

THE UNIVERSITY OF CHICAGO

STRUCTURED NEURAL REPRESENTATIONS FOR
3D SHAPE AND SCENE EDITING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
WILLIAM GAO

CHICAGO, ILLINOIS

AUGUST 2026

Copyright © 2026 by William Gao
All Rights Reserved

To those who remained patient with me

TABLE OF CONTENTS

LIST OF FIGURES	vii
LIST OF TABLES	xii
ACKNOWLEDGMENTS	xiii
ABSTRACT	xv
1 INTRODUCTION	1
2 CONVOLUTIONAL AUTOENCODERS FOR TETRAHEDRAL MESHES . . .	4
2.1 Introduction	4
2.2 Related Work	5
2.2.1 Implicit Representations	6
2.2.2 Tetrahedral Meshes	6
2.2.3 Contrast to Voxels	8
2.3 Method	8
2.3.1 Tetrahedral grid framework	9
2.3.2 TetGAN	11
2.4 Experiments	13
2.4.1 Qualitative results	14
2.4.2 Quantitative results	15
2.4.3 Ablation Study	17
2.5 Discussion	18
3 TEXT-DRIVEN GEOMETRY OPTIMIZATION	20
3.1 Introduction	20
3.2 Related Works	23
3.2.1 Text-Guided 3D Synthesis	23
3.2.2 Neural Shape Deformation	24
3.3 TextDeformer	25
3.3.1 Deformations through Jacobians	26
3.3.2 Language Guidance	26
3.3.3 Jacobian Regularization	27
3.3.4 View Consistency.	28
3.4 Experiments	29
3.4.1 Generality of TextDeformer	30
3.4.2 Expressiveness of TextDeformer	31
3.4.3 Identity Preservation	33
3.4.4 Viewpoint Consistency	34
3.4.5 Effect of Jacobians	35
3.4.6 Qualitative Comparisons	37

3.4.7	Quantitative Evaluation	37
3.5	Discussion	39
4	GENERATIVE 3D EDITING WITHOUT OPTIMIZATION	41
4.1	Introduction	41
4.2	Related Works	43
4.2.1	Large Reconstruction Models	43
4.2.2	Shape Editing	44
4.2.3	Masked Vision Models	44
4.3	Masked LRMs	45
4.3.1	Masked LRM Architecture	45
4.3.2	Supervision	48
4.3.3	Training Stages	49
4.3.4	Mesh Editing	50
4.4	Experiments	52
4.4.1	Quantitative Comparisons	52
4.4.2	Qualitative Evaluations	54
4.4.3	Mesh Editing Characteristics	57
4.4.4	Ablations Studies & Discussion	58
4.5	Discussion	60
5	SCENE DECOMPOSITION WITH MANIPULABLE COMPONENTS	62
5.1	Introduction	62
5.2	Related Works	64
5.2.1	Neural Radiance Fields	64
5.2.2	Compositional and Editable Radiance Fields	65
5.2.3	3D Gaussian Splatting	65
5.3	Method	66
5.3.1	Scene Representation	66
5.3.2	Volume Rendering	68
5.3.3	Supervision	69
5.3.4	Label-free Decomposition	70
5.4	Experiments	72
5.4.1	Instance-Labeled Meshes	73
5.4.2	Comparisons	76
5.4.3	Unsupervised 3D Segmentation	79
5.4.4	Unbounded Real-World Scenes	82
5.5	Discussion	84
6	CONCLUSION	86
	REFERENCES	88

A	CONVOLUTIONAL AUTOENCODERS FOR TETRAHEDRAL MESHES . . .	106
A.1	Architecture Details	106
A.2	Implementation Details	107
A.2.1	Subdivision	107
A.2.2	Hyperparameters	107
A.2.3	Runtime	108
A.3	Experiments	108
A.3.1	Dataset details	108
A.3.2	Additional Quantitative Results	108
A.3.3	Additional Qualitative Results	110
B	GENERATIVE 3D EDITING WITHOUT OPTIMIZATION	114
B.1	Mesh Editing Metrics	114
B.2	Additional Implementation Details	114
B.3	Additional Qualitative Examples	115

LIST OF FIGURES

2.1	Comparison of volumetric shape representations for a torus. Left to right: (a) occupancy extracted from an undeformed 161^3 tetrahedral grid, (b) a deformable 41^3 tetrahedral grid, and (c) a 301^3 voxel representation extracted using marching cubes. Deformable tetrahedral representations achieve comparable geometric fidelity at substantially lower resolution.	6
2.2	TetGAN overview. We represent the input mesh using a deformation field over a tetrahedral grid. The encoder produces a latent vector $\mathbf{Z}$, which is used as input to a decoder which performs a series of convolutions and upsampling operations. The two discriminators learn to classify whether the input shape is fake (generated) or real at the patch/global level.	8
2.3	TetGAN subdivision, pooling, and upsampling operations.	11
2.4	Novel random samples from TetGAN learned latent distribution.	13
2.5	Novel random samples from (a) OccNet, (b) ShapeGAN (VAE), and (c) ShapeGAN, and (d) TetGAN respectively	13
2.6	TetGAN’s latent space interpolation for different categories.	14
2.7	Latent-space arithmetic between meshes. The operation $A - B + C = D$ transfers semantic geometric attributes between shapes. Highlighted regions indicate the transferred component and its insertion into the resulting geometry.	15
2.8	Ablations. Top to bottom: (a) full TetGAN, (b) surface GCN, (c) tetrahedral GCN, (d) no patch discriminator, (e) no global discriminator, (f) VAE only. . .	17
2.9	Our deformation-field weighted smooth (Def. Our Smooth) improves the quality of the final result.	17
3.1	CLIP-Similarity Optimization	20
3.2	TextDeformer Overview. TextDeformer deforms a base mesh by optimizing per-triangle Jacobians using natural language as a guide. We optimize the deformation using three losses: a CLIP-based semantic loss drives the deformation toward the text prompt, a view-consistency loss matches multiple views of the same surface patch to ensure a coherent deformation, and our regularization on the Jacobians controls the fidelity to the base mesh.	25
3.3	Gallery of results. Source meshes inset and target text below.	29
3.4	TextDeformer deforms a source shape into various text-specified targets. The mesh colors visualize the smoothness of the mappings.	30
3.5	Text-driven deformation. Our method can deform the same source into different text-specified geometries, achieving local geometric details (shoe laces and giraffe) and low-frequency shape modifying deformations (body of guitar, shape of animal).	31
3.6	Visualizing iterations of the optimization as the source mesh is deformed to the target (“ <i>Einstein</i> ”, “ <i>Obama</i> ”). Due to our formulation and energy, each facial feature of Max Planck is deformed into the corresponding facial feature of the target (nose to nose, eyes to eyes etc.).	32

3.7	Deformations of different source vases into the same target text, “ <i>a cactus</i> ” and different source meshes into the same target text, “ <i>a skyscraper</i> ”. Our method meaningfully respects the input geometry while conforming to the desired target text.	33
3.8	Identity preservation. The source shape (left) is deformed into a “ <i>giraffe</i> ” (top), “ <i>pagoda</i> ” (bottom) using decreasing amount of weight on the proposed Jacobian regularization. Higher weight encourages preservation of the identity of the source. Note that results with zero Jacobian regularization (right column) may contain artifacts.	34
3.9	Viewpoint Consistency Ablation. Ablation results of removing $\mathcal{L}_{VC}$ for four different source-text pairs. In each example we see instances of incorrect geometry in the shapes deformed without $\mathcal{L}_{VC}$	34
3.10	Deformation Ablation. Top row: TextDeformer with vertex displacements. Bottom row: TextDeformer with Jacobians. Jacobians are crucial to preserving the structure of the input geometry and maintaining high-quality surfaces. . . .	35
3.11	Self-intersections. Comparison results for the shown source and target text “goggles”. Self-intersections are highlighted in red (bottom row). Removing view-consistency (VC) losses causes distortion on the temple arms. Removing Jacobians and optimizing vertices introduces further surface distortion and self-intersections which may impede utility. When applying CLIP-Mesh to this template, we observe the “janus” effect e.g., unrealistic repeated geometry on each side.	36
3.12	Globally-Coherent Deformations. Max Planck deforms into different targets using our method (bottom row), where the front/back view is shown in pairs of left/right. Removing Jacobians and predicting displacements (top row) takes locally sub-optimal steps, which results in distorted shapes with significant artifacts. Jacobians produce global deformations, resulting in cleaner geometries and even prevents the spurious face mirroring (for example, in “ <i>Obama</i> ”).	37
3.13	Qualitative Comparison From top to bottom, the target text prompts are “ <i>an alligator</i> ”, “ <i>a camel</i> ”, “ <i>a comfortable chair</i> ”, “ <i>a chinese lantern</i> ”, “ <i>an octopus</i> ”. Compared to CLIP-Mesh, our method produces more semantically correct and higher quality surfaces. Text2Mesh fails to produce semantically meaningful deformations in all examples.	38
4.1	Large Reconstruction Models: A typical LRM tokenizes posed images and learns a triplane representation, which can be sampled and decoded into a color and density field. A mesh may be extracted from this field or the field may be volumetrically rendered.	42

4.2	Training Pipeline: The images and camera poses are patchified and projected into tokens. A random 3D mask is generated and tokens corresponding to occluded patches are replaced by a learnable mask token. Camera and image tokens are summed and concatenated with learnable triplane tokens to form the transformer input. A clean conditional image is tokenized, forming the cross-attention input. The output triplane tokens are upsampled and decoded into colors and SDF values, which are transformed into densities for volumetric rendering. . . .	46
4.3	Mesh Editing using MaskedLRMs: The inputs comprise front/back views of the source mesh (1^{st} column) and an edited frontal image used as the conditional view. The 2^{nd} column shows the masked area rendered from the front (inset) and a 2D edit. The last column shows our generated mesh from the front/back views.	49
4.4	Multi-view Diffusion vs MaskedLRM: Multi-view diffusion models must infer occluded geometry from the input which leads to artifacts in the multi-view images (incorrect/distorted views). MaskedLRM which does not use multi-view diffusion, receives both the edited image and multi-view information, allowing it to bypass this problem and reconstruct the correct geometry.	51
4.5	Mesh Editing Comparisons: Given a mesh (top left) and various image edits as guidance we demonstrate that our approach is the only one that generates multi-view consistent shape edits that follow the guidance. Colors omitted to clearly visualize the edited geometry.	54
4.6	Mesh Editing Comparisons with Concurrent work: We compare our approach to concurrent work enabling localized 3D editing. While localized approaches better preserve the original structure of the shape, other methods are not able to produce edits as realistic as ours.	56
4.7	Genus changes: Our method unlocks genus-changing edits like adding a handle or a hole to the original vase. We show the output of our model from 2 opposing views in the 3^{rd} column.	57
4.8	Impact of Geometric Supervision: Geometric losses are critical to produce high-quality surfaces. No geometric loss (top) causes severe hole and bump artifacts in the mesh. Depth loss (middle) is not as effective as normal loss (bottom) which allows our model to generate accurate and smooth reconstructions. . . .	59
4.9	Impact of Random Masking: We test our choice of masking strategy by comparing it to masking 25% of patches uniformly at random. The left column shows the conditional image, the middle section shows our results, and the right section shows the results using uniformly random masking. While the model is still capable of generating correct geometry, there is a train-test gap in the masked patches since we define a contiguous 3D region to mask during inference. Thus, the model produces artifacts such as lack of sharp features (in the bird horns and chair slats) in rows 1 and 2, and overall incorrect shape in row 3 (square turtle shell).	60
5.1	Decomposition results on instance-labeled Blender scenes. For each scene we show the reconstructed RGB, segmentation map, Gaussian extents, and per-primitive isolated renders. Each Gaussian specializes to a single object instance.	73

5.2	Scene editing on the desk scene. Each row demonstrates a different transformation across six viewpoints. (1) Shows a uniform $0.5\times$ scale reduction in each axis. (2) and (3) show anisotropic scaling. (4) demonstrates a translation of the lamp across the desk. All edits are done at inference time with no additional optimization required.	74
5.3	Cross-scene composition. A desk, a lamp, a kitchen table, and a drum kit stool, trained from 3 different scenes, are assembled into a coherent composition by placing each primitive at a desired position in a shared world space. Again, no additional optimization is required.	75
5.4	Per-primitive isolated renders under standard volumetric rendering. Despite explicit label supervision, the model fails to learn a clean decomposition. In the drum scene, Gaussians 3 and 6 have absorbed extraneous stand geometry. In the kitchen scene, three primitives collapse entirely. Overall rendering quality is also degraded.	76
5.5	PartNeRF trained on the desk scene with 16 components. Despite plausible reconstruction quality (top), the learned decomposition produces 16 small, overlapping blobs with no semantic coherence. This confirms that geometric regularization alone provides no incentive for parts to align with semantically meaningful regions.	77
5.6	Attempting to edit a PartNeRF decomposition by translating the component containing the lamp. Because parts do not correspond to individual objects, the transformation moves unrelated geometry contained within the same ellipsoidal support, producing implausible results across views.	77
5.7	PartNeRF with label supervision and 3 components on the desk scene. While label supervision partially recovers semantic structure, no component captures the lamp, instead producing two near-duplicate chair primitives. Hard ray-to-part assignment also handles occlusion poorly, producing view-dependent artifacts wherever parts are occluded.	78
5.8	Independent renders of selected objects from our method (left) vs. Gaussian Grouping across three scenes (right) with anisotropic scaling applied. Both methods correctly lift the label information to 3D, but the Gaussian Splatting representation is not robust to transformations like scaling. We observe tear-like artifacts in row 1 throughout the cabinet and row 3 near the edge of the cymbal. In row 2, we see that the texture on the chair is heavily warped by the scaling.	79
5.9	Unsupervised decomposition of the NeRF-Synthetic chair scene using the proposed label-free hierarchical decomposition framework. Each row shows a different camera view. The columns show the per-pixel label assignment derived from spectral clustering, the full composite render of all child primitives, and each child rendered in isolation. The discovered parts include the chair back, armrests, legs, and seat.	80
5.10	Unsupervised decomposition of the NeRF-Synthetic mic scene. The recovered components separate the microphone head, base, and cable.	80

5.11	Unsupervised decomposition of the NeRF-Synthetic drums scene. The recovered components include several different drums, the stool, and cymbals. For the selected 6 cluster setting, we observe that several primitives still contain distinct sub-parts, making them candidates for an additional stage of decomposition.	81
5.12	Recursive subdivision of the pink primitive from Figure 5.11, which contained the bass drum, hi-hat, and several metal stands. The first component containing the bass drum is a candidate for further subdivision to separate the drum from the extraneous stand.	82
5.13	Recursive subdivision of the green primitive from Figure 5.11, which a crash cymbal. The children primitives consist of two parts, the metal legs, and the cymbal connected to the stand.	83
5.14	We train our framework on the MipNeRF-360 garden scene. The columns show the SAM-generated masks, the full composite render, and the table and vase primitives rendered in isolation. While the table is recovered cleanly, noise in the SAM masks leads to imperfect separation of the vase.	84
5.15	Editing in a real-world scene. We translate the recovered vase primitive across the table during inference time.	85
A.1	TetGAN generates meshes with solid interiors.	110
A.2	Additional chair interpolations.	111
A.3	Additional vase interpolations.	111
A.4	Unconditionally sampled benches.	112
A.5	Unconditionally sampled cars. We observe that although the model is able to learn the coarse shape of a car, the discretization is not fine enough to capture details.	113
B.1	Additional Qualitative Examples: Additional qualitative examples editing a person’s head. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.	115
B.2	Additional Qualitative Examples: Additional qualitative examples editing a chair and a full human. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.	116

LIST OF TABLES

2.1	HR/LR refer to grid resolutions of (61^3) and (41^3) respectively. wLS denotes deformation-field weighted Laplacian smoothing. * indicates mesh simplification of outputs. Metrics are computed over extracted surfaces since baseline methods do not generate volumetric meshes.	16
3.1	Quantitative evaluation. We use our text prompts and deformed meshes in a retrieval task to compute R-Precision. We observe that regularizing for view-point consistency improves TextDeformer R-Precision. TextDeformer and CLIP-Mesh achieve quantitatively comparable R-Precision, TextDeformer (Ours) produces higher-quality geometry both qualitatively (see Figure 3.11) and quantitatively (significantly fewer self-intersections). All methods significantly outperform Text2Mesh in R-Precision.	39
4.1	Quantitative Evaluation: We evaluate our model using test-set shapes and compare it to the state-of-the-art LRM and InstantMesh on the ABO and GSO shape datasets, reconstructing the meshes from 6 and 8 posed images. Despite direct reconstruction of new shapes not being our main goal, and our masking introducing extra difficulty in the task, we still achieve better than SoTA metrics on ABO, and comparable to SoTA metrics on the GSO dataset.	53
4.2	Runtime Comparison: Our method is significantly faster than optimization methods as it is feed-forward and also faster than LRM-based approaches that must run multi-view diffusion.	61
A.1	FID↓ scores of TetGAN and baselines on various categories. TetGAN is best in 2 categories and competitive in the rest despite generating volumes (at a smaller object resolution) while baselines only generate surfaces.	109
A.2	Variety metric ↑ for TetGAN and baselines on various categories.	109
A.3	Reconstruction accuracy ↓ for TetGAN and baselines. TetGAN and OccNet both have effective generative capabilities while still enabling reconstruction/encoding. TetGAN offers a significant improvement to OccNet in all categories.	109
B.1	Mesh Editing Comparisons: We provide CLIP cosine similarity metrics of our proposed MaskedLRM approach against two recent 3d editing techniques. .	114

ACKNOWLEDGMENTS

The completion of this PhD marks the end of a long journey of both personal and intellectual growth, one filled with moments of both achievement and difficulty. None of this would have been possible without the people who supported me throughout the years, celebrated my successes, and encouraged me through the more challenging moments.

First and foremost, I want to thank my advisor, Michael Maire. I am deeply grateful for his support and mentorship, giving me the freedom to pursue the research directions that culminated in the completion of this dissertation. I admire his perspective on research and his emphasis on pursuing meaningful, fundamental questions. His mentorship has had a lasting influence on the way I think about computer vision, graphics, and scientific research as a whole.

I would also like to express my gratitude to my committee members Rana Hanocka and Anand Bhattad for their guidance and feedback throughout my PhD. I am especially grateful to Rana, whose early mentorship and collaboration played an important role in introducing me to computer graphics and shaping my research direction.

I am deeply thankful to the exceptional internship mentors and collaborators who supported me throughout the summers of my PhD: Vova Kim, Noam Aigerman, Thibault Groueix, Zekun Hao, Dilin Wang, Nikolaos Sarafianos, and Mingyang Li. Their ideas and technical insight greatly influenced my work and my growth as a researcher over the years.

In addition, I have been fortunate to work alongside a talented group of friends and labmates at UChicago: Xiao Zhang, Sudarshan Babu, David Yunis, Xin Yuan, Ruoxi Jiang, Zain Sarwar, Tewodros Ayalew, Yixuan Luo, and Cheng Yang. Graduate school would have been considerably harder without the conversations, collaboration, humor, and camaraderie we shared over the years.

Beyond academia, I am sincerely grateful to the friends who remained a part of my life throughout my PhD. I want to thank both my close personal friends from college, and

my friends from the communities I became a part of later. Even when distance and busy schedules kept us apart, their presence in my life provided a sense of balance and happiness outside of research.

I owe more than I can express to my family, whose patience, support, and encouragement never wavered, even when the direction of my life was uncertain. Their steady belief in me gave me the confidence to see it through.

Finally, thank you to my best friend Helen. Since we met in college, you have been beside me through nearly every stage of this journey, and this chapter of my life would not have been the same without you.

ABSTRACT

The generation of 3D assets from scratch has become a major area of interest in computer vision and graphics research, driven by applications in several visual mediums, including games, animation, and virtual reality. Despite rapid progress in generative modeling, modifying existing assets in a controllable and interpretable manner remains difficult. Precise 3D manipulation is fundamentally a representation problem: the types of edits a neural system can perform depend on how the scene is processed internally. This dissertation studies how different forms of structure influence the quality, controllability, interpretability, and expressiveness of supported editing operations.

We begin in the second chapter by exploring early representations for generative 3D models, focusing on using tetrahedra as an alternative to dense voxel-based shape synthesis. By training standard variational autoencoders over datasets of 3D assets represented using this discretization, our method learns coherent latent shape embeddings that support shape editing via latent interpolation and arithmetic. Both surface and volumetric geometry can be extracted directly from the learned tetrahedral grid, producing explicit mesh assets suitable for downstream tasks.

However, this class of method is fundamentally limited. Learning meaningful latent embeddings requires training over large collections of 3D assets, while large-scale 3D datasets remain substantially less abundant than image data. Furthermore, edits performed through latent manipulation provide only coarse global control over generated geometry, making precise semantic modifications difficult to specify. These limitations motivated the development of editing approaches that leverage rich semantic information encoded in large multimodal models pretrained on readily available image-text data.

In the third chapter, we study text-based methods for optimizing 3D geometry using natural language as an interface for specifying edits. A common approach in this vein is to directly optimize mesh vertices and differentiably render the resulting shape to obtain

image-text similarity gradients, but this often leads to noisy and unstable deformations. To address this limitation, we adopt an alternative parameterization of the deformation field on vertices that produces smoother and more semantically coherent changes.

Despite our method producing smoother and more coherent deformations, optimization-based editing methods are difficult to control in practice. Direct deformation of explicit geometry also restricts the range of allowable modifications. In the fourth chapter, we address these limitations using a feed-forward masked editing architecture that conditions generation on both the initial input geometry and localized edit specifications. Our method edits in a single forward pass rather than through iterative optimization, producing faster results. The edits are also more predictable as the model directly incorporates the specified edit into the generated 3D geometry, while preserving unedited regions of the original shape.

In the fifth chapter, we shift from editing individual shapes and study structured scene representations for editable neural rendering. This work focuses on recovering semantically meaningful objects and parts within scenes by lifting image-space information into 3D representations. We investigate methods for converting multi-view semantic information into coherent 3D scene decompositions, using either supervised annotations or unsupervised feature affinity methods. The resulting representations enable individual scene components to be isolated and manipulated through operations such as translation, scaling, and scene recomposition.

CHAPTER 1

INTRODUCTION

Methods for 3D content generation have advanced rapidly in recent years, driven by applications in games, animation, simulation, and virtual and augmented reality. Recent progress combines generative learning architectures originally developed for 2D vision with new neural rendering and scene representation techniques designed specifically for 3D and multi-view data.

Early neural approaches to 3D generation often operated directly on voxelized geometry, extending generative frameworks like Variational Autoencoders (VAEs) [103, 132, 43, 104] and Generative Adversarial Networks (GANs) [195, 21] from image synthesis to volumetric shape representations. Although these methods successfully learn 3D generation from latent variables, dense voxel discretization imposes significant memory limitations. This motivates the exploration of alternative, more efficient geometric primitives, including deformable tetrahedra [37, 144]. Many of these early generative methods also exhibit limited editing capabilities through operations performed directly inside learned embedding spaces. By interpolating or modifying latent codes learned over collections of assets, the models can generate combinations and blends of different objects. However, these edits are difficult to control precisely, as latent manipulations alter geometry in broad entangled ways without providing explicit control over which geometric attributes are transferred between shapes.

Furthermore, latent editing techniques rely heavily on learning coherent latent spaces from large collections of 3D assets. Early methods were trained on relatively small datasets such as ShapeNet [16], often restricted to only a few categories, limiting their generalization to arbitrary shapes. Although newer datasets such as Objaverse [30, 31] have significantly expanded the scale of available training assets, the volume of available data is far smaller than that of image and text datasets [139]. This disparity increasingly motivated approaches that leverage semantic supervision derived from large, pretrained multimodal models, such

as CLIP [125] and DINO [14].

One family of approaches uses vision-language foundation models to guide the optimization of a 3D object using natural language descriptions. Because rendering a mesh into images can be done in a differentiable manner, image-text similarity objectives can be backpropagated through the rendered views into underlying geometric parameters. These techniques commonly optimize explicit geometry such as vertex positions [106, 76, 41] and support localized modifications [7]. By leveraging semantic information learned from large-scale image-text datasets, these methods enable open-vocabulary editing without requiring captioned 3D assets, thereby substantially expanding the range of possible edits beyond simple latent interpolation.

CLIP-based optimization methods often remain difficult to control precisely, as text-image similarity objectives can produce noisy and unreliable gradients. In practice, these approaches provide limited control over the final result beyond careful prompt engineering and hyperparameter tuning. Furthermore, because these methods directly optimize existing meshes, they struggle to perform edits that alter the topology of a shape, such as introducing holes or disconnected surfaces.

More recent complementary approaches leverage Large Reconstruction Models (LRMs) [57, 171], which reconstruct 3D assets from multiview images using feed-forward transformer architectures. This formulation not only removes the need for expensive iterative optimization, but also provides more predictable outcomes by reconstructing shapes from a learned prior over realistic 3D geometry. Multiple editing formulations have emerged within this paradigm. Some approaches first learn a multiview inpainting diffusion model before reconstructing a new shape using a pretrained LRM [8]. Other methods instead directly incorporate editing constraints into the reconstruction process itself through masked training [42]. These approaches enable faster and more expressive editing pipelines by regenerating geometry instead of directly deforming an existing mesh. This formulation supports a wider range of

structural modifications, while still preserving consistency with the original shape.

Beyond reconstructing individual objects, recent neural rendering techniques have also enabled high-fidelity reconstruction of full scenes from multiview images. However, while techniques like Neural Radiance Fields [108] and Gaussian Splatting [75] achieve impressive rendering quality, they typically lack explicit object-level semantic structure, making localized manipulation and editing difficult. Recent work has explored lifting semantic information encoded in large pretrained models into neural 3D scene representations [124, 197, 182, 15, 83, 179, 11, 193]. By distilling multiview image features, segmentations, and language-aware embeddings from image space into neural scene representations, these approaches recover semantically meaningful objects directly within the reconstructed scenes.

This dissertation presents several contributions to neural 3D editing across the evolution of geometric and neural rendering representations. Beginning with structured latent generative models, we explore how learned embedding spaces can support semantic manipulation of shapes through latent operations. We then investigate alternative parameterizations for mesh deformation that produce more stable and coherent edits under noisy supervision from large multimodal models. Motivated by the limitations of optimization-based approaches, we introduce a feed-forward masked reconstruction architecture that improves editing speed, predictability, and structural expressiveness by regenerating geometry from learned shape priors. Finally, we investigate a Gaussian-based scene representation that treats objects as first-class primitives, enabling semantically meaningful objects and parts to be isolated and manipulated through scene-level editing operations.

CHAPTER 2

CONVOLUTIONAL AUTOENCODERS FOR TETRAHEDRAL MESHES

This chapter presents the work published in [40].

2.1 Introduction

We approach learning to synthesize and edit 3D shapes through a neural framework designed for *tetrahedral* meshes. Tetrahedral meshes are used widely across a myriad of applications, such as articulated shape deformations [167, 63], solid modeling [29], scientific computing [162], and physics-based simulations [60, 68].

We propose a framework for generating tetrahedral meshes, called TetGAN. Our system learns to produce *novel* and diverse tetrahedral meshes from random noise. Simultaneously, TetGAN provides *controlled* manipulation of existing tetrahedral meshes through latent-encoding capabilities. Our framework trains an encoder to map tetrahedral meshes into a semantically meaningful latent space. This enables performing latent space interpolation or arithmetic operations for effective 3D shape manipulations.

TetGAN leverages an irregular grid of tightly packed tetrahedra within a bounding cuboid to create a spatial structure for learning local correlations in 3D. Akin to a pixel in an image, each tetrahedron is the basic building block for learning over the network input. To represent a 3D shape, we encode each tetrahedron with an *occupancy* value and 3-dimensional *displacement* vector to the closest point on the shape surface. This enables extracting a tetrahedral mesh by removing unoccupied tetrahedra, and displacing vertices on triangles on the surface *boundary*. This representation combines the advantages of implicit representations to handle varying topology, and explicit meshes to obtain a high level of detail efficiently.

The irregular tetrahedral grid has recently started to gain traction as an alternative neural

representation for 3D geometry [110], pioneered by DefTet and DMTet [37, 144]. Using the irregular tetrahedral grid, topological holes can be inserted anywhere easily by removing unoccupied tetrahedra in the grid. Tetrahedral meshes are able to *deform* each triangle by individually displacing vertices since deformed triangles remain planar *and* connected to adjacent faces. Deforming the boundary vertices to coincide with the learned underlying shape will result in a smooth surface with varied topology. This seemingly simple property is non-trivial to obtain with 3D cubic voxel grids (i.e., a hexahedral mesh). Displacing 4-points in a quadrilateral does not necessarily preserve planarity between adjacent quads.

We learn to represent local spatial relationships in 3D shapes (such as corners, curves, and planes), through carefully designed neural network layers. We use a *tetrahedral convolution* layer which shares weights across the entire spatial field, in order to encode local shape cues. Beyond local regions, we progressively incorporate additional 3D *context* through *tetrahedral pooling*, reducing the spatial resolution of the input. The original spatial resolution is restored through *upsampling* layers. The TetGAN layers capture local geometric structure across multiple spatial scales while supporting latent encoding and unconditional shape generation through combined VAE and adversarial training objectives.

2.2 Related Work

Initially, convolutional neural networks built for mesh surfaces were focused on shape analysis tasks such as classification and segmentation [48, 107, 85, 58]. Later, techniques proposed synthesizing vertex locations over a surface mesh with fixed topology [165, 96, 53, 49, 39]. Though meshes can efficiently represent complex geometries, generating meshes with varied genus is challenging [120]. Only a handful of works directly generate or modify surface mesh connectivity [111, 126, 142]. Other techniques combine fixed genus mesh parts to form a more complex topology, such as a composition of primitives [160, 22, 118], parametric patches [45], or assembly of parts [184]. Cubic voxel-based techniques can generate shapes with varied

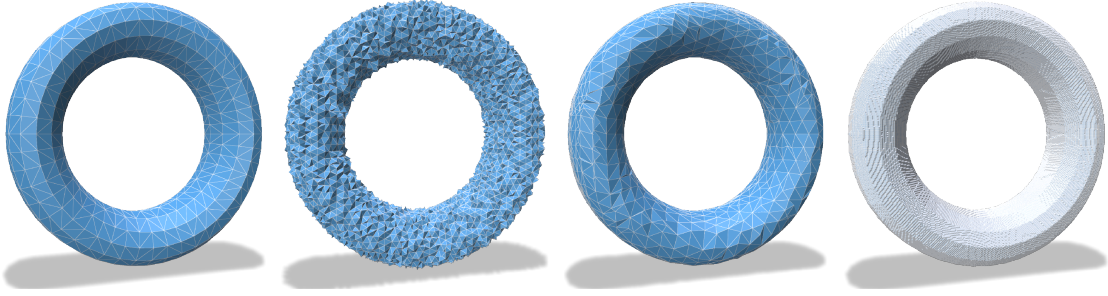


Figure 2.1: Comparison of volumetric shape representations for a torus. Left to right: (a) occupancy extracted from an undeformed 161^3 tetrahedral grid, (b) a deformable 41^3 tetrahedral grid, and (c) a 301^3 voxel representation extracted using marching cubes. Deformable tetrahedral representations achieve comparable geometric fidelity at substantially lower resolution.

genus [23], however planar-face hexahedra cannot be displaced freely.

2.2.1 Implicit Representations

Shapes can also be synthesized using implicit representations, which predict a signed distance or an occupancy value per input point [117, 21, 104, 172]. Implicit representations can generate shapes with any topological genus since the explicit mesh surface is extracted in post-process. ShapeGAN [82] employs DeepSDF [117] as a generator to model a signed distance function, and a discriminator which evaluates the signed distance field produced by a batch of samples alongside latent information. The generated SDF values from the generator are arranged into a voxel volume or input to PointNet.

2.2.2 Tetrahedral Meshes

Tetrahedral meshes are widely used in Finite Element Method (FEM) simulations [60, 115]. Computing realistic and high-quality deformations of articulated shapes commonly relies on a tetrahedral mesh [167, 63]. We use the tool QuarTet [13] to generate the irregular grid of tetrahedra which we use to define the TetGAN convolution, pooling, and upsampling layers. Our use of a tetrahedral grid draws inspiration from DefTet [37], which learned

to reconstruct tetrahedral meshes from point clouds or 2D images. Recently, Deformable Marching Tets (DMTet) [144], learns to reconstruct high-resolution surface meshes from a coarse voxel input.

One of the key differences between our tetrahedral grid representation and that of DefTet [37] and DMTet [144, 110] is how we predict and utilize displacements. In this work, our tetrahedral grid contains a vector-valued displacement to the underlying surface at each tetrahedron’s centroid, which we refer to as a *deformation field*. Deformation fields are a rich representation of the explicit underlying geometry, which provides shape cues over the entire grid. The objective of encoding and predicting a deformation field even aids the network in learning to predict more accurate occupancy fields, suggesting that deformation fields are an effective neural shape representation. After training is complete, we leverage the network-predicted deformation field to *filter* incorrect occupancy predictions and infer weights for *smoothing* the predicted surface geometry.

While prior works apply the tetrahedral grid representation to the tasks of tetrahedral *reconstruction* (DefTet) and mesh super-resolution (DMTet), our method provides *generative* capabilities, a fundamentally different objective. TetGAN enables sampling novel shapes from noise, latent space interpolations, and shape editing, none of which are provided by DefTet/DMTet (nor by inverse rendering [110]). TetGAN achieves this using a novel CNN for tetrahedral meshes, with convolution/pooling blocks that are distinct from DefTet/DMTet components and tailored for the task of generation (inspired by 2D CNNs [102, 73, 74]). Our pooling and upsampling leverages a tetrahedral subdivision structure to learn features over various receptive fields. In a similar spirit, SubDivNet [58] learns using a loop subdivision structure for performing analysis tasks (e.g., classification) over *surface* meshes.

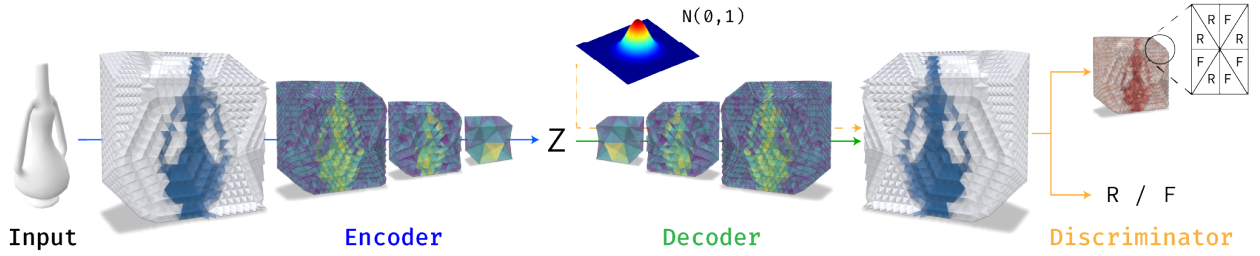


Figure 2.2: TetGAN overview. We represent the input mesh using a deformation field over a tetrahedral grid. The encoder produces a latent vector $\mathbf{Z}$, which is used as input to a decoder which performs a series of convolutions and upsampling operations. The two discriminators learn to classify whether the input shape is fake (generated) or real at the patch/global level.

2.2.3 Contrast to Voxels

Works that synthesize shapes using implicits may opt to predict over a voxel grid [104, 82]. However, the benefits afforded by our deformation field are not trivially amenable to 3D cubic (hexahedral) voxels. Freely displacing the four vertices per-face will break the planar-face property. Alternatively, fitting higher-order curved patches to 4-point sets does not guarantee connectivity between adjacent cells. Moreover, cubic cells do not trivially divide into tetrahedra, and volumetric meshing is an open active area of research [59, 60, 146]. Due to the inherent challenges in converting to tetrahedral meshes (e.g., inevitable trade-offs between geometric accuracy and robustness), several works propose bypassing tetrahedral meshing altogether for a particular application [138, 159]. Compared to cubic voxels, tetrahedral meshes are able to more efficiently adapt to complex geometries, such as high curvature regions as shown in Fig. 2.1. Representing a torus using different volumetric representations (with error under a threshold), requires 301^3 undeformed cubic voxels, compared to 161^3 undeformed tetrahedra or 41^3 deformed tetrahedra.

2.3 Method

We design a generative model over a packed tetrahedral grid to define a coarse inside / outside representation of a shape (i.e., an occupancy). This grid directly represents a tetrahedral

mesh with no additional processing and allows an isosurface to be trivially extracted. We also define a displacement per-vertex in the grid to the underlying shape surface, which enables smoothing the coarse representation to obtain a higher fidelity result.

To learn over this representation, we propose TetGAN, a deep architecture which leverages and fuses ideas from both generative adversarial neural networks (GAN) [44] and Variational Autoencoders (VAE) [80]. First, an encoder performs a series of convolution and pooling layers (on a tetrahedral grid) to aggregate the spatial information over the 3D shape into a latent vector. Next, a decoder processes the latent vector through a series of convolution and upsampling layers to reconstruct the 3D shape. This stage is supervised by ground truth features computed over the training data. Finally, to improve the diversity and smoothness of the learned latent space, we make use of an additional adversarial training, where the VAE decoder acts as the generator of a GAN. This enables TetGAN to directly learn more point samples inside the latent space. To synthesize a new tetrahedral mesh, we sample from the learned latent space and use the decoder to generate the occupancy and displacement fields. The overall model is illustrated in Figure 2.2.

2.3.1 Tetrahedral grid framework

Tetrahedral grid. Our framework represents shapes using an occupancy and a displacement encoded within an irregular grid of tetrahedra. First, we compute an initial tetrahedralization of a cube using QuarTet [13] to obtain a tetrahedral grid $\mathbf{T} = \{\mathbf{T}_1, \dots, \mathbf{T}_K\}$ consisting of K tetrahedra. Each tetrahedron $\mathbf{T}_k$ is represented by its four vertices: $\{v_{k_a}, v_{k_b}, v_{k_c}, v_{k_d}\}$, where each $v_{k_i} \in \mathbb{R}^3$. To represent an input surface mesh $\mathbf{M}$, we first normalize it to fit inside the unit cube. We calculate the occupancy value (inside or outside) at each tetrahedron’s centroid. We also calculate the displacement from the tetrahedron centroid to the closest point on the mesh surface, i.e., the displacement field.

Tetrahedral convolution. Despite being an irregular grid $\mathbf{T}$, a unique structure nat-

urally admits learning a convolution. Every tetrahedron in the grid is adjacent to exactly four tetrahedra, which defines the spatial support of the convolutional filter. Specifically, for each tetrahedron $\mathbf{T}_k \in \mathbf{T}$, we perform a weighted sum of $\mathbf{T}_k$ and its four adjacent neighbors $\mathbf{T}_j \in \mathbf{T}$ (i.e., $\mathbf{T}_k$ and $\mathbf{T}_j$ share a triangle face). A convolution on the tetrahedral grid is defined as $\mathbf{W}_0 \cdot \phi(\mathbf{T}_k) + \sum_{j \in \mathcal{N}_k} \mathbf{W}_j \cdot \phi(\mathbf{T}_j)$, where $\mathcal{N}_k$ denotes the set of adjacent neighbors of $\mathbf{T}_k$, $\mathbf{W}$ denotes the convolution weights, and $\phi(\mathbf{T}_k)$ denotes the features for tetrahedron $\mathbf{T}_k$. The ordering of the four neighbors in the convolution is prescribed by the tetrahedral grid.

Subdivision structure for pooling and upsampling. To perform pooling and upsampling on the tetrahedral grid, we need to first define how to down/upsample, for which we introduce a subdivision structure. For any $\mathbf{T}_j \in \mathbf{T}$, we can subdivide it into eight tetrahedra, represented as a *supercell* $\mathcal{S}_j = \{\mathbf{T}_k\}$. A supercell is created by adding vertices at the midpoints of each edge. Starting with a low resolution grid $\mathbf{T}^{(1)}$, we produce successive grids $\mathbf{T}^{(2)}, \mathbf{T}^{(3)}, \dots, \mathbf{T}^{(N)}$ by subdividing each tetrahedron. Naturally, pooling of features ϕ from $\mathbf{T}^{(n)}$ to $\mathbf{T}^{(n-1)}$ is defined as

$$\phi^{(n-1)}(\mathbf{T}_j^{(n-1)}) = \mathbf{F} \left(\left\{ \phi^{(n)}(\mathbf{T}_k^{(n)}) \mid \forall \mathbf{T}_k^{(n)} \in \mathcal{S}_j^{(n-1)} \right\} \right), \quad (2.1)$$

where $\mathbf{F}$ is a symmetric aggregation function (either a max or average).

To upsample features $\phi^{(n)}$ from $\mathbf{T}^{(n-1)}$ to $\mathbf{T}^{(n)}$, one may define techniques analogous to those in 2D computer vision. We propose an alternative method analogous to interpolation upsampling. The upsampled features of $\mathbf{T}_k^{(n)} \in \mathcal{S}_j$ are an average of the features of the neighbors of its parent $\mathbf{T}_j^{(n-1)}$, weighted by the inverse distance between centroids:

$$\phi^{(n)}(\mathbf{T}_k^{(n)}) = \sum_{\ell \in \mathcal{N}(\mathbf{T}_j^{(n-1)})} \frac{\alpha_1}{\|c_\ell^{(n-1)} - c_k^{(n)}\|} \phi(T_\ell^{(n-1)}), \quad (2.2)$$

where α_1 is a scalar that normalizes the coefficients to sum to one. For an illustration of the downsampling and upsampling processes, see Figure 2.3

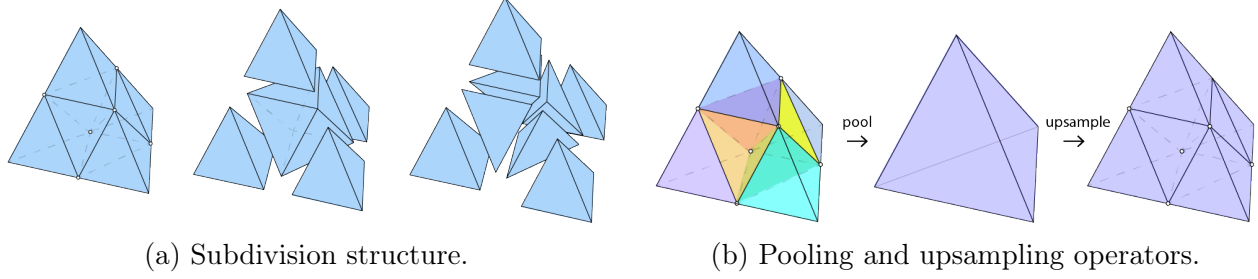


Figure 2.3: TetGAN subdivision, pooling, and upsampling operations.

2.3.2 TetGAN

Problem formulation. We represent the 3D shape by inserting it inside the irregular grid $\mathbf{T}^{(N)}$ and computing an occupancy and a displacement per tetrahedron. The occupancy $\mathbf{O}$ is given by: $\mathbf{O}(\mathbf{T}_k^{(N)}) = \mathbf{1}[w_{\mathbf{M}}(c_k^{(N)}) > 0]$, where $w(\cdot)$ gives the winding number of any point with respect to the surface $\mathbf{M}$ and $\mathbf{1}$ denotes the indicator function. Here, $c_k^{(N)}$ denotes the centroid of $\mathbf{T}_k^{(N)}$ given by the average coordinate of its four vertices. We define a deformation on a *tetrahedron* $\mathbf{T}_k^{(N)}$ as the difference between $c_k^{(N)}$ and its closest point p on $\mathbf{M}$. In order to obtain a deformation over the *vertices*, we compute the average displacement over each tetrahedra incident to each vertex $v^{(N)}$, weighted by the inverse distance:

$$\mathbf{D}(v^{(N)}) = \sum_{k \in \mathcal{I}(v^{(N)})} \frac{\alpha_2}{\|v^{(N)} - c_k^{(N)}\|} \arg \min_{p \in \mathbf{M}} \|c_k^{(N)} - p\| - c_k^{(N)}, \quad (2.3)$$

where $\mathcal{I}(v)$ denotes a set of incident tetrahedra indices of vertex v and α_2 again is a normalizing scalar. We directly penalize the deformation per-vertex against the ground-truth deformation field. To represent a *tetrahedral* mesh $\mathbf{M}$ we have $\mathbf{X} = [\mathbf{O}, \mathbf{D}]$. $\mathbf{O}$ trivially defines a *triangular mesh* consisting of all faces shared by tetrahedra with different occupancies. Our goal is to learn a generative model $p(\mathbf{X})$ from which $\mathbf{X}$ can be sampled.

Generative model. Our generative model is a hybrid between a VAE and a GAN. Starting with a VAE, we have an encoder f_{enc} , a decoder f_{dec} , and a prior function $p(\mathbf{Z})$. Both encoder and decoder are deep neural networks consisting of the proposed tetrahedral convolution layers; exact architecture details are deferred to the supplemental. To generate

a new mesh during inference, we can easily sample a latent vector $\mathbf{Z} \sim p(\mathbf{Z})$ and pass it through the decoder $f_{\text{dec}}(\mathbf{Z})$. We train the VAE using binary cross-entropy on the occupancies and mean squared loss on the deformations. To enable a more densely sampled and smoothly varying latent space, we additionally include global and local (WGAN-GP [46]) adversarial losses. This introduces two discriminator networks f_{dis}^l and f_{dis}^g , which operate over local and global attributes. The local discriminator is a fully convolutional *patch discriminator* [88] which outputs a ‘real’ or ‘fake’ score per tetrahedron. On the other hand, the global discriminator makes use of pooling layers to aggregate information across the entire grid into a ‘real’ or ‘fake’ score.

Deformation-Field Weighted Laplacian Smoothing. We apply Laplacian smoothing to the output mesh using weights computed with the predicted deformation field. Specifically, we first update the vertices by applying the deformation field over them, i.e., $\hat{v}_i^{(N)} = v_i^{(N)} + D(v_i^{(N)})$. Next, these displaced vertices are smoothed by performing a weighted average with their neighbors:

$$\tilde{v}_i^{(N)} = \beta \cdot \hat{v}_i^{(N)} + (1 - \beta) \cdot \sum_{\hat{v}_j^{(N)} \in \mathcal{N}(\hat{v}_i^{(N)})} \left| \cos \left(\mathbf{D}(v_i^{(N)}), \mathbf{D}(v_j^{(N)}) \right) \right| \cdot \hat{v}_j^{(N)} \quad (2.4)$$

where $\cos$ denotes the cosine similarity, $\mathcal{N}(v_i^{(N)})$ denotes the neighboring vertices of $\hat{v}_i^{(N)}$ and $\beta \in [0, 1]$ is a hyperparameter that controls how much influence the original position of $\hat{v}_i^{(N)}$ exerts on its smoothed position $\tilde{v}_i^{(N)}$.

Deformation field filtering. We observe that for tetrahedra that are truly on the surface of the underlying shape, the corresponding deformation should be relatively small in magnitude. Thus, we can use the predicted deformation field to filter incorrect occupancies. In particular, we compute the average norm of surface tetrahedra deformations in the dataset:

$$\mu = \frac{1}{|D|} \sum_{\mathbf{M} \in D} \frac{1}{|\mathbf{M}|} \sum_{\substack{T_k \in \mathbf{T}, \mathbf{O}_{\mathbf{M}}(T_k)=1 \\ \exists T_j \in \mathcal{N}_k: \mathbf{O}_{\mathbf{M}}(T_j)=0}} \|\mathbf{D}(T_k)\| \quad (2.5)$$

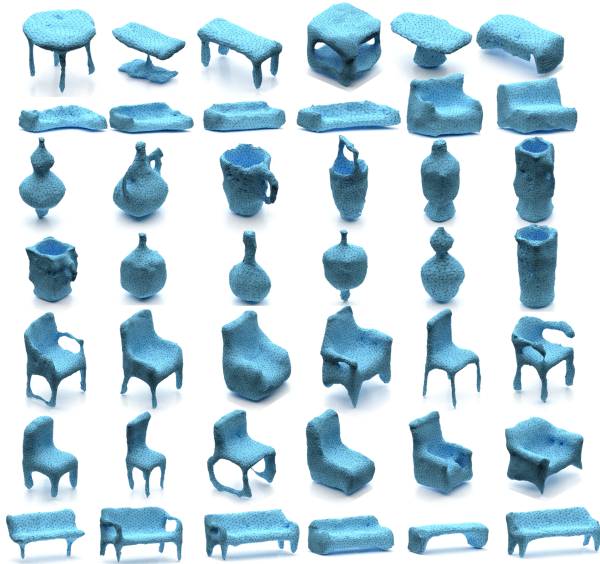


Figure 2.4: Novel random samples from TetGAN learned latent distribution.

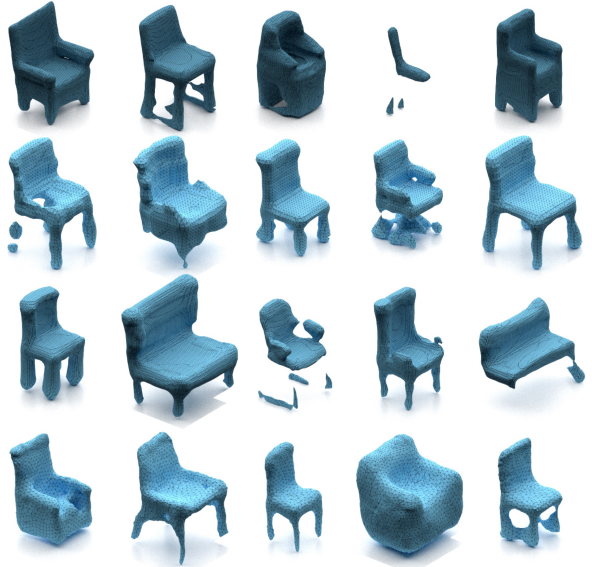


Figure 2.5: Novel random samples from (a) OccNet, (b) ShapeGAN (VAE), and (c) ShapeGAN, and (d) TetGAN respectively

where $|\mathbf{M}|$ gives the number of occupied tetrahedra on the surface of $\mathbf{M}$. During inference (but not during training), we simply discard surface tetrahedra with a deformation larger than $\gamma\mu$ where γ is a tunable parameter.

2.4 Experiments

In our experiments, we assess the ability of the encoder to effectively map input shapes to the learned latent space, and the decoder to accurately reconstruct them. Next, we evaluate the variety of the generated shape samples when sampling from the learned latent distribution. We compare TetGAN against implicit networks [104, 82] quantitatively using Frechet Inception Distance (FID). Third, we qualitatively explore the latent space through latent space arithmetic and interpolation. Finally, we conduct ablation studies on our own system to quantify the impact of the proposed components, e.g., without adversarial losses. Note that there are no existing works that take tetrahedral meshes as input directly. Therefore, when comparing to baselines, we use their respective choice of discretization.

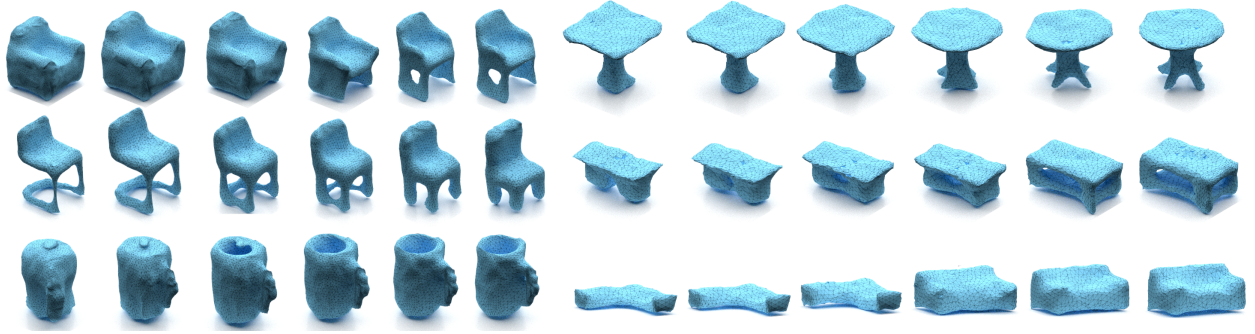


Figure 2.6: TetGAN’s latent space interpolation for different categories.

Dataset. We evaluate TetGAN on five ShapeNet [16] categories (chair, sofa, benches, table, and car), and vases from COSEG [168].

To prepare the training and validation data, we use a randomized 5/1 train/validation split on the dataset. We process the ShapeNet models to be manifold and watertight using the method by [61], and normalized to a unit bounding cube. We compute occupancy and deformation features for each shape as specified in Section 2.3.2.

2.4.1 Qualitative results

Visual Sample Quality. To demonstrate the generative capabilities of TetGAN, in Figure 2.4 we show random samples over the categories of chair, sofa, table, benches, and vase. These samples are generated unconditionally (i.e., purely from noise). TetGAN is trained separately for each category. We observe that TetGAN is capable of generating a diverse set of shapes with varied topology. We also show qualitative comparison with the OccNet [104] and ShapeGAN [82] in Figure 2.5.

Latent Space Exploration. We experiment with the latent space TetGAN learns over shapes. In Figure 2.6 we demonstrate TetGAN’s ability to produce semantically meaningful intermediates between two shapes. We further demonstrate TetGAN encodes high-level semantics of objects and parts through latent space arithmetic.

As illustrated in Figure 2.7, the difference between two meshes can be added to another

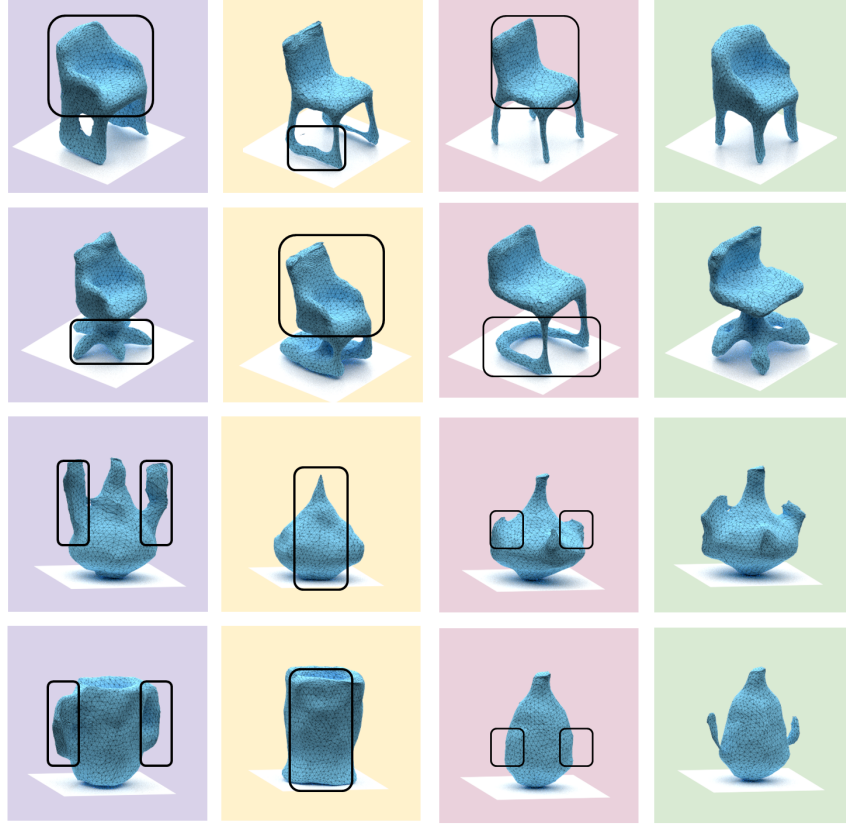


Figure 2.7: Latent-space arithmetic between meshes. The operation $A - B + C = D$ transfers semantic geometric attributes between shapes. Highlighted regions indicate the transferred component and its insertion into the resulting geometry.

mesh resulting in a completely new object. For example, in the first row, by subtracting the encodings of two similar chairs (in purple and yellow backgrounds) with one main difference (in the chair legs), we are able to transfer that difference onto a third chair (in pink) resulting in a brand-new chair (in green).

2.4.2 Quantitative results

Table 2.1 contains quantitative metrics evaluating reconstruction and generation of chairs. Note that all metrics evaluate *surface quality* since baseline generative methods, unlike TetGAN, do not produce volumetric meshes. See supplemental for additional quantitative evaluation.

Model	Recon. (Chamfer)↓	Avg. # Faces↓	FID↓	Variety↑
OccNet	0.0015	86901	1.06	0.0041
ShapeGAN (GAN)	-	25837	3.38	0.0036
ShapeGAN (VAE)	0.0011	6115	8.66	0.0017
OccNet*	0.019	5500	1.59	0.0039
ShapeGAN (GAN*)	-	5500	3.87	0.0038
ShapeGAN (VAE*)	0.0012	5500	9.25	0.0016
TetGAN – HR/wLS	0.0040	5263	2.18	0.0040
TetGAN – HR/No wLS	0.0035	5263	1.35	0.0028
TetGAN – LR/wLS	0.0032	3062	5.01	0.0038
TetGAN – LR/No wLS	0.0029	3062	2.09	0.0033

Table 2.1: HR/LR refer to grid resolutions of (61^3) and (41^3) respectively. wLS denotes deformation-field weighted Laplacian smoothing. * indicates mesh simplification of outputs. Metrics are computed over extracted surfaces since baseline methods do not generate volumetric meshes.

Reconstruction. We evaluate how accurately TetGAN reconstructs shapes on a test set using Chamfer Distance as a metric for accuracy. Table 2.1 shows the Chamfer distance between the reconstructed and the ground truth objects as well as the number of faces in the reconstructed object of TetGAN and baseline works. We observe that both OccNet and the VAE version of ShapeGAN beat TetGAN in reconstruction. However, for a fair comparison, we simplify OccNet generated meshes to a comparable number of faces as those produced by TetGAN. In this setting, TetGAN outperforms OccNet in reconstruction. We also note that, although TetGAN does not beat ShapeGAN (VAE) this metric, reconstruction quality does not necessarily correlate with visual sample quality.

Variety. We evaluate the diversity of TetGAN’s shape generation by measuring the most similar pairs in a set of generated shapes. We generate k pairs of shapes and take the average Chamfer distance from the n most similar pairs. In our experiments, we use $k = 250$ and $n = 25$. We run the experiment 5 times to account for variance. We observe that TetGAN produces more varied shapes than both versions of ShapeGAN and comes close to the variety of OccNet, beating the simplified version.

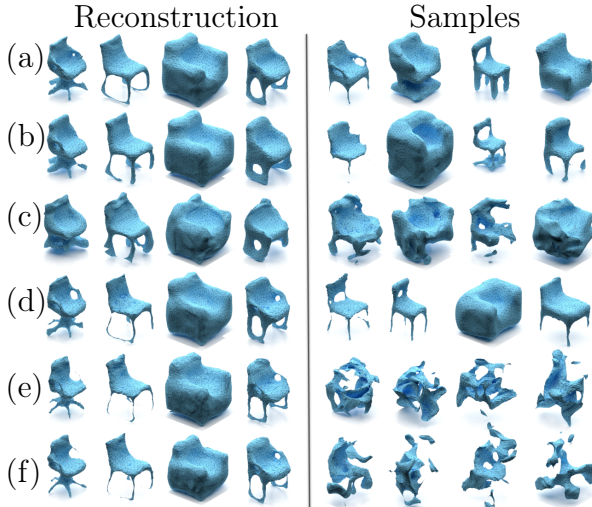


Figure 2.8: Ablations. Top to bottom: (a) full TetGAN, (b) surface GCN, (c) tetrahedral GCN, (d) no patch discriminator, (e) no global discriminator, (f) VAE only.

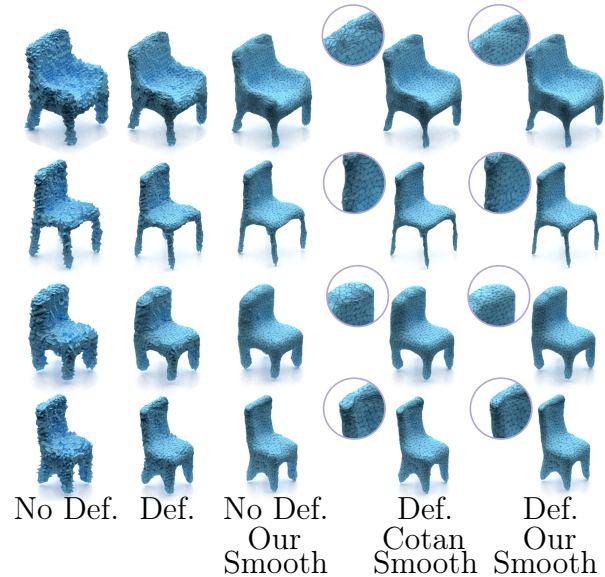


Figure 2.9: Our deformation-field weighted smooth (Def. Our Smooth) improves the quality of the final result.

Frechet Inception Distance. We use **FID** [55] between a set of generated shapes and the hold-out set to measure quality of generated shapes. To compute this distance, we use deep features from a PointNet++ [122] architecture trained for classification. We observe that at a high grid resolution, TetGAN without Laplacian smoothing comes close to the FID that OccNet achieves and beats both versions of ShapeGAN. TetGAN again beats the simplified version of OccNet. TetGAN at low grid resolution also produces shapes of reasonable quality and FID score. We observe that Laplacian smoothing, although produces desirable results, harms the FID score. This is likely due to the smoothing producing unnaturally thin pieces.

2.4.3 Ablation Study

Adversarial ablations. We evaluate the effectiveness of the components in TetGAN. TetGAN performs best using the full system, although the global discriminator appears far more important to the quality of generated results. Visually, as seen in Figure 2.8, removing the

local discriminator (row 4) causes floating artifacts. Removing the global discriminator (row 5) ruins the generative ability of TetGAN. We find that the global discriminator is crucial to learning the global structure of shapes. This is supported by a sharp increase in FID (23.35 without the global discriminator, only 12.27 without the local discriminator).

Graph convolutions. In Figure 2.8, row (b), we train using the GCN used in DefTet [37]. Their method trains a GCN to predict displacements over the vertices of a surface extracted from the ground-truth meshes. In this setting, we observe that the predicted deformations tend to be quite small and not well-suited for a noisy occupancy field. In Figure 2.8 row (c), we learn a tetrahedral GCN (learning the same weight for each the four neighbors), and observe significant quality degradation.

Deformation and smoothing. We experimented with different methods of smoothing the output of TetGAN. As seen in Figure 2.9, the learned displacements were often were not sufficient to create a smooth mesh. However, combining the vertex displacements with an additional weighted Laplacian smoothing (wLS) procedure yields desirable results, smoother and better preservation of detail compared to LS alone. Comparing the last two columns, we also observe that our wLS creates sharper features compared to cotangent LS.

2.5 Discussion

In this chapter, we presented TetGAN, a framework for generative modeling over tetrahedral mesh representations. By encoding occupancy and displacement fields over an irregular tetrahedral grid, our method enables synthesis of explicit geometry with variable topology while supporting semantically meaningful latent-space interpolation and manipulation. We introduced convolution, pooling, and upsampling operations defined directly on tetrahedral grids, allowing the network to capture spatial geometric structure across multiple scales. Combined with adversarial and variational training objectives, the resulting latent representation supports generation of diverse tetrahedral meshes and simple shape editing operations

through latent arithmetic and interpolation.

However, these latent editing operations remain fundamentally coarse and difficult to control precisely. As illustrated in Figure 2.7, latent arithmetic can transfer broad semantic features between shapes, but offers imprecise control over how those features manifest in the final geometry. The general concept may be preserved, but the user cannot specify its exact placement, scale, or form on the edited shape. These limitations motivate the exploration of other editing paradigms that incorporate more explicit semantic supervision.

CHAPTER 3

TEXT-DRIVEN GEOMETRY OPTIMIZATION

This chapter presents the work published in [41].

3.1 Introduction

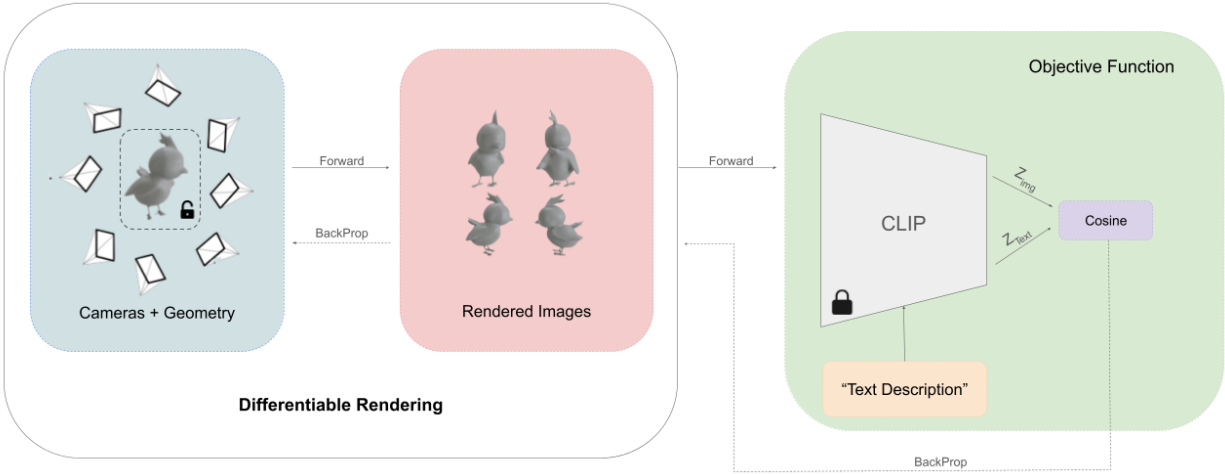


Figure 3.1: **CLIP-Similarity Optimization**

The latent space editing operations explored in the previous chapter demonstrate that neural shape representations can capture semantically meaningful geometric structure. However, the demonstrated manipulations were coarse and difficult to control precisely. Advances in vision-language models like CLIP [125] introduced a new paradigm for controllable visual generation and editing, using natural language as an interface. Large-scale text-driven image generation systems such as [133] and [127] have been trained on massive datasets of captioned web images. Recent research in 3D generative modeling [106, 76, 164, 113, 121] is similarly text-driven.

However, due to the relative lack of labeled 3D training data, these methods leverage differentiable rendering to provide semantic supervision for modifying and generating geometry. Figure 3.1 shows a prototypical example of this family of methods. Renderings of

a source mesh, along with a target text description, are encoded into latent vectors using CLIP, and the average cosine similarity between the image vectors and the text vector is computed as the objective. Gradients are backpropagated through the rendering process to the optimizable parameters in the mesh, often including vertex positions and shape textures. This process encourages the source mesh to change in a way such that the rendered images are more similar to the text prompt, in CLIP’s latent space.

In contrast to previous text-guided works that either generate geometry from scratch like [121, 164] or focus on adding textural detail such as [106], in this chapter we instead focus on shape editing, and specifically shape deformation. Our framework, TextDeformer, produces deformations that capture both large-scale structure and fine geometric detail, while remaining consistent with the input shape. Unlike methods that generate geometry entirely from scratch, deformation-based editing requires maintaining coherent correspondence between the source and edited geometry (e.g., a leg deforms into a semantically corresponding leg rather than an unrelated structure). Thus, the editing process must simultaneously 1) produce high-quality surface geometry, with minimal self-intersections and noisy normals; 2) produce plausible results which match the text description; 3) remain faithful to the underlying geometric structure (e.g., deform the source’s head into the target’s head and not into body).

Straightforward optimization of mesh vertices through differentiable rendering, as previous text-to-3D methods displayed, often converges to undesirable local minima, and gradient steps often turn parts of the mesh inside-out, introducing significant artifacts. The crux of the difficulty lies in the fact that the back-propagated gradient from CLIP is noisy, with many undesirable local minima and arbitrary directions. Thus, instead of displacing vertices, we take inspiration from Neural Jacobian Fields [1] to devise a more robust representation of deformations. We optimize matrices representing the deformation’s gradients, i.e., the Jacobians of each of the triangles, and compute the deformed vertex positions from them, by

solving Poisson’s equation. Our key observation is that representing the deformation through Jacobians in this way leads to a representation that favors smoother, large deformations while introducing global relation between vertices and pixels, thus avoiding localized noisy gradients. More precisely: i) smooth Jacobians represent low-frequency, large-scale deformations, and ii) Poisson’s equation leads to each Jacobian affecting all vertices, and in turn all pixels of the rendering. Thus, when CLIP’s gradients back-propagate to the Jacobians, each pixel’s gradient has a global effect, leading to a more regularized solution.

Second, we observe that CLIP-derived per-pixel embedded features are unfortunately view-dependent and the same 3D coordinate may be assigned significantly different features in different 2D viewpoints. This in turn implies a given vertex on the mesh may receive conflicting gradients from different viewpoints (conceptually, when deforming a cow into an elephant, one viewpoint may try to tug on a vertex to grow a trunk out of it, while the other will try to use it as a tip of a tusk). This leads to incoherent results, and, in some cases, a lack of global consistency (e.g., adding multiple trunks when deforming a cow into an elephant). To counter that, we devise a novel loss, which encourages vertices to achieve similar CLIP features from different viewpoints, thereby leading to global coherency in the deformations.

Third, to ensure the deformed shape still lies in semantic correspondence to the input shape, we add an identity-preserving term, which ensures that the deformation optimization step does not stray too far from the initial input mesh, thereby preventing the optimization from ignoring the input geometry.

Through experiments, we show we can apply text-driven deformations to a large class of source shapes and desired targets (from organic to man-made shapes). Our method produces plausible shapes while preserving properties unique to deformation-based frameworks, such as significant resemblance to the input shape, and providing meaningful correspondences between the source and deformed shapes.

3.2 Related Works

There has been a large variety of works in the space of text-guided content synthesis driven by CLIP [125], a foundational model which learns a joint embedding space for text and images. Many generative models such as StyleCLIP [119], GLIDE [112], and DALL-E-2 [127] leverage text during training by computing distance between text and images in the embedding space of CLIP. Additionally, there has been work on using CLIP guidance in fine-tuning the latest state-of-the-art diffusion models to achieve even higher quality results [77].

3.2.1 Text-Guided 3D Synthesis

In comparison to text guided image generation, text-to-3D is relatively undeveloped. While there are some works that propose training joint embeddings of text descriptions and 3D objects [18], these works are lacking in scale as the largest captioned 3D dataset (the recent Objaverse dataset with 800k assets [30]) is several orders of magnitude smaller than the LAION-5B dataset [139]. Nevertheless, there have been large strides in text-to-3D leveraging large pre-trained 2D models such as CLIP.

CLIP-Forge [136] overcomes the lack of pre-trained counterpart to CLIP for 3D by using renderings of training shapes to bridge the gap between text and 3D data. They first train a voxel encoder and an implicit decoder on available 3D datasets using CLIP image embeddings, then swap image embeddings for text embeddings at inference time. However, their method is still limited by the availability of 3D datasets used to train their autoencoder. Point-E [113] proposes to generate an image from text using a 2D diffusion model, then trains a point cloud diffusion model conditioned on images on a private dataset of millions of 3D shapes. While not on par with the state-of-the-art in terms of shape quality, their approach can generate 3D shapes significantly faster.

Several approaches tackle zero-shot geometry synthesis, bypassing the need of a 3D dataset. Dreamfields [65] leverage volume rendering with a Neural Radiance Field (NeRF) [108]

to directly optimize views of a 3D shape against a desired text prompt in CLIP’s embedding space. Recently, leveraging 2D diffusion models [133, 135], DreamFusion [121] and [164] propose to distill such models as a differentiable image-based loss. Extracting and editing an explicit mesh from these works is not straightforward, since NeRFs represent shapes through network weights.

Other works have used surface-based differentiable rendering in order to pass views of explicit 3D objects to CLIP. Using this method, Text2Mesh [106] employs a network to predict colors and deformations along the normals of a template mesh. Their objective is to *stylize* the template mesh while *preserving* the initial content. In contrast, CLIP-Mesh [76] proposes deforming the vertices of a sphere in accordance to an input text prompt to synthesize a completely new geometry. Magic3D [94] combines both representations by first optimizing a radiance field using score-distillation similar to DreamFusion, then extracts an explicit mesh from the radiance field and optimizing its vertices via differentiable surface rendering and score distillation. The code is not public at the time of submission. Our work also leverages differentiable rendering and CLIP, but focuses on the problem of *deforming* explicit geometry rather than generating it from scratch.

3.2.2 Neural Shape Deformation

Shape deformation has been traditionally approached by providing a user with handles that control a deformation space, either through an energy-minimizing formulation [148, 147] or through skinning the mesh with weights that interpolate coordinates with respect to the handles [64, 35]. Skinning methods have been used extensively in learning context [177, 178, 56, 90], point handles [97], and cages [183], while variational formulations are used as regularizers, e.g., ARAP [150] or the Laplacian [72]. Both of these approaches span only a subset of possible deformations and prevent fine-grained control over details. Other methods focus on learning tasks over one template mesh [152, 38], and assign per-vertex

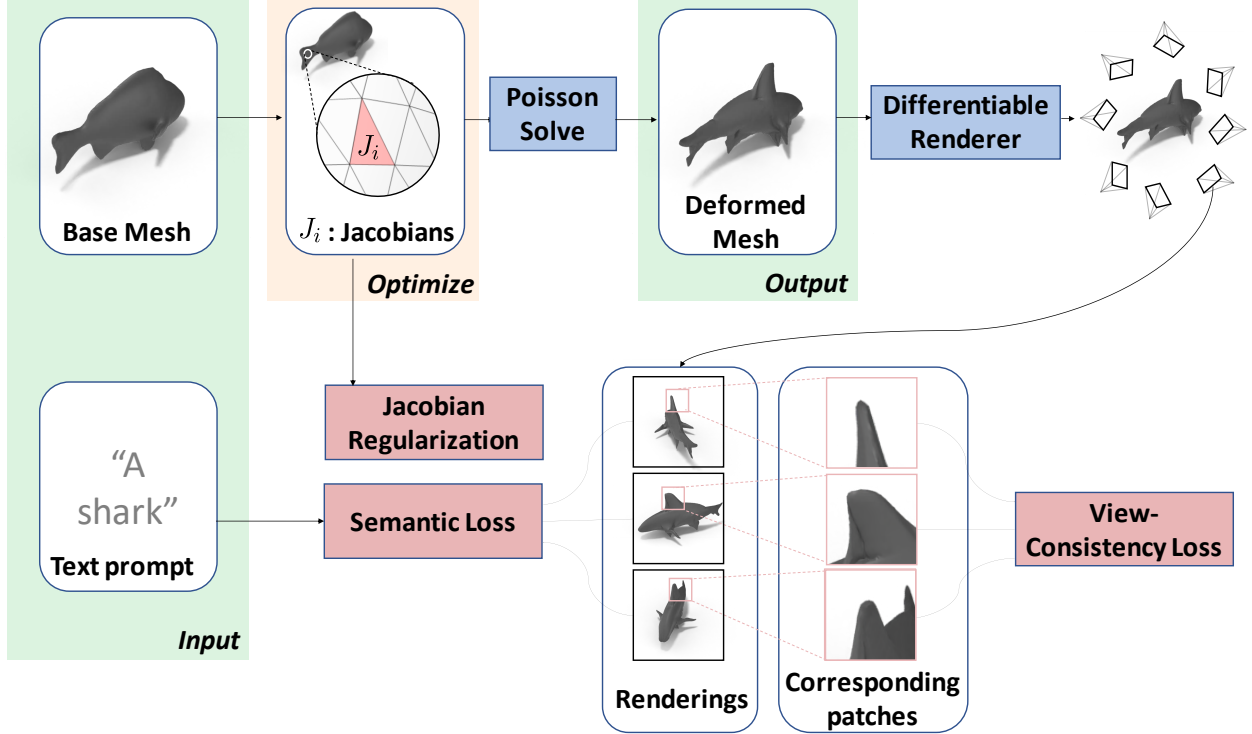


Figure 3.2: **TextDeformer Overview.** TextDeformer deforms a base mesh by optimizing per-triangle Jacobians using natural language as a guide. We optimize the deformation using three losses: a CLIP-based semantic loss drives the deformation toward the text prompt, a view-consistency loss matches multiple views of the same surface patch to ensure a coherent deformation, and our regularization on the Jacobians controls the fidelity to the base mesh.

coordinates [143] or offsets from a simpler (e.g., linear) model [3, 4, 134, 194, 185].

While some recent works propose data-driven approaches to predict realistic deformations [1, 66, 183, 47, 189], the semantic capabilities of all these works are limited once again by the lack of 3D datasets pairing shapes and captions. Our work is similar in spirit to these methods, but instead of requiring explicit supervision, we leverage differentiable rendering and powerful visual models such as CLIP to drive deformations of a template shape.

3.3 TextDeformer

Figure 3.2 shows an overview of our method. Given an input shape, TextDeformer enables manipulating the geometry guided by a user-specified text description. We represent the

geometry of the input shape using a mesh $\mathcal{M}$ defined by a set of vertices $\mathcal{V} \in \mathbb{R}^{n \times 3}$ and faces $\mathcal{F}$. We optimize a displacement map $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ over the vertices through differentiable rendering.

3.3.1 Deformations through Jacobians

A naive optimization of Φ would simply entail directly displacing each vertex $\mathcal{V}$, which may overly distort the original shape, especially when the target text describes a highly-detailed texture. This is due to this representation exposing high-frequency, oscillatory modes of deformation. Thus, in this work we opt for a different parameterization of deformations. Inspired by Neural Jacobian Fields [1], we parameterize the shape using a set of per-triangle Jacobians which define a deformation. Specifically, we represent per-triangle jacobians by matrices $J_i \in \mathbb{R}^{3 \times 3}$ for every face $f_i \in \mathcal{F}$. Following [1], we solve a Poisson problem to compute a deformation map Φ^* as the mapping with Jacobian matrices for each face that are closest to $\{J_i\}$ in the least-squares sense, that is:

$$\Phi^* = \min_{\Phi} \sum_{f_i \in \mathcal{F}} |f_i| \|\nabla_i(\Phi) - J_i\|_2^2 \quad (3.1)$$

where $\nabla_i(\Phi)$ denotes the Jacobian of Φ at triangle f_i and $|f_i|$ is the area of that triangle. Hence, we may optimize the deformation mapping Φ indirectly by optimizing the matrices $\{J_i\}$ which define Φ^* . These Jacobians are initialized to identity matrices, thereby initializing Φ^* as the identity mapping. Please refer to [1] for the full technical details.

3.3.2 Language Guidance

Our objective is to use text to guide the deformation of the source shape. We leverage the pre-trained vision-language CLIP [125], which provides a shared embedding space between images and text. In order to connect geometry to images, we pass our shape through

a differentiable renderer $\mathcal{R}$ [86]. Hence, we may differentiably embed the renders of the deformed shape

$$e_{\mathcal{M}} = \text{CLIP}(\Phi^*(\mathcal{M})) \in \mathbb{R}^{512}$$

We abuse notation and the rendering function $\mathcal{R}$ is understood to be implicit when passing shapes to CLIP. The desired deformation, described with a natural language prompt $\mathcal{P}$, is also embedded $e_{\mathcal{P}} = \text{CLIP}(\mathcal{P}) \in \mathbb{R}^{512}$. Then, we may optimize Φ^* such that $e_{\mathcal{M}}$ and $e_{\mathcal{P}}$ agree, by maximizing the cosine similarity between the embeddings:

$$\mathcal{L}_{\mathcal{P}}(\Phi^*, \mathcal{M}, \mathcal{P}) = \text{sim}(e_{\mathcal{M}}, e_{\mathcal{P}}) \quad (3.2)$$

where $\text{sim}(\cdot, \cdot)$ stands for cosine similarity. Similarly to StyleCLIP [119], we find that incorporating relative directions in CLIP’s embedding space can give stronger signals when the optimization landscape between Φ^* and $\mathcal{P}$ is unclear. Given a base caption $\mathcal{P}_0$ that describes $\mathcal{M}$, we compute the direction between the target prompt and the base prompt: $\Delta\text{CLIP}(\mathcal{P}, \mathcal{P}_0) = \text{CLIP}(\mathcal{P}) - \text{CLIP}(\mathcal{P}_0)$. We compute the direction of the deformations and aim to optimize:

$$\mathcal{L}_{\Delta\mathcal{P}}(\Phi^*, \mathcal{P}, \mathcal{P}_0) = \text{sim}(\Delta\text{CLIP}(\mathcal{P}, \mathcal{P}_0), \Delta\text{CLIP}(\Phi^*(\mathcal{M}), \mathcal{M})). \quad (3.3)$$

3.3.3 Jacobian Regularization

To prevent the deformation from straying too far from the input undeformed geometry, we introduce another regularization term on the predicted Jacobians, which penalizes the difference between the Jacobians $\{J_i\}$ and the identity, i.e., no deformation:

$$\mathcal{L}_I(t_j) = \alpha \sum_{i=1}^{|\mathcal{F}|} \|J_i - I\|_2 \quad (3.4)$$

where α is a hyper-parameter which may be tuned to control the strength of the deformations defined by $\{J_i\}$.

3.3.4 View Consistency.

A common problem when performing multi-view optimization through CLIP is the lack of view-consistency, i.e., a particular view may pull the shape toward a specific deformation while another view pulls toward a different deformation. Averaging gradients on the Jacobians for each view does not necessarily lead to a coherent 3D deformation which may manifest in various artifacts, including muddled details and incorrect geometry.

We introduce another regularization term to tackle this problem by utilizing the patch-level deep features of CLIP’s vision transformer (ViT). In ViTs, the image is split into non-overlapping patches $P_0, P_1, \dots, P_n$, which are then projected into a higher-dimensional space and passed through transformer encoder blocks $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n$. For each vertex $v \in \mathcal{V}$, and each render $r \in \mathcal{R}(\mathcal{M})$, if v is visible, we compute the pixel $p(v, r)$ in r that contains v . Then, by associating $p(v, r)$ with the nearest corresponding patch center $P(v, r)$, we extract a deep feature vector corresponding to v and r and encourage vertices to have similar deep features across renders from different viewpoints:

$$\mathcal{L}_{\text{VC}}(v) = \sum_{i=1}^{|\mathcal{R}(\mathcal{M})|} \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{R}(\mathcal{M})|} \text{sim}(\mathcal{T}_k(P(v, r_i)), \mathcal{T}_k(P(v, r_j))) \quad (3.5)$$

for some chosen layer $\mathcal{T}_k$. In practice, we choose to use the *token* output of the final transformer block. Then, we simply penalize this loss over all vertices $v \in \mathcal{M}$:

$$\mathcal{L}_{\text{VC}}(\mathcal{M}) = \beta \sum_{v \in \mathcal{V}} \mathcal{L}_{\text{VC}}(v) \quad (3.6)$$

where β is another tunable hyper-parameter. To compute $P(v, r)$, we follow [2] by mod-

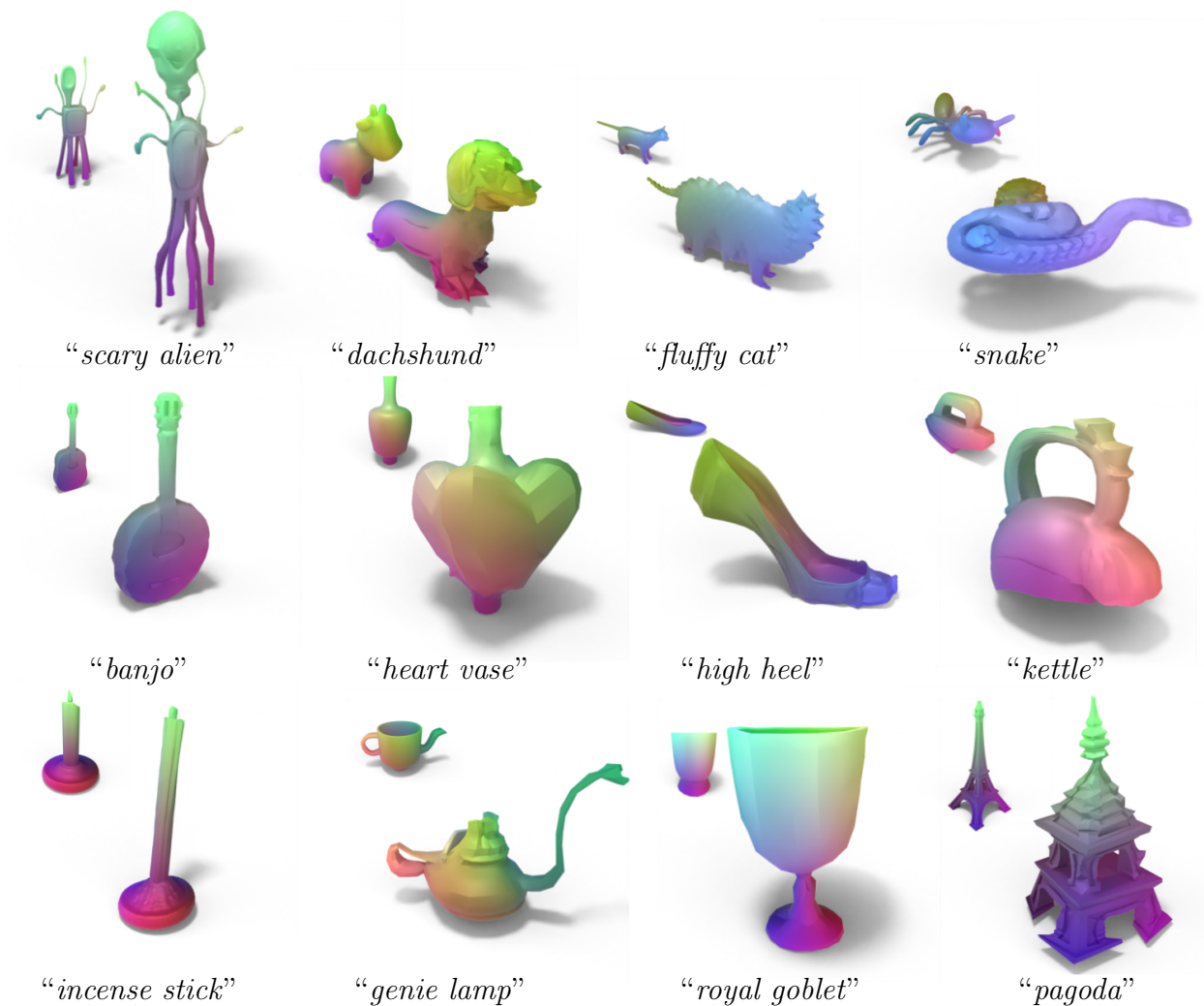


Figure 3.3: **Gallery of results.** Source meshes inset and target text below.

ifying CLIP’s ViT to use a smaller stride in its initial convolution, obtaining *overlapping* patches. Then, by interpolating the positional encoding, we achieve finer-resolution deep features.

3.4 Experiments

We run TextDeformer on a variety of text prompt, source mesh pairs. Each pair takes approximately 1.5 hours for 5000 iterations.

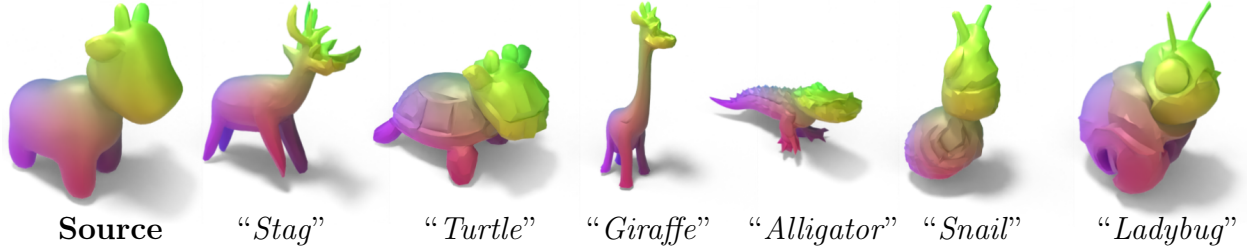


Figure 3.4: TextDeformer deforms a source shape into various text-specified targets. The mesh colors visualize the smoothness of the mappings.

3.4.1 Generality of TextDeformer

We show that TextDeformer can handle a wide variety of source and target prompts. We represent the source mesh as a small inset next to the deformed shapes unless otherwise specified. Figure 3.3 shows one such collection of source and targets. We see that TextDeformer is capable of producing high-quality results for different types of source and target pairs.

Adjective Targets We see that TextDeformer is capable of deforming source meshes in accordance to a target *adjective*. In Figure reffig:gallery, we see that TextDeformer deforms a cat to be “*fluffy*,” a vase to be “*heart-shaped*,” an alien to be “*scary*,” and a shoe to be “*high-heeled*.” For this style of target text, our method produces deformation maps that preserve the overall structure of the source shape.

Related Targets TextDeformer is also capable of deforming source meshes into *related* shapes, which are not exactly descriptors of the original mesh, but also not too semantically different. For example, in Figure 3.3, we see that TextDeformer is able to deform an acoustic guitar into a “*banjo*,” a candle into an “*incense stick*,” the Eiffel tower into a “*pagoda*,” a vase into a “*royal goblet*,” and so on.

Unrelated Targets Finally, we observe that TextDeformer is capable of deforming source meshes into completely *unrelated* shapes, which are far from the source mesh. This capability is illustrated in Figure 3.3 where an ant is deformed into a “*snake*,” and a clothing iron is

deformed into a “*kettle*,” as well as in Figure 3.4 where a cow is deformed into various unrelated animals such as a “*turtle*”.

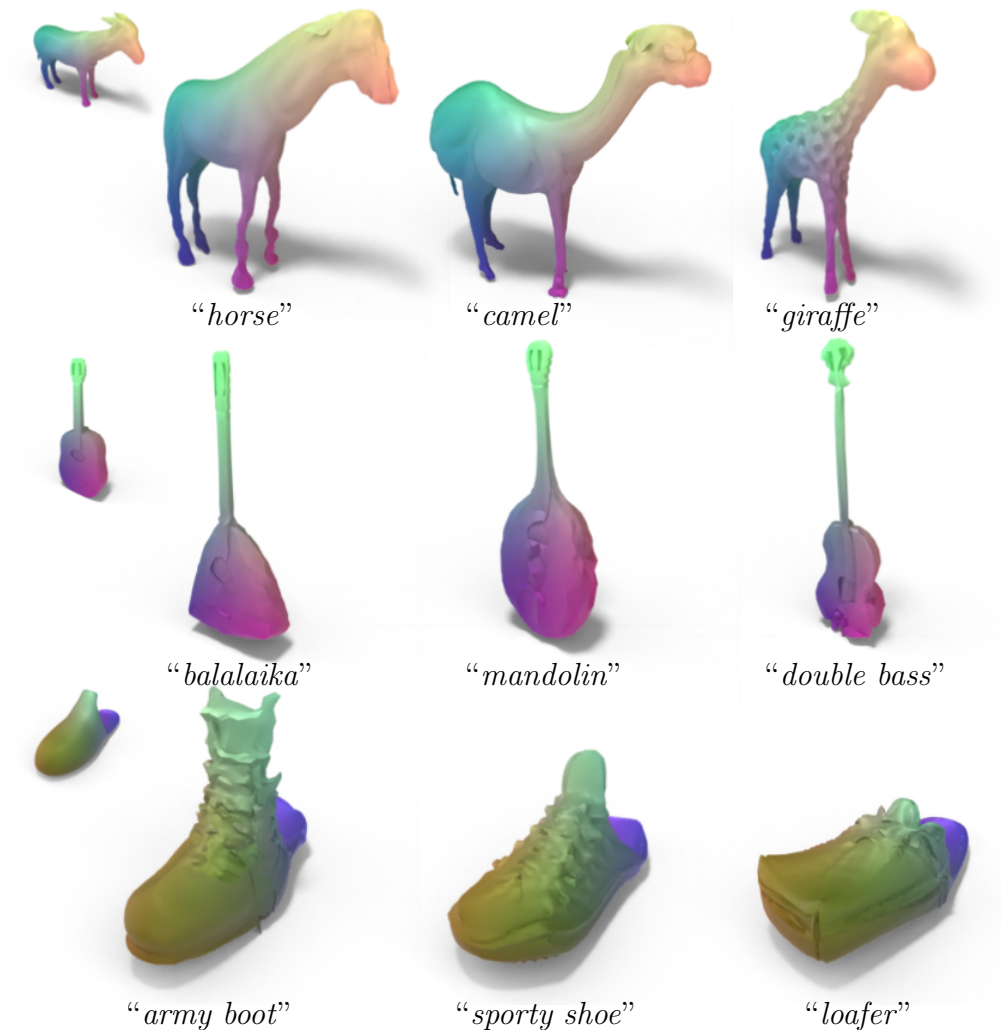


Figure 3.5: **Text-driven deformation.** Our method can deform the same source into different text-specified geometries, achieving local geometric details (shoe laces and giraffe) and low-frequency shape modifying deformations (body of guitar, shape of animal).

3.4.2 Expressiveness of TextDeformer

Frequency We study the expressiveness of our method by experimenting with different text prompts for the same source mesh. We observe that TextDeformer is powerful enough to express both high-frequency texture details as well as low-frequency deformations to the

shape structure. In Figure 3.5, we see in the top row that TextDeformer can produce the requisite low-frequency deformations to change the donkey into various other animal shapes, but also add high-frequency deformations to emulate “*giraffe*” spots. Similarly in middle row, TextDeformer produces different low-frequency maps to change the shape of the guitar body while preserving the neck. Alternately in the last row, TextDeformer produces high-frequency deformations to emulate fine details such as the laces of an “*army boot*” or the creases in a “*sporty shoe*.” Similarly, we observe TextDeformer producing high-frequency deformations in Figure 3.4, particular in the “*alligator*” example.

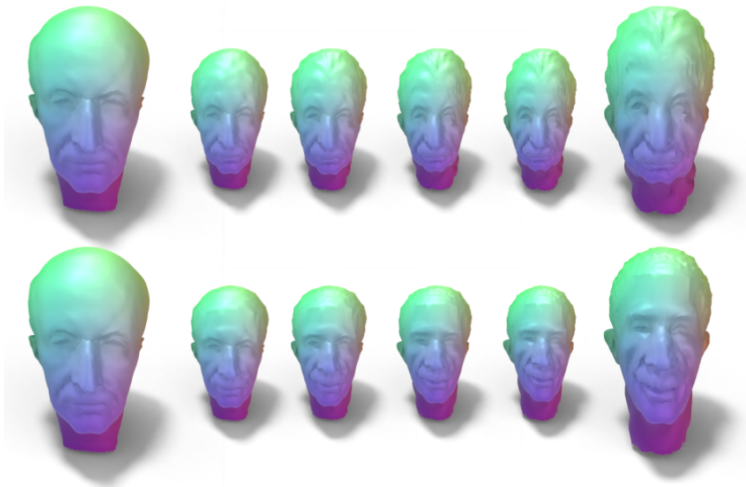


Figure 3.6: Visualizing iterations of the optimization as the source mesh is deformed to the target (“*Einstein*”, “*Obama*”). Due to our formulation and energy, each facial feature of Max Planck is deformed into the corresponding facial feature of the target (nose to nose, eyes to eyes etc.).

Dense Matching In Figure 3.6, we demonstrate the ability of TextDeformer to preserve highly detailed semantics throughout the optimization process. Not only does TextDeformer correctly deform features of Max Planck to the corresponding features of the target (“*Einstein*,” “*Obama*”), it is also consistent at every step. We observe at different intermediate faces that the corresponding features are always mapped correctly.

3.4.3 Identity Preservation

We demonstrate our method’s ability to preserve the source shape in two ways.

Impact of the Input Geometry

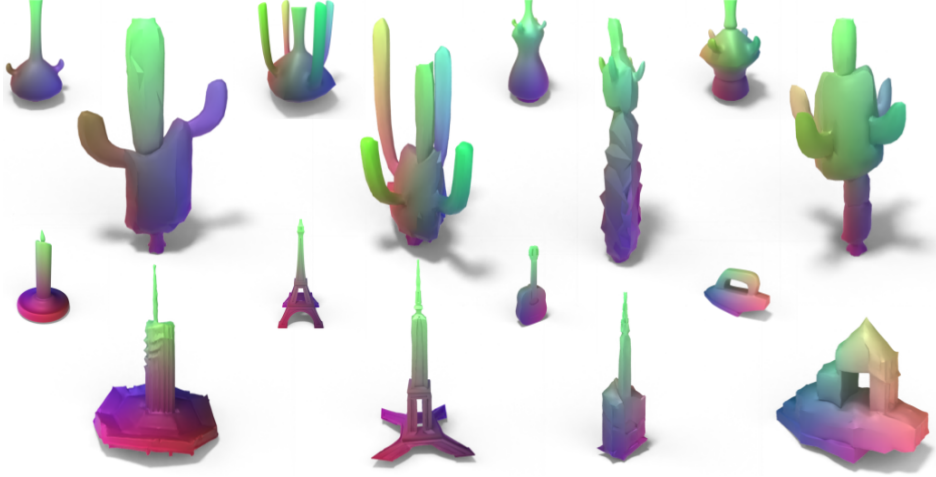


Figure 3.7: Deformations of different source vases into the same target text, “*a cactus*” and different source meshes into the same target text, “*a skyscraper*”. Our method meaningfully respects the input geometry while conforming to the desired target text.

In Figure 3.7, we use TextDeformer to deform different vases with various appendages into “*a cactus*.” Note that the resulting deformation map grows cactus branches depending on where appendages are on the source mesh. Hence, TextDeformer is able to produce a deformation map that conforms to the target text, “*cactus*,” in an adaptive manner, borrowing coarse structures from the input geometry. We also observe a similar effect when deforming semantically diverse meshes into the target text, “*skyscraper*”. TextDeformer preserves different aspects of each mesh, such as the base of the candle and the body of the guitar. It also produces interesting variation e.g., deforming the flatter iron into a dome structure instead of a thinner needle.

Jacobian Regularization Recall that we also define a Jacobian regularization term $\mathcal{L}_I$ in Equation 3.2 which is scaled by a hyper-parameter α . In Figure 3.8, we show that adjusting α controls how far the deformation map Φ^* is allowed to deviate from the source mesh. With

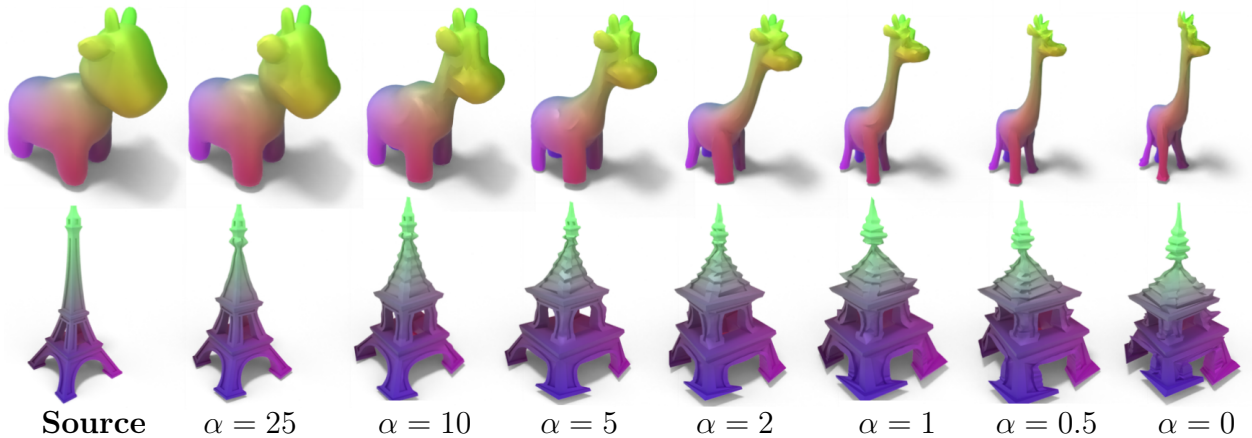


Figure 3.8: **Identity preservation.** The source shape (left) is deformed into a “*giraffe*” (top), “*pagoda*” (bottom) using decreasing amount of weight on the proposed Jacobian regularization. Higher weight encourages preservation of the identity of the source. Note that results with zero Jacobian regularization (right column) may contain artifacts.

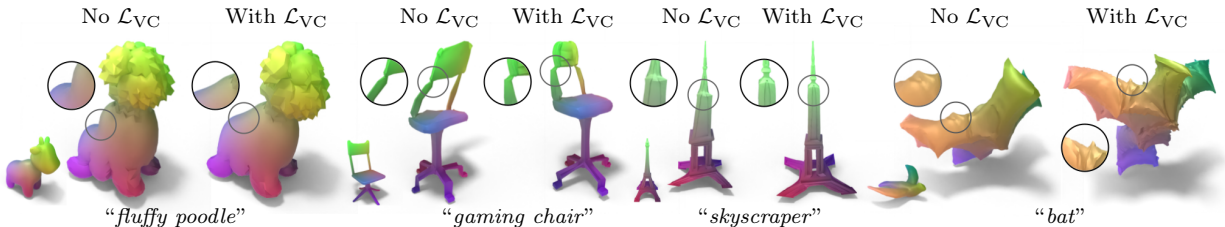


Figure 3.9: **Viewpoint Consistency Ablation.** Ablation results of removing $\mathcal{L}_{VC}$ for four different source-text pairs. In each example we see instances of incorrect geometry in the shapes deformed without $\mathcal{L}_{VC}$.

$\alpha = 25$, we see that the cow and the Eiffel tower do not change meaningfully in accordance to their respective text prompts (“*giraffe*” and “*pagoda*”), while setting $\alpha = 0$ may result in some artifacts in the deformed shape. We observe that setting α to intermediate values offers the best results.

3.4.4 Viewpoint Consistency

We experiment with the qualitative effect of the viewpoint consistency loss $\mathcal{L}_{VC}$ by using TextDeformer with and without this loss term and noting the differences in the deformed meshes. We observe in Figure 3.9 that the deformations produced without $\mathcal{L}_{VC}$, although smooth due to our choice in representation, often contain unrealistic geometric features. Such

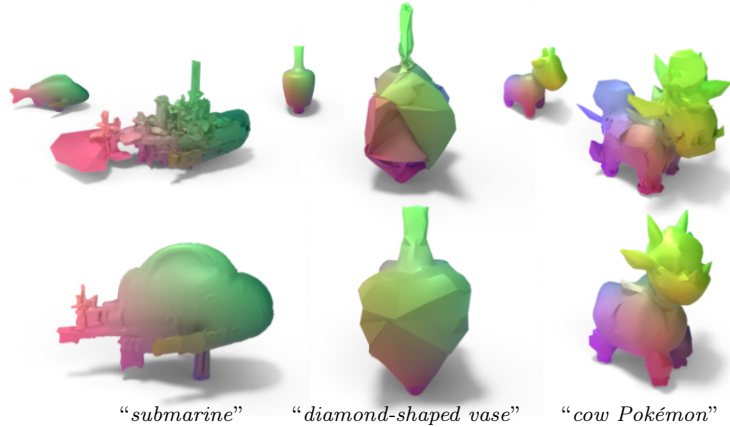


Figure 3.10: **Deformation Ablation.** Top row: TextDeformer with vertex displacements. Bottom row: TextDeformer with Jacobians. Jacobians are crucial to preserving the structure of the input geometry and maintaining high-quality surfaces.

abnormalities can be caused by outliers in the sampled camera views during optimization. This problem is especially apparent in the “*gaming chair*,” in which the backrest is crooked. This inaccuracy may appear correct in some perspectives, but is overall an undesirable feature. Another example is the tip of the “*skyscraper*,” which is pulled to one side of the building during optimization. Finally, we observe that the back of the “*fluffy poodle*,” and the wings of the “*bat*” are incorrectly curved inwards when they are optimized without $\mathcal{L}_{VC}$. We provide a quantitative evaluation of these observations in Section 3.4.7.

3.4.5 Effect of Jacobians

We also experiment with replacing Jacobians with vertex displacements in our pipeline, but keeping the additional losses we introduce, $\mathcal{L}_{\Delta\mathcal{P}}$ and $\mathcal{L}_{VC}$.

Surface Quality We observe that our use of Jacobians plays a key in obtaining high-quality surface geometries. In Figure 3.10, we show that replacing Jacobians with vertex displacements in TextDeformer drastically decreases the resulting surface quality for three different examples. In the “*diamond-shaped vase*,” vertex displacements cause the shape to collapse inwards. In the other two examples, vertex displacements create details which

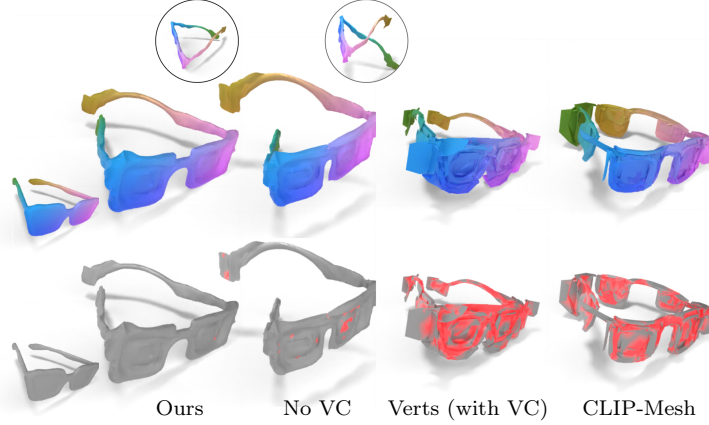


Figure 3.11: **Self-intersections.** Comparison results for the shown source and target text “goggles”. Self-intersections are highlighted in red (bottom row). Removing view-consistency (VC) losses causes distortion on the temple arms. Removing Jacobians and optimizing vertices introduces further surface distortion and self-intersections which may impede utility. When applying CLIP-Mesh to this template, we observe the “janus” effect e.g., unrealistic repeated geometry on each side.

do not respect the initial shape template and contain many self-intersections. The bottom row of Figure 3.11 highlights this deterioration of surface quality for another example. We observe that vertex displacements cause a large number of self-intersections, even in areas where the resulting surface appears to require a relatively simple deformation, such as the lens of the “goggles.”

Globally Coherent Deformations Recall from Equation 3.1 that we solve a global Poisson system to compute the deformation map from the Jacobians. Hence, unlike the gradients of vertices which only affect one point, gradients propagated through Jacobians influence a large surface mesh area, leading to a *globally-coherent* deformation. In Figure 3.12, in which a bust of Max Planck is deformed to various other faces, we observe that vertex displacements take local, sub-optimal steps, not only contributing to poor surface quality, but also, in the example of “Obama,” causing spurious face mirroring on the back of the head. Since there are few camera views in which the front and back of the head are visible, the optimization process naturally leads to this result. However, we observe that Jacobians do not experience this artifact.

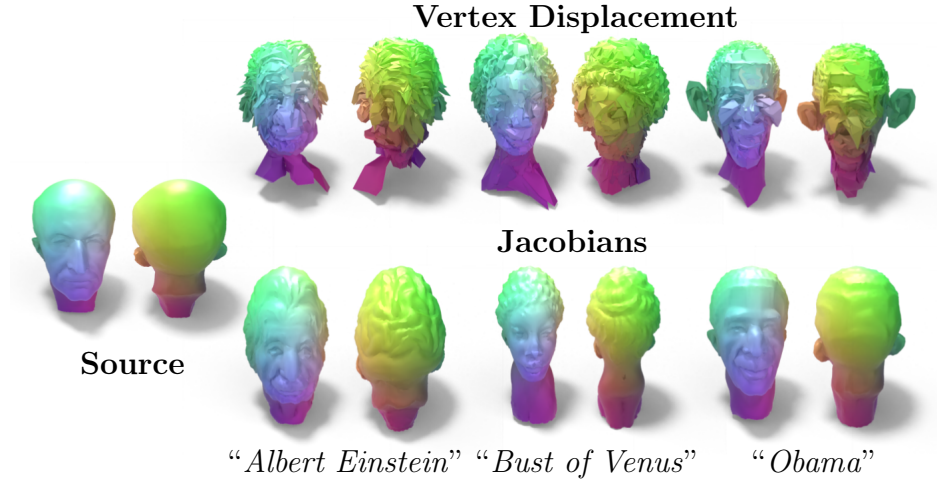


Figure 3.12: **Globally-Coherent Deformations.** Max Planck deforms into different targets using our method (bottom row), where the front/back view is shown in pairs of left/right. Removing Jacobians and predicting displacements (top row) takes locally sub-optimal steps, which results in distorted shapes with significant artifacts. Jacobians produce global deformations, resulting in cleaner geometries and even prevents the spurious face mirroring (for example, in “Obama”).

3.4.6 Qualitative Comparisons

Beyond ablations, we also qualitatively evaluate our method against two existing text-based methods for 3D synthesis and editing, CLIP-Mesh and Text2Mesh. In Figure 3.13, we show results for 5 different target text prompts: “*an alligator*”, “*a camel*”, “*a chinese lantern*”, “*a comfortable chair*”, and “*an octopus*”. TextDeformer outperforms the baselines in multiple aspects. First, our method produces more semantically accurate deformations: a more elongated “*alligator*”, a larger “*comfortable chair*”, and a rounder “*chinese lantern*”. In the examples of “*a camel*” and “*an octopus*” where CLIP-Mesh produces semantically correct results, we observe that it produces poor quality surfaces with irregular triangulation and self-intersections, whereas TextDeformer produces smoother, more realistic geometry.

3.4.7 Quantitative Evaluation

We evaluate TextDeformer quantitatively in two ways, comparing to CLIP-Mesh [76], Text2Mesh [106], as well as our method without $\mathcal{L}_{VC}$ and our method but using vertex displacements.

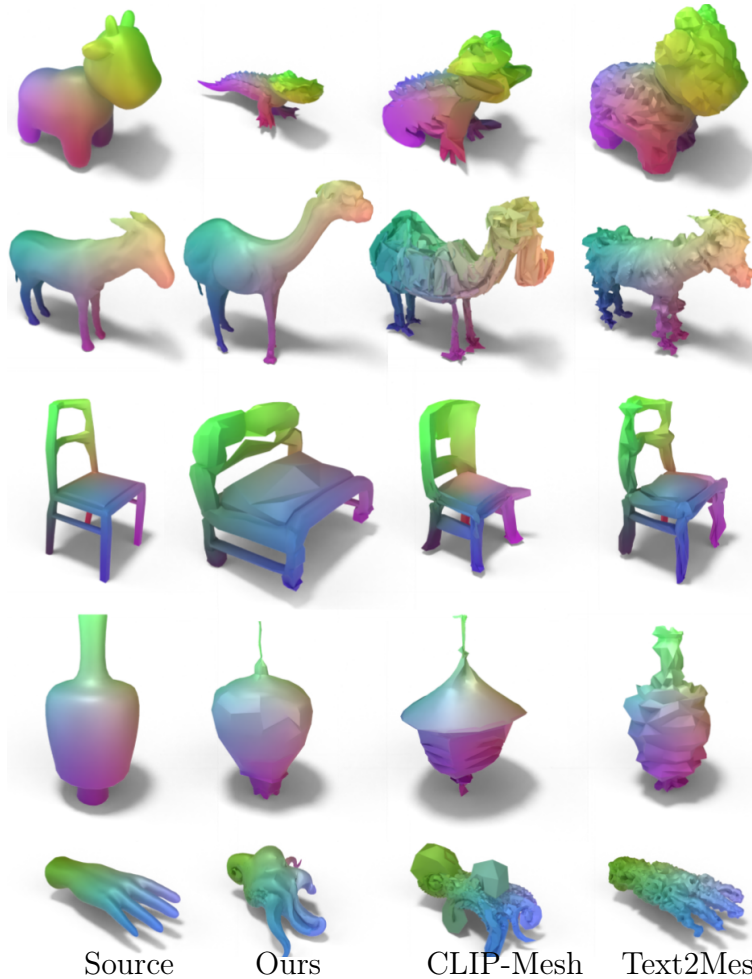


Figure 3.13: **Qualitative Comparison** From top to bottom, the target text prompts are “an alligator”, “a camel”, “a comfortable chair”, “a chinese lantern”, “an octopus”. Compared to CLIP-Mesh, our method produces more semantically correct and higher quality surfaces. Text2Mesh fails to produce semantically meaningful deformations in all examples.

Retrieval Precision First, following CLIP-Mesh [76], we compute the R-Precision of CLIP-L/14 on a retrieval task. Specifically, we use a set of 111 text prompt, source mesh pairs to generate a set of deformed meshes. Then, we retrieve deformed shapes for each text prompt through CLIP (L/14) cosine similarity. The first column of Table 3.1 shows the results of this experiment. We observe that the viewpoint consistency loss $\mathcal{L}_{VC}$ increases the deformation quality of the model, affirming our observations in Section 3.4.4. We also observe that the pipelines using vertex displacements achieve the highest R-Precision scores. This is to be expected as vertex displacements give a high amount of freedom when deforming the

	CLIP R-Precision (L/14) $\uparrow$	Intersections $\downarrow$
Ours	55.2%	3.2%
Ours-noVP	51.5%	3.3%
Ours-Verts	55.4%	67.7%
CLIP-Mesh	57.4%	62.8%
Text2Mesh	12.7%	17.3%

Table 3.1: **Quantitative evaluation.** We use our text prompts and deformed meshes in a retrieval task to compute R-Precision. We observe that regularizing for viewpoint consistency improves TextDeformer R-Precision. TextDeformer and CLIP-Mesh achieve quantitatively comparable R-Precision, TextDeformer (Ours) produces higher-quality geometry both qualitatively (see Figure 3.11) and quantitatively (significantly fewer self-intersections). All methods significantly outperform Text2Mesh in R-Precision.

shape towards the text prompt, without respect to the initial template geometry. Finally, all methods outperform Text2Mesh, which is more suited for stylization and texturization problems rather than our semantic editing problem.

Geometric Quality (Self-Intersection) We also measure the ratio between the number of self-intersections in the deformed mesh and the number of faces. This validates the qualitative evaluation of in Figure 3.10, Figure 3.11, and Figure 3.13 that the vertex displacement pipelines disregard the triangulation of the shape in order to optimize CLIP cosine-similarity.

3.5 Discussion

This chapter presented the approach of backpropagating through text-image foundation models to optimize 3D geometry, along with our contributions in [41]. Our key observation is that CLIP-similarity objectives produce noisy gradients which are poorly suited for direct per-vertex optimization. To address this, we optimize per-face *Jacobians* instead of vertex locations, predicting smoother deformations and mitigating sharp local artifacts. We additionally introduce a view consistency loss that reduces over-fitting to salient views by encouraging semantically consistent interpretations of geometry across all views. Finally, we

propose an identity regularization term that allows users to control the magnitude of the deformation while maintaining correspondence to the source shape. Together, these components enable expressive text-guided deformations while preserving coherent global structure.

Since TextDeformer, subsequent works have continued to explore regularization strategies for text-guided geometric optimization. DreamMesh [180] generates high-quality textured meshes via a multi-stage pipeline, leveraging our Jacobian parameterization to stabilize optimization and to avoid degenerate local minima. MagicClay [7] instead uses a hybrid representation that maintains both a textured mesh and a network that represents a signed distance field (SDF). The authors observe that the SDF is more robust to the noisy text-based optimization process, while the explicit mesh preserves topological properties of the original shape. Their method further imposes a consistency loss between the two representations to enforce consistency.

Despite this progress, the optimization approach to shape editing is fundamentally limited. Large structural deformations are difficult to produce reliably, especially in regions with low face density. For example, in Figure 3.3, TextDeformer successfully arches a flat shoe into a high heel, but struggles to extrude the heel geometry itself. While dynamic remeshing partially alleviates this issue as in [7], these methods still have difficulty performing substantial topological modifications. More broadly, surface optimization approaches cannot represent edits that alter the surface topology. For instance, these techniques cannot add a handle to a cup or put a hole through a chair. Finally, iterative optimization remains computationally expensive and often unpredictable, with editing sessions frequently requiring long runtimes and repeated prompt tuning.

Motivated by these limitations, we turn towards feed-forward generative methods in 3D, which can perform arbitrary edits through a single forward pass.

CHAPTER 4

GENERATIVE 3D EDITING WITHOUT OPTIMIZATION

This chapter presents the work published in [42].

4.1 Introduction

In this chapter, we investigate feed-forward reconstruction models for 3D editing as an alternative to the optimization-based approaches discussed in the previous chapter.

A common strategy to edit 3D models without optimization often utilize a two-stage approach, delegating semantic generation and editing primarily to pretrained 2D generative models. In these methods, edited images are first synthesized and subsequently lifted into 3D geometry using reconstruction pipelines [89, 175]. As a result, the quality and consistency of the final 3D edit depend heavily on the multiview consistency of the intermediate representations.

Many of these approaches rely on diffusion models that synthesize multiple views [98, 145, 101, 100] before reconstructing geometry using a feed-forward Large Reconstruction Model (LRM) [171, 57]. While editing a single image is increasingly reliable, multi-view generation often suffers from ambiguous structure in occluded regions and fails to maintain globally consistent geometry across viewpoints. Efforts have been made to adapt multi-view diffusion models specifically for text-driven editing rather than single-view-to-multiview synthesis [8, 34] though, as we qualitatively demonstrate, editing multiview images in a realistic manner remains a challenging task.

To overcome these challenges, we introduce an alternative method that lifts 2D edits into 3D. Rather than using a 3D reconstruction model purely as a geometric lifting stage, our approach directly trains the LRM itself to perform masked geometric completion. This completion process is learned directly within the 3D reconstruction model rather than through

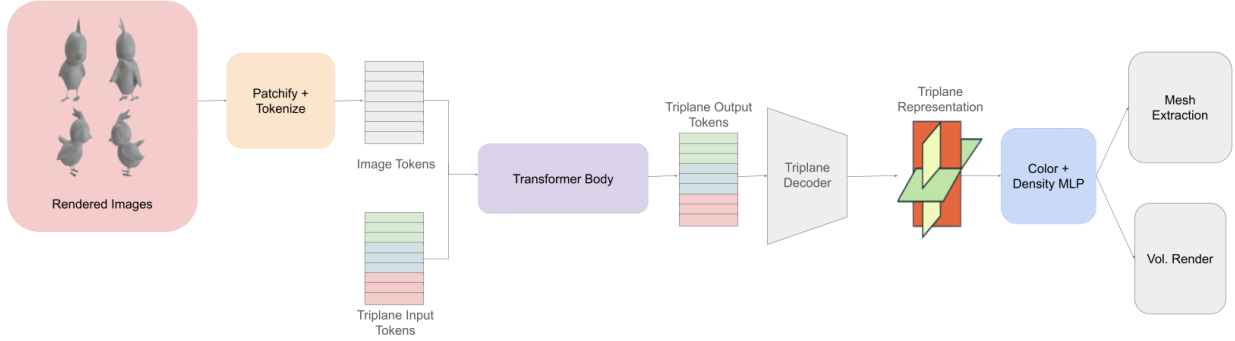


Figure 4.1: **Large Reconstruction Models:** A typical LRM tokenizes posed images and learns a triplane representation, which can be sampled and decoded into a color and density field. A mesh may be extracted from this field or the field may be volumetrically rendered.

independent multiview editing. The model receives a set of masked renders together with a single clean conditioning image used to infer the missing geometric content.

In contrast to optimization methods, reconstruction is performed through a single forward pass, producing substantially faster and more predictable edits. Because the model is explicitly trained for masked geometric reconstruction, users can control edits directly through the conditioning image while preserving consistency with the original shape outside of the masked region. By conditioning reconstruction on an edited view while propagating that information consistently across all views, the model mitigates many of the ambiguity and multiview consistency issues present in prior reconstruction pipelines.

A central challenge is designing a training procedure that encourages the model to propagate conditional information across all viewpoints. To accomplish this, we introduce a new 3D masking strategy in which occlusions are generated by renering randomly generated explicit geometry. Supervising both occluded and visible regions with multiview reconstruction targets encourages the model to jointly infer the missing geometry while accurately reconstructing the visible region.

Unlike previous works such as NeRFiller [170] which used fixed masks for each scene, training with randomly generated masks allows our model to generalize to arbitrary shapes and test-time masks. We demonstrate that this training method allows our model to be

used downstream for editing tasks while maintaining strong quantitative performance on reconstruction baselines. By manually defining an editing region analogous to the train-time occlusions, and using a single edited canonical view, users can generate edits that remain faithful to both the original geometry and the edited content.

4.2 Related Works

4.2.1 *Large Reconstruction Models*

LRM [57] and its recently introduced variants [89, 176, 166, 171, 190, 175, 52, 158, 12] showcase the solid capabilities of the transformer architecture for sparse reconstruction. Trained on large-scale 3D [30, 31] and multi-view image datasets [187], these models reconstruct geometry and texture details from sparse inputs or a single image in a feed-forward manner. A typical LRM architecture is shown in Figure 4.1. Most LRMs focus on reconstructing radiance fields, which cannot be consumed by standard graphics pipelines for editing. MeshLRM [171] and InstantMesh [175] extract mesh and texture maps, but it remains a challenging to perform shape editing in an intuitive and fast manner. Furthermore, while these models achieve quite high reconstruction quality when given at least four complete views as input [89, 190], the problem is much more ambiguous when given only a single (possibly edited) image [57]. In this chapter, we investigate how to utilize the LRM representation power for 3D shape editing, given a handful incomplete views as input for the shape reconstruction. This makes the reconstructed geometry of the non-edited content match significantly better to the original geometry, while ensuring view-consistency of the edited parts.

4.2.2 *Shape Editing*

Editing 3D shapes has been an active area of research for at least four decades. Early works focused on deformation [140, 27], cut and paste [128, 10], Laplacian surface editing [36, 95, 148] or fusion [71]. Recent works have tackled this task from different viewpoints depending on the geometry representation, the losses, and the downstream application. Regarding representation, research has been conducted on implicit surface editing [54, 25], voxels [141], NeRF [50, 170, 188, 17, 5, 67, 70] and Gaussian splatting [169, 20, 84], and mesh-based deformations as discussed in Chapter 2 [41, 78, 137, 7]. Other recent approaches build upon progress in LRMs performing multi-view editing using diffusion models and then using LRMs to reconstruct an edited shape [8, 123, 34]. In contrast, our work introduces a novel architecture trained on multi-view consistent masked data that bypasses the need for inconsistent diffusion editing and enables 3D mesh editing within seconds.

4.2.3 *Masked Vision Models*

The original Denoising Autoencoder [163] used random noise as a “masking” method on images during training with the objective of learning better semantic representations. More recently, methods using transformers convert images into sequences and predict unknown pixels [6, 19] which culminated in the development of the Vision Transformer (ViT) [32] as the backbone of modern masked image models. Models like the Masked Autoencoder [51] use ViTs to process randomly masked tokens, where every token represents a distinct, non-overlapping image patch. Research in diffusion models which also uses random noise as a “masking” procedure, has exploded in popularity, producing increasingly impressive generated images. By taking random Gaussian noise and constraining it to a specific region, diffusion models can be used for image inpainting [87, 28]. Masked autoencoders have been built for 3D data types such as point clouds [116, 191, 69], meshes [93], and NeRFs [62], with each work developing a different way to “patchify” their respective 3D representations.

Point clouds have the most natural representation for next token prediction [151, 113], while efforts have also been made into tokenizing triangle meshes for generation as sequences of vertices and faces [111, 154]. Our paper presents a new approach to combine masking with LRMs for editing.

4.3 Masked LRMs

Our large reconstruction model, shown in Figure 4.2, reconstructs a 3D shape from input images. Specifically, the model maps a sparse set of renders from various viewpoints into a latent triplane representation. We sample this representation to obtain latent features at different 3D points, which are then decoded into distance and RGB values for volumetric rendering. At training, we predict output renders from arbitrary camera poses. During inference, we use marching cubes to produce the reconstructed geometry. Unlike existing LRMs, our model uses a conditional branch to accept an additional view of the target shape. The inputs are then corrupted by a random masking procedure, forcing the model to learn to “inpaint” the input views using the conditional branch signal.

4.3.1 Masked LRM Architecture

Image and Pose Tokens The raw input to our model is a set of images with known camera parameters. During both training and inference, the input shapes are oriented in an arbitrary manner. Since we cannot guarantee a canonical viewpoint in the data, we remove the dependence on absolute poses by computing all camera parameters relative to the first randomly selected image which we use as the conditional input. These camera parameters are represented by Plücker rays, forming a 6-channel grid with the same spatial dimensions as RGB pixels. We apply the standard ViT tokenization [32] to the image and the Plücker rays independently, dividing both grids into non-overlapping patches, and linearly projecting them into a high-dimensional embedding.

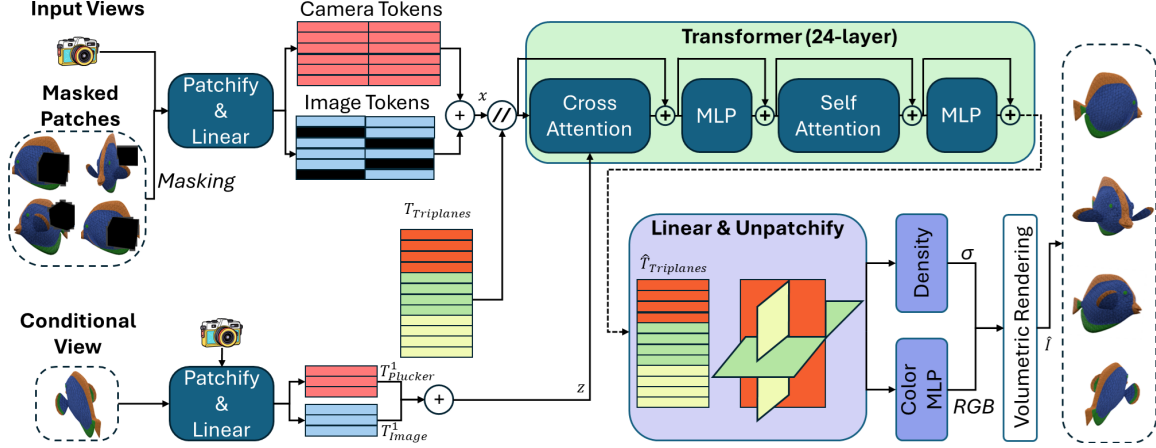


Figure 4.2: **Training Pipeline:** The images and camera poses are patchified and projected into tokens. A random 3D mask is generated and tokens corresponding to occluded patches are replaced by a learnable mask token. Camera and image tokens are summed and concatenated with learnable triplane tokens to form the transformer input. A clean conditional image is tokenized, forming the cross-attention input. The output triplane tokens are up-sampled and decoded into colors and SDF values, which are transformed into densities for volumetric rendering.

Masking After the input images are tokenized, we randomly select a subset of tokens to mask out. For general masked image modeling, [51] demonstrated that dropping out random patches from the encoded image enabled a desirable balance between reconstruction and learned representation quality. However, since our goal is to train a model that fills in the missing geometry from the content of a single clean view, it is not suitable to occlude random patches since they lack correspondence for each input view. Instead, we require a structured, *3D-consistent* form of occlusion.

Specifically, we generate a 3D rectangular mesh with uniformly random side lengths. We then render the depth map of this mesh from the same cameras as the input images, obtaining a set of multi-view consistent occlusions. Patches containing pixels that would be occluded by this random mesh are masked out. Instead of dropping the masked patches entirely as in [51], we propose to replace them with a learnable token. This does not suffer the same train-test gap, as occluded images are passed to the model during inference as well. This allows the model to maintain the 3D spatial context of the occlusion. Hence, our

masking strategy is specifically designed with downstream editing of an occluded shape in mind.

Model Formulation Using the above input tokenization and masking procedures, we can write a complete description of our model. Let $\mathcal{S}$ be a shape rendered from n camera poses described by the Plücker ray coordinates $\{\mathbf{C}_i\}_{i=1}^n$ producing RGB renders $\{\mathbf{I}_i\}_{i=1}^n$. The input token sequence to our model for any image are given by:

$$T_{\text{Image}}^i = \mathbf{PatchEmbed}(\mathbf{I}_i), T_{\text{Plücker}}^i = \mathbf{PatchEmbed}(\mathbf{C}_i), \quad (4.1)$$

where **PatchEmbed** is the operation of splitting images into non-overlapping patches and applying a linear layer to the channels.

We reserve T_{Image}^1 and $T_{\text{Plücker}}^1$ for the clean conditional signal. Now, we sample a random rectangular mesh $\mathcal{O}$ and render it from the same camera poses as $\mathcal{S}$. Comparing the depth maps of $\mathcal{S}$ and $\mathcal{O}$, we produce modified tokens $\tilde{T}_{\text{Image}}^i$ where the token for any patch that contains an occluded pixel is replaced by a learnable mask token. Then, the input tokens to the model are constructed as:

$$\mathbf{x} = \left\| \right\|_{i=2}^n (\tilde{T}_{\text{Image}}^i + T_{\text{Plücker}}^i), \quad \mathbf{z} = T_{\text{Image}}^1 + T_{\text{Plücker}}^1, \quad (4.2)$$

where z is the condition passed to the model and $\|$ the iterated concatenation operation. We choose to add the Plücker ray tokens after masking such that the model can differentiate between different occluded patches. Note that adding a Plücker ray token for each patch means the model does not need a positional embedding to differentiate patches.

We use three sequences of learnable tokens $\mathbf{T}_{\text{Triplanes}} = \mathbf{T}_{xy} \| \mathbf{T}_{yz} \| \mathbf{T}_{xz}$ to produce the predicted triplanes. These tokens are passed to the transformer body of the model, which comprises of iterated standard cross-attention and self-attention operations equipped with MLPs and residual connections:

$$\hat{\mathbf{x}}, \hat{\mathbf{T}}_{\text{Triplanes}} = \mathbf{Self-Att} \left(\mathbf{Cross-Att} \left(\mathbf{x} \| \mathbf{T}_{\text{Triplanes}} \right), \mathbf{z} \right), \quad (4.3)$$

with $\mathbf{x}$ and $\mathbf{T}_{\text{Triplanes}}$ coming from the previous transformer block (or the input). Finally,

we upsample each triplane token to a patch using a single layer MLP, evaluate the learned triplanes at densely sampled points, and decode the latents using MLPs. We obtain predicted images and $\hat{\mathbf{I}}$ pixel-ray opacity maps $\hat{\mathbf{M}}$ through volumetric rendering, and normal maps $\hat{\mathbf{N}}$ by estimating normalized SDF gradients:

$$\begin{aligned}\mathbf{Triplanes} &= \mathbf{MLP}_{\text{Upsample}}(\hat{\mathbf{T}}_{\text{Triplanes}}) \\ \mathbf{SDF}(x, y, z) &= \mathbf{MLP}_{\text{Distance}}(\mathbf{Triplanes}(x, y, z)) \\ \mathbf{RGB}(x, y, z) &= \mathbf{MLP}_{\text{Color}}(\mathbf{Triplanes}(x, y, z)) \\ \sigma &= \mathbf{Density}(\mathbf{SDF}) \\ \hat{\mathbf{N}} &= \mathbf{NormGrad}(\mathbf{SDF}) \\ \hat{\mathbf{I}}, \hat{\mathbf{M}} &= \mathbf{VolRender}(\sigma, \mathbf{RGB})\end{aligned}$$

where we convert the SDF values to densities σ for rendering following [181]. The learned image tokens $\hat{\mathbf{x}}$ are not used for any remaining task and are thus discarded.

4.3.2 Supervision

Our LRM is trained with L2 reconstruction and LPIPS perceptual losses. Given a ground truth image $\mathbf{I}$, normal map $\mathbf{N}$, and binary silhouette mask $\mathbf{M}$, we use the following losses:

$$\mathcal{L}_{\text{Recon}} = w_I \|\hat{\mathbf{I}} - \mathbf{I}\|_2^2 + w_N \|\hat{\mathbf{N}} - \mathbf{N}\|_2^2 + w_M \|\hat{\mathbf{M}} - \mathbf{M}\|_2^2$$

$$\mathcal{L}_{\text{Percep}} = w_P \mathcal{L}_{\text{LPIPS}}(\hat{\mathbf{I}}, \mathbf{I})$$

where w_I, w_N, w_M, w_P are tunable weights, and $\mathcal{L}_{\text{LPIPS}}$ is the image perceptual similarity loss proposed in [192]. For the results in this paper, we choose simply $w_I = w_M = w_P = 1$ and $w_N = 0$ or 1 depending on the stage of training.

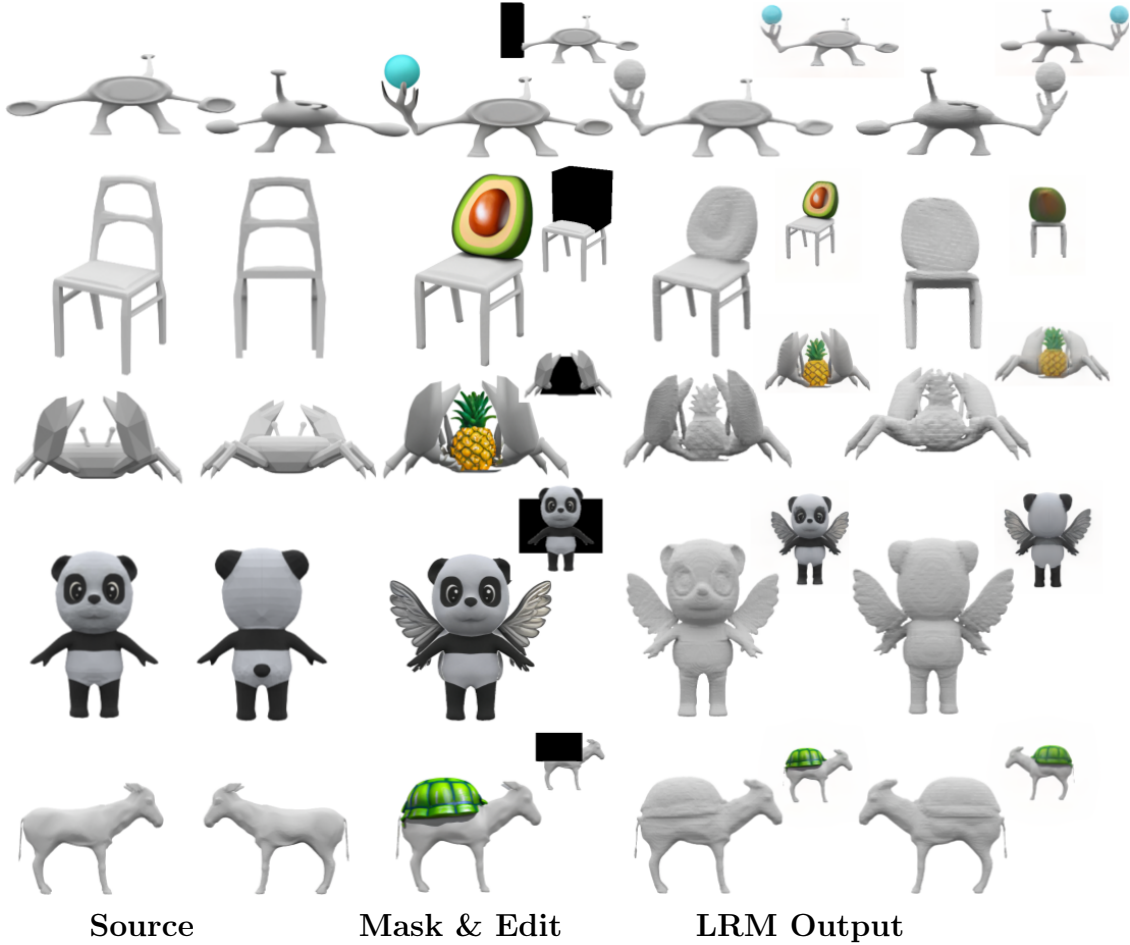


Figure 4.3: **Mesh Editing using MaskedLRMs:** The inputs comprise front/back views of the source mesh (1^{st} column) and an edited frontal image used as the conditional view. The 2^{nd} column shows the masked area rendered from the front (inset) and a 2D edit. The last column shows our generated mesh from the front/back views.

4.3.3 Training Stages

Our model is trained in stages following [171], for training efficiency. However, the purpose of our stages differ. Since fully rendering 512×512 output images is computationally expensive, for every stage, we sample a random 128×128 crop from each output image to use as supervision. We maintain the full images for the input throughout every stage.

In stage 1, we downsample the output images to a 256×256 resolution, allowing the random crops to supervise 25% of the image. We use 128 samples per ray for the volumetric rendering. In this initial stage, we observe that the geometric supervision from the normal

maps is not yet necessary, so we drop this portion of the reconstruction loss by setting $w_N = 0$ enabling a more efficient backwards pass.

In stage 2, we downsample the output images to 384×384 , meaning that the random crops now only supervise 11% of the image and increase the samples per ray to 512. By increasing the rendering fidelity and decreasing the proportion of the image supervision, we train the model to focus more sharply on geometric details. We observed that without any geometric supervision, the LRM may produce degenerate solutions by generating textures that hide geometric artifacts. Thus, we introduce the normal loss by setting $w_N = 1$.

4.3.4 *Mesh Editing*

Since our LRM is trained with 3D-consistent, multi-view occlusions, using the conditional branch to complete the partial observations, it is straightforward to use it for shape editing. Given a shape $\mathcal{S}$, we manually define an occlusion $\mathcal{O}$ that occludes the region of interest for editing. Then, we edit a representative image within the pixels that are occluded from its camera viewpoint. This may be done a variety of ways – for our results, we use a text-to-image masked diffusion method [87, 28]. The image edit is used as a conditional signal, while the rest of the occluded images are fed to the main transformer body of the LRM. The LRM is trained to inpaint the occluded region using the content from the conditional branch, and as such it propagates the 2D edit into 3D space. Figure 4.3 shows examples of this process.

This approach to shape editing is much faster than optimization-based methods (see Table 4.2), requiring only a single forward pass to lift the desired edit into 3D. Our model also produces more realistic shapes since it is trained on a large collection of scanned objects instead of relying on diffusion model guidance and complex regularizations. It is also more expressive than optimization-based methods as it can generate arbitrary geometry based on the input condition. For example, it can change the geometric genus of a shape (adding a handle or a hole as in Figure 4.7), which deformation-based optimization methods cannot

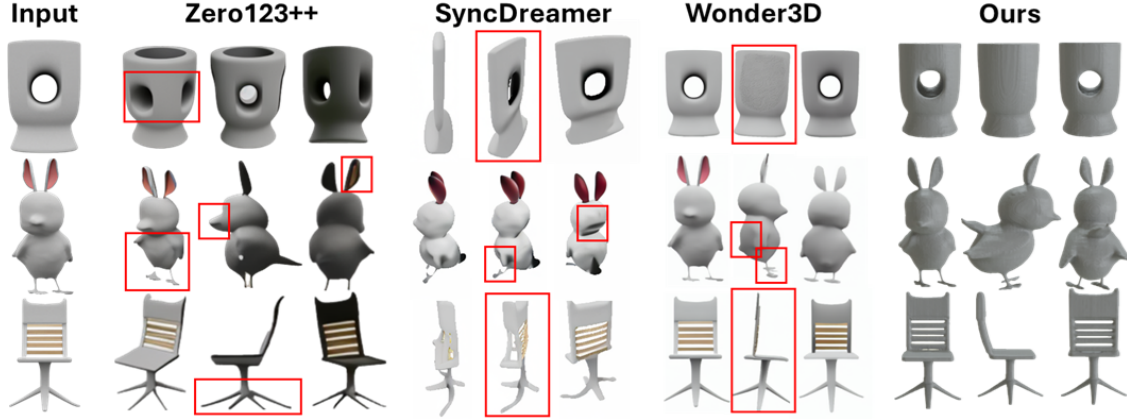


Figure 4.4: **Multi-view Diffusion vs MaskedLRM**: Multi-view diffusion models must infer occluded geometry from the input which leads to artifacts in the multi-view images (incorrect/distorted views). MaskedLRM which does not use multi-view diffusion, receives both the edited image and multi-view information, allowing it to bypass this problem and reconstruct the correct geometry.

do as genus changes are not differentiable.

Generative methods using LRMs such as InstantMesh [175] rely on methods such as Zero-123++ [145] to generate multi-view images, introducing view-consistency artifacts. In Figure 4.4 we show examples of such artifacts generated by recent models. Zero-123++ hallucinates additional holes in the vase, and generates a distorted and incorrect bird anatomy. SyncDreamer [100] generates unrealistically distorted views, such as a completely flattened vase, poor bird anatomy, and a warped chair. Wonder3D [101] is better, but it cannot capture the correct bird anatomy and chair structure. In contrast, our model requires only a single view as conditioning and uses the prior from the dataset to construct the shape in a consistent manner. Some recent concurrent work tackles editing directly in the multi-view image space. While this also handles ambiguity, we show in Figure 4.6 that our method produces more realistic edits.

4.4 Experiments

Training Data We train our Masked LRM on a the Objaverse dataset [30] containing shapes collected from a wide variety of sources. Each shape is normalized to fit within a sphere of fixed radius. Our training data consists of 40 512×512 images of each shape, rendered from randomly sampled cameras. We also render the corresponding depth and normal maps for these camera poses. Every iteration, the model inputs and reconstruction targets are chosen randomly from these pre-fixed sets of images.

Evaluation We evaluate the reconstruction quality of our model on the GSO [33] and ABO [26] datasets and compare the state-of-the-art MeshLRM [171]. Since MeshLRM cannot be easily repurposed for our editing task, we also compare the reconstruction quality with InstantMesh. We use PSNR, SSIM, and LPIPS on the output renders from novel poses as metrics. To remain consistent with the training setting, we randomly generate a rectangular mask to occlude the input views and provide a different clean view as conditioning for our method. Finally, we qualitatively demonstrate the main contribution of our model, the ability to propagate 2D edits from a single viewpoint into 3D. We compare our results to prior works for text and image based 3D generation.

4.4.1 Quantitative Comparisons

Table 4.1 shows novel-view synthesis metrics of our method when compared to InstantMesh and MeshLRM. Since our main goal is to edit existing shapes and not to completely generate shapes from scratch, we choose to train our model by randomly selecting 6-8 input views along with one conditional view, giving our LRM a denser input than MeshLRM. We show metrics for both 6 and 8 input views and we compute those on another set of 10 camera poses, different from the input poses.

Our method is competitive with the state-of-the-art model on reconstruction, achieving

Method	ABO Dataset			GSO dataset		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
InstantMesh [175] (NeRF)	-	-	-	23.14	0.898	0.119
InstantMesh [175] (Mesh)	-	-	-	22.79	0.897	0.120
MeshLRM [171]	26.09	0.898	0.102	27.93	0.925	0.081
Ours (6 views)	28.37	0.946	0.081	27.24	0.931	0.088
Ours (8 views)	28.65	0.947	0.078	27.58	0.933	0.085

Table 4.1: **Quantitative Evaluation:** We evaluate our model using test-set shapes and compare it to the state-of-the-art LRM and InstantMesh on the ABO and GSO shape datasets, reconstructing the meshes from 6 and 8 posed images. Despite direct reconstruction of new shapes not being our main goal, and our masking introducing extra difficulty in the task, we still achieve better than SoTA metrics on ABO, and comparable to SoTA metrics on the GSO dataset.

a 2.56 PSNR improvement on the ABO dataset, and a comparable PSNR on GSO. We observe the same phenomenon in perceptual quality measured by LPIPS, where our method significantly outperforms on ABO shapes, and is comparable on GSO shapes. As expected, using 6 views under-performs using 8 views, but only by a slight margin. Furthermore, our method significantly outperforms InstantMesh. This is to be expected, as InstantMesh infers everything from a single view, while both MeshLRM and MaskedLRM access multi-view information.

Our model achieves performance on par with SoTA on reconstructing a diverse set of output poses, indicating that it has learned to effectively “in-paint” the randomly occluded regions in the input views using context from the available unoccluded signal. Since our end goal is mesh shape editing, it is *not critical that we surpass the reconstruction quality* of prior works, as we only need to ensure a high quality baseline for the output geometry. We further demonstrate qualitatively in Sec. 4.4.3 that our model indeed learns to inpaint using the conditional signal, instead of only the context from multi-view images, thereby accomplishing feed-forward shape editing through a single view.

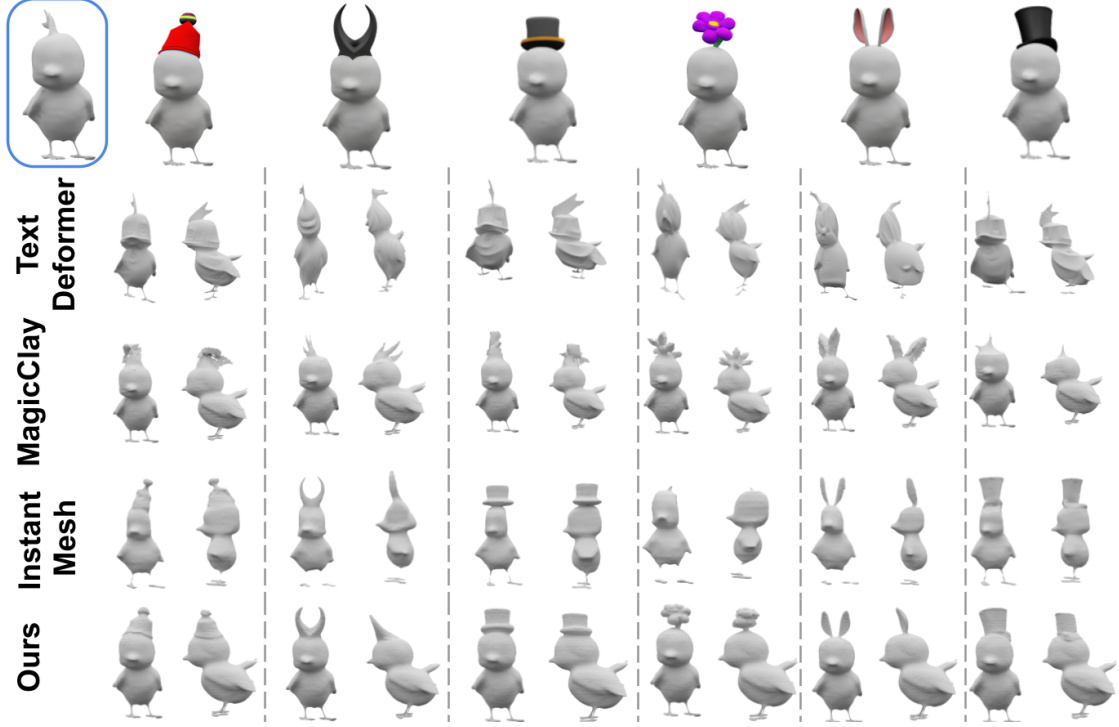


Figure 4.5: **Mesh Editing Comparisons:** Given a mesh (top left) and various image edits as guidance we demonstrate that our approach is the only one that generates multi-view consistent shape edits that follow the guidance. Colors omitted to clearly visualize the edited geometry.

4.4.2 Qualitative Evaluations

Using a bird mesh generated from a text-to-3D model, and several editing targets, we compare our method against other 3D editing methods. Full results are shown in Figure 4.5. We define a masked region to edit on the head of the bird (omitted from the figure for brevity). Conditional signals provided to our method generated by masked diffusion are shown in the 1st row, and our results are in the last row.

Optimization methods In the 2nd and 3rd rows of Figure 4.5, we show the results of two text-based mesh optimization methods. Instead of using the edited images themselves, we use the text prompts that we passed to the diffusion model as guidance. The first optimization-based method we compare to (2nd row) is TextDeformer [41] which uses a Jacobian representation [1] for deformations to constrain the edits instead of explicit lo-

calization. TextDeformer struggles with the highly localized nature of our task and globally distorts the mesh, failing to produce an output of acceptable quality in all examples.

We also compare with MagicClay [7], which optimizes both an SDF and a triangle-mesh representation. It also optionally uses a manual “seed” edit so that the optimization task is easier. However, since this requires an additional layer of modeling expertise (i.e., a manual user intervention) our method does not require, we opt out of this step for our comparison. Unlike TextDeformer, MagicClay selects a subset of vertices to deform to combat noisy SDS [121] gradients. Since we have a 3D mask, we simply choose the vertices that lie within that region. Although this selection serves to localize the editing process, we observe that the deformations are still noisy. While sometimes MagicClay edits are semantically correct (flower and rabbit ears), in other cases such as the fedora (3rd column) and top hat (6th column), the optimization process collapses completely.

In both cases, noisy gradients from text-guidance result in results in optimizations that are both unpredictable and uncontrollable. In contrast, the output of our LRM is highly predictable from the selected conditional view, which may be re-generated until desirable or even manually edited.

LRM-Based Methods We compare our method against recent top-performing methods that combine multi-view diffusion and reconstruction models. InstantMesh [175] is one such pipeline that may be used for shape editing. It relies on using Zero-123++ [145] to generate multi-view images from a single view and then passing these images to an LRM. To edit, we simply pass an edited image of the original shape. As shown in the 4th row of Figure 4.5, this results in a poorly reconstructed shape that is particularly thin when compared to the ground truth and the output quality suffers due to the inability of Zero-123++ [145] to generate faithful multi-view images, as discussed in Section 4.3.4. Methods that rely directly on a separate diffusion module to generate the LRM inputs will run the risk of generating artifacts from inaccurate multiview generation. In comparison, our method does not suffer

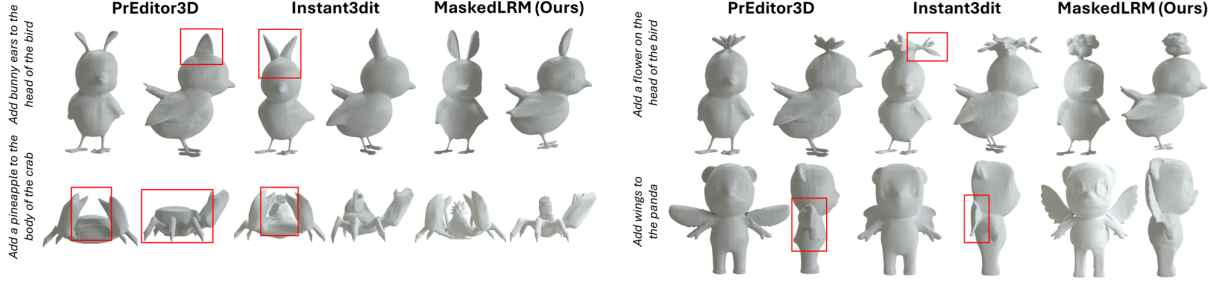


Figure 4.6: **Mesh Editing Comparisons with Concurrent work:** We compare our approach to concurrent work enabling localized 3D editing. While localized approaches better preserve the original structure of the shape, other methods are not able to produce edits as realistic as ours.

any such reconstruction artifacts since it uses trivially consistent ground truth renders as the main LRM inputs.

In Figure 4.6 we also compare to two concurrent works namely PrEditor3D [34] and Instant3Dit [8] that tackle localized editing using multi-view diffusion via text prompting¹. PrEditor3D [34] performs multi-view editing by first inverting a multi-view diffusion model to obtain Gaussian noise images, and then using the forward process to edit the desired region with a separate text prompt. While PrEditor3D generates semantically correct edits based on the prompts, it produces undesirable artifacts in several examples. In particular, many of the edits lack detail, such as the bunny ears in the top left and the wings in the bottom right. It also fails to produce a pineapple body in the bottom left.

Instant3Dit [8] uses masking in multi-view diffusion training instead of LRM training. They use their text-prompted multi-view inpainting model to edit and then use an LRM to generate an edited mesh. Similar to PrEditor3D, it produces semantically correct edits that lack realism due to artifacts. Instead of producing sharp bunny ears, Instant3Dit is only capable of adding vague pointed structures to the bird. In the second column, the flower it generates is plant-like but unrealistic. In the bottom row, we see that the pineapple and wings are again semantically correct but lacking in detail.

¹. While the code is not yet available, the authors of these works offered to run their method on examples we provided which enabled these comparisons.

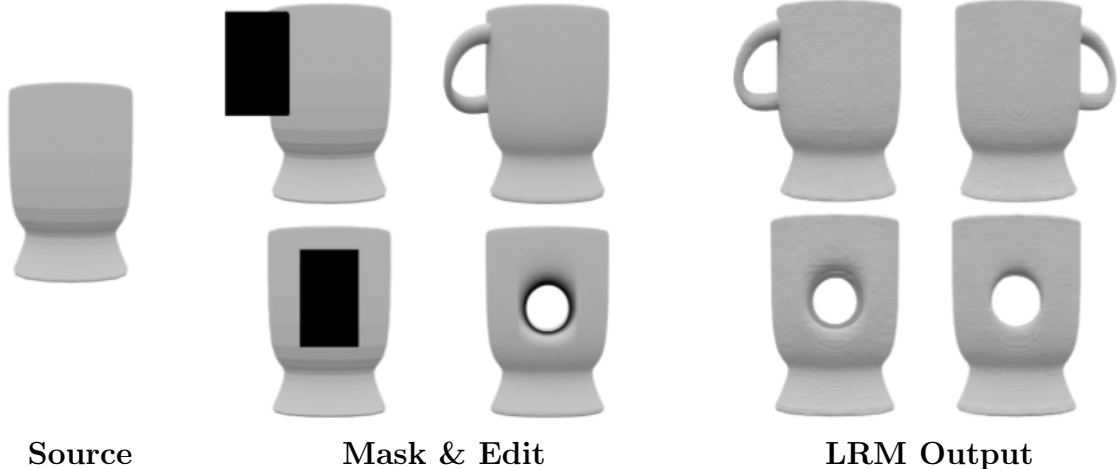


Figure 4.7: **Genus changes:** Our method unlocks genus-changing edits like adding a handle or a hole to the original vase. We show the output of our model from 2 opposing views in the 3rd column.

4.4.3 Mesh Editing Characteristics

In Figures 4.3 and 4.7, we show mesh editing examples demonstrating the capabilities of our method. The 1st column shows the source mesh rendered from different viewpoints. The 2nd column shows the edited conditional image with a render of the original masked region inset. Our LRM accepts the edited view along with a set of occluded ground-truth renders (omitted from the figure) and predicts an SDF. The last column shows the mesh extracted from the output while the insets depict volumetric renders of the predicted SDF.

Expressiveness The edits throughout this paper show the expressiveness of our method. The meshes used in Figures 4.5 as well as in row 1 of Figure 4.3 are examples of non-standard shapes – a unique bird mesh generated from a text-to-multiview diffusion model [155] and a “Tele-alien” from the COSEG [168] dataset. Despite being novel shapes, our model is able to give the alien a staff and the bird a hat. The other four rows of 4.3 consist of edits that are “unnatural” – creating an avocado backrest, replacing the body of a crab with a pineapple, giving a panda wings, and giving a donkey a turtle shell. In every example, our method successfully translates the 2D edit into geometry in a realistic manner.

The edits in Figure 4.7 show a critical benefit of our method. Since the final mesh is

constructed entirely from the output of a neural network, there are no geometric restrictions on the type of edit we can do. The last two rows demonstrate the ability of our network to change the genus of a shape by adding a handle or a hole through the middle, which would be impossible for geometry optimization-based methods.

Identity Preservation Although our model discards the initial shape in order to bypass editing limitations, we observe that the LRM still achieves highly faithful reconstructions of the initial geometry outside of the region of interest. This confirms our quantitative observations that our method has near-SoTA reconstruction quality. This also indicates that, due to multi-view masking, our method is able to constrain the edit inside selected 3D region without needing to perform expensive learning over explicit 3D signals.

4.4.4 Ablations Studies & Discussion

Geometric Supervision We investigate the effect of using geometric losses during training. Figure 4.8 compares three LRMs: a model trained by the pipeline described in Sec. 4.3, a model trained with no geometric supervision, and a model trained by replacing the normal map loss with a depth map loss.

Using no geometric supervision results in poor surface quality, highlighted in the red boxes. Since the main training objective is multi-view image reconstruction, the model hallucinates correct geometry using colors, without producing an accurate surface. Supervising the predicted depth somewhat mitigates this issue, but the effect is weak and the surfaces are still incomplete. Normal map supervision gives high quality surfaces as shown in the green boxes.

Random Masking In order to justify our masking strategy, we train our model masking patches uniformly randomly instead of using 3D occlusions. Figure 4.9 compares meshes extracted from a model trained using our 3D masking versus masking 25% of patches uniformly at random. We observe that uniformly random patch masking can still generate

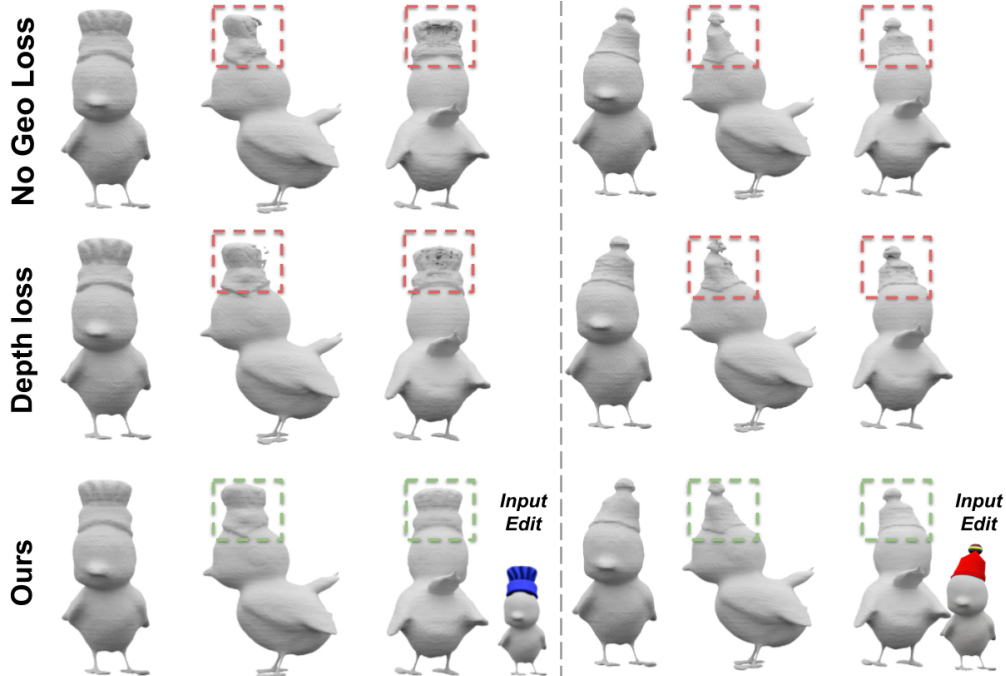


Figure 4.8: **Impact of Geometric Supervision:** Geometric losses are critical to produce high-quality surfaces. No geometric loss (top) causes severe hole and bump artifacts in the mesh. Depth loss (middle) is not as effective as normal loss (bottom) which allows our model to generate accurate and smooth reconstructions.

“roughly correct” shapes, especially adding a moustache to the face in the last row. This is because we add camera pose embeddings after masking, so the model can differentiate masked and non-masked tokens, regardless of their distribution within the image. Furthermore, the reconstruction outside of the masked region is still accurate. However, there still exists a train-test gap between random patches and contiguous patches created by selecting an editing region, which causes significant artifacts in the other three examples. In the first and second rows, we observe a blurring artifact, where the model cannot generate sharp features in the horns on the bird and between the slats of the chair. In the third row, using random patches causes the shape of the turtle shell to be malformed. In comparison, using our masking method produces accurate and sharp geometry in all examples.

Runtime In Table 4.2 we provide performance comparisons between our approach and several top-performing recent works. Our method is not only much faster than optimization-

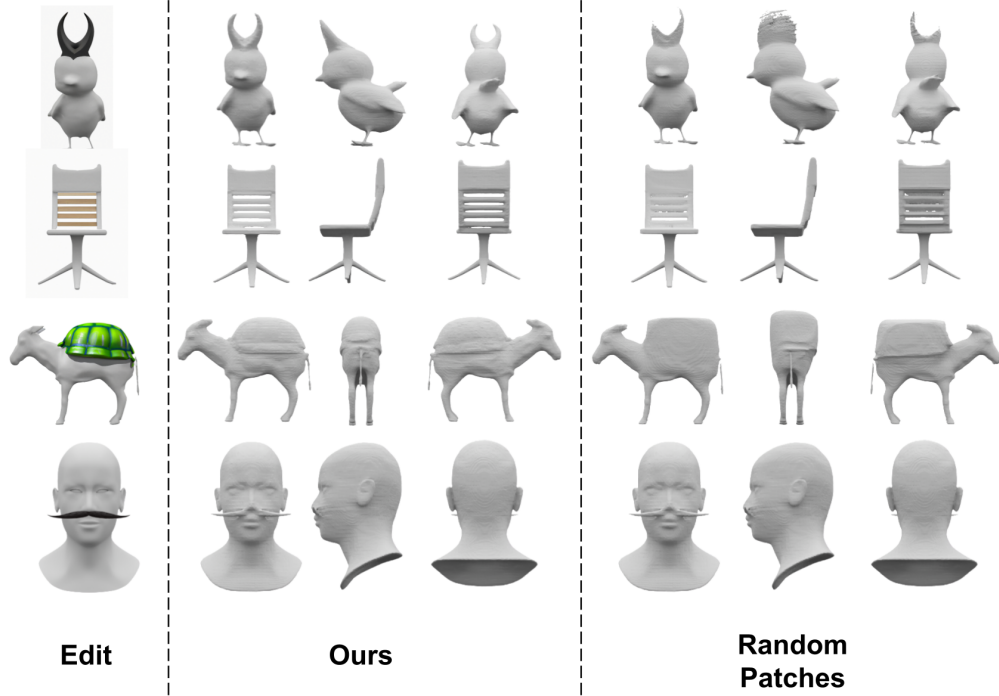


Figure 4.9: **Impact of Random Masking:** We test our choice of masking strategy by comparing it to masking 25% of patches uniformly at random. The left column shows the conditional image, the middle section shows our results, and the right section shows the results using uniformly random masking. While the model is still capable of generating correct geometry, there is a train-test gap in the masked patches since we define a contiguous 3D region to mask during inference. Thus, the model produces artifacts such as lack of sharp features (in the bird horns and chair slats) in rows 1 and 2, and overall incorrect shape in row 3 (square turtle shell).

based approaches [7, 41] as it requires only one forward pass, but it also outperforms LRM approaches [175, 8] that make use of a multi-view generation model. PrEditor3D [34] requires the forward pass of several large pre-trained models [129, 99] resulting in a longer runtime.

4.5 Discussion

In this chapter we presented our work in [42] to perform 3D shape editing via large-scale pre-trained models. Our method builds upon recent progress in Large Reconstruction Models by introducing a multi-view input masking strategy that trains the reconstruction model itself to perform geometric completion. Our LRM is trained to “inpaint” the masked region using

	Optimization-based		LRM-based			
	TextDeformer	MagicClay	InstantMesh	Instant3dit	PrEditor3D	Ours
Runtime ↓	~20mins	~1hour	30sec	~6sec	80sec	< 3sec

Table 4.2: **Runtime Comparison:** Our method is significantly faster than optimization methods as it is feed-forward and also faster than LRM-based approaches that must run multi-view diffusion.

a clean conditional viewpoint to reconstruct the missing information. At inference time, a single edited conditioning image can guide reconstruction of the edited geometry through a single forward pass. This formulation enables fast and controllable editing without requiring iterative optimization or explicit 3D modeling expertise.

However, the model remains constrained by the quality and expressiveness of the conditioning edit. While text-to-image models can create a wide range of results, precisely producing a desired edit may still require substantial prompt iteration. Our method is limited both by Marching Cubes triangulation quality, and the resolution of the reconstruction model which makes performing edits that require extremely intricate details challenging. Fine geometric details, in regions such as faces, may reconstruct imperfectly and exhibit blurry artifacts. Furthermore, while our model accurately reconstructs geometry, it replaces the triangulation with one derived from Marching Cubes, which may be undesirable. Recent work such as [34] explores techniques to merge edited 3D regions with the original shape, which may be leveraged to alleviate this issue.

CHAPTER 5

SCENE DECOMPOSITION WITH MANIPULABLE COMPONENTS

5.1 Introduction

The previous chapters addressed editing at the level of individual objects, by embedding them in a high-dimensional space and using latent arithmetic, by optimizing vertex deformations, or by generating new content in masked regions using learned priors. In each case, the subject of the edit was a single, isolated shape, with the central task to produce a geometrically coherent modification given some form of semantic guidance. This chapter focuses on the multi-object scene setting, where the goal is to learn a representation that jointly reconstructs a scene while decomposing into object-level components. Rather than segmenting or grouping small primitives post-hoc, we work towards building a decomposition directly into the representation. By design, each primitive corresponds to a distinct object or part that can be manipulated.

Neural Radiance Fields [108] have achieved impressive quality in reconstructing 3D scenes from multiview images. However, these techniques treat the scene as a standalone object. NeRF encodes the entire scene into a single implicit function, providing no mechanism to interact with individual components. Subsequent works sought to make objects a structural component of the representation. Compositional NeRF methods [11, 174] decompose scenes into object-level fields, but editing and manipulation remain cumbersome relative to manipulating objects directly as primitives. Nerflets [193] represents scenes as local mixtures of small radiance fields, but still relies on oversegmenting the scene and post-hoc grouping. PartNeRF [156] is the most closely related prior work: it represents objects as local ellipsoidal NeRFs and achieves semantic alignment by training over large datasets of shapes within a category. However, applied to a more complex single scene, its geometric regularizers alone

provide no incentive for parts to align with meaningful structures.

In parallel, subsequent research on Gaussian Splatting has also added semantic embeddings or feature fields to the representation [182, 15, 198]. However, semantic grouping still requires post-hoc clustering of collections of small splats. Furthermore, editing selected groups with simple geometric transformations such as translating or scaling produces gaps and surface artifacts as each splat’s appearance was optimized to render at a fixed position, orientation, and resolution. Correcting these artifacts requires additional finetuning for each edit.

To address limitations within these paradigms, we look to design a framework that treats objects as first-class primitives. We opt to adopt the representation of a scene as a collection of Gaussian primitives, each equipped with an independent NeRF. Each Gaussian defines an ellipsoidal bounding volume with a mean and per-axis scale and rays intersecting a Gaussian are sampled within the intersection interval and queried against the local NeRF. A key design choice is the composite rendering scheme. Rather than a mixture of experts or a hard assignment for every ray, we use a mixed rendering formulation. Each Gaussian’s per-sample opacity is computed independently, but these opacities are composited jointly in a global depth-sorted order via the standard transmittance product. This scheme gives each Gaussian a semi-independent accumulation of rendering weight, preventing early optimization from starving later-appearing Gaussians, while still allowing late-stage object specialization.

When semantic labels are available, a cross-entropy classification loss on volume-rendered label composites lifts them into 3D. When per-pixel labels are unavailable, we investigate a label-free decomposition pipeline. Given a trained model over the whole scene, we voxelize the local coordinate frame of each Gaussian and construct a graph over occupied voxels. We build a graph by connecting neighborhoods of occupied voxels, with edge weights combining a geometric proximity term and a semantic term derived from accumulating DINOv2 features into the grid. Spectral clustering on this graph produces a volumetric partition of the

Gaussian into coherent subregions which may be used to spawn child primitive that are trained to represent objects in the scene.

We present experiments on instance-labeled Blender scenes, demonstrating our framework recovers one-primitive-per-object decompositions with no post-hoc processing and that scene editing operations such as translation, scaling, and cross-scene composition follow naturally directly as a single rendering operation. We also show preliminary results applying the label-free pipeline to NeRF-Synthetic scenes without semantic information, and initial experiments on real-world unbounded scenes from Mip-NeRF 360 [9].

5.2 Related Works

5.2.1 *Neural Radiance Fields*

NeRF [108] introduced differentiable volume rendering of scenes represented by a continuous radiance field parameterized by a neural network. This paradigm for scene representation and novel view synthesis became a foundation for a large body of research, with extensions addressing unbounded scenes [9], rendering speed [109], and generalization across datasets of scenes [186]. Another line of work decomposed scenes spatially into small local networks [130, 131] or per-block subnetworks [161, 153], and Switch-Nerf [105, 196] applies a learned mixture-of-experts routing network to NeRF training. These decompositions improve efficiency or scale but are not driven by scene structure. The representations do not align with object boundaries and do not produce semantically coherent units. In designing our method, we use the volume rendering representation of NeRF but move away from arbitrary decompositions, instead representing a scene as a small number of Gaussian-anchored primitives, each intended to correspond to a distinct object or part.

5.2.2 *Compositional and Editable Radiance Fields*

Several works sought to decompose NeRF scenes into object-level components [179, 11, 173, 174, 149], enabling object manipulation. Nerflets [193] represents a scene as a collection of local radiance fields each enhanced with a logit vector contributing to a semantic render. However, Nerflets still relies on over-segmentation and post-hoc grouping to recover coherent objects. PartNeRF [156] is another work in this line that is most closely related to our proposed method. It represents objects as a small set of local ellipsoidal NeRF parts, supporting part-level transformations. Our method shares this core representation, but differs in key aspects. PartNeRF’s primitives are regularized for coverage and non-overlap, but this is again purely geometric. Semantic coherence emerges because the authors opt to train an architecture across many shapes of the same category, requiring a curated dataset. Overfitting to single scenes, PartNeRF can produce many uninterpretable parts. Furthermore, their hard assignment of rays to single parts is incompatible with external semantic supervision. Unassigned parts receive zero gradient per ray, leading to unstable specialization even with explicit labels. In contrast, we design regularizers that encourage specialization while keeping assignment soft, allowing external supervision to shape the decomposition before ownership converges. Finally, we also use spectral clustering inside each primitive’s local coordinate frame to produce semantically meaningful decomposition for arbitrary objects without the need of annotations or datasets.

5.2.3 *3D Gaussian Splatting*

Gaussian Splatting [75] introduced a newer representation for scenes, replacing NeRF network queries with explicit, fast-rendering anisotropic Gaussian primitives, further advancing real-time rendering and reconstruction quality. Subsequent work distilled foundation-model feature fields into the representation [197] and embedded LLM features for open-vocabulary querying [124, 91, 92, 157]. Gaussian Grouping [182] added identity embeddings using Seg-

ment Anything [81, 24] supervision to enable control over explicit objects, while SAGA [15] added scale-gated affinity features to enable promptable, multi-granularity segmentation. ObjectGS [198] used object-aware anchors that generate neural Gaussians and explicit object IDs. Although these works enable varying degrees of scene control, they share a common design: the underlying representation consists of up to millions of tiny primitives, each optimized purely for appearance. Semantic grouping is a post-hoc operation and editing can be brittle: transforming selected groups of splats creates gaps and boundary artifacts, which may require further finetuning to fix.

5.3 Method

Our method represents a scene as a set of K Gaussians, each equipped with an independent neural radiance field (NeRF). Similarly to PartNeRF, each Gaussian is equipped with a mean and a scale that define a local coordinate system. A ray cast into the scene intersects a subset of these primitives and each NeRF is queried within these local intersections. However, in contrast with PartNeRF, the contributions of each Gaussian are composited via a modified volume rendering scheme that preserves per-Gaussian opacity while still using global depth sorting for transmittance. Unlike approaches that recover semantics by grouping several primitives together, our design enforces a strong spatial locality prior, making “objects” a first-class primitive with no post-processing required. A single Gaussian in our framework represents a distinct component of the scene.

5.3.1 Scene Representation

Gaussian Primitives We represent the scene with K ellipsoidal Gaussians parametrized by means μ_k and per-axis scales $s_k \in \mathbb{R}^3$ for $k = 1, 2, \dots, K$. Given a 3D point $\mathbf{x}$ in world

coordinates, its local coordinates with respect to the Gaussian k are given by

$$\mathbf{x}_k^{\text{loc}} = (\mathbf{x} - \mu_k) \oslash s_k$$

where $\oslash$ denotes the Hadamard division. The local Gaussian density of $\mathbf{x}$ is then

$$\rho_k(\mathbf{x}) = \exp\left(-\frac{1}{2}\left\|\mathbf{x}_k^{\text{loc}}\right\|^2\right)$$

Rays are intersected with these Gaussians via an analytical ellipsoid intersection. A ray $\mathbf{o} + t\mathbf{d}$ intersects Gaussian k within a radius threshold r when the discriminant of the quadratic $at^2 + 2bt + c = r^2$ is positive, where

$$\begin{aligned} a &= \|\mathbf{d} \oslash s_k\|^2 \\ b &= \langle \mathbf{d} \oslash s_k, (\mathbf{o} - \mu_k) \oslash s_k \rangle \\ c &= \|(\mathbf{o} - \mu_k) \oslash s_k\|^2 \end{aligned}$$

The intersection interval $[t_k^{\text{enter}}, t_k^{\text{exit}}]$ is given by the two roots of this quadratic and defines the sampling interval for the local NeRF.

Per-Gaussian NeRFs Each Gaussian k is equipped with a NeRF denoted by ϕ_k . The NeRF receives normalized coordinates $\mathbf{x}_k^{\text{loc}}/r$ as the positional input along with a normalized viewing direction $\mathbf{d}$. The normalization of the location input restricts it to $[-1, 1]$ and makes each NeRF’s input location distribution invariant to the world-geometry of its respective Gaussian. Following the standard NeRF architecture, both inputs are encoded using a Fourier feature embedding using L log-space frequency bands before being passed to the network. The predicted density σ_k uses only view-independent layers ϕ_k^σ while $\mathbf{d}$ is injected into a small subnetwork that predicts color. The final volumetric density for a Gaussian k at a world-coordinate $\mathbf{x}$ is given by the product

$$\sigma_k(\mathbf{x}) = \gamma \cdot \rho_k(\mathbf{x}) \cdot \phi_k^\sigma(\mathbf{x}_k^{\text{loc}}/r)$$

where γ is an appropriately chosen hyper-parameter.

5.3.2 Volume Rendering

Given a ray $\mathbf{o} + t\mathbf{d}$, a standard NeRF samples N points along the ray given by $x_i = \mathbf{o} + t_i\mathbf{d}$. Then the standard volume integral accumulates color along the ray as a weighted sum over these samples:

$$\begin{aligned}\alpha_i &= 1 - \exp(-\sigma_i \Delta t_i) \\ T_i &= \prod_{j=1}^{i-1} (1 - \alpha_j) \\ \hat{c} &= \sum_{i=1}^N T_i \alpha_i c_i\end{aligned}$$

where c_i and σ_i are network predictions of color and density respectively for coordinates x_i and $\Delta t_i = t_{i+1} - t_i$ is the step-size between samples. Our framework follows the same volume integral, but with samples drawn per-Gaussian from their corresponding intersection intervals $[t_k^{\text{enter}}, t_k^{\text{exit}}]$. All samples are then globally sorted by depth along the ray before compositing.

Composition of Multiple Gaussians The standard NeRF integral using a global sort of the per-Gaussian samples would result in the step size Δt_i for a sample in Gaussian k to be potentially measured against a sample in a different Gaussian $j \neq k$. This results in potentially reduced opacity α_i^k for any sample, especially if the overlaps between Gaussians are large. We instead compute α_i^k independently within Gaussian k . The step size Δt_i^k depends only on consecutive samples within that Gaussian’s own interval. The resulting independent alphas are then composited in the global depth-sorted order via the standard transmittance prefix product. This hybrid of independent opacity but joint transmittance ensures that each Gaussian accumulates rendering weight semi-independently, preventing Gaussians whose samples happen to appear later along a ray of being systematically starved of transmittance early during optimization.

5.3.3 Supervision

As in standard NeRF, the first training loss is simply the mean squared error between rendered and ground truth pixels for a batch of rays $\mathcal{R}$:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \|\hat{c}(r) - c(r)\|^2$$

Weight Entropy Regularization For many scenes, a pixel can be explained by the first surface its corresponding ray hits. Since our composite rendering scheme attempts to keep all K Gaussians alive early during optimization, we must impose a prior encouraging each ray to be explained by a single component. To do so, we introduce an entropy loss that penalizes diffuse ray ownership. For each ray, we compute the per-Gaussian rendering weights $\{w_k\}_{k=1}^K$ by summing the rendering weights over all samples belonging to Gaussian k . To complete the rendering distribution, we include the background weight $w_{K+1} = 1 - \sum_{k=1}^K w_k$ that absorbs unaccounted transmittance. The entropy loss is then

$$\mathcal{L}_{\text{ent}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \sum_{k=1}^{K+1} w_k(r) \log w_k(r)$$

Semantic Decomposition Entropy regularization encourages each ray to be owned by a single primitive, but does not provide signal for decomposition. When per-pixel semantic labels $y(r)$ are available, we resolve this ambiguity by following prior works [193, 182] and supervising the decomposition directly via a cross-entropy loss. Specifically, each Gaussian k is associated with a learnable semantic logit vector $\ell_k \in \mathbb{R}^C$ where C is the number of unique labels. These logit vectors are volume-rendered along with color into a per-pixel classification prediction

$$\hat{\ell}(r) = \sum_{k=1}^K w_k(r) \ell_k$$

and supervised via the loss $\mathcal{L}_{\text{CE}} = \text{CE}(\hat{\ell}(r), y(r))$. Using the combined training objective

$$\mathcal{L}_{\text{total}} = \lambda_{\text{MSE}} \mathcal{L}_{\text{MSE}} + \lambda_{\text{ent}} \mathcal{L}_{\text{ent}} + \lambda_{\text{CE}} \mathcal{L}_{\text{CE}}$$

our framework is able to lift pre-existing labels into a 3D representation where each Gaussian

primitive corresponds to exactly one semantic class. However, in many practical settings, dense per-pixel labels are unavailable, or a finer-grained decomposition beyond the provided labels is desired. In the following sections, we describe a label-free pipeline that addresses these settings, enabling unsupervised decomposition of scenes.

Scale Shrinkwrapping To ensure each foreground Gaussian’s bounding volume tracks its true density support, we periodically apply a hard geometry update. During training, we accumulate rendering-weight weighted world positions into unnormalized moments S_0, S_1, S_2 . After a set period, the mean is stepped towards the density-weighted centroid S_1/S_0 and the variances are stepped towards $S_2/S_0 - (S_1/S_0)^2$.

5.3.4 Label-free Decomposition

Given a trained model, we can use the local coordinate system of each Gaussian primitive to perform label-free decomposition. Recall that each Gaussian k defines a normalized coordinate frame $[-1, 1]^3$ which serves as positional input to the local NeRF. We construct a graph over this volume by voxelizing the local coordinate system and connecting occupied voxels. Edge weights are informed by two signals: a geometric term based on spatial proximity, and a semantic term derived from DINOv2 [114] features accumulated into the voxel grid. Spectral clustering on this graph yields a partition of the Gaussian’s learned density field into semantically meaningful parts, each of which can be spawned and fine-tuned as an independent child primitive. This recursive procedure gives a tree-structured decomposition of the scene.

Voxelizing the Parent Field For a parent Gaussian p equipped with NeRF parameters ϕ_p , we evaluate the learned density field $\rho_p \cdot \phi_p^\sigma$ on a regular voxel grid in local coordinates $[-1, 1]^3$. To augment geometry with semantic information, we accumulate DINOv2 features into this grid across training views. For each view, we cast rays through the parent Gaussian and evaluate rendering weights at each sample along the ray. Each point with non-negligible

rendering weight is transformed into the parent’s local coordinate frame and accumulated into the nearest voxel, weighted by rendering weight. The accumulated feature for a voxel i is

$$f_i = \frac{\sum_{v,r} w_{ivr} d_{vr}}{\sum_{v,r} w_{ivr} + \epsilon}$$

where w_{ivr} indicates the total rendering weight contributed to voxel i by samples along ray r in view v .

Spectral Partition We construct a weighted graph $G = (\mathcal{V}, \mathcal{E})$ over occupied voxels by connecting each to its k nearest neighbors with respect to Euclidean distance. Edge weights combine geometric proximity and semantic terms derived from the voxel grid:

$$w_{i,j} = \exp\left(-\frac{\|u_i - u_j\|^2}{2\tau_{\text{pos}}^2}\right) \cdot \exp\left(\frac{\cos(f_i, f_j)}{\tau_{\text{DINO}}}\right)$$

Following standard spectral clustering, we compute the normalized graph Laplacian $L_{\text{sym}} = I - D^{-1/2} W D^{-1/2}$ and take the eigenvectors corresponding to the smallest d_{spectral} nontrivial eigenvalues as columns of a matrix $Z \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{spectral}}}$. Each row $z_i \in \mathbb{R}_{\text{spectral}}^d$ is the spectral embedding of voxel i . Agglomerative clustering on these embeddings partitions the voxels into K clusters, producing a label-free volumetric decomposition of the parent Gaussian field. Each cluster corresponds to a semantically coherent region and is converted into child Gaussian primitives.

Child Initialization For a cluster C_k , the child Gaussian mean is set to the density-weighted centroid of its assigned voxels:

$$\mu_k = \frac{\sum_{i \in C_k} \sigma_i \mathbf{x}_i}{\sum_{i \in C_k} \sigma_i}$$

where the world-space coordinates $\mathbf{x}_i$ are recovered by inverting the normalization procedure defined in 5.3.1: $\mathbf{x}_i = u_i \odot (r \cdot s_p) + \mu_p$. Note that we include an extra r factor for the radius threshold since the NeRF inputs are further normalized by this threshold to fall in the $[-1, 1]$ interval.

The child Gaussian scales are then just the density-weighted per-axis standard deviations:

$$s_k = \sqrt{\frac{\sum_{i \in C_k} \sigma_i (\mathbf{x}_i - \mu_k)^2}{\sum_{i \in C_k} \sigma_i}}$$

Since r defines how many standard deviations within which rays intersections are sampled, this choice of s_k ensures each child’s sampling region matches the coverage the parent provided.

Semantic Supervision of Child Primitives We supervise each child Gaussian through induced 2D segmentations obtained by projecting the spectral clustering result into images. For any training ray r , we accumulate the rendering weights into a label probability vector $q(r) \in \mathbb{R}^K$

$$q(r) = \sum_{i=1}^S w_p(r, i) \mathbf{e}_{\ell(i)}$$

where S is the samples along the ray, $w_p(r, i)$ is the rendering weight of sample i in the parent Gaussian p of ray r , and $\mathbf{e}_{\ell(i)}$ is a one-hot vector giving the voxel label of the sample. The ray is assigned the argmax label of this probability vector, and then the set of child Gaussians is supervised with the same combined objective $\mathcal{L}_{\text{total}}$ defined in Section 5.3.3.

5.4 Experiments

We evaluate on three classes of multi-view image data. The first is a set of custom rendered Blender meshes with instance-level labels. This data provides a controlled setting for verifying our framework recovers coherent, one-primitive-per-instance decompositions without relying on post-hoc grouping of tiny primitives. The second is the standard NeRF-Synthetic dataset, consisting of rendered Blender meshes without labels. We use these scenes to test our decomposition method in an unsupervised setting with known geometry. Finally, we experiment with real-world unbounded scenes used by MipNeRF-360 [9] to test our object recovery in a setting with unbounded depth, unknown geometry, and complex backgrounds.

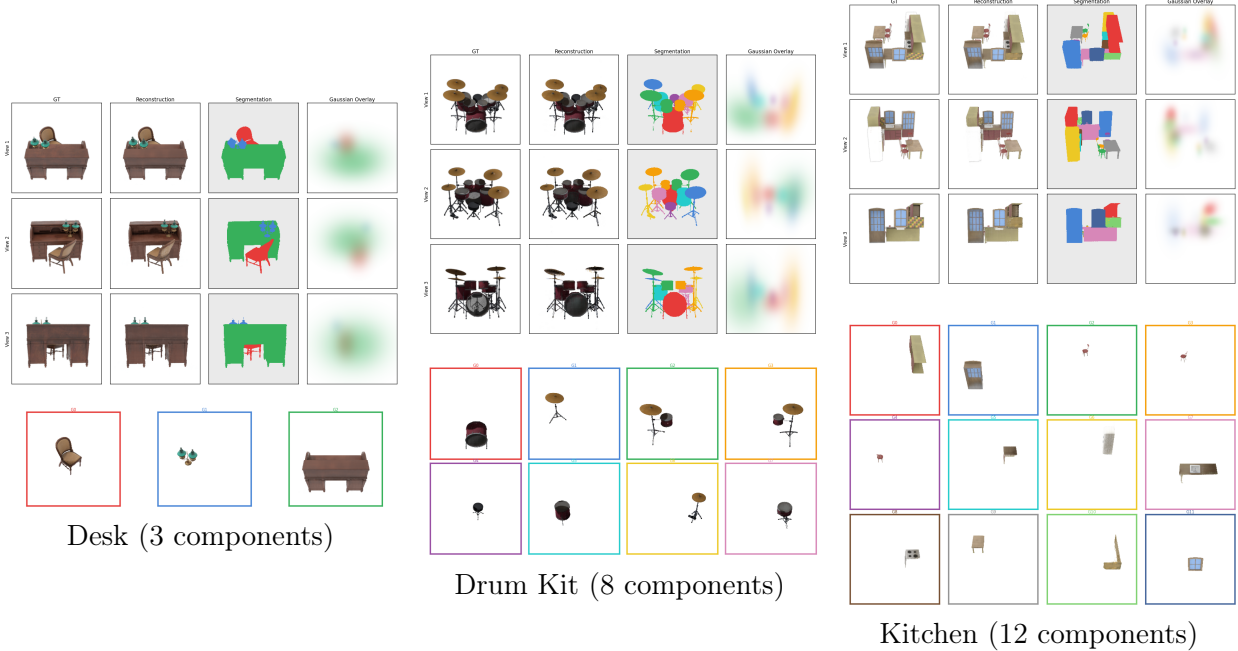


Figure 5.1: Decomposition results on instance-labeled Blender scenes. For each scene we show the reconstructed RGB, segmentation map, Gaussian extents, and per-primitive isolated renders. Each Gaussian specializes to a single object instance.

5.4.1 Instance-Labeled Meshes

Reconstruction and Object Retrieval We render three custom labeled scenes in Blender: a desk scene with a large component dominating the other two, a kitchen scene with many components, and a drum scene with complex occlusions. Figure 5.1 shows that our method recovers a clean decomposition in all three cases. Each Gaussian primitive represents an entire object instance, and the independent component renders demonstrate that each primitive learns an occlusion-free appearance. Objects that are partially occluded in the training views are recovered cleanly in isolation.

Editing and Composition The design of our framework induces a natural set of editing operations that require no post-hoc optimization or post-processing. Since each Gaussian primitive maintains a local coordinate system, and the corresponding NeRF renders using those local coordinates, transformations applied to a Gaussian’s position and scale are automatically reflected in the final image. Concretely, a translation a Gaussian’s mean translates



Figure 5.2: Scene editing on the desk scene. Each row demonstrates a different transformation across six viewpoints. (1) Shows a uniform $0.5\times$ scale reduction in each axis. (2) and (3) show anisotropic scaling. (4) demonstrates a translation of the lamp across the desk. All edits are done at inference time with no additional optimization required.

the object in world space, while scaling its axes stretches or shrinks the object along the corresponding direction. Figure 5.2 illustrates this capability on the desk scene. Each column shows a test viewpoint and each column demonstrates a distinct transformation. The first three rows edit only the desk primitive, showing different scales: (1) is an isotropic shrink, (2) and (3) are anisotropic scalings in different axes. The last row demonstrates a translation of the lamp primitive across the desk. Crucially, these edits are all done at inference only by adjusting the Gaussian parameters, with no additional finetuning.

The object-centric structure of our rendering scheme also enables cross-scene composition. Because each Gaussian primitive maintains its own independent opacity and local coordinate frame, primitives sourced from entirely different trained models can be



Figure 5.3: Cross-scene composition. A desk, a lamp, a kitchen table, and a drum kit stool, trained from 3 different scenes, are assembled into a coherent composition by placing each primitive at a desired position in a shared world space. Again, no additional optimization is required.

assembled into a coherent scene simply by placing them at desired positions in a shared world space, while retaining their learned appearances. Figure 5.3 illustrates this: the desk and lamp primitives are taken from the desk scene and the lamp is shifted to a new position on the desk surface. A kitchen table is scaled down and placed as a prop, and a stool from the drum kit scene is positioned below the desk. Despite being drawn from three independently trained models, the assembled scene renders coherently from novel viewpoints with correct mutual occlusion between all four primitives.

Rendering Ablation We use the labeled kitchen and drum scenes to demonstrate the necessity of our modified volume rendering method. Under standard volumetric rendering, samples from all Gaussians are depth-sorted and composited through a single shared transmittance chain. Because Gaussians overlap, particularly before specialization has emerged, their samples interleave along each ray, and each primitive’s step size is not determined by its own geometry, but by the aggregate densities of all overlapping primitives. Thus, a primitive’s opacity field cannot be interpreted in isolation as it is entangled with the fields of neighboring Gaussians. This introduces rendering instability that label supervision alone cannot resolve, as demonstrated in Figure 5.4. In the labeled drums scene, Gaussians 3 and 6 absorb the metal stand geometry of the drum modeled by Gaussian 7. In the kitchen scene, three primitives collapse entirely, rendering blank images in isolation.

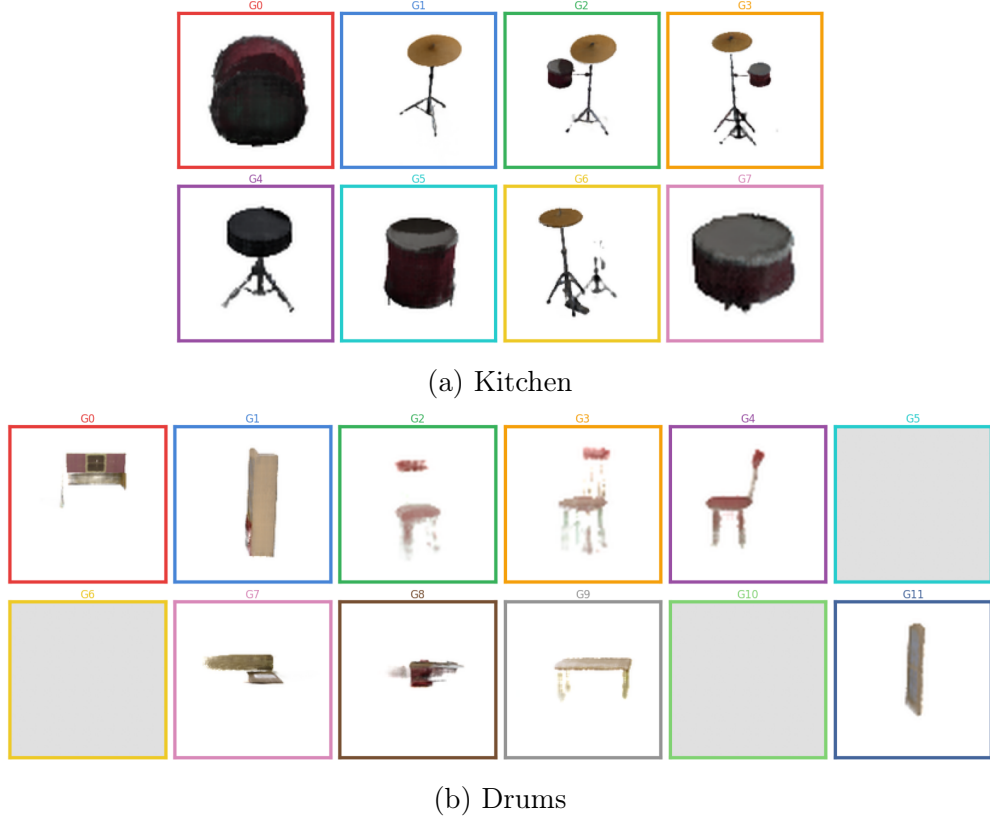


Figure 5.4: Per-primitive isolated renders under standard volumetric rendering. Despite explicit label supervision, the model fails to learn a clean decomposition. In the drum scene, Gaussians 3 and 6 have absorbed extraneous stand geometry. In the kitchen scene, three primitives collapse entirely. Overall rendering quality is also degraded.

5.4.2 Comparisons

PartNeRF PartNeRF [156] is the most closely related work to our proposed method, also using ellipsoidal primitives to support part-level transformations. We evaluate whether PartNeRF produces semantically interpretable components when applied at the scene level. Figure 5.5 visualizes the full optimized scene from different views, as well as each of the 16 parts rendered independently. Despite plausible reconstruction quality (top row), none of the 16 components isolates a semantically recognizable object or part. Attempting to edit the scene by translating the component containing the lamp (as we do in Figure 5.2 row 4) illustrates the limitation even further. Because parts do not correspond to individual

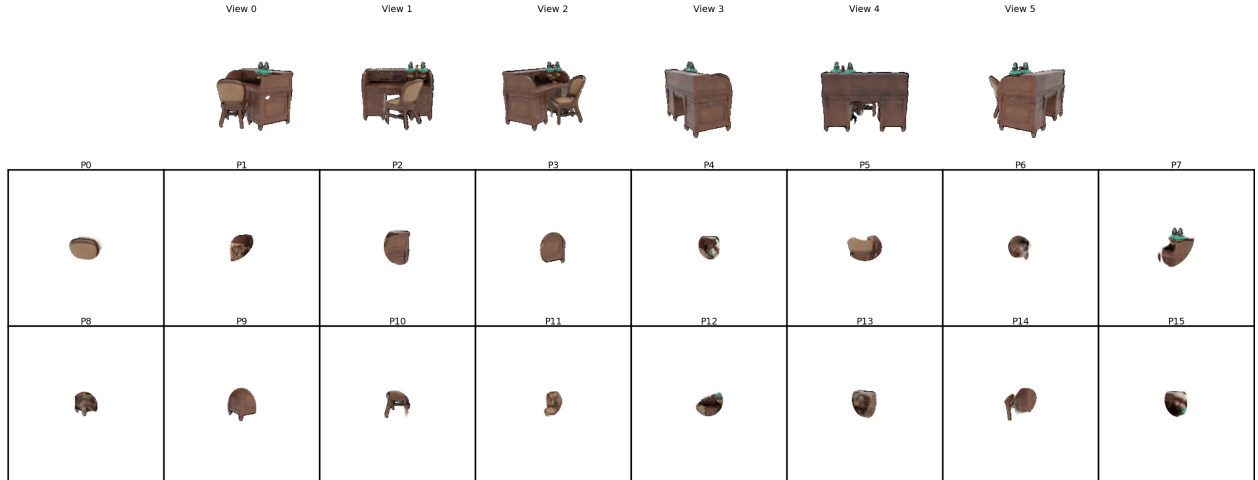


Figure 5.5: PartNeRF trained on the desk scene with 16 components. Despite plausible reconstruction quality (top), the learned decomposition produces 16 small, overlapping blobs with no semantic coherence. This confirms that geometric regularization alone provides no incentive for parts to align with semantically meaningful regions.



Figure 5.6: Attempting to edit a PartNeRF decomposition by translating the component containing the lamp. Because parts do not correspond to individual objects, the transformation moves unrelated geometry contained within the same ellipsoidal support, producing implausible results across views.

objects, the transformation moves unrelated geometry sharing the same support, producing implausible results. As we show in Figure 5.6, the translation also displaces a portion of the desk, while leaving a green artifact at the original location of the lamp.

We additionally evaluate a label-supervised variant of PartNeRF with 3 components, matching the number of distinct objects in the scene. As shown in Figure 5.7, label supervision partially recovers the correct structure. However, we observe that the desk and lamp components are grouped together, while the model produces two duplicate chair primitives. This is consistent with their hard-assignment rendering. In short, PartNeRF opts to render only the first primitive that a ray hits with non-negligible occupancy. Since the lamp



Figure 5.7: PartNeRF with label supervision and 3 components on the desk scene. While label supervision partially recovers semantic structure, no component captures the lamp, instead producing two near-duplicate chair primitives. Hard ray-to-part assignment also handles occlusion poorly, producing view-dependent artifacts wherever parts are occluded.

is small and frequently occluded, it never accumulates sufficient rays to claim a dedicated hard-assigned primitive. Furthermore, hard-assignment also handles occlusions poorly. For any view in which the chair occludes the desk, we observe significant distortion in the desk component (for example, columns 1 and 2 in the first row). We observe a similar effect for views in which the chair is occluded by the desk (for example, columns 4, 5, 6 in rows 2 and 3).

Gaussian Grouping Unlike PartNeRF, Gaussian Grouping uses label information to successfully produce semantically meaningful groups. The selected components in Figure 5.8 are recognizably a cabinet, a chair, and a cymbal. We evaluate the robustness of Gaussian grouping to geometric transformation compared to our method by applying anisotropic scaling to each object. Because each Gaussian’s appearance is optimized at the original scale, orientation, and total Gaussian resolution, scaling the splats creates artifacts in the resulting renders. In the first row we observe tears throughout the entire cabinet that reveal the white



Figure 5.8: Independent renders of selected objects from our method (left) vs. Gaussian Grouping across three scenes (right) with anisotropic scaling applied. Both methods correctly lift the label information to 3D, but the Gaussian Splatting representation is not robust to transformations like scaling. We observe tear-like artifacts in row 1 throughout the cabinet and row 3 near the edge for the cymbal. In row 2, we see that the texture on the chair is heavily warped by the scaling.

background. Similarly in row 3, we observe tears near the edge of the cymbal. In row 2, there is heavy texture warping on the chair. In contrast, our method represents each object as a Gaussian-anchored radiance field queried at arbitrary local sample positions within the primitive’s coordinate frame. Because appearance is a function of local position rather than a fixed per-splat attribute, geometric transformations such as scaling do not corrupt the rendered output. As shown in Figures 5.2 and 5.8, our method handles such transformations cleanly without reoptimization.

5.4.3 Unsupervised 3D Segmentation

We next investigate the proposed label-free decomposition described in Section 5.3.4 using Nerf-Synthetic Blender scenes which provide multiview renders without instance information. Figures 5.9, 5.10, and 5.11 show part decompositions of the chair, microphone, and drum kit scenes along with the segmentations derived from spectral clustering. Despite the absence of semantic supervision, we observe that the recovered components often align with semantically meaningful object or part structure. In the chair scene, the decomposition separates the chair back, armrests, seat, and two supporting legs into distinct primitives. In the microphone



Figure 5.9: Unsupervised decomposition of the NeRF-Synthetic chair scene using the proposed label-free hierarchical decomposition framework. Each row is shows a different camera view. The columns show the per-pixel label assignment derived from spectral clustering, the full composite render of all child primitives, and each child rendered in isolation. The discovered parts include the chair back, armrests, legs, and seat.



Figure 5.10: Unsupervised decomposition of the NeRF-Synthetic mic scene. The recovered components separate the microphone head, base, and cable.

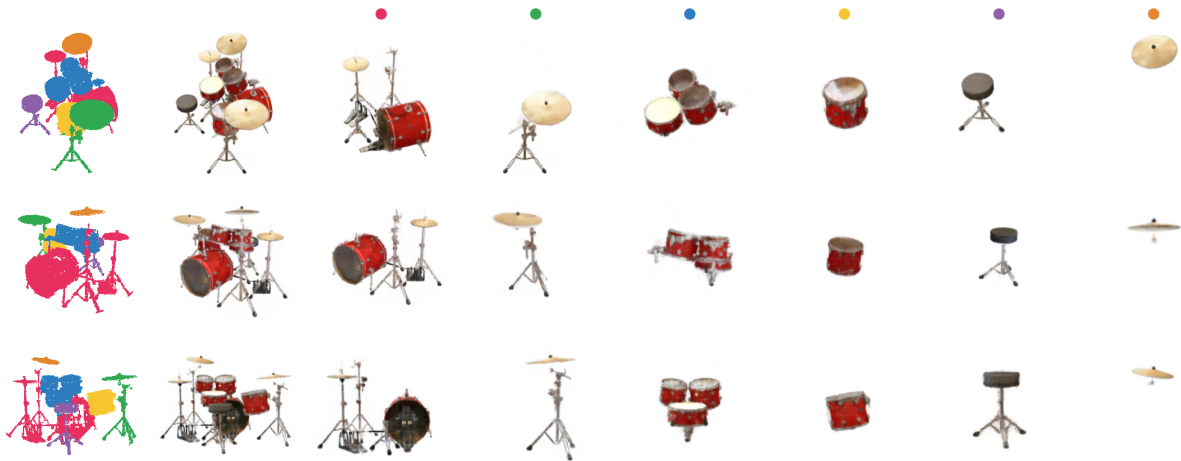


Figure 5.11: Unsupervised decomposition of the NeRF-Synthetic drums scene. The recovered components include several different drums, the stool, and cymbals. For the selected 6 cluster setting, we observe that several primitives still contain distinct sub-parts, making them candidates for an additional stage of decomposition.

scene, the recovered components include the microphone head, supporting base, and wire, despite the thin geometry of these structures. The drum kit, however, presents a substantially more challenging setting due to the dense arrangement of objects in the scene. Nevertheless, our decomposition method still recovers coherent regions corresponding to different drums, cymbals, and the stool. However, the selected 6-cluster decomposition still results in under-segmented child primitives, which contain multiple nearby structures. This coarse granularity motivates the use of a recursive hierarchical subdivision to further separate complex regions.

To address this, we apply a second stage of our spectral decomposition procedure to the pink primitive in Figure 5.11. This primitive rendered the bass drum, hi-hat, and surrounding metal stands as a single component. As demonstrated in Figure 5.12, subdividing this component recovers additional children: the hi-hat cymbal (pink), the hi-hat stand (orange), an additional metal stand (teal) and the bass drum paired with another stand (purple). Across Figures 5.11, 5.12, and 5.13, the learned primitives operate at differing levels of semantic granularity. A hi-hat cymbal and stand correspond to part-level decomposition of a single instrument instance, while the bass drum paired with a metal stand forms a



Figure 5.12: Recursive subdivision of the pink primitive from Figure 5.11, which contained the bass drum, hi-hat, and several metal stands. The first component containing the bass drum is a candidate for further subdivision to separate the drum from the extraneous stand.

compound of two spatially close but unrelated objects. Our method is designed such that any primitive can be recursively subdivided until the desired level of detail is reached.

While the recursive decomposition recovers additional structure, densely packed objects and thin geometric elements remain challenging. At practical resolutions, closely adjacent objects maybe become merged after voxelization and subsequent 2D label projection. This can be seen in Figure 5.12, where the orange and teal labels blend together near the ground, causing the orange primitive to capture a small portion of the adjacent teal structure.

5.4.4 *Unbounded Real-World Scenes*

Finally, we explore the application of our framework to unbounded real-world scenes from the MipNeRF-360 dataset. Unlike the synthetic datasets considered previously, these scenes are comprised of multiview photographs of real environments rather than rendered images. While our framework supports unsupervised decomposition as described in Section 5.3.4, the unbounded nature of these scenes presents a practical challenge for our voxelization-based clustering approach. Our decomposition approach begins by discretizing the spatial support



Figure 5.13: Recursive subdivision of the green primitive from Figure 5.11, which is a crash cymbal. The children primitives consist of two parts, the metal legs, and the cymbal connected to the stand.

of a primitive and constructing a graph over occupied voxels. In MipNeRF-360 scenes, however, background geometry often extends far outward in all directions. As a result, the root primitive may span a large spatial region even though much of the scene structure is concentrated near the center. When voxelized at a fixed resolution, the central objects occupy only a small fraction of the available voxels, while most of the grid is dedicated to distant background geometry and empty space. Recovering meaningful object boundaries in this regime requires prohibitively high resolutions, making direct application of our procedure impractical.

For this reason, our experiments use masks generated by the Segment Anything Model (SAM) [81], following common practice in recent object-centric reconstruction works. This allows us to investigate whether our framework can recover real-world objects as isolated primitives and whether these primitives support the same object-level manipulations demonstrated in Section 5.4.1. Figure 5.14 shows results on the garden MipNeRF-360 garden scene. We observe that the table is recovered as a coherent object that can be rendered independently from the surrounding environment. However, the vase primitive contains some blurry artifacts due to imperfections in the SAM masks. Despite this limitation in reconstruction

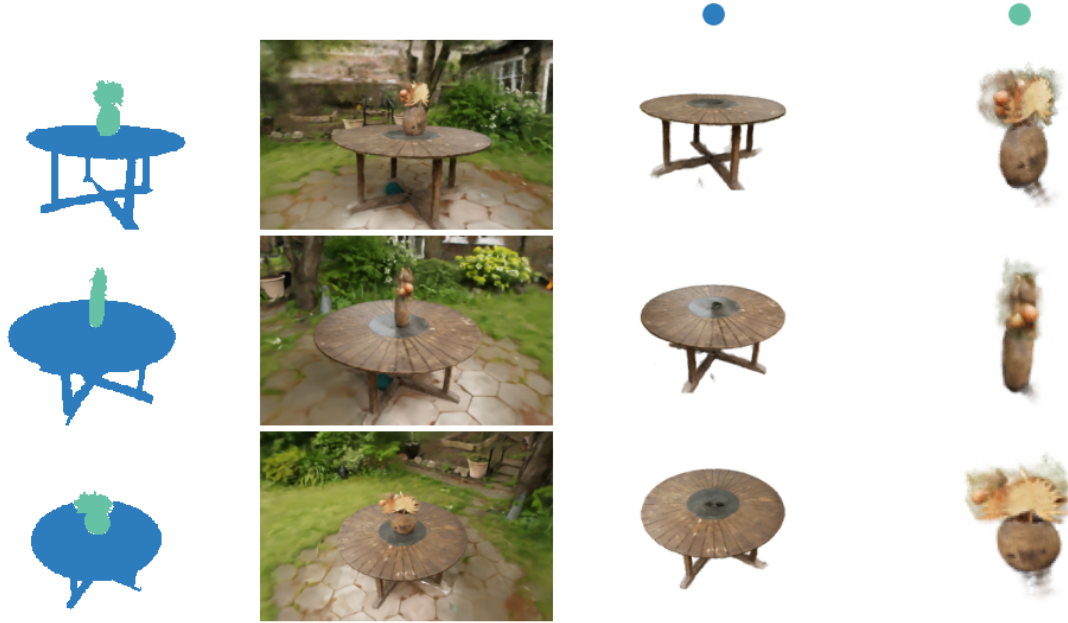


Figure 5.14: We train our framework on the MipNeRF-360 garden scene. The columns show the SAM-generated masks, the full composite render, and the table and vase primitives rendered in isolation. While the table is recovered cleanly, noise in the SAM masks leads to imperfect separation of the vase.

quality, both objects still function as local primitives that can be manipulated.

In Figure 5.15, the vase is translated across the table by modifying only the corresponding Gaussian mean at inference time. Despite imperfections in the decomposition, the edit still produces a plausible modified scene. This suggests that the object manipulation capabilities of our framework shown previously in synthetic scenes extend to real-world scenes when suitable semantic supervision is available.

5.5 Discussion

In this chapter, we introduced a scene representation that combines large Gaussian primitives with independent NeRFs. Unlike prior approaches, our method jointly provides semantic decomposition and object-level editing within a single representation. We also propose a hierarchical subdivision procedure when semantic supervision is not available. The re-



Figure 5.15: Editing in a real-world scene. We translate the recovered vase primitive across the table during inference time.

sulting primitives can be manipulated directly through their Gaussian parameters, enabling transformations, scaling, and cross-scene composition without any additional finetuning.

Despite these advantages, several limitations remain. The quality of the recovered primitives relies heavily on the available training segmentations or clustering procedure. In densely packed scenes, our unsupervised clustering method may struggle to separate nearby objects at practical voxel resolutions. Large unbounded scenes present an additional challenge, as the scene volume can greatly exceed the size of the objects of interest, making voxel-based partitioning difficult. In these settings, we rely on externally provided mask proposals such as those from SAM, whose quality may ultimately limit the final representations.

CHAPTER 6

CONCLUSION

This dissertation explores the problem of controllable 3D editing across several iterations of neural representations for shapes and scenes. As 3D generative models have improved, the challenge has shifted from producing high-quality geometry, to modifying that geometry in precise and plausible ways. Throughout this work, we demonstrate that the capabilities of a neural editing system are tied to the representation design. The types of edits a technique can perform, the precision it offers, and the limitations it suffers are all influenced by how 3D information is encoded and propagated internally.

In the first chapter, we investigate editing within a learned latent space, where semantic manipulations arise naturally through operations on shape embeddings. However, this system provides only limited control over the specific changes induced by a latent edit. To enable more expressive modifications, we turn to optimization-based approaches guided by large multimodal models, which leverage the vast amount semantic information available in image-text datasets. We show that alternative parameterizations for optimizing mesh deformations can improve the stability of this editing paradigm under noisy supervision. Nevertheless, these approaches are fundamentally slow and unpredictable, often requiring many iterations and careful tuning to obtain a desired result. Motivated by these limitations, we investigated the possibility of feed-forward editing through Large Reconstruction Models. We frame editing as a multiview completion problem allowing modified geometry to be generated directly from a learned prior over realistic shapes.

We then extend our scope beyond individual shapes to full scenes. In this setting, the focus is no longer just modifying geometry in isolation, but recovering semantic object structure that can be rendered and edited either independently or within the scene context. To this end, we introduce a framework based on Gaussian primitives and local radiance fields that supports object-level manipulation and a hierarchical semantic decomposition of scenes.

Collectively, the contributions in this dissertation demonstrate how neural representations play a key role in the controllability and expressiveness of 3D editing systems. As representations become richer, they enable increasingly powerful forms of interaction ranging from latent shape interpolation to multi-scene recomposition. Further advances in generative modeling and neural rendering will continue to reduce the barrier to entry for creating and editing 3D assets, enabling users to generate novel content through intuitive modifications to existing geometry and scenes.

REFERENCES

- [1] Noam Aigerman, Kunal Gupta, Vladimir G. Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural Jacobian Fields: Learning Intrinsic Mappings of Arbitrary Meshes. *ACM Transactions on Graphics*, 2022.
- [2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep ViT Features as Dense Visual Descriptors. *ECCVW What is Motion For?*, 2022.
- [3] Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O’Brien. Fast and Deep Deformation Approximations. *ACM Transactions on Graphics*, 2018.
- [4] Stephen W Bailey, Dalton Omens, Paul Dilorenzo, and James F O’Brien. Fast and Deep Facial Deformations. *ACM Transactions on Graphics*, 2020.
- [5] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. SINE: Semantic-driven Image-based NeRF Editing with Prior-guided Editing Field. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023.
- [6] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-training of Image Transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [7] Amir Barda, Vladimir G. Kim, Noam Aigerman, Amit Bermano, and Thibault Groueix. MagicClay: Sculpting Meshes With Generative Neural Fields. In *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2024.
- [8] Amir Barda, Matheus Gadelha, Vladimir G. Kim, Noam Aigerman, Amit H. Bermano, and Thibault Groueix. Instant3dit: Multiview Inpainting for Fast Editing of 3D Objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [9] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
- [10] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM transactions on graphics (TOG)*, 21(3):312–321, 2002.
- [11] Wang Bing, Lu Chen, and Bo Yang. DM-NeRF: 3D Scene Geometry Decomposition and Manipulation from 2D Images. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

- [12] Mark Boss, Zixuan Huang, Aaryaman Vasishta, and Varun Jampani. SF3D: Stable Fast 3D Mesh Reconstruction with UV-Unwrapping and Illumination Disentanglement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [13] Robert Bridson and Crawford Doran. Quartet. <https://github.com/crawforddor-an/quartet>, 2013.
- [14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [15] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment Any 3D Gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.
- [16] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
- [17] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. NeuralEditor: Editing neural radiance fields via manipulating point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [18] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. *arXiv preprint arXiv:1803.08495*, 2018.
- [19] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative Pretraining from Pixels. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.
- [20] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhonggang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [21] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2019.
- [22] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

- [23] Zhiqin Chen, Vladimir G Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. DECOR-GAN: 3D Shape Detailization by Conditional Refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [24] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking Anything with Decoupled Video Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [25] Chong Bao and Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.
- [26] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. ABO: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022.
- [27] Sabine Coquillart. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, 1990.
- [28] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. DiffEdit: Diffusion-based Semantic Image Editing with Mask Guidance. In *International Conference on Learning Representations (ICLR)*, 2023.
- [29] Barbara Cutler, Julie Dorsey, Leonard McMillan, Matthias Müller, and Robert Jagnow. A Procedural Approach to Authoring Solid Models. *ACM Transactions on Graphics*, 2002.
- [30] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli Vanderbilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.
- [31] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-XL: A Universe of 10m+ 3D Objects. *Advances in Neural Information Processing Systems*, 36, 2024.
- [32] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

- [33] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google Scanned Objects: A High-quality Dataset of 3D Scanned Household Items. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.
- [34] Ziya Erkoç, Can Gümeli, Chaoyang Wang, Matthias Nießner, Angela Dai, Peter Wonka, Hsin-Ying Lee, and Peiye Zhuang. PrEditor3D: Fast and Precise 3D Shape Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [35] Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin, and Alec Jacobson. Latent-space Dynamics for Reduced Deformable Simulation. *Computer Graphics Forum*, 2019.
- [36] Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. iWIRES: An Analyze-and-edit Approach to Shape Manipulation. *ACM Transactions on Graphics*, 2009.
- [37] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, Sanja Fidler, Huan Ling, David Acuna, Karsten Kreis, Seung Kim, et al. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. *Advances in Neural Information Processing Systems*, 2020.
- [38] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic Unpaired Shape Deformation Transfer. *ACM Transactions on Graphics*, 2018.
- [39] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics*, 2019.
- [40] William Gao, April Wang, Gal Metzer, Raymond A. Yeh, and Rana Hanocka. TetGAN: A convolutional neural network for tetrahedral mesh generation. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2022.
- [41] William Gao, Noam Aigerman, Groueix Thibault, Vladimir Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM Transactions on Graphics (SIGGRAPH)*, 2023.
- [42] William Gao, Dilin Wang, Yuchen Fan, Aljaz Bozic, Tuur Stuyck, Zhengqin Li, Zhao Dong, Rakesh Ranjan, and Nikolaos Sarafianos. 3D Mesh Editing using Masked LRMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [43] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a Predictable and Generative Vector Representation for Objects. In *Proceedings of the European Conference on Computer Vision*, 2016.

- [44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2014.
- [45] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A Papier-mâché Approach to Learning 3D Surface Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [46] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 2017.
- [47] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. ALIGNet: Partial-Shape Agnostic Alignment via Unsupervised Learning. *ACM Transactions on Graphics*, 2018.
- [48] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics*, 2019.
- [49] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2Mesh: A self-prior for deformable meshes. *ACM Transactions on Graphics*, 2020.
- [50] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [51] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [52] Zexin He and Tengfei Wang. OpenLRM: Open-Source Large Reconstruction Models. <https://github.com/3DTopia/OpenLRM>, 2023.
- [53] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep Geometric Texture Synthesis. *ACM Transactions on Graphics*, 2020.
- [54] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing Implicit Shapes Through Part Aware Generation. *ACM Transactions on Graphics*, 2022.
- [55] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

- [56] Daniel Holden, Jun Saito, and Taku Komura. Learning an Inverse Rig Mapping for Character Animation. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2015.
- [57] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large Reconstruction Model for Single Image to 3D. In *Proceedings of the International Conference on Learning Representations*, 2024.
- [58] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R. Martin. Subdivision-Based Mesh Convolution Networks. *ACM Transactions on Graphics*, 2021.
- [59] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral Meshing in the Wild. *ACM Transactions on Graphics*, 2018.
- [60] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast Tetrahedral Meshing in the Wild. *ACM Transactions on Graphics*, 2020.
- [61] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698*, 2018.
- [62] Muhammad Zubair Irshad, Sergey Zakharov, Vitor Guizilini, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. NeRF-MAE: Masked Autoencoders for Self-supervised 3D Representation Learning for Neural Radiance Fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2025.
- [63] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded Biharmonic Weights for Real-time Deformation. *ACM Transactions on Graphics*, 2011.
- [64] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH 2014 Courses*, 2014.
- [65] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-Shot Text-Guided Object Generation with Dream Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [66] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. KeypointDeformer: Unsupervised 3D Keypoint Discovery for Shape Control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [67] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George Drettakis. NeRFshop: Interactive Editing of Neural Radiance Fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2023.

- [68] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, et al. gradSim: Differentiable Simulation for System Identification and Visuomotor Control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [69] Jincen Jiang, Xuequan Lu, Lizhi Zhao, Richard Dazaley, and Meili Wang. Masked Autoencoders in 3D Point Cloud Representation Learning. *IEEE Transactions on Multimedia*, 2023.
- [70] Hyunyoung Jung, Seonghyeon Nam, Nikolaos Sarafianos, Sungjoo Yoo, Alexander Sorkine-Hornung, and Rakesh Ranjan. Geometry Transfer for Stylizing Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [71] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive Mesh Fusion Based on Local 3D Metamorphosis. In *Graphics interface*, 1999.
- [72] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning Category-Specific Mesh Reconstruction from Image Collections. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [73] Tero Karras, Samuli Laine, and Timo Aila. A Style-based Generator Architecture for Generative Adversarial Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [74] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-Free Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 2021.
- [75] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 2023.
- [76] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. CLIP-Mesh: Generating Textured Meshes from Text using Pretrained Image-text Models. *SIGGRAPH Asia 2022 Conference Papers*, December 2022.
- [77] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [78] Hyun Woo Kim, Itai Lang, Thibault Groueix, Noam Aigerman, Vladimir G. Kim, and Rana Hanocka. MeshUp: Multi-Target Mesh Deformation via Blended Score Distillation. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2025.

- [79] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [80] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.
- [81] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything. In *Proceedings of the IEEE/CVR Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [82] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial Generation of Continuous Implicit Shape Representations. In *Proceedings of the Conference of the European Association for Computer Graphics*, 2020.
- [83] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. In *Advances in Neural Information Processing Systems*, 2022.
- [84] Dmytro Kotovenko, Olga Grebenkova, Nikolaos Sarafianos, Avinash Paliwal, Pingchuan Ma, Omid Poursaeed, Sreyas Mohan, Yuchen Fan, Yilei Li, Rakesh Ranjan, et al. WaSt-3D: Wasserstein-2 Distance for Scene-to-Scene Stylization on 3D Gaussians. In *Proceedings of the European Conference on Computer Vision*, 2024.
- [85] Alon Lahav and Ayellet Tal. MeshWalker: Deep Mesh Understanding by Random Walks. *ACM Transactions on Graphics*, 2020.
- [86] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics*, 2020.
- [87] Eran Levin and Ohad Fried. Differential Diffusion: Giving Each Pixel Its Strength. *arXiv preprint arXiv:2306.00950*, 2023.
- [88] Chuan Li and Michael Wand. Precomputed Real-time Texture Synthesis with Markovian Generative Adversarial Networks. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [89] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3D: Fast Text-to-3D with Sparse-view Generation and Large Reconstruction Model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [90] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning Skeletal Articulations with Neural Blend Shapes. *ACM Transactions on Graphics*, 2021.

- [91] Wanhua Li, Yujie Zhao, Minghan Qin, Yang Liu, Yuanhao Cai, Chuang Gan, and Hanspeter Pfister. LangSplatV2: High-dimensional 3D Language Gaussian Splatting with 450+ FPS. *Advances in Neural Information Processing Systems*, 2025.
- [92] Wanhua Li, Renping Zhou, Jiawei Zhou, Yingwei Song, Johannes Herter, Minghan Qin, Gao Huang, and Hanspeter Pfister. 4D LangSplat: 4D Language Gaussian Splatting via Multimodal Large Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [93] Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. MeshMAE: Masked Autoencoders for 3D Mesh Data Analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.
- [94] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [95] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. Differential Coordinates for Interactive Mesh Editing. In *Proceedings of Shape Modeling International*, 2004.
- [96] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural Subdivision. *ACM Transactions on Graphics*, 2020.
- [97] Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. DeepMetaHandles: Learning Deformation Meta-Handles of 3D Meshes with Biharmonic Coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [98] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2023.
- [99] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [100] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. SyncDreamer: Generating Multiview-consistent Images from a Single-view Image. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [101] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3D: Single Image to 3D using Cross-Domain Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- [102] Mehran Mehralian and Babak Karasfi. RDCGAN: Unsupervised Representation Learning With Regularized Deep Convolutional Generative Adversarial Networks. In *Proceedings of the Conference on Artificial Intelligence and Robotics*, 2018.
- [103] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. VV-Net: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019.
- [104] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [105] Zhenxing Mi and Dan Xu. Switch-NeRF: Learning Scene Decomposition with Mixture of Experts for Large-scale Neural Radiance Fields. In *International Conference on Learning Representations (ICLR)*, 2023.
- [106] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2Mesh: Text-Driven Neural Stylization for Meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [107] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-Dual Mesh Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2020.
- [108] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [109] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*, 2022.
- [110] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [111] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An Autoregressive Generative Model of 3D Meshes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.
- [112] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

- [113] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. *arXiv preprint arXiv:2212.08751*, 2022.
- [114] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [115] Gilles-Philippe Paillé, Nicolas Ray, Pierre Poulin, Alla Sheffer, and Bruno Lévy. Dihedral Angle-based Maps of Tetrahedral Meshes. *ACM Transactions on Graphics*, 2015.
- [116] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked Autoencoders for Point Cloud Self-Supervised Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.
- [117] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [118] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [119] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [120] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A Differentiable Poisson Solver. In *Advances in Neural Information Processing Systems*, 2021.
- [121] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3d using 2d diffusion. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.
- [122] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [123] Zhangyang Qi, Yunhan Yang, Mengchen Zhang, Long Xing, Xiaoyang Wu, Tong Wu, Dahua Lin, Xihui Liu, Jiaqi Wang, and Hengshuang Zhao. Tailor3D: Customized 3D Assets Editing and Generation with Dual-Side Images, 2024.
- [124] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [125] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
- [126] Marie-Julie Rakotosaona, Noam Aigerman, Niloy J Mitra, Maks Ovsjanikov, and Paul Guerrero. Differentiable Surface Triangulation. *ACM Transactions on Graphics*, 2021.
- [127] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *ArXiv*, abs/2204.06125, 2022.
- [128] Mervi Ranta, Masatomo Inui, Fumihiko Kimura, and Martti Mäntylä. Cut and Paste Based Modeling with Boundary Features. In *Proceedings on the Second ACM Symposium on Solid Modeling and Applications*, 1993.
- [129] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.
- [130] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [131] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [132] Abdrakhmanov Renat and Kerimbek Imangali. Learning Latent Representations for 3D Voxel Grid Generation using Variational Autoencoders. In *IEEE AITU: Digital Generation*, 2024.
- [133] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

- [134] Cristian Romero, Dan Casas, Jesus Perez, and Miguel A. Otaduy. Learning Contact Corrections for Handle-Based Subspace Dynamics. *ACM Transactions on Graphics*, 2021.
- [135] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *Advances in Neural Information Processing Systems*, 2022.
- [136] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-Forge: Towards Zero-shot Text-to-shape Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [137] Nikolaos Sarafianos, Tuur Stuyck, Xiaoyu Xiang, Yilei Li, Jovan Popovic, and Rakesh Ranjan. Garment3dgen: 3d garment stylization and texture generation. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2025.
- [138] Rohan Sawhney and Keenan Crane. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Transactions on Graphics*, 2020.
- [139] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An Open Large-scale Dataset for Training Next Generation Image-text Models. *Advances in Neural Information Processing Systems*, 2022.
- [140] Thomas W Sederberg and Scott R Parry. Free-form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, 1986.
- [141] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-E: Text-guided Voxel Editing of 3D Objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [142] Nicholas Sharp and Maks Ovsjanikov. PointTriNet: Learned Triangulation of 3D Point Sets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [143] Siyuan Shen, Yin Yang, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. High-order Differentiable Autoencoder for Nonlinear Model Reduction. *ACM Transactions on Graphics*, 2021.
- [144] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems*, 2021.

- [145] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a Single Image to Consistent Multi-view Diffusion Base Model, 2023.
- [146] Hang Si. TetGen, a Delaunay-based Quality Tetrahedral Mesh Generator. *ACM Transactions on Mathematical Software*, 2015.
- [147] Olga Sorkine and Marc Alexa. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, 2007.
- [148] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian Surface Editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, 2004.
- [149] Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation. *arXiv:2104.01148*, 2021.
- [150] Bo Sun, Xiangru Huang, Qixing Huang, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. ARAPReg: An As-Rigid-As Possible Regularization Loss for Learning Deformable Shape Generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [151] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively Learned Point Cloud Generation with Self-attention. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2020.
- [152] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational Autoencoders for Deforming 3D Mesh Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [153] Matthew Tancik, Vincent Casser, Xinchun Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [154] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive Auto-encoder for Artistic Mesh Generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.
- [155] Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. MVDiffusion++: A Dense High-resolution Multi-view Diffusion Model for Single or Sparse-view 3D Object Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

- [156] Konstantinos Tertikas, Despoina Paschalidou, Boxiao Pan, Jeong Joon Park, Mikaela Angelina Uy, Ioannis Emiris, Yannis Avrithis, and Leonidas Guibas. Generating Part-Aware Editable 3D Shapes without 3D Supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [157] Anh Thai, Songyou Peng, Kyle Genova, Leonidas Guibas, and Thomas Funkhouser. SplatTalk: 3D VQA with Gaussian Splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [158] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. TripoSR: Fast 3D Object Reconstruction From a Single Image. *arXiv preprint arXiv:2403.02151*, 2024.
- [159] Ty Trusty, Honglin Chen, and David I.W. Levin. The Shape Matching Element Method: Direct Animation of Curved Surface Models. *ACM Transactions on Graphics*, 2021.
- [160] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning Shape Abstractions by Assembling Volumetric Primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [161] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [162] Dirce Uesu, Louis Bavoil, Shachar Fleishman, Jason Shepherd, and Cláudio T Silva. Simplification of Unstructured Tetrahedral Meshes by Point Sampling. In *Intl. Workshop on Volume Graphics*, 2005.
- [163] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.
- [164] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [165] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [166] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

- [167] Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. Linear Subspace Design for Real-time Shape Deformation. *ACM Transactions on Graphics*, 2015.
- [168] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active Co-analysis of a Set of Shapes. *ACM Transactions on Graphics*, 2012.
- [169] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-Consistent 3D Editing with Gaussian Splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [170] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snively, Abhishek Kar, and Angjoo Kanazawa. NeRFiller: Completing Scenes via Generative 3D Inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [171] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. MeshLRM: Large Reconstruction Model for High-Quality Mesh. *arXiv preprint arXiv:2404.12385*, 2024.
- [172] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Advances in Neural Information Processing Systems*, 2016.
- [173] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-compositional Neural Implicit Surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.
- [174] Qianyi Wu, Kaisiyuan Wang, Kejie Li, Jianmin Zheng, and Jianfei Cai. ObjectSDF++: Improved Object-Compositional Neural Implicit Surfaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2023.
- [175] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models. *arXiv preprint arXiv:2404.07191*, 2024.
- [176] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3D: Denoising Multi-View Diffusion using 3D Large Reconstruction Model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [177] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting Animation Skeletons for 3D Articulated Models via Volumetric Nets. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2019.
- [178] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural Rigging for Articulated Characters. *ACM Transactions on Graphics*, 2020.

- [179] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [180] Haibo Yang, Yang Chen, Yingwei Pan, Ting Yao, Zhineng Chen, Zuxuan Wu, Yu-Gang Jiang, and Tao Mei. DreamMesh: Jointly Manipulating and Texturing Triangle Meshes for Text-to-3D Generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [181] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. *Advances in Neural Information Processing Systems*, 2021.
- [182] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [183] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural Cages for Detail-Preserving 3D Deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [184] Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir G Kim, and Hao Zhang. COALESCE: Component Assembly by Learning to Synthesize Connections. In *Proceedings of the International Conference on 3D Vision*, 2020.
- [185] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.
- [186] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields from One or Few Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [187] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. MVImgNet: A Large-scale Dataset of Multi-view Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [188] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. NeRF-Editing: Geometry Editing of Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [189] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. Semantic Shape Editing using Deformation Handles. *ACM Transactions on Graphics*, 2015.

- [190] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large Reconstruction Model for 3D Gaussian Splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2025.
- [191] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale Masked Autoencoders for Hierarchical Point Cloud Pre-training. *Advances in Neural Information Processing Systems*, 2022.
- [192] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [193] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local Radiance Fields for Efficient Structure-Aware 3D Scene Representation from 2D Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [194] Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbic. A Deep Emulator for Secondary Motion of 3D Characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [195] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation. In *Computer Graphics Forum*, 2022.
- [196] Xue Xiao Zhenxing Mi, Ping Yin and Dan Xu. Learning Heterogeneous Mixture of Scene Experts for Large-scale Neural Radiance Fields. In *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2025.
- [197] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suyu You, Zhangyang Wang, and Achuta Kadambi. Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [198] Ruijie Zhu, Mulin Yu, Linning Xu, Lihan Jiang, Yixuan Li, Tianzhu Zhang, Jiangmiao Pang, and Bo Dai. ObjectGS: Object-aware Scene Reconstruction and Scene Understanding via Gaussian Splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

APPENDIX A

CONVOLUTIONAL AUTOENCODERS FOR TETRAHEDRAL MESHES

A.1 Architecture Details

The network architecture of TetGAN combines a convolutional variational autoencoder and two convolutional Wasserstein discriminators, which are denoted as f_{enc} , f_{dec} , f_{dis}^l , and f_{dis}^g .

The encoder f_{enc} consists of N convolutional blocks of 4 convolution layers, divided by $N - 1$ pooling layers that downsample the irregular grid. After the final convolution, TetGAN can be adapted to use a variety of modules to compute an encoding. In the variational setting, we also use 2 additional linear layers, for the output layers of f_{enc}^μ and $f_{\text{enc}}^{\sigma^2}$. All layers except the final output layers are followed by instance normalization and a leaky ReLU nonlinearity.

The decoder f_{dec} is nearly a reversal of the encoder. It consists of N convolutional blocks of 4 convolution layers, divided by $N - 1$ upsampling layers that increase the grid resolution. Before the first convolution, we reverse the module that produces the final encoding. Again, all layers except the final output convolutional layer are followed by instance normalization and a leaky ReLU nonlinearity.

TetGAN is fairly robust to the choice in module used in computing the final encoding. In this manuscript, we chose to use a large linear layer across the entire grid for the high-resolution models (making one latent code per mesh) and a shared MLP per-tetrahedron for the low-resolution models (making one latent code *per tetrahedron*). In our experiments, using a Tetrahedral convolution to compute the final encoding also works well, although it produces thinner meshes. We choose a latent code size of 512.

The local discriminator f_{dis}^l is a PatchGAN [88] discriminator outputting a probability per tetrahedron $\mathbf{T}_i^{(N)} \in \mathbf{T}^{(N)}$. A final probability is computed by averaging over the tetra-

hedra. Specifically, f_{dis}^l has a fully convolutional architecture, where the final layer has 1 output channel, representing the WGAN score per tetrahedron. Each convolutional layer except the final is followed by instance normalization and a leaky ReLU nonlinearity.

The global discriminator f_{dis}^g is a more standard convolutional discriminator outputting a single probability for the entire grid $\mathbf{T}^{(N)}$. Its architecture mirrors the encoder architecture, consisting of N convolutional blocks of 3 convolution layers, divided by $N - 1$ pooling layers. After the final convolution, we apply a fully connected linear layer that outputs a scalar WGAN score. As before, each layer, except the final linear layer, is followed by instance normalization and a leaky ReLU nonlinearity.

A.2 Implementation Details

A.2.1 Subdivision

Note that there is no canonical way to subdivide a tetrahedron. The diagonal in the subdivision grid is selected based on the initial ordering of the vertices/edges in the low resolution grid (and remains fixed/unchanged throughout).

A.2.2 Hyperparameters

We subdivide the base tetrahedral grid 3 times, i.e., choosing $N = 4$. We trained TetGAN using the Adam optimizer [79] with learning rate of 0.0001 and $\beta_1, \beta_2 = 0, 0.9$ as suggested by Gulrajani et al. [46]. We choose a batch size of 30. We choose 0.5 for the weighted smoothing hyperparameter β (main text, equation 4) and 4 for the deformation field filtering hyperparameter γ (main text, equation 5).

A.2.3 Runtime

Each shape file takes on average 1-2 seconds to preprocess, depending on the size of the file. The total training time is approximately two days on two NVIDIA A40 GPUs for the larger categories (such as chairs). Smaller categories (such as benches) will give reasonable results in one day, and may not benefit from training any longer.

A.3 Experiments

A.3.1 Dataset details

Dataset size. The size of the dataset in COSEG is only 300 vases. For the ShapeNet categories, there are 6778 chairs, 1813 benches, 3173 sofas, 3514 cars, and 8436 tables.

Non-manifold edges. Our method may contain non-manifold edges. We ran some statistics on our data and found that 98.9% of edges in the ground-truth (chair) dataset are manifold, and 98.5% of edges in the generated tetrahedral meshes are manifold.

A.3.2 Additional Quantitative Results

We present additional quantitative results evaluating TetGAN in unconditional generation using ShapeNet benches, sofas, tables, and COSEG vases. We train high-resolution (approx. 61^3) models for benches and vases, and low-resolution (approx. 41^3) models for sofas and tables, choosing resolution based on perceived necessity (i.e., our belief of how much resolution is necessary to fully capture details of the category). We compare against unsimplified versions of the baselines.

FID Recall that we use we use deep features from a PointNet++ [122] architecture trained for classification to calculate FID score. Table A.1 shows the scores for TetGAN and baselines across multiple categories. We observe that TetGAN beats baselines in both the Shapenet

Model	Benches	Sofas	Tables	Vases
OccNet	10.91	1.54	2.47	7.42
ShapeGAN (GAN)	7.46	1.57	0.48	5.46
ShapeGAN (VAE)	7.74	8.30	6.48	20.51
TetGAN	2.67	3.35	2.13	2.57

Table A.1: FID↓ scores of TetGAN and baselines on various categories. TetGAN is best in 2 categories and competitive in the rest despite generating volumes (at a smaller object resolution) while baselines only generate surfaces.

benches and COSEG vases categories, models which are trained at a high grid resolution. TetGAN is able to remain competitive with the *surface-based* baselines, while task of generating *coarse volumetric* meshes i.e., tetrahedral meshes with fewer surface faces.

Model	Benches	Sofas	Tables	Vases
OccNet	0.0014	0.0017	0.0053	0.0021
ShapeGAN (GAN)	0.0017	0.0013	0.0047	0.0022
ShapeGAN (VAE)	0.0003	0.0004	0.0032	0.0009
TetGAN	0.0014	0.0017	0.0045	0.0023

Table A.2: Variety metric ↑ for TetGAN and baselines on various categories.

Variety Recall we evaluate variety by averaging the Chamfer distance between the k closest pairs of generated meshes within a sample of size N (choosing $k = 25$, $N = 250$ in practice). Table A.2 shows the variety metrics for TetGAN and baselines. We observe that TetGAN again beats baselines in 2 categories, sofas and vases and is relatively competitive in the other two.

Model	Benches	Sofas	Tables	Vases
ShapeGAN (VAE)	0.0010	0.0018	0.0014	0.0048
OccNet	0.0095	0.0118	0.0222	0.0187
TetGAN	0.0046	0.0024	0.0029	0.0055

Table A.3: Reconstruction accuracy ↓ for TetGAN and baselines. TetGAN and OccNet both have effective generative capabilities while still enabling reconstruction/encoding. TetGAN offers a significant improvement to OccNet in all categories.

Reconstruction Recall we use Chamfer distance to measure reconstruction quality. Table A.3 shows the average Chamfer distance between reconstructed meshes and ground truth meshes for TetGAN and baselines over multiple categories. The VAE version of ShapeGAN, which is a pure VAE, beats the alternatives handily. However, TetGAN offers a significant improvement over OccNet in all categories.

A.3.3 Additional Qualitative Results

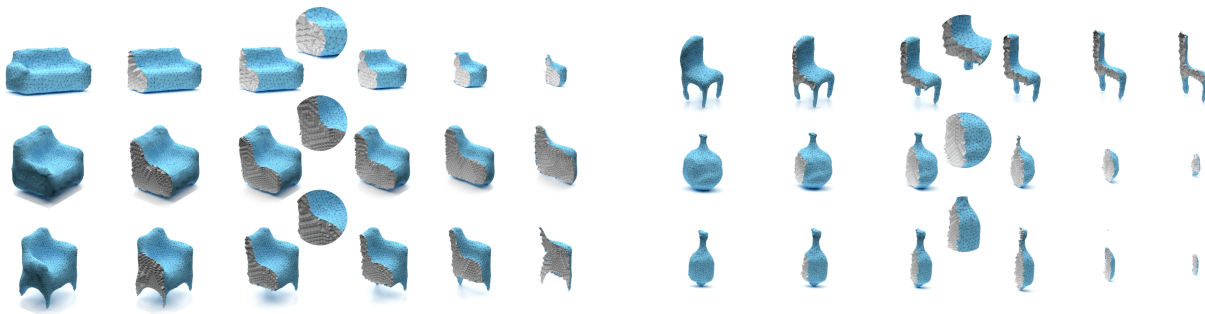


Figure A.1: TetGAN generates meshes with solid interiors.

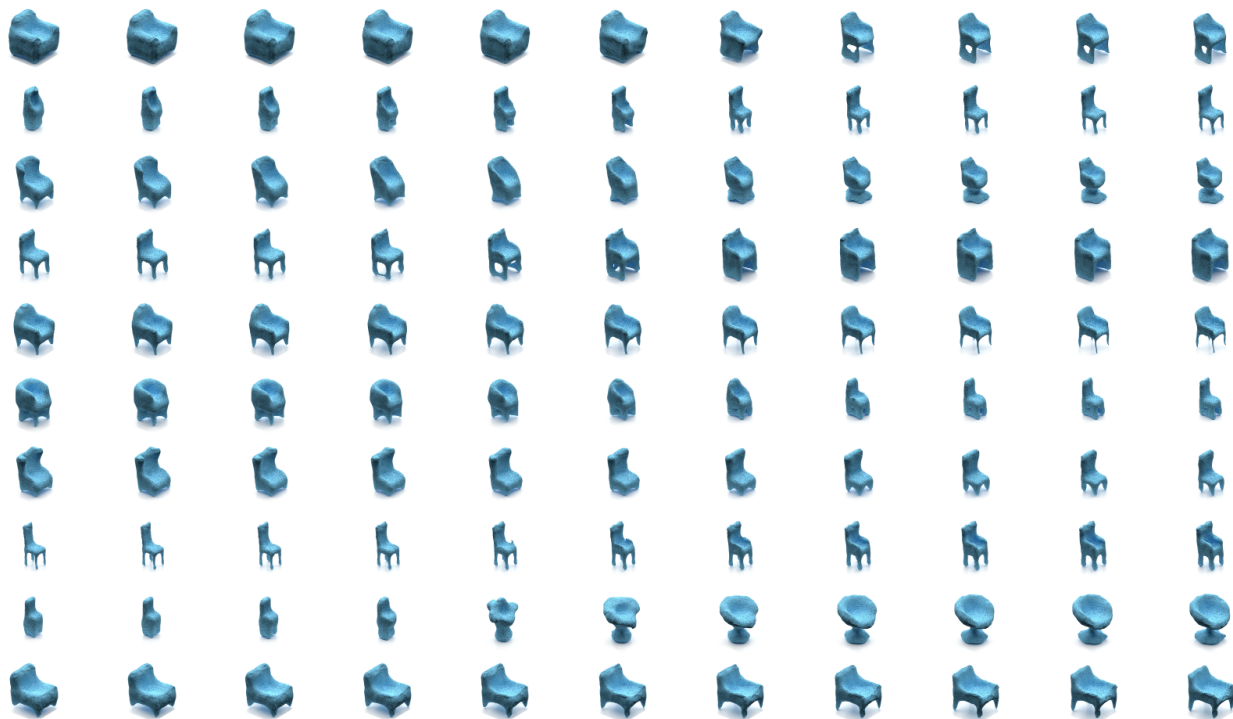


Figure A.2: Additional chair interpolations.

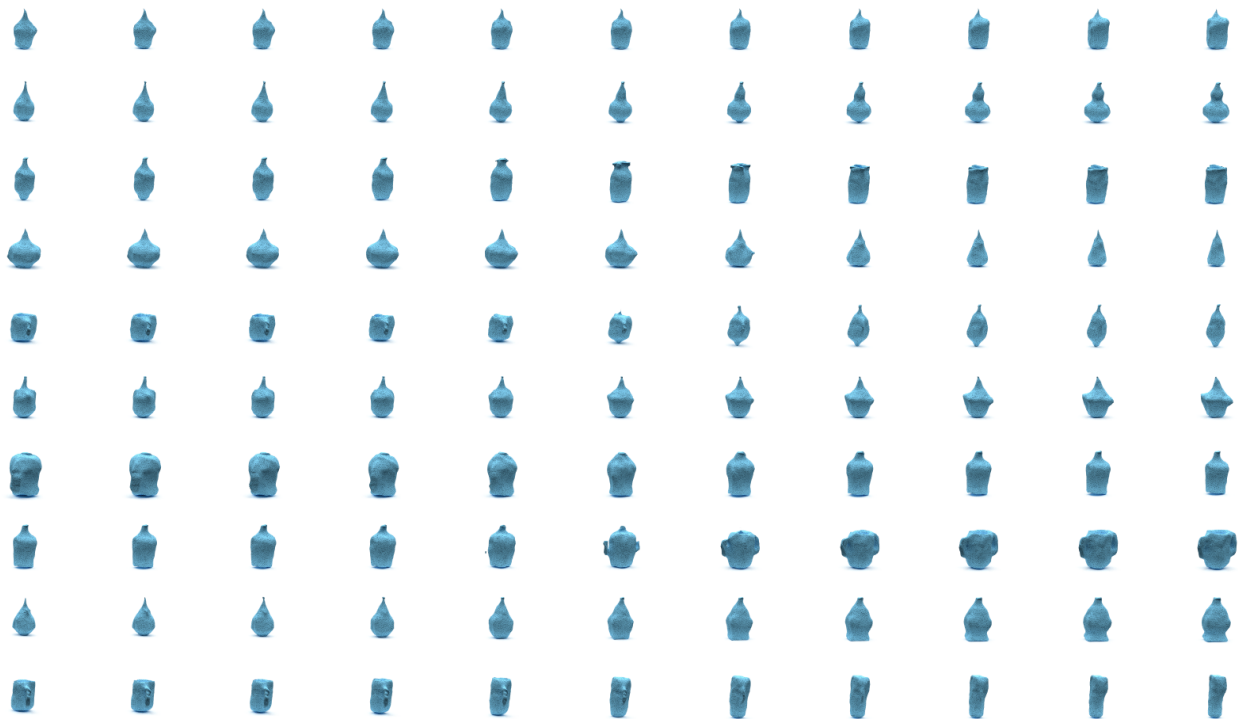


Figure A.3: Additional vase interpolations.

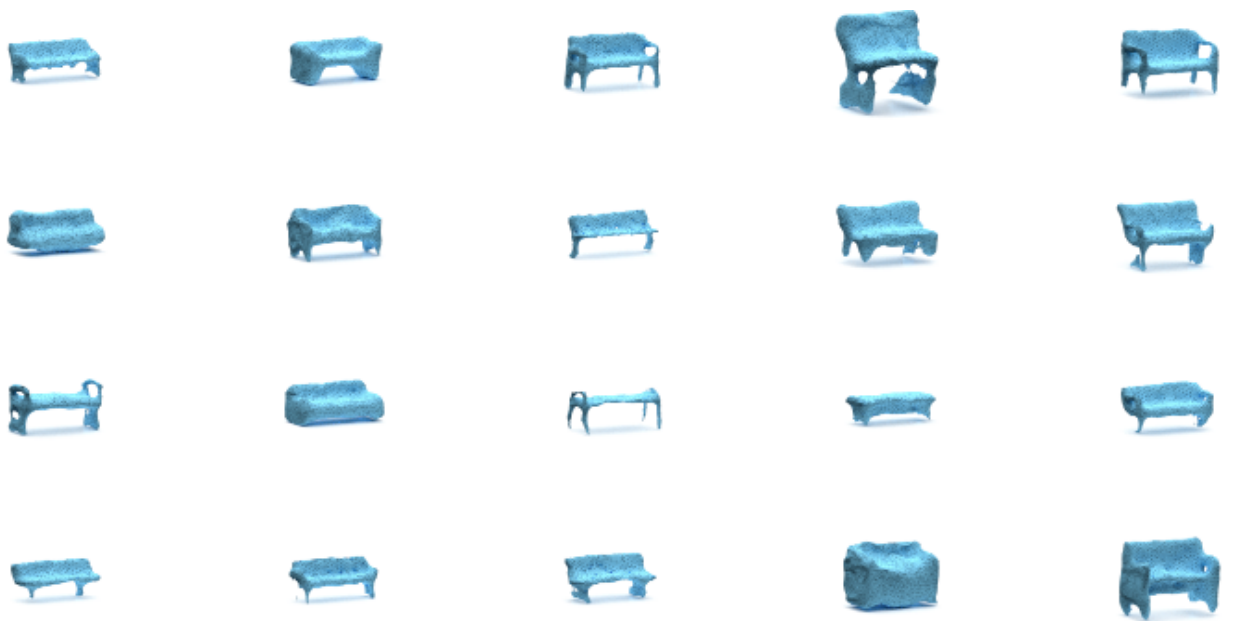


Figure A.4: Unconditionally sampled benches.



Figure A.5: Unconditionally sampled cars. We observe that although the model is able to learn the coarse shape of a car, the discretization is not fine enough to capture details.

APPENDIX B

GENERATIVE 3D EDITING WITHOUT OPTIMIZATION

B.1 Mesh Editing Metrics

Mesh Editing Comparisons						
ViT-L-14			ViT-BigG-14			
	Ours	Instant3Dit	MagicClay	Ours	Instant3Dit	MagicClay
CLIP Cos.Sim. $\uparrow$	0.323	0.303	0.285	0.337	0.309	0.286

Table B.1: **Mesh Editing Comparisons:** We provide CLIP cosine similarity metrics of our proposed MaskedLRM approach against two recent 3d editing techniques.

Large scale quantitative editing comparisons are difficult as there is no standard benchmark. In Table B.1 we compare against MagicClay [7] and Instant3Dit [8], generating edits using a text prompt. We then use the same text input for generating our conditional edited views and use CLIP similarity averaged across multiple views to measure the faithfulness of each method to the prompt.

B.2 Additional Implementation Details

Our model implementation details are based mostly on [171]. Our model tokenizes 16×16 sized patches. The token embedding size and transformer width are 1024. The transformer depth is 24 layers. Each attention and cross attention module use multi-head attention with 16 heads. Our model uses LayerNorm and GeLU activations with a Pre-LN architecture.

We trained our models using 64 H100 GPUs with 80GB of RAM each. We use an AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$ and a weight decay of 0.01. During stage 1 of training, we train for 30 epochs. For each batch consisting of 12 shapes, we randomly sample the number of input views for the batch uniformly at random between 6 and 8, not including

the 1 view for the conditional view. We use another 4 views for supervision. Over the first 1500 iterations, we linearly warm up to a peak learning rate of $4e - 4$ and then use cosine learning rate decay. During stage 2, to account for increased rendering costs, we reduce the batch size to 8 shapes. We train for 20 epochs, with a peak learning rate of $5e - 6$.

B.3 Additional Qualitative Examples

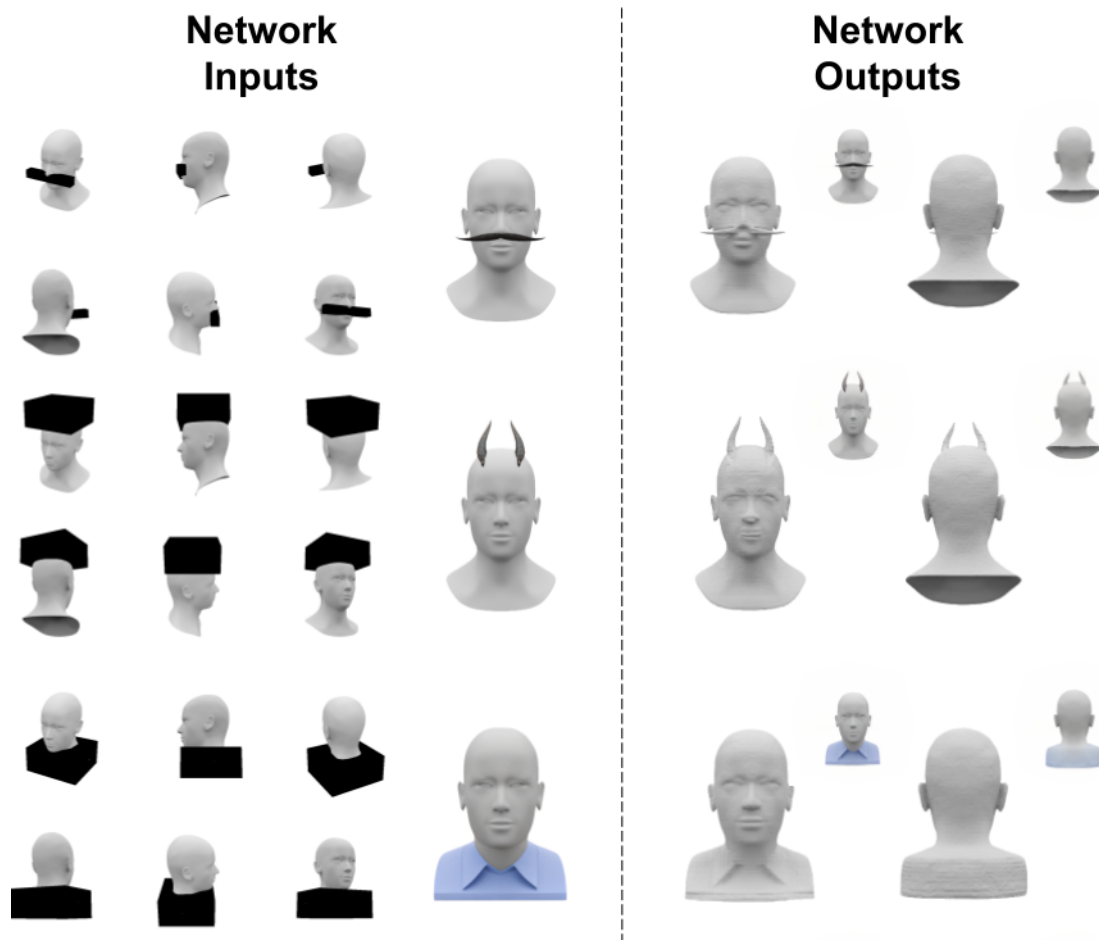


Figure B.1: **Additional Qualitative Examples:** Additional qualitative examples editing a person’s head. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.



Figure B.2: **Additional Qualitative Examples:** Additional qualitative examples editing a chair and a full human. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.