0.2592
	0.1	0.049	2.69	0.2607	0.057	3.51	0.2778	0.053	3.10	0.2689
	0.3	0.045	**2.57**	0.2703	0.050	3.32	0.2882	0.047	2.95	0.2794
	0.4	0.043	2.58	0.2709	0.047	**3.29**	**0.2922**	0.045	**2.94**	**0.2818**
	0.5	0.041	2.62	**0.2718**	0.043	3.30	0.2885	0.042	2.96	0.2803
	0.6	0.039	2.70	0.2685	0.040	3.40	0.2819	0.039	3.05	0.2753

Table 6.4: Ablation studies. We conduct ablations for our method on the synthetic data. "non-strict" means we use a less strict mapping. Best results are in bold.

Analyzing the results presented in Table 6.4, it becomes apparent that our method, when trained with O2M, demonstrates a greater capacity to generate diverse shapes compared to the version trained without O2M. This observation highlights the efficacy of the O2M strategy in enhancing the diversity of the generated shapes. Additionally, when opting for a less strict mapping, the diversity level further increases, indicating our method's ability to capture the varying degrees of diversity present in the ground-truth data.

Furthermore, as the constraint weight increases, the TMD decreases, highlighting the constraint module is effective in restricting the sampling process. On the other hand, with the constraint weight increasing from 0.0 to 0.4, the generation quality increases, indicating the positive impact of the constraint module on generating higher-quality shapes. However, when further increasing the constraint weight from 0.4 to 0.6, the generation quality shows a decline. This observation suggests that the primary contribution to high-quality generation comes from the conditional autoregressive generation pipeline rather than the constraint module. The average TMD within the similar groups (refer to chapter 4) is 0.051, serving as an upper limit for the TMD of the generated results. Consequently, we opt for a constraint weight of 0.2, as it yields an average TMD of 0.050 and further increasing the diversity may at the cost of compromising the generation quality.

6.3 Qualitative Samples

We utilize the Marching Cubes [30] implementation provided by PyMCubes [71] to extract surface meshes from the predicted distance field volumes.

In Figure 6.1 and 6.2, we present qualitative comparisons of synthetic data among Pix2Vox, AutoSDF, and our method. The visual inspection clearly demonstrates that our method excels in generating higher-fidelity shapes from a single RGB image compared to the other two baselines. Notably, in most circumstances, each hypothesis generated by our method exhibits two key characteristics: (1) distinctiveness from one another; (2) a robust alignment with the visual content of the target object in the input image.

In addition to the results obtained from synthetic data, we also present qualitative samples from real-world data in Figure 6.3, 6.4, 6.5 and 6.6. Analyzing these visualizations reveals that our method consistently generates higher-quality shapes compared to AutoSDF in both *chair* and *table* categories. In terms of generation diversity, in the second set of comparison from Figure 6.3, given a chair image taken from the frontal view, our method is able to produce diverse and plausible chair backs for each hypothesis, especially compared with those produced by AutoSDF.

6.4 Limitations

In this thesis, our primary experiments are based on synthetic data. Despite our efforts to simulate realistic scenarios, a gap remains between the synthetic environment and real-world scenes. Real-world images typically include backgrounds, and objects may experience multiple occlusions. Additionally, there is greater variability in light energy in real-world images. In future work, we aim to take these factors into account to enhance the model's adaptability when transferring to the real-world domain.

Besides, our experiments are currently focused on the *chair* and *table* categories, with a limited number of baselines for comparison. We plan to expand our study to include more object categories and compare our method with a broader range of state-of-the-art models in future research.

In terms of results, one noticeable issue is that our model struggles generating accurate holes, as observed in the first comparisons in Figure 6.3 and 6.4. Additionally, there are still inconsistencies between the generated shapes and the visual content of the input images, exemplified by the last hypothesis in the second comparisons from Figure 6.1. Addressing these issues will be an another focus of our future work.

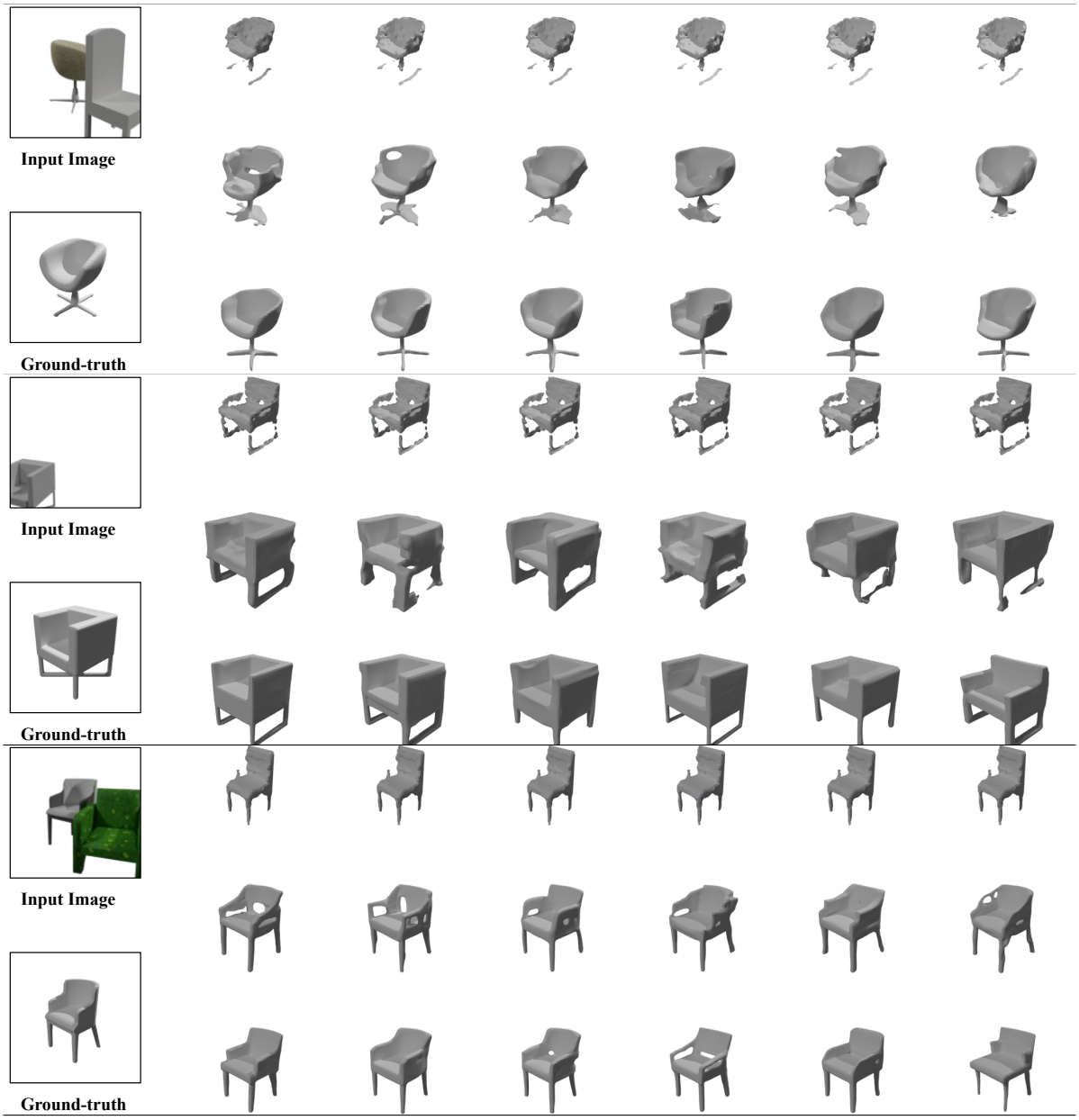


Figure 6.1: Qualitative results comparisons on the *chair* category of synthetic data. For each row within in a set of comparisons, 1st: Pix2Vox; 2nd: AutoSDF; 3rd: Ours. All methods are trained with O2M.

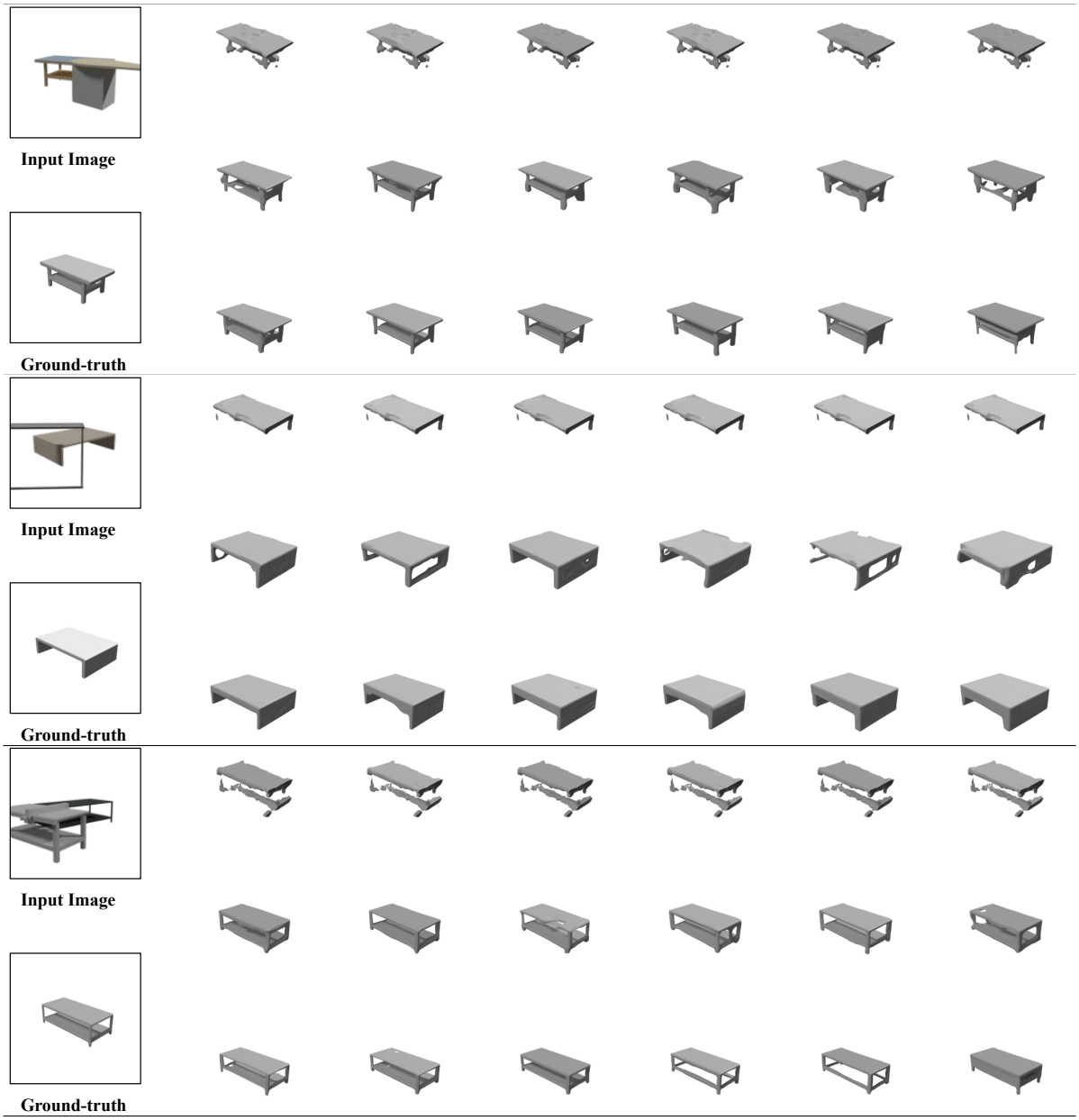


Figure 6.2: Qualitative results comparisons on the *table* category of synthetic data. For each row within in a set of comparisons, 1st: Pix2Vox; 2nd: AutoSDF; 3rd: Ours. All methods are trained with O2M.

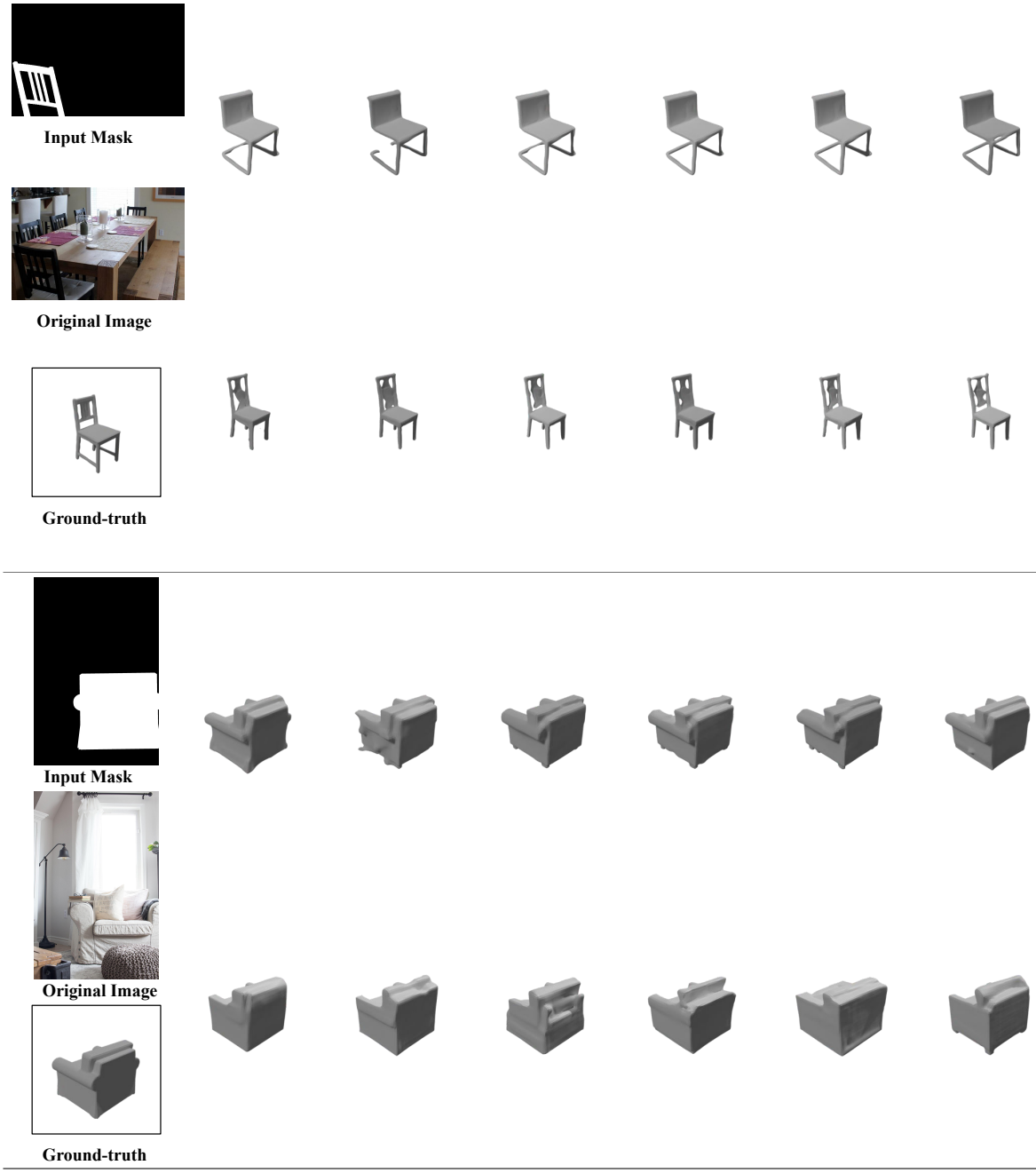


Figure 6.3: Qualitative results comparisons on the *chair* category of real-world data. For each row within in a set of comparisons, 1st: AutoSDF; 2nd: Ours.

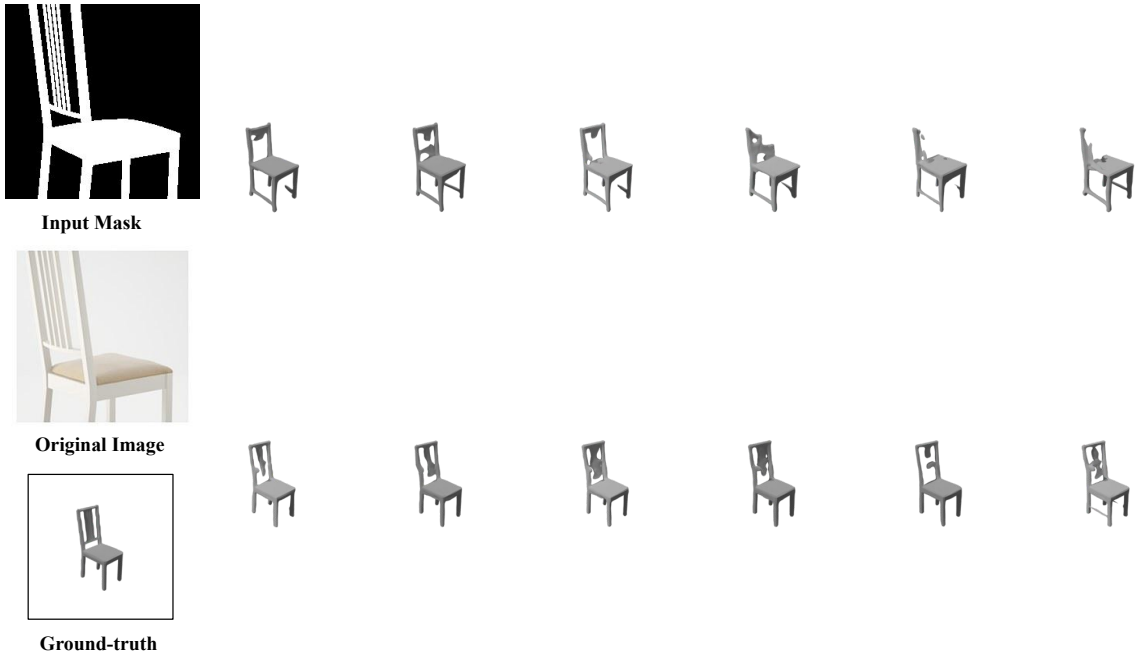


Figure 6.4: Qualitative results comparisons on the *chair* category of real-world data. For each row within in a set of comparisons, 1st: AutoSDF; 2nd: Ours.

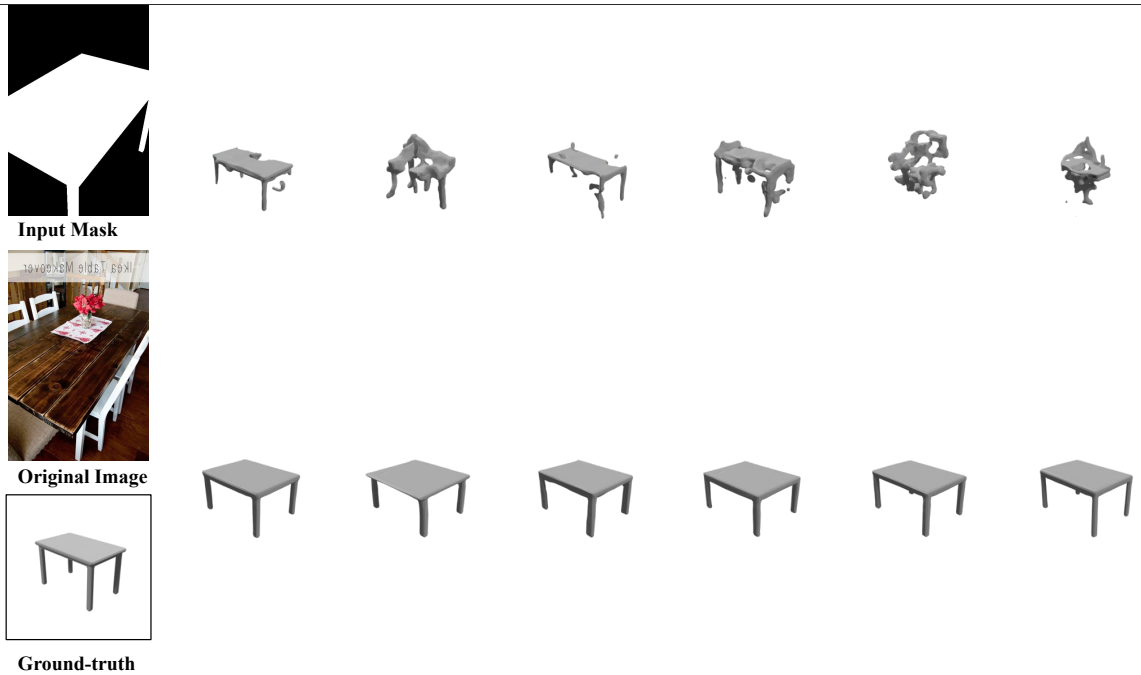


Figure 6.5: Qualitative results comparisons on the *table* category of real-world data. For each row within in a set of comparisons, 1st: AutoSDF; 2nd: Ours.

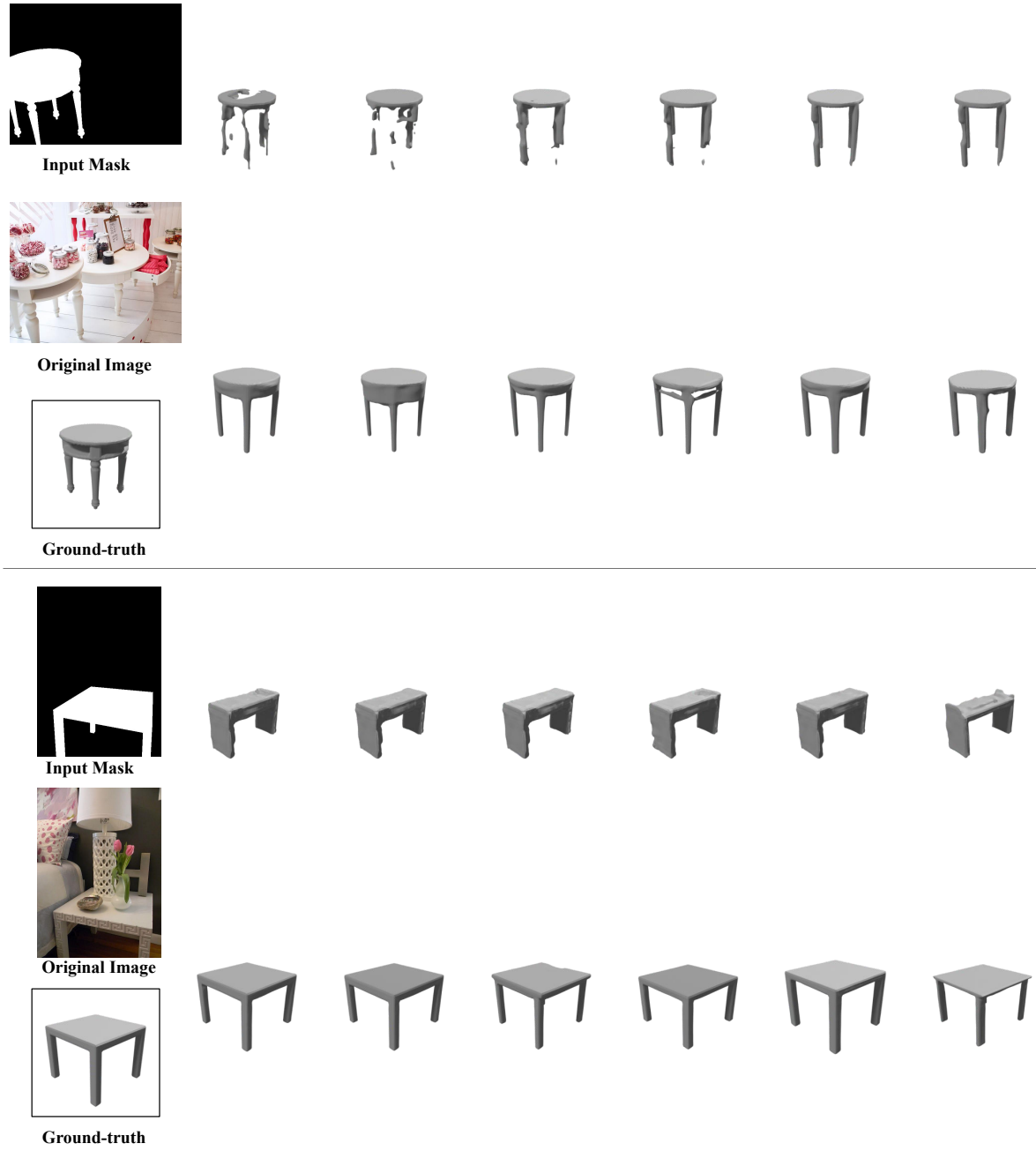


Figure 6.6: Qualitative results comparisons on the *table* category of real-world data. For each row within in a set of comparisons, 1st: AutoSDF; 2nd: Ours.

7 Conclusion

In conclusion, we present a data-driven method rooted in deep learning that is capable of generating the probabilistic distribution of shapes conditioned on a single RGB image. By sampling from this distribution, our approach facilitates the generation of multiple plausible 3D shapes corresponding to the input image.

Our model is trained on **newly generated training pairs** tailored to our specific use case. Additionally, we employ the **O2M** training strategy, allowing one input image to be mapped to multiple ground-truth shapes. This strategy proves effective in enhancing the diversity among the generated shapes. To further improve generation quality, we incorporate the **conditional module** driven by the **cross-attention mechanism**, along with the **constraint module**. The integration of these techniques enables the generation of shapes from an RGB image that meets two key criteria: (1) a high degree of diversity and (2) a strong alignment with the visual content present in the input image.

Our method demonstrates promising results in generating diverse and visually aligned 3D shapes from single RGB images. As we continue to explore and refine our approach, we aim to address the identified limitations and extend our evaluations to cover a wider range of object categories. We sincerely aspire that the outcomes of this work will serve as a valuable contribution to diverse practical applications in the future, ranging from content creation to robotics and self-driving cars, among others.

List of Figures

1.1 A chair with high self-occlusion. 1
1.2 Examples for content creation. 2
1.3 Multiple forms of invisibility. 4

2.1 Different 3D representations. 8

3.1 Patch-wised encoding VQ-VAE overview. 13
3.2 Conditional shape generation pipeline overview. 15

4.1 ShapeNet CAD model samples. 18
4.2 Fine-grained instance-level segmentation visualization from PartNet [24]. . . . 19
4.3 Two strategies to induce ambiguity. 19
4.4 Intermediate outputs for generating ground-truth mappings. 21
4.5 Sample of image to many ground-truth shapes mapping. 22
4.6 Signed distance field representation of a polygonal shape (from [66]). 22

5.1 Cross-attention conditional module overview. 26

6.1 Qualitative results comparisons on the *chair* category of synthetic data. 36
6.2 Qualitative results comparisons on the *table* category of synthetic data. 37
6.3 Qualitative results comparisons on the *chair* category of real-world data. . . . 38
6.4 Qualitative results comparisons on the *chair* category of real-world data. . . . 39
6.5 Qualitative results comparisons on the *table* category of real-world data. . . . 39
6.6 Qualitative results comparisons on the *table* category of real-world data. . . . 40

List of Tables

5.1 Architecture for the constraint module. 27

6.1 Results comparisons on synthetic data. 32
6.2 Results comparisons on real-world data. 33
6.3 Analysis of performance changes on real-world data in relation to the number of epochs during fine-tuning our pretrained model. 33
6.4 Ablation studies. 34

Bibliography

[1] P. Mittal, Y.-C. Cheng, M. Singh, and S. Tulsiani. "Autosdf: Shape priors for 3d completion, reconstruction and generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 306–315.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[4] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[6] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[7] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]`.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2016. arXiv: `1511.00561 [cs.CV]`.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: `1606.00915 [cs.CV]`.

[10] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. *Pyramid Scene Parsing Network*. 2017. arXiv: `1612.01105 [cs.CV]`.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[12] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2147–2154.

[13] R. Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[14] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[15] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer. 2016, pp. 628–644.

[16] S. R. Richter and S. Roth. "Matryoshka networks: Predicting 3d geometry via nested shape layers". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1936–1944.

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

[18] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[19] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction". In: *Advances in neural information processing systems* 32 (2019).

[20] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang. "Pix2vox: Context-aware 3d reconstruction from single and multi-view images". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2690–2698.

[21] H. Xie, H. Yao, S. Zhang, S. Zhou, and W. Sun. "Pix2Vox++: Multi-scale context-aware 3D object reconstruction from single and multiple images". In: *International Journal of Computer Vision* 128.12 (2020), pp. 2919–2935.

[22] D. Maturana and S. Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 922–928.

[23] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[24] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 909–918.

[25] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. "Pix3d: Dataset and methods for single-image 3d shape modeling". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2974–2983.

[26] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[27] Y. Xu, S. Peng, C. Yang, Y. Shen, and B. Zhou. "3d-aware image synthesis via learning structural and textural representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18430–18439.

[28] K. Schwarz, A. Sauer, M. Niemeyer, Y. Liao, and A. Geiger. "Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 33999–34011.

[29] Z. Zhang. "Microsoft kinect sensor and its effect". In: *IEEE multimedia* 19.2 (2012), pp. 4–10.

[30] L. We. "Marching cubes: A high resolution 3d surface construction algorithm". In: *Comput Graph* 21 (1987), pp. 163–169.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[32] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems* 29 (2016).

[33] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[34] E. J. Smith and D. Meger. "Improved adversarial systems for 3d object generation and reconstruction". In: *Conference on Robot Learning*. PMLR. 2017, pp. 87–96.

[35] V. A. Knyaz, V. V. Kniaz, and F. Remondino. "Image-to-voxel model translation with conditional adversarial networks". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

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

[37] J. Ho, A. Jain, and P. Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[38] P. Dhariwal and A. Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[39] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. *Cascaded Diffusion Models for High Fidelity Image Generation*. 2021. arXiv: `2106.15282 [cs.CV]`.

[40] A. Nichol and P. Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].

[41] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. "Image super-resolution via iterative refinement". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2022), pp. 4713–4726.

[42] S.-I. Cheng, Y.-J. Chen, W.-C. Chiu, H.-Y. Tseng, and H.-Y. Lee. "Adaptively-realistic image generation from stroke and sketch with diffusion model". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 4054–4062.

[43] O. Avrahami, D. Lischinski, and O. Fried. "Blended diffusion for text-driven editing of natural images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18208–18218.

[44] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani. *Imagic: Text-Based Real Image Editing with Diffusion Models*. 2023. arXiv: 2210.09276 [cs.CV].

[45] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. 2022. arXiv: 2112.10741 [cs.CV].

[46] T. Rahman, H.-Y. Lee, J. Ren, S. Tulyakov, S. Mahajan, and L. Sigal. *Make-A-Story: Visual Memory Conditioned Consistent Story Generation*. 2023. arXiv: 2211.13319 [cs.CV].

[47] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV].

[48] L. Zhou, Y. Du, and J. Wu. "3d shape generation and completion through point-voxel diffusion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5826–5835.

[49] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[50] S. Luo and W. Hu. "Diffusion probabilistic models for 3d point cloud generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2837–2845.

[51] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis. *LION: Latent Point Diffusion Models for 3D Shape Generation*. 2022. arXiv: 2210.06978 [cs.CV].

[52] Y.-C. Cheng, H.-Y. Lee, S. Tulyakov, A. Schwing, and L. Gui. *SDFusion: Multimodal 3D Shape Completion, Reconstruction, and Generation*. 2023. arXiv: 2212.04493 [cs.CV].

[53] C. Sbrolli, P. Cudrano, M. Frosi, and M. Matteucci. *IC3D: Image-Conditioned 3D Diffusion for Shape Generation*. 2023. arXiv: 2211.10865 [cs.CV].

[54] J. Ho and T. Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].

[55] Y. Bengio and S. Bengio. "Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

[56] B. Uria, I. Murray, and H. Larochelle. *RNADE: The real-valued neural autoregressive density-estimator*. 2014. arXiv: 1306.0186 [stat.ML].

[57] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. *PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications*. 2017. arXiv: 1701.05517 [cs.LG].

[58] Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma. *PointGrow: Autoregressively Learned Point Cloud Generation with Self-Attention*. 2019. arXiv: 1810.05591 [cs.CV].

[59] M. Ibing, G. Kobsik, and L. Kobbelt. *Octree Transformer: Autoregressive 3D Shape Generation on Hierarchically Structured Sequences*. 2021. arXiv: 2111.12480 [cs.CV].

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[61] X. Yan, L. Lin, N. J. Mitra, D. Lischinski, D. Cohen-Or, and H. Huang. *ShapeFormer: Transformer-based Shape Completion via Sparse Representation*. 2022. arXiv: 2201.10326 [cs.CV].

[62] A. Razavi, A. van den Oord, and O. Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf.

[63] P. Esser, R. Rombach, and B. Ommer. *Taming Transformers for High-Resolution Image Synthesis*. 2021. arXiv: 2012.09841 [cs.CV].

[64] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. *Neural Discrete Representation Learning*. 2018. arXiv: 1711.00937 [cs.LG].

[65] G. A. Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[66] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner. "An optimization approach to rough terrain locomotion". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 3589–3595.

[67] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[68] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[69] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*. 2020. arXiv: 2006.10739 [cs.CV].

[70] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. *What Do Single-view 3D Reconstruction Networks Learn?* 2019. arXiv: 1905.03678 [cs.CV].

[71] *PyMCubes*. URL: https://pypi.org/project/PyMCubes/.