3D SHAPE GENERATION AND MANIPULATION

by

Alara Dirik

M.S., Computer Engineering, Boğaziçi University, 2022

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in MS in Computer Engineering
Boğaziçi University
2022

ACKNOWLEDGEMENTS

I would like to thank to my thesis supervisor Assoc. Prof. Emre Uğur and co-superviser Dr. Pınar Yanardağ for their guidance, valuable advice and support throughout my studies. I would also like to thank my family and friends who supported me and kindly offered to review and give feedback on my thesis.

This research has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TUBITAK (Project No: 118c321). We also acknowledge the support of NVIDIA Corporation through the donation of the TITAN RTX GPU and GCP research credits from Google.

ABSTRACT

3D SHAPE GENERATION AND MANIPULATION

Computer graphics, 3D computer vision and robotics communities have produced multiple approaches to represent and generate 3D shapes, as well as a vast number of use cases. These use cases include, but are not limited to, data encoding and compression, shape completion and reconstruction from partial 3D views. However, controllable 3D shape generation and single-view reconstruction remain relatively unexplored topics that are tightly intertwined and can unlock new design approaches. In this work, we propose a unified 3D shape manipulation and single-view reconstruction framework that builds upon Deep Implicit Templates [1], a 3D generative model that can also generate correspondence heat maps for a set of 3D shapes belonging to the same category. For this purpose, we start by providing a comprehensive overview of 3D shape representations and related work, and then describe our framework and proposed methods. Our framework uses ShapeNetV2 [2] as the core dataset and enables finding both unsupervised and supervised directions within Deep Implicit Templates. More specifically, we use PCA to find unsupervised directions within Deep Implicit Templates, which are shown to encode a variety of local and global changes across each shape category. In addition, we use the latent codes of encoded shapes and metadata of the ShapeNet dataset to train linear SVMs and perform supervised manipulation of 3D shapes. Finally, we propose a novel framework that leverages the intermediate latent spaces of Vision Transformer (ViT) [3] and a joint image-text representational model, CLIP [4], for fast and efficient Single View Reconstruction (SVR). More specifically, we propose a novel mapping network architecture that learns a mapping between the latent spaces ViT and CLIP, and DIT. Our results show that our method is both view-agnostic and enables high-quality and real-time SVR.

ÖZET

3 BOYUTLU MODEL JENERASYONU VE MANİPÜLASYONU

Bilgisayar grafikleri, 3B bilgisayarlı görü ve robotik komüniteleri 3B şekilleri ifade etmek, modellemek ve jenere etmek için pek çok yöntem geliştirmiş ve kullanım alanı oluşturmuştur. Bu kullanım alanlarının bazıları 3B şekilleri kodlama ve sıkıştırma ve kısmi 3B şekillerin tamamlanması olup; 3B şekil manipülasyonu ve tek resimden 3B şekil üretme hala görece az çalışılmış konulardır. 3B şekil manipülasyonu ve tek resimden 3B şekil üretme konuları birbirleriyle ilişkili olup, bu konular üzerindeki çalışmalar yeni dizayn metodolojilerini mümkün kılacaktır. Bu tezde 3B şekil manüpülasyonu için bir çerçeve geliştirip, baz model olarak Deep Implicit Templates'i [1] kullandık. Bu model 3B şekil üretmenin haricinde, aynı kategoriye ait şekiller için topolojik benzerlik haritaları çıkarabilmektedir. Bunun için öncelikle 3B şekil temsil formatlarını ve ilgili araştırmaları anlatmakla başlayıp, daha sonra geliştirdiğimiz metodları ve çerçeveyi anlattık. Tezimizde ana veriseti olarak ShapeNetV2'yi [2] kullanarak Deep Implicit Templates katmanları içinde denetimli ve denetimsiz yönler bulduk. Deneylerimiz sonucunda, PCA uygulayarak bulduğumuz denetimsiz yönlerin pek çok lokal ve global özelliği temsil ettiğini gördük: sandalye yüksekliği, araba uzunluğu, dizayn trendleri ve şekil alt kategorileri gibi. Ek olarak, ShapeNetV2 meta verisini ve öğrenilmiş şekil kodlarını kullanarak eğitilmiş lineer SVM modelleriyle başarılı şekilde denetimli manipülasyon yapabileceğimizi gösterdik. Son olarak, eğitilmiş bir Vision Transformer (ViT) [3] modelinin ve eğitilmiş bir birleşik resim-yazı temsili modeli olan, CLIP'in [4] ara katmanlarını kullanarak gerçek zamanlı ve efektif bir tek resimden 3B şekil üretme metodu geliştirdik. Geliştirdiğimiz metod, ViT ve CLIP'in ara katmanlarıyla DIT'in ara katmanları arasında köprü görevi görmekte olup, poz ve perspektiften bağımsız olarak gerçek zamanlı ve yüksek kalitede resimleri 3B şekillere dönüştürebilmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
CLIP	Contrastive Language-Image Pretraining
CNN	Convolutional Neural Networks
DIT	Deep Implicit Templates
EMD	Earth Mover's Distance
LIDAR	Laser Imaging Detection and Ranging
PCA	Principal Components Analysis
SDF	Signed Distance Function
SLAM	Simultaneous Localization and Mapping
SVM	Support Vector Machine
ViT	Vision Transformer

1. INTRODUCTION

Generative Adversarial Networks (GANs) [7] are generative models that revolutionized computer vision. Today, GANs are widely used for various visual tasks due to their success in generating high-quality images. To name a few, image generation [8,9], image manipulation [10], image denoising [11, 12], image super-resolution [13] and domain translation [6] are only some of the many creative uses of generative models.

Despite the impressive capabilities of GANs, these advances are overwhelmingly limited to 2D image generation tasks whereas 3D generation and manipulation remains challenging. This is mainly because one of the greatest challenges to the development of generative models for 3D content is converging on the right representation as it is often not possible to apply image-based methods to 3D data. Whereas RGB images have become the quasi-standard in 2D vision tasks, common 3D representations such as point clouds, meshes and other compact surface representations all pose significant problems during training, which makes it difficult to agree on a common representation. Preprocessing images to train models is relatively simple compared to preprocessing 3D shapes, as the input 3D shapes require to be aligned and scaled, in addition to each 3D representation posing its unique additional challenges. For example, direct extension of 2D pixel grids to 3D voxel grids [14] to train convolutional networks significantly limits resolution due to the high memory demands of 3D convolutions. Mesh datasets, on the other hand, often require remeshing of mesh objects to create an even number of faces and vertices across the dataset, which often leads to a loss in detail depending on the shape topologies and the remeshing method used. Similarly, while point clouds are very popular for generative tasks [15–17], they are limited in their ability to model sharp features or high-resolution color textures. Hence, the quality, flexibility and fidelity of 3D encoding and generation approaches are limited by the representation used. To address these issues, several alternative representations such as NeRF [18] and DeepSDF [5], as well as hybrid methods collectively known as neural implicit representations, have been proposed in the last 3 years.

These methods use neural networks to define implicit surface or volume representations. While most recent works on 3D shape generation adopt implicit representations to generate high-quality shapes and overcome the shortcomings of previous work, these methods are difficult to deploy in practice with no means of real-time generation and manipulation. We note that the majority of previous work focus solely on high-quality and high-fidelity 3D generation without exploring 3D manipulation and controllable generation.

In this thesis, we focus on the task of interactive and controllable 3D generation, and propose a unified framework for unsupervised and supervised manipulation of 3D generative models, one-shot co-segmentation and Single View Reconstruction (SVR). Unlike previous work that directly operates on explicit 3D representations, our proposed framework solely operates within the latent space of a base 3D generative model, Deep Implicit Templates (DIT) [1] to enable real-time reconstruction and manipulation. More specifically, we use PCA and linear SVM models trained on shape metadata to find unsupervised and supervised manipulation directions within the latent space of DIT. Moreover, we propose a highly effective and fast method that leverages Vision Transformer (ViT) [3] and CLIP [4], a joint image-text representation model trained on a huge dataset, to map images to the latent space of DIT in a view-agnostic manner. Furthermore, we propose a one-shot category-level consistent segmentation method that directly capitalizes the information generated by DIT. While we apply the proposed manipulation and SVR methods on DIT, we note that these methods are applicable to any differentiable generative model that either learns an implicit representation and has intermediate latent layers that can be manipulated, or directly used to generate explicit 3D representations. For all experiments, we use the car, airplane, sofa and chair categories of the ShapeNetV2 dataset [2] and compare our proposed SVR method to state-of-the-art competitor methods: AtlasNet [19] and IM-NET [20]. We note that we are not able to perform comparative experiments for shape manipulation tasks as previous works that do not involve per-shape optimization schemes confine shape manipulation to simple interpolation.

We show through extensive experiments that our framework achieves fast and highly effective manipulations and reconstruction while preserving generated shape quality and fidelity. We list our main contributions as follows:

- We propose unsupervised and supervised methods to discover meaningful, global and effective semantic directions in the latent space of a 3D generative model, DIT.
- We show that our method is able to find several distinct and fine-grained directions for a variety of categories such as cars, airplanes, chairs, and sofas.
- We propose a novel, view-agnostic SVR network architecture that can map input images to the learned latent space of a 3D generative model, enabling image-based reconstruction and direct manipulation.
- We propose a simple yet highly effective category-level, one-shot 3D segmentation method.

2. RELATED WORK

2.1. Classical 3D Shape Representations

One of the most pressing research questions in the domain of 3D vision is how to best represent 3D data. Unlike the 2D vision domain, where RGB pixel representations have become the standard for research, 3D vision research still uses a wide variety of explicit (e.g. point clouds, voxels, meshes), implicit and hybrid shape representations, mostly due to each representation having its own set of advantages and disadvantages.

One of the most popular explicit shape representations is the point cloud, which represents shapes as a set of 3D coordinates in (x, y, z) format. As many sensors used in the industry, such as LIDAR and depth cameras, output point clouds, the point cloud format is extensively used in popular 3D vision problems such as reconstruction of 3D shapes [21], 3D object classification [22, 23], and segmentation [23]. While popular, point clouds provides no information on how the points are connected, are order invariant and often yield noisy reconstructions and generations. Another very popular 3D representation is the mesh format, which describes each shape as a set of triangles, where each edge of each face is a connection between two vertices. While meshes are better suited to describe the topology, they pose significant problems such as the question of how to deal with shapes with unequal number of vertices and faces. Voxel format is yet another popular 3D representation format that describes objects as a fixed-sized volume occupancy matrix. While this dense grid structure is particularly suitable for CNN-based architectures, voxel format requires high-resolution in order to describe fine-grained surface descriptions and hence, cannot be generalized to articulate shapes.

2.2. Neural Implicit Representations

Finally, there have been a large number of neural implicit representations proposed in the recent years. These methods seek to overcome the shortcomings of the classical 3D shape representations as described in Section 2.1. Neural implicit representations represent 3D shapes as learned functions that map 3D coordinates to a signed distance function (SDF) [5], or a binary occupancy value [20, 24]. While SDF values denote how far a given point (x,y,z) is from the closest normal surface point, occupancy fields tell if the query point (x, y, z) is within the shape surface boundaries. Hence, both SDF based and occupancy based representations aim to create a lightweight and continuous shape representation that is infinitely scalable in resolution. Despite their significant advantages, a drawback of implicit representations is they require aggressive sampling and querying of 3D coordinates in order to construct explicit surfaces. Neural volume rendering methods such as NeRF [25] are tangent to this problem as NeRF proposes a method to synthesize views with user specified camera intrinsics and perspective without explicitly constructing a surface. In practice, this is achieved by generating radiance fields along specified ray paths. We note that while works that use implicit representations for 3D shape generation have achieved tremendous success, the model outputs require conversion to an explicit representation such as the mesh or point cloud format in order to be used in downstream application. This conversion process is often performed via ray-marching or spherical tracing, which are not only time-consuming but also non-differentiable.

2.3. 3D Shape Generation

3D generation and reconstruction methods can be classified into two broad categories based on their main motivations: shape encoding and generation, and reconstruction from single or multi-view images. Shape encoding is arguably the most popular 3D vision task as learning compact 3D representations is important in a vast number of applications and areas such as object recognition, data compression and retrieval, navigation (e.g. SLAM) and symbol generation.

Previous work on shape encoding use a wide array of shape representations and methods. PointNet [22,23], for example, processes point cloud data and extracts global shape features via max-pooling operations for object classification. The shape features extracted by PointNet are also widely used as an encoder for point and occupancy generation networks [20]. As described in Section 2.1, an important limitation of working with point clouds is that they consist of unordered points and converting them to meshes often fail to produce watertight surfaces. Other works such as [26] propose using convolutional restricted Boltzmann machines for unsupervised learning of mesh features, which can be used for training linear and non-linear 3D object classifiers. AtlasNet [19] proposes representing 3D shapes as a collection of parametric surface elements and tries to learn a mapping from 2D squares sampled from images or point clouds coupled with learned latent shape features to 3D points. However, their approach does not exploit locality information, resulting in lower fidelity generation. Works such as [1,5,24] propose conditional frameworks to learn implicit surface or volume functions while simultaneously learning an encoding for each training shape. Similarly, IM-GAN [20] proposes to use an implicit decoder IM-NET in conjunction with features learned by a latent-GAN model [15] to yield a general 3D generative model. Because the outputs of this method and related methods are implicit in the network weights, high-resolution results typically cannot be visualized or exported in real time. Other work, [27,28] propose overfitting a neural implicit function per shape to encode and reconstruct objects as needed. Another branch of research focuses on learning continuous data distributions for novel shape synthesis. For example, SP-GAN [17] proposes an unsupervised sphere-guided generative model for directly synthesize point clouds. Others propose various CNN-based GAN architectures [15,29,30] to generate novel shapes in the point cloud format.

2.4. 3D Shape Reconstruction

Another important line of research in 3D computer vision is 3D reconstruction from single or multi 2D views, or partial 3D views, often in the point cloud format.

As 3D data is often acquired from LIDAR sensors in the form of points clouds, and thus suffer from issues such as occlusion, most work on 3D reconstruction focuses on reconstruction from partial point clouds or even multi-view partial point clouds. Works such as PoinTr++ [31] and IM-NET [20] operate on point clouds and employ iterative refinement schemes that seek to minimize the average Chamfer distance between the completed point cloud to the ground truth shape. Moreover, SDF and occupancy based representations have been successfully used for reconstruction and 3D shape completion using partial point clouds [24, 32, 33]

Another line of work seeks to reconstruct 3D shapes from single or multi-view images. For example, works such as [34, 35] leverage the grid-like architecture of the voxel format to train CNNs on single and multi 2D views respectively to reconstruct the 3D object in voxel format. Other work such as Pixel2point [36] and AtlasNet propose CNN-based architectures that directly maps pixels of a single-view image to 3D points to create point cloud and mesh outputs respectively. Similarly, work such as DeepI2P [37] tries to register each pixel in a single-view to a point in a LIDAR acquired point cloud for SLAM purposes. In our work, we show that it is possible to perform higher quality reconstructions without a time-consuming optimization scheme by restricting the mapping process to the dense latent space of the generative model instead of working directly with meshes or point clouds. For the sake of brevity, we exclude multi-view reconstruction from related work.

2.5. Latent Space Manipulation

While 3D shape manipulation is a largely unexplored topic, several methods have already been proposed to exploit the latent space of generative models for image manipulation. These methods are collectively known as latent space manipulation methods and can be further divided into two categories: supervised and unsupervised methods. Supervised manipulation methods often leverage pre-trained binary classifiers to optimize an input sample in order to enhance or remove the target attribute, or to learn meaningful style directions that can be applied to unseen samples.

Alternatively, they use small labeled datasets to train binary classifiers in an effort to learn a boundary for the target attribute [38, 39]. Other work investigate whether it is possible to find unsupervised directions in the latent space of pre-trained GANs that encode meaningful attributes [40, 41]. For example, GANSpace [42] proposes a simple unsupervised manipulation method that applies principal component analysis (PCA) [43] to randomly selected latent vectors of the intermediate layers of BigGAN [44] and StyleGAN [8] models. The components obtained are then used to manipulate random or real images embedded in the latent space. SeFA [45] proposes a closed-form optimization method to directly optimize the intermediate weight matrix of a pre-trained GAN. Typically, these methods use the found directions to modify the image semantics by shifting the latent code by a certain amount in the identified direction to strengthen, negate, or remove the attribute of interest.

3. METHODOLOGY

In this chapter, we provide an overview of our proposed manipulation and SVR framework. We start by giving background information on DeepSDF, DIT, ViT and CLIP, and then describe our unsupervised and supervised manipulation methods. We then describe our SVR framework and real-to-synthetic image translation module in detail, as well as our one-shot co-segmenntation method.

3.1. Background on DeepSDF and DIT

A Signed Distance Function (SDF) is a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ where $d = f(\mathbf{x}, \mathbf{y}, \mathbf{z})$ denotes the shortest signed distance from a point $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ to a watertight surface $\mathcal{S}$ belonging to a shape. Moreover, the sign of d indicates whether $(\mathrm{x}, \mathrm{y}, \mathrm{z})$ is within or outside the surface, and points on the surface are implicitly represented with f as formulated as

$$\mathcal{S} = \{f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0\}, \tag{3.1}$$

where $\mathcal{S}$ denotes the shape surface. DeepSDF proposes learning an implicit surface representation such that a neural network f_θ learns an SDF via training and can be used to reconstruct an object via coordinate sampling and forward propagating the samples through f_θ. Hence, for a given 3D coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, the signed distance can be computed with $f_\theta((\mathbf{x}, \mathbf{y}, \mathbf{z})) = \widehat{d}$. The parameters θ are optimized with the loss $J(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z}),d}\mathcal{L}\left(f_\theta((\mathbf{x}, \mathbf{y}, \mathbf{z})), d\right)$, where d is the ground-truth signed distance and $\mathcal{L}$ is a traditional distance metric such as the L^2 distance. However, this formulation is not sufficient to learn a single function that encodes multiple shapes. Hence, DeepSDF proposes an auto-decoder architecture to learn a function $f_\theta((\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s}))$ where $\mathbf{s} \in \mathbb{R}^m$ is an input shape feature vector that is unique to each shape. In practice, the shape feature vectors are randomly initialized for each shape separately and learned via the back-propagation of a 3D loss during training. Overall architecture of DeepSDF to learn a single or multiple shapes is shown in Figure 3.1.

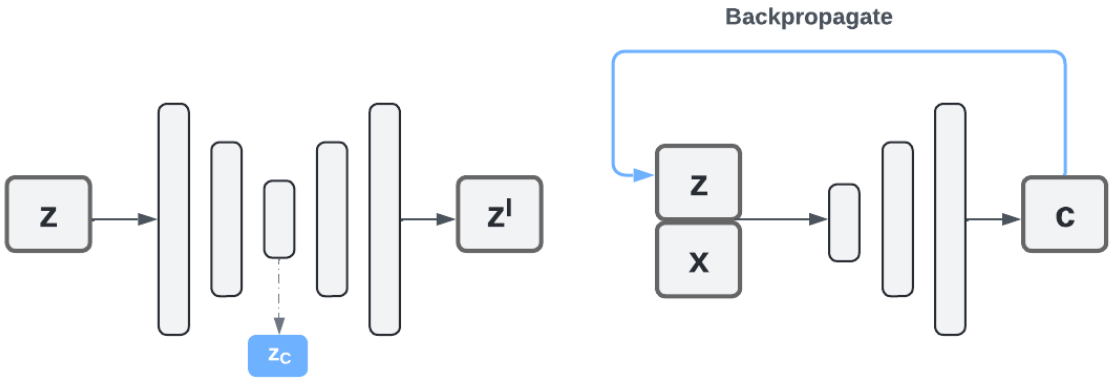

Figure 3.1. An overview of the DeepSDF architecture [5]: DeepSDF network predicts the SDF value of the input 3D query location. The network can be trained (left) on a single shape or (right) conditioned with a latent vector that allows DeepSDF to model a large space of shapes, where each latent vector encodes a unique shape and optimized through training.

Deep Implicit Templates (DIT) is an extension of DeepSDF and proposes using an intermediate mapping between the input (x, y, z, s) to a global shape template $\mathcal{T}$ to learn the average shape of a shape category (e.g. cars, planes) to improve encoding quality and prevent generation artifacts. In practice, this intermediate mapping takes the form of an LSTM network where the input is an ordered sequence of sampled coordinates (x, y, z) and a shape feature vector s. While DIT produces significantly better results than DeepSDF in categories such as *cars*, where there is a prominent common shape template, it tends to produce more artifacts in more diverse categories such as *chairs*.

3.2. Background on ViT and CLIP

In our work, we leverage two pre-trained vision models: ViT and CLIP. Vision Transformers (ViT) [3] are powerful models that have emerged as an alternative to CNN-based architectures for visual tasks such as image classification and object detection, and quickly achieved state-of-the-art results in numerous visual tasks. Furthermore, ViT-based models have shown better robustness to occlusions and adversarial attacks, as well as less texture bias compared to CNN-based architectures [46].

The ViT architecture represents an image as n non-overlapping patches $\{p_i\}_{i\in 1..n}$. These patches are linearly projected to a multi-dimensional space and concatenated with positional embeddings that are learned through training to generate *spatial tokens*. An additional token is added in order to represent global visual properties and then the tokens are forward propagated through the transformer's encoder layers, where each layer $l \in L$ consists of Multihead Self-Attention modules (MS), normalization layers (L_{Norm}) and MLP blocks with skip connections. This process is formulated as

$$\begin{aligned} \hat{\mathcal{T}}^l &= \mathsf{MS}(\mathsf{L_{Norm}}(\mathcal{T}^{l-1})) + T^{l-1} \\ \mathcal{T}^l &= \mathsf{MLP}(\mathsf{L_{Norm}}(\hat{\mathcal{T}}^l)) + \hat{\mathcal{T}}^l, \end{aligned} \tag{3.2}$$

where $\mathcal{T}^l = \left[\sqcup_0^l, \ldots, \sqcup_n^l\right]$ denote layer l's output tokens. Furthermore, each Multihead Self-Attention block linearly projects T^l into queries, keys and values as follows

$$q_i^l = W_q^l \cdot \sqcup_i^{l-1}, \quad k_i^l = W_k^l \cdot \sqcup_i^{l-1}, \quad v_i^l = W_v^l \cdot \sqcup_i^{l-1}, \tag{3.3}$$

where l: $\{k_i^l, q_i^l, v_i^l, t_i^l\}$ is the set of key, query, value and token that directly corresponds to the image patch p_i at each layer l. The richness of ViT's latent space and ViT's SOTA results across several vision tasks led to the exploration of using deep ViT features as general dense visual descriptors [47, 48].

In addition to ViT, we leverage CLIP [4] - a powerful joint image-text representation model - to map single-view images to the latent space of pre-trained DIT, our base 3D generative model. CLIP is a multi-modal contrastive learning framework that consists of a text encoder module and an image encoder module. The two modules are trained simultaneously with the joint objective of mapping the image and text inputs to a shared embedding space. To achieve this, CLIP is trained on over 400 million image-text pairs with the goal of maximizing the similarity between the embeddings of matched image and text instances and minimizing those of unmatched pairs, resulting in a powerful joint representation model. Moreover, previous work has shown that CLIP is robust to pose variation and much more sensitive to content differences. In our work, we exploit this finding and use CLIP, to extract dense pose invariant features, which are then used to train a view-independent SVR network.

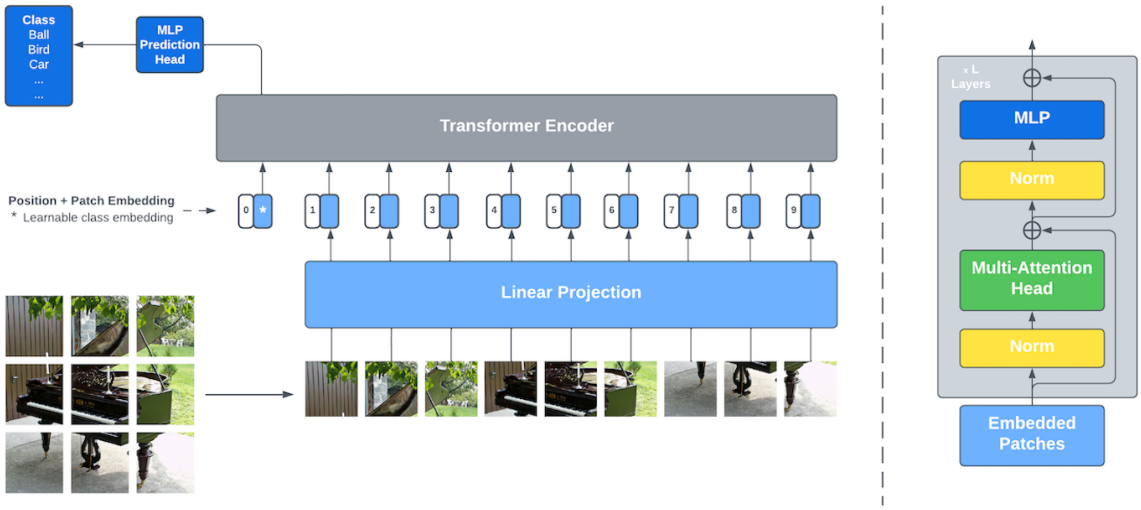

Figure 3.2. An overview of the ViT architecture [3]: Each image is split into n non-overlapping patches and are sub-sequentially forward propagated through positional embedding and transformer layers. Each image patch p_i has a corresponding set of unique features in each layer: $\{k_i^l, q_i^l, v_i^l, t_i^l\}$. Each feature set at each layer can be used as a dense feature descriptor.

We note that it is also possible to directly train CLIP-like multi-modal models to jointly represent encoded 3D shape, text and image triplets. For example, AudioCLIP [49] and Wav2CLIP [50], are recent CLIP-based multi-modal works that seek to jointly encode audio and image data. While AudioCLIP extends CLIP with a third branch that jointly encodes audio along with text and images, Wav2CLIP uses the trained CLIP model to distill the weights of the image encoder and train a new CLIP model for joint audio-image representation with audio-image streams extracted from videos, removing text from the equation. We refrain from directly training a CLIP-based model as our base DIT models are trained on each shape category separately and not suitable for contrastive learning schemes.

3.3. Unsupervised Manipulation

We start by exploring unsupervised manipulation for controlled 3D generation by finding global linear directions in the latent space of DIT.

To do this, we use the randomly initialized and learned latent vectors that encode each shape. Given a set of n latent vectors p with shape $(1, 512)$, we concatenate the vectors to create a matrix of shape $(n, 512)$ and apply PCA to this matrix to obtain k number of principal components with each component representing a new transformed axis that is a linear combination of original features. Moreover, the principal components are ranked by how much variation they encode, hence higher ranked components represent more prominent and less entangled directions.

After acquiring the principal components, we can manipulate a randomly generated latent vector or the encoding of a shape by directly applying a component as a style direction vector to steer the generation and enhance or remove the target attribute encoded by the component. This transformation is formulated as follows

$$p^{'} = p + (\alpha \times t \times \mathrm{PC}_i), \tag{3.4}$$

such that α denotes the step size, t denotes the number of steps, and PC_i is the i_{th} principal component (direction) used. Once the new latent vector $p^{'}$ is computed, it can be used as input to DIT to generate a mesh with the desired attributes. We note that the attributes encoded by each principal component is determined upon manual visual inspection.

3.4. Supervised Manipulation with SVMs

The ShapeNet dataset does not only consist of shapes but also include metadata such as text descriptions and sparse labels for 3D models. We leverage this metadata to create binary labeled datasets that consists of learned latent vectors for each shape and their corresponding value for the target attribute (e.g. 0 or 1 for "sports car"). Inspired by InterfaceGAN [51], we assume that for any binary attribute, we can find a hyperplane in latent space that separates positively labeled shape encodings from negatively labeled ones. Our manipulation method is as follows: Assume we have a attribute scoring function $f_S : p \rightarrow \mathcal{S}$, where $\mathcal{S} \subseteq \mathbb{R}^m$ represents the semantic space with m attributes and p represents the encoded shape vectors learned by DIT.

We can bridge the latent space p and the semantic space $\mathcal{S}$ with $\mathbf{s} = f_S(p)$, where s and p denote the attribute scores and the sampled latent code respectively. Assuming $\mathcal{S}$ changes continuously, we can train a Support Vector Machine (SVM) on latent vectors and corresponding binary labels, and use the normal vector to the separation hyperplane to enhance or diminish the target attribute. More formally, assume we found a separation hyperplane that has a unit normal vector $\mathbf{n} \in \mathbb{R}^d$. Then, we can formulate the distance between a sample latent vector p and this hyperplane as follows

$$\mathrm{d}(\mathbf{n}, p) = \mathbf{n}^T p, \tag{3.5}$$

where the distance may be negative depending on which side of the hyperplane p lies. We note that when p is near the separation hyperplane and is steered toward or away from the hyperplane, both the distance and the semantic evaluation vary accordingly. In practice, we perform this control / manipulation with:

$$p^{'} = p + (\alpha \times t \times \mathbf{n}^T), \tag{3.6}$$

where α is the step size, t is the number of steps, and $\mathbf{n}^T$ is the unit normal vector to the hyperplane. $p^{'}$ can be used as input to DIT to generate shapes that are manipulated in the target semantic direction. In our experiments, we train five linear SVMs on various attributes where sufficient number of labels are available within the metadata: “sports car”, “jeep” for cars, “cantilever” for chairs and “L shaped” for sofas.

3.5. Single-View Reconstruction with Multi-Modal and Positional Cues

In this section, we propose our multi-module Single-View Reconstruction framework: 3D-LatentMapper. Before describing our framework, which is trained in two stages, we first describe our dataset generation process, as well as the preprocessing steps to extract features from pre-trained ViT and CLIP models. Finally, we describe our framework, and training procedure, as well as our real-to-synthetic image translation module in detail.

3.5.1. Dataset Generation

We train our proposed SVR network using sets of matching synthetic images and encoded 3D shapes for each shape category. Based on the size of the ShapeNetV2 categories, intra-category diversity of shapes and availability of real-world image datasets, we choose to use *car* and *airplane* categories and create synthetic datasets for these categories as follows. We first encode each 3D shape into DIT's latent space by initializing a latent vector and optimizing it via back-propagation. To this end, we randomly sample $10,000$ 3D coordinates from a unit sphere and forward propagate the sampled coordinates through the trained DIT model to acquire their corresponding predicted SDF values. We then compute the L1 loss between the ground truth and predicted SDF values of the query coordinates, and optimize the latent vector via back-propagation using the Adam optimizer. After converging to an optimized latent vector, we use the latent vector to compute the SDF values of randomly sampled coordinates and acquire its corresponding mesh via ray-marching of SDF values. We save the latent vectors (encoded 3D shapes) along with the meshes for quantitative evaluation purposes. In order to train a view-agnostic SVR model, we render N=9 views of each generated mesh from a fixed camera view by rotating each mesh $\mathcal{M}$ by 0, 30, 60, 90, 120, 150, 180, 210 and 240 degrees around the y-axis. Our reasoning in rendering generated meshes instead of ground truth meshes is generated meshes are better representatives of of DIT's learned latent space and are easier to learn from.

In addition to synthetic image datasets, we use two publicly available car and airplane image datasets to be used for real-to-synthetic image translation as described in Section 3.5.3: the Car Connection Dataset, which consists of 66,079 car images, and the FGVC-Aircraft dataset, which consists of 10,200 airplane images. While the FGVC-Aircraft dataset is a manually labeled dataset that does not require additional preprocessing, the Car Connection Dataset is automatically scraped from various websites and includes partial car images and detail views such as the car interior or zoomed-in interior and exterior detail images. As these images lead to non-optimal training results, we seek to filter them out.

In order to filter faulty/irrelevant images, we use a set of engineered text templates that describes the target category (e.g. "Photo of a car") and compute the average cosine distance between the CLIP embeddings of each image and the set of text templates. A full list of the text templates used can be found in Appendix B.1. This process is formulated as follows

$$\mathcal{D}_{\text{CLIP}} = \frac{\sum_{j=1}^{N} D_{\text{CLIP}}(\mathcal{I}, \mathcal{T}_{\text{j}})}{N}, \tag{3.7}$$

where $\mathcal{I}$ is an image from the dataset to be filtered, $\mathcal{T}_j$ is a text description of the target shape category embedded in a text template from a list of N templates. Once the average CLIP distance of each image is computed, we use the mean (μ) and the standard deviation (σ) of the computed distances to set a threshold such that images that yield distances that are larger than the threshold are filtered out. The filtering process outputs a binary label $\mathcal{C}_{\mathcal{I}}$:

$$\mathcal{C}_{\mathcal{I}} = \begin{cases} 0, & \text{if } D_{\text{CLIP}}(\mathcal{I}) > \mu + 2 * \sigma. \\ 1, & \text{otherwise,} \end{cases} \tag{3.8}$$

such that images with a distance score of more than two standard deviations from the mean are assigned a label of 0, and 1 vice versa.

3.5.2. Single-View Reconstruction

Next, we propose a novel SVR network architecture that leverages ViT and CLIP to map single-view images to the learned latent space of DIT. Our motivation is as follows: DeepSDF and DIT learn SDF functions and do not generate meshes. Instead, they rely on aggressive point sampling, SDF value retrieval and ray marching to generate a mesh. This mesh generation process is not only time-consuming (90 seconds per generation) but also non-differentiable, meaning we cannot optimize the generated shapes directly. In our work, we aim to overcome this issue by directly predicting the latent code that corresponds to the input image without generating a mesh. Additionally, the predicted latent vector can be directly manipulated with the unsupervised and supervised manipulation methods proposed in Section 3.3 and Section 3.4 respectively.

Given a set of 3D shapes $(m_0, m_1, ..., m_N)$ and a DIT model $\mathcal{G}$ trained on a target shape category, let $\mathbf{c_i} \in \mathcal{R}^d$ denote a d-dimensional latent vector that corresponds to the ith 3D shape such that $0 \leq i \leq N$. Our goal is to map a single-view image $\mathcal{I}_i$ to a latent vector $\mathbf{c_i}$, such that concatenating $\mathbf{c_i}$ with sampled 3D coordinates and feeding it into DIT and sub-sequently ray-marching the predicted SDF values reconstructs the corresponding 3D shape. To achieve this, we propose a two-stage mapping network architecture, which we denote as 3D-LatentMapper, and leverage ViT and CLIP as dense feature descriptors. We use CLIP and ViT instead of popular dense feature descriptors such as ResNet trained on large datasets for two main reasons: (i) CLIP has been shown to provide relatively pose invariant embeddings, which are crucial for view-agnostic 3D reconstruction from images, a notoriously difficult task; (ii) The ViT architecture provides a rich latent space with positional information and has been show to be more robust to variations and less biased to textures and lighting. This provides a significant opportunity to use ViT as a dense feature descriptor. As stated above, training 3D-LatentMapper requires preprocessing images to extract dense features followed by a two-stage training process.

For preprocessing, we use a pretrained CLIP model's image encoder to extract 512-dimensional embeddings of the input images: $CLIP(\mathcal{I}) = \mathbf{z}_{CLIP}$ such that $\mathbf{z}_{CLIP} \in \mathcal{R}^{512}$. Additionally, we use a pre-trained ViT model and leverage the findings of previous work to extract dense features from images by forward propagating the input image $\mathcal{I}_i$ through ViT. Unlike CNN-based models, where earlier layers represent coarse features such as edges or local textures, and deeper layers capture higher level concepts, it has been shown [48] that ViTs feature a different type of representation hierarchy. The latent spaces of the shallow layers mainly capture the position of the image patch, while this position bias is reduced in deeper layers in favor of semantic features.

As described in Section 3.2, ViT represents each image as a collection of n patches, where each image patch p_i is directly associated with its own query, key, value, and token in each layer l: $\{q_i^l, k_i^l, v_i^l, t_i^l\}$.

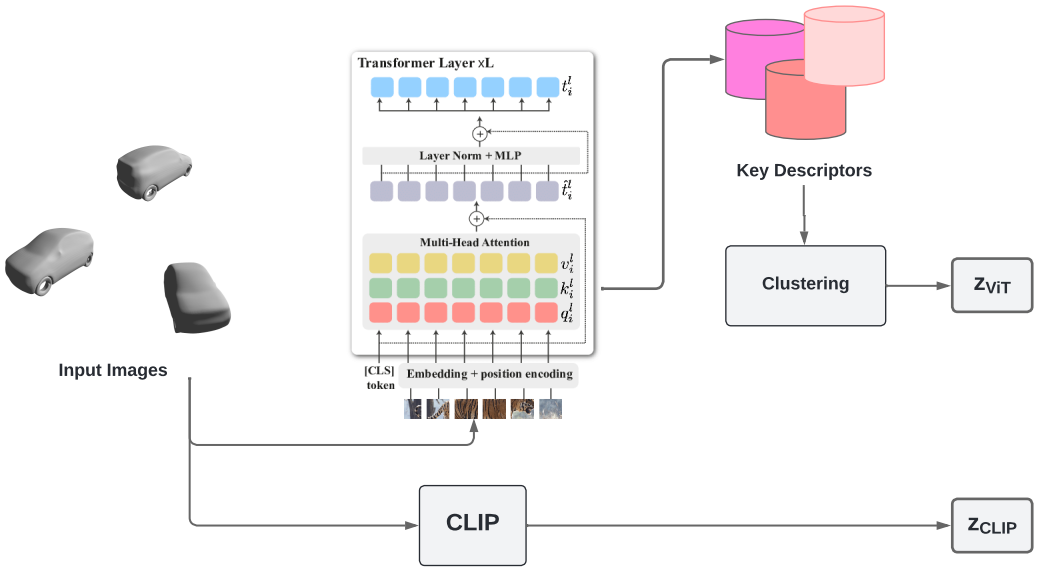

Figure 3.3. An overview of the preprocessing pipeline: each image is fed into CLIP's image encoder to extract image embeddings and into ViT to extract key descriptors from all layers. The extracted key descriptors from $N = 100$ images are then clustered to assign descriptor labels and create dense descriptors.

Leveraging the findings of previous work on ViT's latent spaces [48], we extract the key values k_i^l across all layers and all images to use as dense ViT descriptors, as they have been shown to be less sensitive to background noise in images. As the extracted keys are across multiple layers, it is not possible to train a model directly on the descriptors due to memory constraints. To overcome this issue: (i) we extract all key descriptors from $N = 500$ randomly sampled synthetic images and additionally subsample 20% of all descriptors; (ii) we adopt a bag-of-descriptors approach and cluster the descriptors using the K-means algorithms with the number of clusters set to $K = 20$. We note that clustering the descriptors of an image $\mathcal{I}_i$ yields a dense feature vector $\mathbf{z}_{ViT}$ such that $\mathbf{z}_{ViT} \in \mathcal{R}^{4235}$. An overview of our preprocessing pipeline is shown in Figure 3.3. After extracting descriptors from synthetic images, we train a three-stage, multi-module framework to map the descriptors of each image to the latent space of DIT. A diagram of our proposed framework is shown in Figure 3.4 and the architecture schemes of the two modules are provided in Appendix A.1 and A.2.

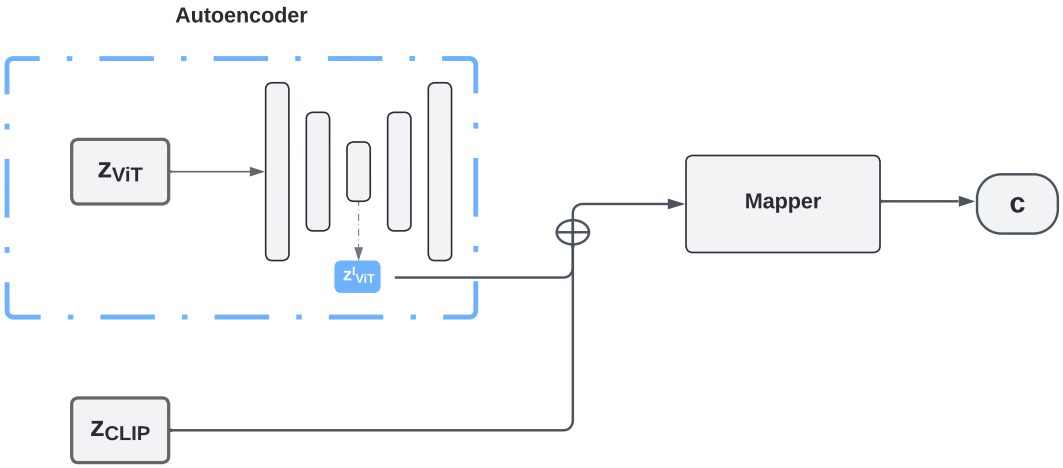

Figure 3.4. An overview of our framework: (i) an autoencoder is trained to compress and reconstruct ViT descriptors. Once trained, the autoencoder's weights are frozen and the model is used to extract dense features from the bottleneck layer. (ii) Compressed ViT descriptors and CLIP embeddings are concatenated and used as input to a mapping module that maps the combined dense features to the latent space of DIT.

3D-LatentMapper consists of an autoencoder with feed-forward layers and a feed-forward network. The autoencoder's encoder and decoder each consist of a single hidden layer and is trained on extracted ViT features $\mathbf{z}_{ViT}$ for 100 epochs. By training an autoencoder, we seek to further compress the dense ViT features to a 512-dimensional space such that, after the autoencoder is trained, we freeze the model weights and use it to extract latent vectors from the autoencoder's bottleneck. We denote these latent vectors as $\bar{\mathbf{z}}_{ViT}$ and concatenate them CLIP embeddings to create a combined descriptor $\mathbf{z} \in \mathcal{R}^{1024}$. Finally, we use the combined dense feature descriptors as input and corresponding DIT latent vectors c as output into a feed-forward network with two hidden layers with hyperbolic tangent activations. We train this model using an L1 loss and the Adam optimizer for 1000 epochs. Once trained, our proposed model learns a mapping $z \rightarrow c$, such that the predicted latent vector can be directly fed into DIT to generate a mesh or can be manipulated using our proposed unsupervised and supervised manipulation methods.

3.5.3. Real-to-Synthetic Image Translation

While our trained SVR model is highly effective at reconstructing 3D shapes from single-view synthetic images, a real-word use case would require using real images as input. Hence, we propose an additional module to convert real images to synthetic images, such that the translated images can be used as input to our 3D-LatentMapper framework. To this end, we use an existing bidirectional image translation framework, CycleGAN [6], which consists of two generators and two discriminators that are trained simultaneously.

More specifically, CycleGAN learns a bidirectional mapping between two unaligned image datasets of images, X and Y, using the generators $G : X \rightarrow Y$ and $F : Y \rightarrow X$. The two generators are trained jointly with discriminators D_x, D_y that encourage the generation of realistic images in the corresponding domain, with a cycle consistency loss $F(G(x)) \approx x, G(F(y)) \approx y$ for $x \in X, y \in Y$ in addition to the GAN losses of G and F. The cyclic loss ensures that translated images of each domain can be translated back into their original domain without a significant loss. For a schematic representation of CycleGAN, see Figure 3.5. The composite loss of CycleGAN is formulated as

$$\begin{aligned} L_{\text{CycleGAN}}\ (G, F, D_x, D_y) = & L_{\text{GAN}}\ (G, D_Y, X, Y) \\ & + L_{\text{GAN}}\ (F, D_X, Y, X) \\ & + \lambda_{\text{cycle}}\, L_{\text{cycle}}\ (G, F), \end{aligned} \tag{3.9}$$

where $L_{\text{cycle}}\ (G, F)$ is the cyclic loss and is given as

$$\begin{aligned} L_{\text{cyc}}\ (G, F) = & \mathbb{E}_{x \sim p_{\text{data}}\ (x)}\left[\|F(G(x)) - x\|_1\right] \\ & + \mathbb{E}_{y \sim p_{\text{data}}\ (y)}\left[\|G(F(y)) - y\|_1\right], \end{aligned} \tag{3.10}$$

such that cyclic loss enforces a realistic bi-directional mapping. In practice, we use CycleGAN to solely map real car images to synthetic images to be used as input to our proposed SVR model.

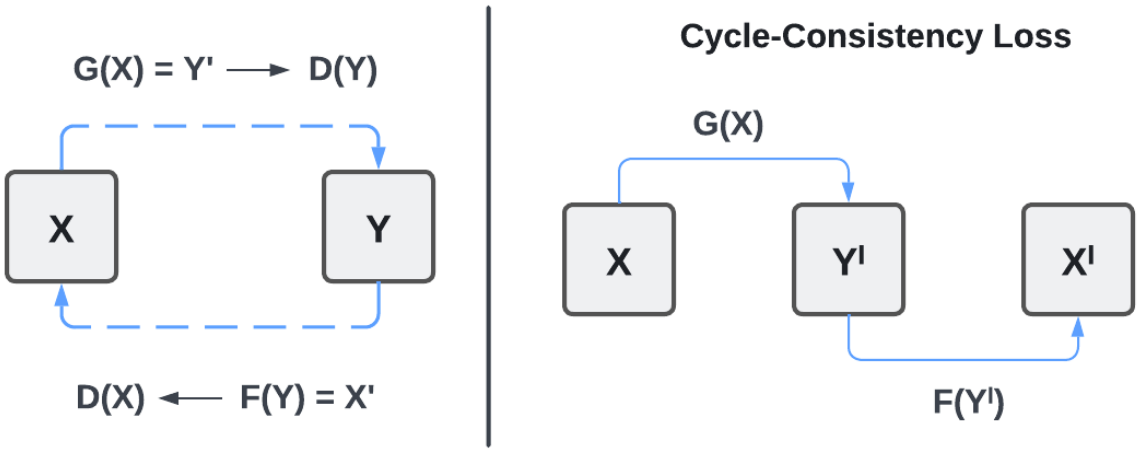

Figure 3.5. An overview of the CycleGAN architecture [6]: The model consists of two generators $G(X)$ and $F(Y)$, and two discriminators D_Y and D_X. D_Y and D_X encourages G and F to translate input samples from their respective domains to samples that are indistinguishable from their respective target domains.

3.6. One-Shot Segmentation

An important benefit of generating implicit dense correspondence maps is that it enables part co-segmentation on a category of 3D shapes such that shapes belonging to a specific category such as cars can be simultaneously segmented into semantically-consistent parts. Achieving consistent co-segmentation, in return, can enable part-based manipulations (i.e changing the size of the wheels of a car without changing irrelevant attributes). The intuition behind this is as follows: DIT learns an average category template via training and generates a dense correspondence map that encodes where each point of the generate shape correspond to in the learned template. Hence, surface points across different encoded shapes are said to correspond to the same semantic part if they have the same RGB value. In our work, we leverage this observation for one-shot segmentation. More specifically, we start by creating a 3D annotation map of a single randomly selected generated mesh in Blender, a popular 3D software, by selecting and assigning a color to generic parts such as doors, wheels, hood, and windows. Next, we unwrap both the original and annotated meshes via spherical projection to create 2D geometry images that are aligned with each other.

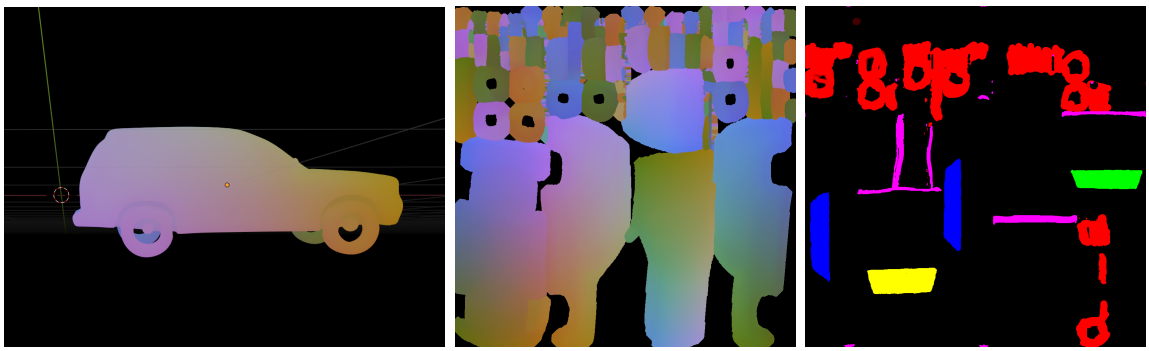

Figure 3.6. From left to right: a sample car mesh with projected correspondence heat map, its corresponding raw geometry image and annotated geometry image with discreet labels.

A sample mesh with projected correspondence heat map, and its corresponding unwrapped geometry image and annotations are shown in Figure 3.6.

We use the unwrapped geometry image and the corresponding segmentation map to map each RGB value to a discreet label and hash this mapping. To segment unseen shapes, we use the saved mapping to map RGB values to discrete labels: $f(r, g, b) = d$ where (r, g, b) denote the RGB value of a given generated mesh vertex and d denotes the assigned discreet label. In case of unseen RGB values (due to the varying number of faces and vertices), we identify the closest labeled mesh vertex and assign its label. If the vertex has multiple labeled neighbors, we use the rule of consensus to assign labels.

4. EXPERIMENTS

In this section, we start by describing our dataset, experimental setup and evaluation metrics. We then present the experimental results of the proposed unsupervised and supervised manipulation methods, SVR framework and one-shot segmentation method. We evaluate our results based on reconstruction fidelity and manipulation quality. Furthermore, we compare our SVR framework to two state-of-the-art methods - AtlasNet and IM-NET - and show that our method achieves higher quality reconstructions without compromising from speed.

4.1. Datasets

For the quantitative comparisons, we use the car, airplane and chair categories of the ShapeNetV2 dataset [2], consisting of 3509, 4045 and 3500 shapes respectively. We choose these three categories as they contain sufficiently structurally distinct shapes and demonstrate the range of our method. For Real-to-Synthetic image conversion, we use two publicly available datasets - the Car Connection Dataset, which consists of 66,079 car images, and the FGVC-Aircraft dataset, which consists of 10,200 airplane images. While the FGVC-Aircraft dataset is a manually labeled dataset that does not require additional preprocessing, the Car Connection Dataset is automatically scraped from various websites and includes partial car images and detail views (e.g. interior, texture details). As described in Section 3.5, we use CLIP to filter the dataset and create a dataset of 41,122 car images.

4.2. Evaluation Metrics

For manipulation experiments, we perform qualitative evaluation as it is not possible to compare our manipulation methods to previous work. For SVR tasks, we compare the generated shapes to ground truth shapes using Chamfer distance (CD) and Earth Mover's Distance (EMD) metrics.

CD and EMD metrics are computed by sampling 5000 points from the generated and ground truth shapes, matching the nearest points across the sets of points and computing their sum of Euclidean and Manhattan distances respectively. In addition to CD and EMD, we use the Light Field Distance (LFD) [52] as a visual similarity metric. To compute LFD, the generated and ground truth shapes are rendered from various camera angles. The rendered images are then encoded using Zernike moments and Fourier descriptors to compute similarity. For reconstruction from single-view real images, we perform qualitative and quantitative evaluation and compare our method to state-of-the-art SVR methods. Finally, we report the average inference time of ours and competing methods as real-time manipulation and reconstruction is critical for interactive applications, such as shape exploration and design.

4.3. Experimental Setup

For all experiments, we train generative, manipulation and reconstruction models on each shape category separately. We use ShapeNetV2 and we use create training, validation and test sets with a split ratio of 80-10-10% for each shape category. We use the official implementation of Deep Implicit Templates and train models from scratch for the "Car", "Airplane", "Chair" and "Sofa" categories. For DIT training, we create an improved preprocessing pipeline to eliminate non-watertight meshes, implement a new point sampling strategy to increasingly aggressive sampling near the surface and set the latent vector size the 512 and keep the rest of hyperparameters the same. To encode ground truth 3D meshes into the latent space of DIT, we use a fixed 500-step optimization process and use the Adam optimizer with the default hyperparameters.

For manipulation experiments, we use the Scikit-Learn implementations of PCA and SVM to find unsupervised and supervised directions within the latent space of DIT. We perform PCA and train linear SVM models on the training sets and use the found directions to manipulate encoded test shapes (latent vectors), and use the validation set to select the best SVM model. We set the manipulation step size α to 0.1 for PCA experiments and to 0.02 for SVM experiments.

To extract CLIP features, we use the official implementation of CLIP [4] and the pre-trained RN50x64 model, which uses ResNet [53] to extract dense features and train the image encoder module. Similarly, we use the official implementation of Dino-ViT [48] to extract key descriptors from input images. Due to memory constraints, we use Scikit-Learn's Mini Batch K-means implementation to cluster the extracted key descriptors. Furthermore, we apply random zoom-in and zoom-out transformations to all images to improve the generalization capabilities of our method to real images. For the autoencoder module of our SVR framework, we use L2 loss and the Adam optimizer with default hyperparameters and train the model for 100 epochs for each category. For the mapper module of our SVR framework, we use L1 loss and the Adam optimizer with default hyperparameters and train the model for 1000 epochs with early stopping enabled after 5 epochs to avoid overfitting. To map real images to synthetic images and vice versa, we train a bidirectional image-to-image translation model separately for car and airplane shape categories using the official implementation of CycleGAN [6]. Each model is trained on unaligned pairs of real images of the target category and rendered images of encoded 3D shapes. We train each CycleGAN model for 500 epochs with batch size of 8, using the default hyperparameters.

For comparisons with other 3D generation and SVR methods, we use the official Pytorch implementations of AtlasNet and IM-NET with the default hyperparameters. Additional details of single-view reconstruction experiments with AtlasNet and IM-NET are provided in Appendix A.3. We run all experiments, including model training and inference on a single NVIDIA TitanRX GPU.

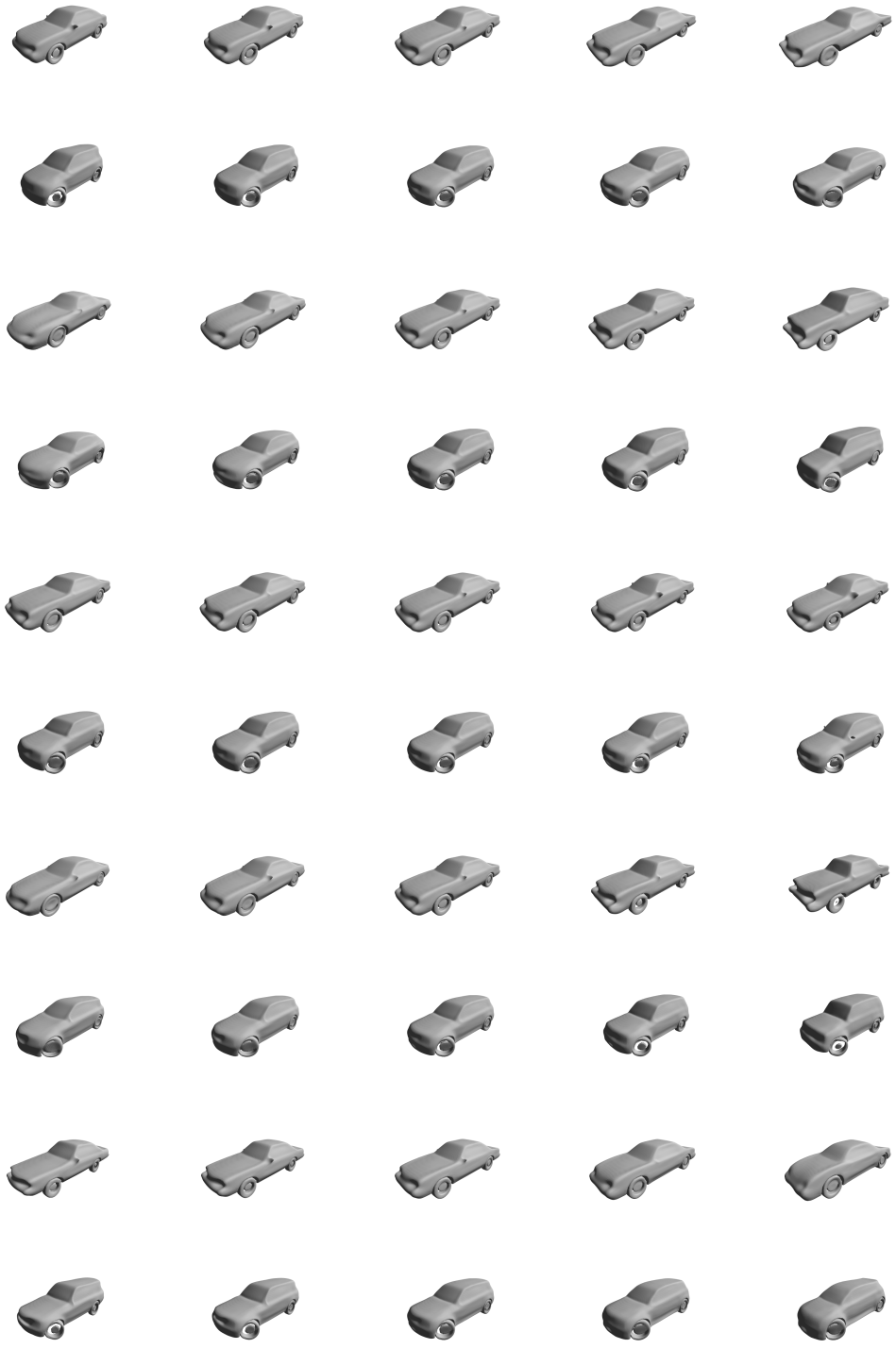

Figure 4.1. From top to bottom: top 5 principal components encoding *length*, *roof height*, *width*, *roundness* and *design era* applied to a pair of car meshes. The middle image of each row denotes the original, left and right images denote the manipulations applied in negative and positive directions with varying number of steps.

Figure 4.2. From top to bottom: top 4 principal components applied to a pair of test sofa meshes. For each row, the middle image denotes the original, left and right images denote the manipulations applied in negative and positive directions with increasing number of steps with strength parameter $\alpha = 0.1$. From top to bottom: components encode *width*, *L-shaped*, *leg/cushion height*, and *backrest height* respectively.

4.4. Unsupervised Manipulation

We start by applying PCA on the encoded shape vectors of a set training shapes and retrain the 10 highest ranking principal components. Next, we test the generalization capability of the found components by applying them in the negative and positive directions with varying number of steps with a fixed strength parameter α as described in Section 3.3. We show the manipulation results for the first 5 components on two random and unseen encoded car meshes in Figure 4.1. As can be seen in the figure, our method is capable of performing consistent and high quality edits without creating artifacts. We note that while the manipulation results indicate that the attributes encoded by each principal component are not completely disentangled (affecting multiple attributes), the component succeed in encoding important directions such as "length", "width", "roundness" and prominent subcategories (i.e. SUV type cars and classic cars from the 50s and 60s are common within the ShapeNetV2 Car category). In addition to cars, we perform the same experiment on the sofa category ShapeNetV2, a significantly less diverse category compared to cars, and present the results in 4.2. Similar to the first experiment, we show the manipulation results for the first 5 components on two random and unseen encoded sofas and interpret the components as *width*, *L-shaped*, *leg/cushion height*, and *backrest height* directions. We note that, unlike the car, which is more diverse, the shapes of the sofa category can significantly deviate from the average category shape template and feature outliers such as L-shaped sofas, daybeds and love seats. Furthemore, our experiments show that pushing latent vectors too far along the target direction causes unexpected changes, such as the over-application of the *width* direction transforming the input shapes into semi- U-shaped sofas, a very rare sub-category within the ShapeNetV2 dataset.

4.5. Supervised Manipulation

Next, we train linear SVMs using both manually and automatically (using ShapeNet metadata) labeled subsets of encoded shape vectors learned through training by DIT and their corresponding binary labels (i.e. 1 or 0 for "sports car").

However, we note that the metadata of ShapeNet is automatically scraped from 3D model captions and often inadequate or inaccurate. Due to lack of high-quality metadata, we train 4 SVM models for the binary attributes “sports car”, “jeep car”, “cantilever chair”, and “L shaped sofa”. Furthermore, we note that all SVM models are trained with less than 50 positive samples. To perform supervised manipulations, we apply the normal vectors to separation hyperplanes of trained SVMs as manipulation directions and move the encoded shape vectors along the normal vectors. We present the results of the supervised manipulation experiments in Figure 4.3.

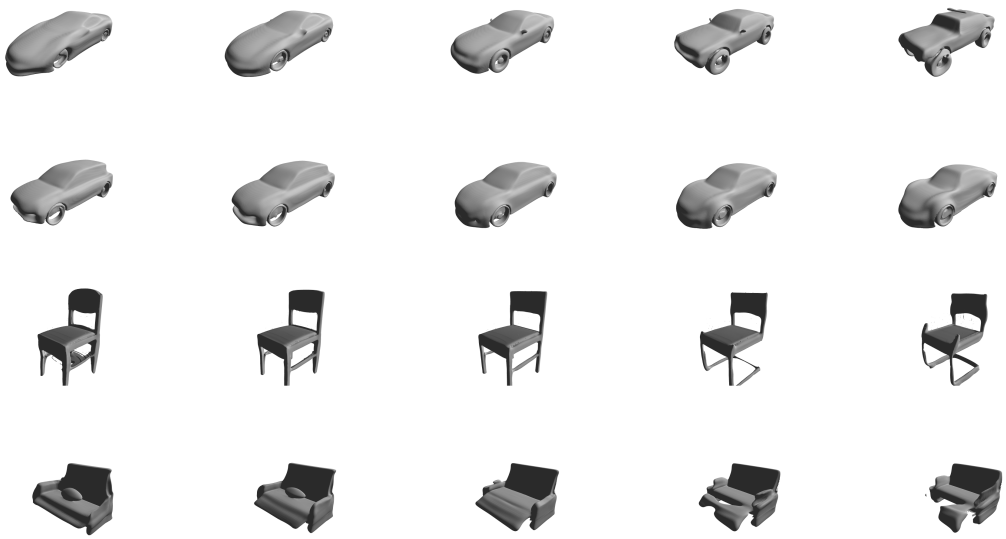

Figure 4.3. From top to bottom: “jeep”, “race car”, “cantilever chair”, “L-shaped sofa” manipulations. For each row, the middle image denotes the original, left and right images denote the manipulations applied in negative and positive directions with increasing number of steps with strength parameter $\alpha = 0.02$.

Despite using tiny datasets to train the SVM models, Figure 4.3 shows that our method can perform accurate and highly complex manipulations without creating significant artifacts. For example, the *Jeep* manipulation changes not only the overall shape of the car but introduces more subtle changes that are specific to the target manipulation, such as increasing car tire size and adding headlights and bumpers to the shape. Similarly, moving the encoded shape along the negative direction has the opposite effect, such as decreasing the car height and roof height.

On the other hand, manipulations such as *cantilever chair*, while successfully achieving the target edit, creates artifacts. We believe this is due to the chair category's lack of a common shape template as the height and width of chairs vary significantly within ShapeNet.

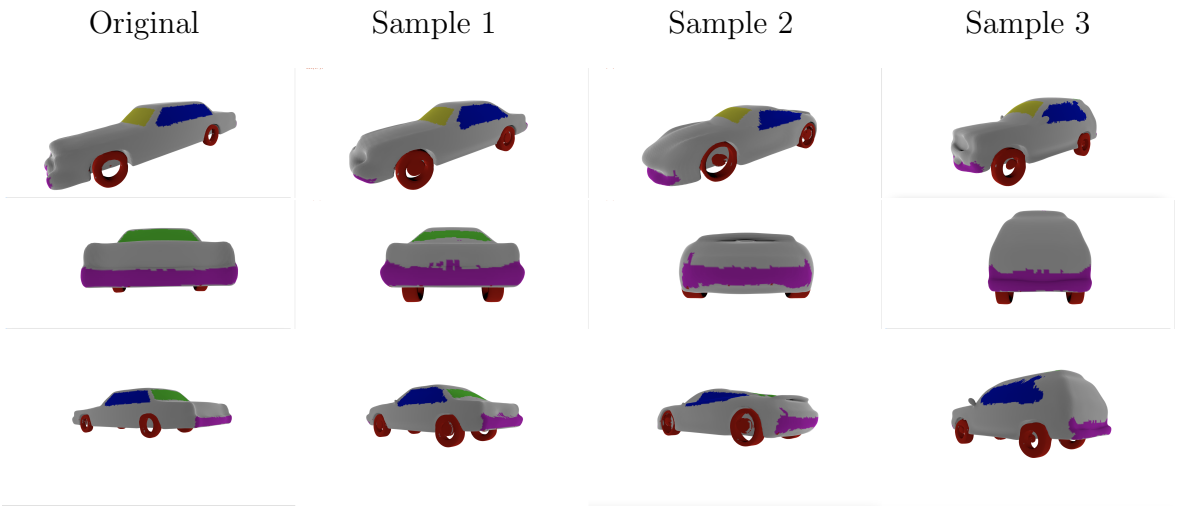

Figure 4.4. One-shot segmentation using corresponding heat maps. From top left clockwise: Original mesh, annotated original mesh, segmented mesh samples.

4.6. One-Shot Segmentation

Next, we perform one-shot segmentation by annotating the generated correspondence heat map of a single shape and applying the continuous-to-discreet mapping described in Section 3.6 to new shapes. For this experiment, we randomly select a generated car mesh and unwrap its correspondence heat map and assign discreet labels to 6 semantic regions: rear window, front window, side windows, bumpers, wheels and body. We then used to generated continuous-to-discreet mapping to segment unseen cars and present the results in Figure 4.4. As shown in Figure 4.4, our one-shot segmentation method can be successfully applied to generate 3D segmentation maps of unseen shapes. However, we note that our method requires a one-to-one part mapping constraint such that each part to be segmented needs to exist across all shapes within the category. For example, the bottom right corner of Figure 4.4 shows that shapes that are drastically different from the average shape, such as a sports car with a convertible top, might lead to partial segmentation failures.

We note that since generative models synthesize novel samples, we cannot quantitatively evaluate the segmentation outputs of generated meshes as there is no ground-truth data available. Furthermore, we note that our method is limited to shape categories with a prominent common shape template, such that samples of the category do not deviate too much from the template and have equal number of parts. Hence, our method is more suitable to categories such as *cars* and *airplanes* and less suitable to categories such as *chairs* and *handbags*.

4.7. Single-View Reconstruction

We evaluate our SVR model's performance on single-view reconstruction tasks and present both quantitative and qualitative results. We compare our method to two state-of-the-art 3D shape generation methods that use triangular mesh and implicit representation formats respectively: AtlasNet [19] (with 1-Sphere and 25-Squares templates) and IM-NET [20]. Our results show that our method achieves significantly better results both in terms of reconstruction quality and fidelity without compromising from speed. We note that we omit comparing our method to voxel-based SVR methods, which suffer from much lower visual fidelity due to resolution constraints inherent to voxel-based generative methods. Finally, we show that our method enables reconstruction from real images without any preprocessing.

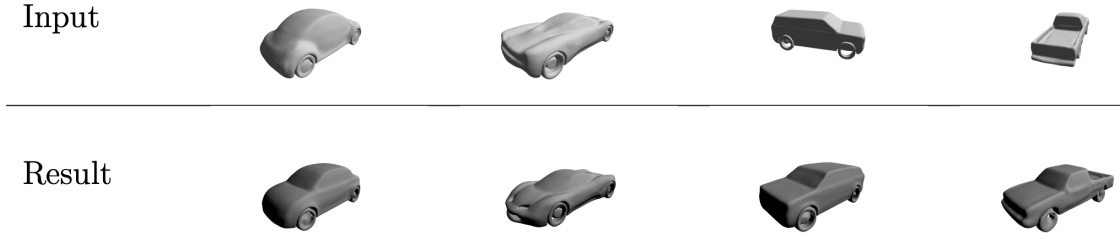

Figure 4.5. Qualitative Reconstruction Results on Cars: our method can successfully reconstruct 3D shapes from a variety of views.

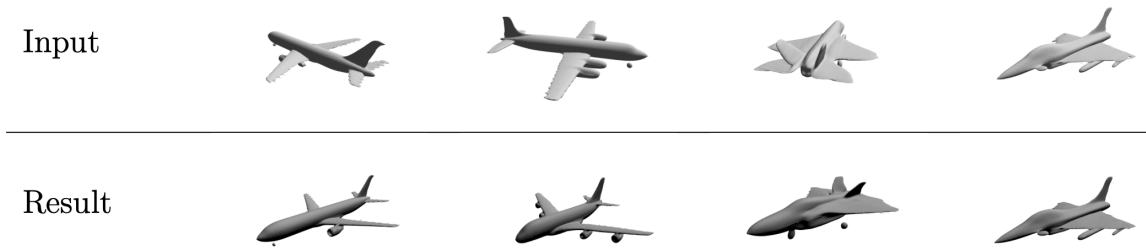

Figure 4.6. Qualitative Reconstruction Results on Planes: our method can successfully reconstruct 3D shapes from a variety of views.

We start with a qualitative results of single-view reconstruction and present reconstructions samples of our method for cars and airplanes in Figure 4.5 and Figure 4.6 respectively. As shown in Figure 4.5, our method not only reconstructs cars with high fidelity but also can infer unseen features such how the front of a sports car should look like. Similarly, a seemingly uninformative input image that depicts truck's trunk is sufficient to reconstruct a truck. We also include cases where training set includes no similar samples, such as the Tortoise Beetle car depicted on the far left. In this case, we see that our method reconstructs a similar car that is able to capture the high-level semantics of the input image such as the overall size of the car, roundness of the roof and hood and to a certain extent the profile curves around the wheels. While the results on planes produce slightly more artifacts, Figure 4.6 shows that our method can successfully reconstruct a variety of airplanes from different views. Note how our method is able to capture fine details such as the number of the engines and artifacts present in the input image.

Furthermore, we present a comparative qualitative analysis AtlasNet (with 1-Sphere and 25-Squares templates) and IM-NET using fixed-angle images and present the results in Figure 4.7. As shown in Figure 4.7, our method's performance significantly surpasses both AtlasNet 1-Sphere and 25-Squares methods and produces finer and accurate surface details compared to IM-NET. We note that while IM-NET's performance is close to our method in terms of capturing the overall shape, it produces more artifacts. Moreover, IM-NET is not view-agnostic and is trained on fixed-view synthetic images.

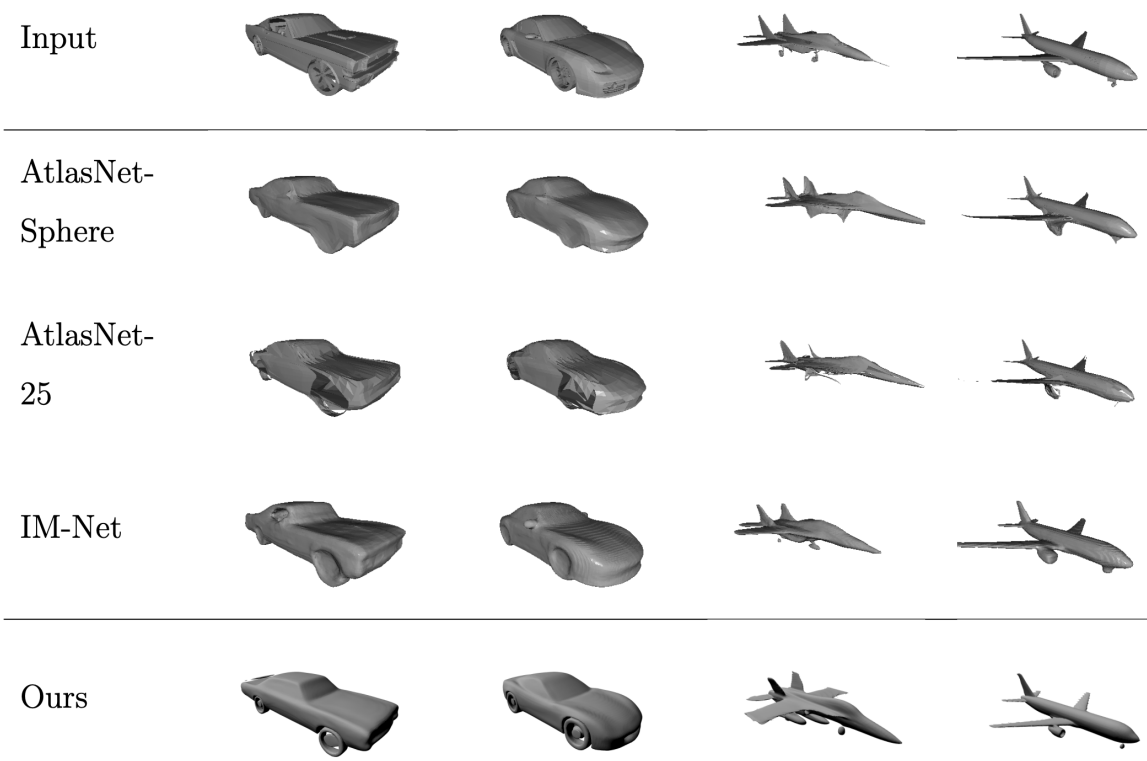

Figure 4.7. Qualitative Reconstruction Results: note that our method achieves the highest-fidelity 3D reconstruction and runs in real-time. Input images for SVR are shown in top row.

We also notice that even when our method cannot accurately reconstruct details such as the prominent front bumper of the car in the first input image, it successfully captures the overall design and style of the car such as the characteristic sharp profile curves. Similarly, we see that our method can successfully reconstruct fine details competitor methods cannot capture, such as the visible front landing gear seen in the commercial airplane image input. Interestingly, we see in the third sample of a fighter plane that our method introduces torpedoes to the reconstruction, a common feature within the fighter plane sub-category but not present in the input image. Hence, we deduce that our trained 3D-LatentMapper acts as a conceptual search engine and can map the input image to the closest/most relevant point within the latent space. In future work, we aim study how text and position supervised features are mapped to 3D dense features and how we can leverage them to train 3D generative models.

In order to validate our qualitative results, we next perform a quantitative analysis to measure the reconstruction fidelity all our method and all competitor methods. To this end, we use the ground-truth meshes that correspond to the input synthetic images and compute the Earth Mover's Distance(EMD) and the Chamfer Distance between each reconstructed shape and the ground truth shape, and report the average Chamfer's Distance and EMD values in Table 4.1.

Table 4.1. Quantitative results on Single View Reconstruction. Note that the CD values are multiplied by 10^3 and EMD are multiplied by 10^2.

	SVR error			
	Chamfer ↓		EM Dist. ↓	
	Car	Airplane	Car	Airplane
AtlasNet-Sph	92.48	46.17	21.03	14.32
AtlasNet-25	82.34	38.45	18.23	13.29
IM-Net	3.539	4.166	3.24	3.04
Ours	**2.438**	**3.144**	**2.62**	**2.16**

Results in Table 4.1 show that our method achieves the best CD and EMD on the car and airplane categories. We note that this is especially significant as both AtlasNet and IM-NET models are trained on fixed-view images for a fair comparison whereas our method is view-agnostic and trained on multi-view images. Given that airplanes often have many sharp and spiky features, view-agnostic SVR is a more difficult task for the plane category. These results also demonstrate that our method is qualitatively and quantitatively superior to competitor methods. In addition to quantitative and qualitative analysis, we measure and report the run times of our SVR framework and all competitor methods, and present the results in Table 4.3. For a fair comparison, we take both inference time to predict the latent vector and the time required for ray-marching to construct a mesh into account.

Table 4.2. Comparison of 3D generative methods by the capabilities and inference time. We include quantitative and qualitative comparisons with AtlasNet and IM-NET methods in our experiments.

	Ours	**AtlasNet**	**IM-NET**
Real-time generation	No	Yes	No
Real-time manipulation	**Yes**	Yes	No
High-quality surface	**Yes**	No	Yes
High-quality reconstruction	**Yes**	No	Yes
View-agnostic reconstruction	**Yes**	No	No

As shown in Table 4.3, our method is significantly faster than IM-NET while achieving better reconstruction results. While AtlasNet is much faster than our method due to directly outputting meshes, we note that our method is vastly superior to AtlasNet's performance in terms of reconstruction fidelity and quality.

Table 4.3. Comparison of inference time of our method and all competitor methods.

	Inference Time (s)
AtlasNet-Sph	0.078
AtlasNet-25	0.112
IM-Net	13.280
Ours	8.030

Finally, we perform SVR using unseen real images, a notoriously difficult task. We note that our real image datasets are solely filtered in order to exclude irrelevant images and are not processed in any other way (e.g. background removal). This process is described in Section 3.5.3 in detail. For this experiment, we use our Real-to-Synthetic image translation models and translate real images to synthetic images. We then feed the synthetic image into our 3D-LatentMapper framework to predict the corresponding latent vector and reconstruct a mesh via ray-marching. We present the input images, predicted synthetic versions and reconstruction results of sample cars in Figure 4.8.

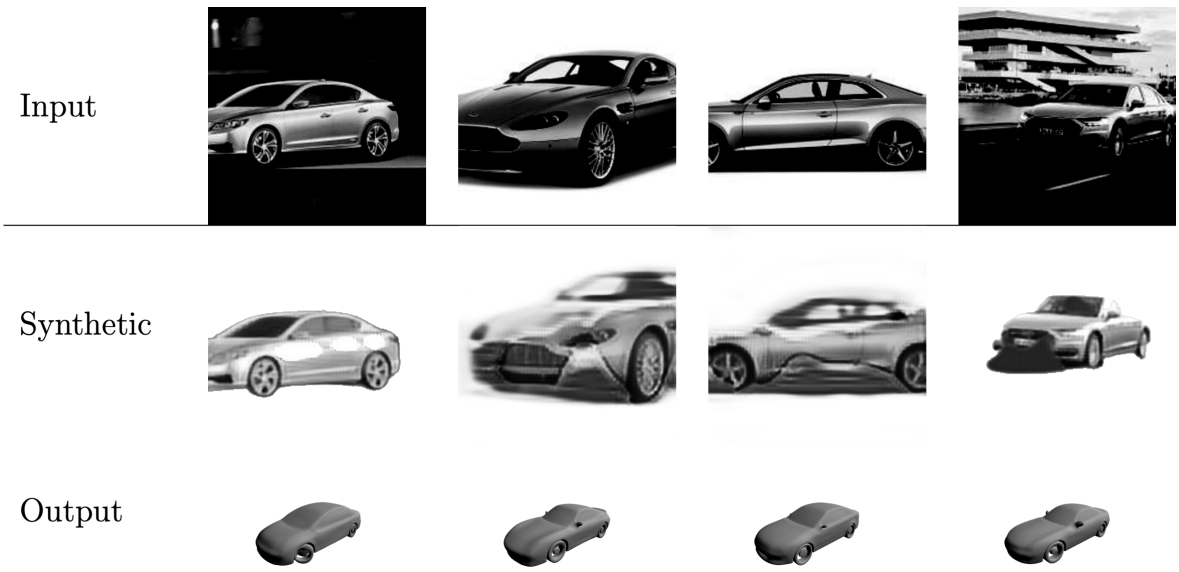

Figure 4.8. Reconstruction from real images: we first perform real-to-synthetic image translation and use the predicted synthetic images as input to our 3D-LatentMapper framework.

As shown in Figure 4.8, while our real-to-synthetic image translation model does not produce plausible synthetic images, it successfully learns to remove background clutter and change the color scale of the target object (cars) to resemble that of our synthetic image dataset. Furthermore, despite the shortcomings of the image translation module, we find that our 3D-LatentFramework can still capture important, definitive attributes of the input images. Notice how in the first sample, the reconstructed car resembles both a Sedan and Hatchback car. Similarly, the reconstruction of the second sample, which is an Aston Martin, features characteristic Aston Martin attributes such as the front hood curves. However, the reconstruction of the fourth sample is not nearly as successful and does not feature any of the prominent attributes in the input image. We note that this is partly due to reconstructing from unseen views and partly due to rather low-quality image translation.

5. CONCLUSION AND FUTURE WORK

5.1. CONCLUSION

We propose a unified 3D shape reconstruction and manipulation framework that directly operates within the learned latent space of a 3D generative model, DIT. While previous work 3D shape manipulation is limited to shape interpolation, we show that our method can find interesting, meaningful and disentangled directions with the latent space. Furthermore, we propose a novel Single View Reconstruction framework that is view-agnostic and can effectively map both synthetic and real images into the latent space of DIT in real-time while outperforming state-of-the-art models both qualitatively and quantitatively. To the best of our knowledge, our method is the first that leverages deep positional and multi-modal features for Single-View Reconstruction. Our key takeaways are as follows:

- As in image generative models, it is possible to find accurate and disentangled directions within the latent space of a 3D generative model. However, the disentanglement of the found directions largely depends on the diversity of the dataset.
- It is possible to find accurate supervised directions in a few-shot manner, indicating that most features are linearly separable within the latent space of DIT.
- CLIP is an effective tool to extract pose-invariant features and can be successfully used to train view-agnostic SVR models. Furthermore, key descriptors extracted from the intermediate layers of ViT act as highly effective dense descriptors.
- All of the proposed methods are limited by the capabilities and expressiveness of the base 3D generative model's latent space. Hence, we believe a deeper network with a richer latent space would lead to better results.

5.2. FUTURE WORK

While the results of our experiments are very promising, we note that the auto-decoder architecture of DIT makes it difficult to learn continuous data distributions and generate novel, realistic shapes since they seek to learn a shape encoding latent vector for each shape. Hence, we believe a 3D generative network with a GAN architecture could yield significantly better results.

Secondly, our current work is completely dependent on the information encoded in the intermediate latent space of models. Given that most implicit representation models feature shallow architectures that consist of 1 or 2 hidden layers, we believe we could strongly benefit from exploring 3D generative models with a richer latent space. Furthermore, we note that constructing a mesh via ray-marching takes around 8 seconds on a single GPU, making direct optimization of meshes very time-consuming. Hence, we propose an alternative method for faster and more fine-grained manipulation of meshes using DIT: extending DIT with a differentiable ray-marching method that would enable directly mapping a set of SDF values to a mesh in a differentiable manner. Finally, our experiments are limited to the ShapeNet dataset, which consists of rigid objects. In the future, we would like to extend our method to other rigid 3D datasets, as well as morphable 3D datasets such as human faces or human bodies.

REFERENCES

1. Zheng, Z., T. Yu, Q. Dai and Y. Liu, "Deep Implicit Templates for 3D Shape Representation", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1429–1439, 2021.

2. Chang, A. X., T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository", *ArXiv*, Vol. abs/1512.03012, 2015.

3. Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", *ArXiv*, Vol. abs/2010.11929, 2021.

4. Radford, A., J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krüger and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision", *ArXiv*, Vol. abs/2103.00020, 2021.

5. Park, J. J., P. R. Florence, J. Straub, R. A. Newcombe and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019.

6. Zhu, J.-Y., T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks", *IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

7. Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville and Y. Bengio, "Generative Adversarial Nets", *Conference on*

Neural Information Processing Systems (NeurIPS), 2014.

8. Karras, T., S. Laine and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019.

9. Karras, T., S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116, 2020.

10. Wang, T.-C., M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz and B. Catanzaro, "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8798–8807, 2018.

11. Wang, T., X. Yang, K. Xu, S. Chen, Q. Zhang and R. W. H. Lau, "Spatial Attentive Single-Image Deraining with a High Quality Real Rain Dataset", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12262–12271, 2019.

12. Li, S., I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. M. C. Junior, J. Zhang, X. Guo and X. Cao, "Single Image Deraining: A Comprehensive Benchmark Analysis", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3833–3842, 2019.

13. Sun, W. and Z. Chen, "Learned Image Downscaling for Upscaling Using Content Adaptive Resampler", *IEEE Transactions on Image Processing*, Vol. 29, pp. 4027–4040, 2020.

14. Wu, J., C. Zhang, T. Xue, B. Freeman and J. B. Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling", *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

15. Achlioptas, P., O. Diamanti, I. Mitliagkas and L. J. Guibas, "Learning Representations and Generative Models for 3D Point Clouds", *International Conference on Machine Learning (ICML)*, 2018.

16. Cai, R., G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely and B. Hariharan, "Learning Gradient Fields for Shape Generation", *European Conference on Computer Vision (ECCV)*, pp. 364–381, Springer, 2020.

17. Li, R., X. Li, K.-H. Hui and C.-W. Fu, "SP-GAN: Sphere-Guided 3D Shape Generation and Manipulation", *ArXiv*, Vol. abs/2108.04476, 2021.

18. Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", *European Conference on Computer Vision (ECCV)*, 2020.

19. Groueix, T., M. Fisher, V. G. Kim, B. Russell and M. Aubry, "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

20. Chen, Z. and H. Zhang, "Learning Implicit Fields for Generative Shape Modeling", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5932–5941, 2019.

21. Park, J., H. Kim, Y.-W. Tai, M. S. Brown and I.-S. Kweon, "High Quality Depth Map Upsampling for 3D-TOF Cameras", *International Conference on Computer Vision*, pp. 1623–1630, 2011.

22. Fan, H., H. Su and L. J. Guibas, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 605–613, 2017.

23. Qi, C. R., H. Su, K. Mo and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", *IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR), pp. 652–660, 2017.

24. Mescheder, L. M., M. Oechsle, M. Niemeyer, S. Nowozin and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4455–4465, 2019.

25. Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", *European Conference on Computer Vision (ECCV)*, 2020.

26. Han, Z., Z. Liu, J. Han, C.-M. Vong, S. Bu and C. L. P. Chen, "Mesh Convolutional Restricted Boltzmann Machines for Unsupervised Learning of Features With Structure Preservation on 3-D Meshes", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 10, pp. 2268–2281, 2016.

27. Sitzmann, V., J. N. P. Martel, A. W. Bergman, D. B. Lindell and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions", *ArXiv*, Vol. abs/2006.09661, 2020.

28. Takikawa, T., J. Litalien, K. Yin, K. Kreis, C. T. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire and S. Fidler, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11353–11362, 2021.

29. Hui, L., R. Xu, J. Xie, J. Qian and J. Yang, "Progressive Point Cloud Deconvolution Generation Network", *European Conference on Computer Vision (ECCV)*, 2020.

30. Shu, D. W., S. W. Park and J. Kwon, "3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions", *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3858–3867, 2019.

31. Yu, X., Y. Rao, Z. Wang, Z. Liu, J. Lu and J. Zhou, "PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers", *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12478–12487, 2021.

32. Xu, Q., W. Wang, D. Ceylan, R. Mech and U. Neumann, "DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction", *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

33. Chibane, J., T. Alldieck and G. Pons-Moll, "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6968–6979, 2020.

34. Kar, A., S. Tulsiani, J. Carreira and J. Malik, "Category-Specific Object Reconstruction from a Single Image", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1966–1974, 2015.

35. Choy, C. B., D. Xu, J. Gwak, K. Chen and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-View 3D Object Reconstruction", *European Conference on Computer Vision (ECCV)*, pp. 628–644, Springer, 2016.

36. Afifi, A. J., J. Magnusson, T. A. Soomro and O. Hellwich, "Pixel2Point: 3D Object Reconstruction From a Single Image Using CNN and Initial Sphere", *IEEE Access*, Vol. 9, pp. 110–121, 2021.

37. Li, J. and G. H. Lee, "DeepI2P: Image-to-Point Cloud Registration via Deep Classification", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15955–15964, 2021.

38. Goetschalckx, L., A. Andonian, A. Oliva and P. Isola, "GANalyze: Toward Visual Definitions of Cognitive Image Properties", *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5744–5753, 2019.

39. Shen, Y., C. Yang, X. Tang and B. Zhou, "InterFaceGAN: Interpreting the Disen-

tangled Face Representation Learned by GANs", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 44, pp. 2004–2018, 2022.

40. Voynov, A. and A. Babenko, "Unsupervised Discovery of Interpretable Directions in the GAN Latent Space", *ArXiv*, Vol. abs/2002.03754, 2020.

41. Jahanian, A., L. Chai and P. Isola, "On the Steerability of Generative Adversarial Networks", *ArXiv*, Vol. abs/1907.07171, 2020.

42. Härkönen, E., A. Hertzmann, J. Lehtinen and S. Paris, "GANSpace: Discovering Interpretable GAN Controls", *ArXiv*, Vol. abs/2004.02546, 2020.

43. Wold, S., K. Esbensen and P. Geladi, "Principal Component Analysis", *Chemometrics and Intelligent Laboratory Systems*, Vol. 2, No. 1-3, pp. 37–52, 1987.

44. Brock, A., J. Donahue and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", *ArXiv*, Vol. abs/1809.11096, 2019.

45. Shen, Y. and B. Zhou, "Closed-Form Factorization of Latent Semantics in GANs", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1532–1540, 2021.

46. Naseer, M., K. Ranasinghe, S. Khan, M. Hayat, F. S. Khan and M.-H. Yang, "Intriguing Properties of Vision Transformers", *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

47. Caron, M., H. Touvron, I. Misra, H. J'egou, J. Mairal, P. Bojanowski and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers", *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9630–9640, 2021.

48. Amir, S., Y. Gandelsman, S. Bagon and T. Dekel, "Deep ViT Features as Dense Visual Descriptors", *ArXiv*, Vol. abs/2112.05814, 2021.

49. Guzhov, A., F. Raue, J. Hees and A. R. Dengel, "AudioCLIP: Extending CLIP to Image, Text and Audio", *ArXiv*, Vol. abs/2106.13043, 2021.

50. Wu, H.-H., P. Seetharaman, K. Kumar and J. P. Bello, "Wav2CLIP: Learning Robust Audio Representations From CLIP", *ArXiv*, Vol. abs/2110.11499, 2021.

51. Shen, Y., J. Gu, X. Tang and B. Zhou, "Interpreting the Latent Space of GANs for Semantic Face Editing", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9240–9249, 2020.

52. Chen, D.-Y., X.-P. Tian, E. Y.-T. Shen and M. Ouhyoung, "On Visual Similarity Based 3D Model Retrieval", *Computer Graphics Forum*, Vol. 22, 2003.

53. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

APPENDIX A: SVR FRAMEWORK

A.1. Deep ViT Autoencoder Architecture

We use an autoencoder architecture with feed forward layers to further compress extracted ViT descriptors. Our proposed network consists of 2 dense layers followed by ReLU activations. Once the model is trained, we use the frozen model and extract dense features from the bottleneck (output of the encoder).

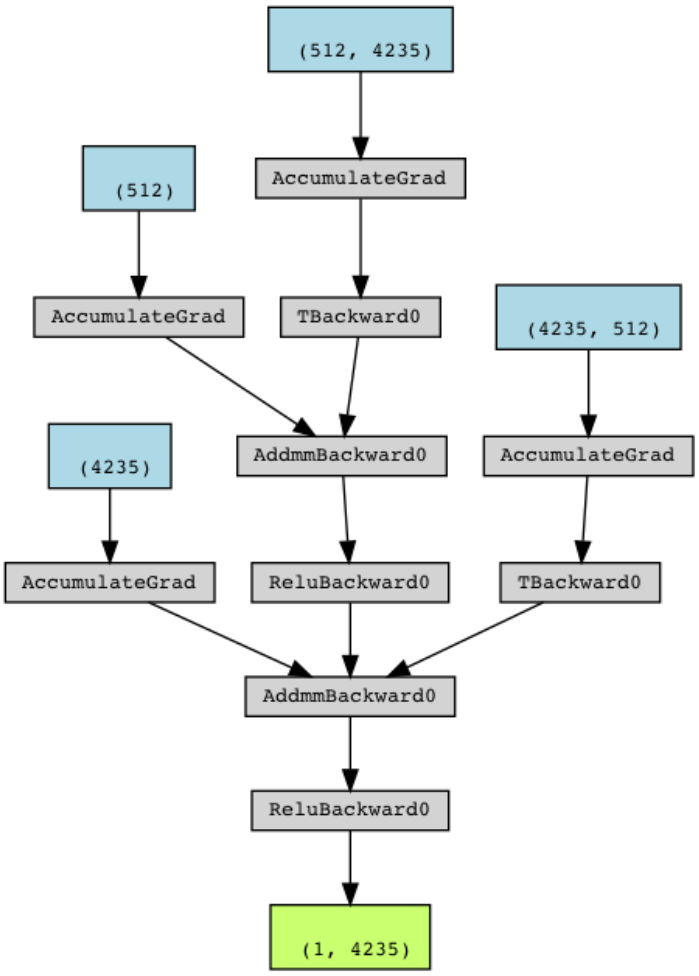

Figure A.1. A diagram of our ViT autonetwork to compress dense descriptors: the model consists of 2 dense layers followed by ReLU activations.

A.2. Latent Mapper Architecture

We employ a dense neural network to map dense embeddings extracted from ViT and CLIP to the latent space of DIT. Our proposed network consists of 3 dense layers followed by hyberbolic tangent activations. We note that we use the hyberbolic tangent function instead of more ReLU and such, since the learned latent space of DIT is observed to have a Gaussian distribution with a mean of 0.

A.3. Single View Reconstruction Experiment

For SVR with AtlasNet [19] and IM-NET [20], we used the official implementations and the renderings of ShapeNet data provided by 3D-R2N2 [35] with a 90-10% split for training and testing. For AtlasNet experiments, we trained image encoders on *car* and *airplane* categories and used the decoder from the respective autoencoder with fixed parameters to compute a Chamfer distance loss between the resulting mesh and the ground truth mesh corresponding to the input image. For IM-NET experiments, we trained ResNet encoders on *car* and *airplane* categories and used the trained implicit decoder with fixed parameters to train a mapping network that maps images to the latent space of IM-NET. More specifically, we used grayscale images as input and trained ResNET encoders to minimize the mean squared loss between the predicted feature vectors and the ground truth feature vectors encoded by the pre-trained autoencoder.

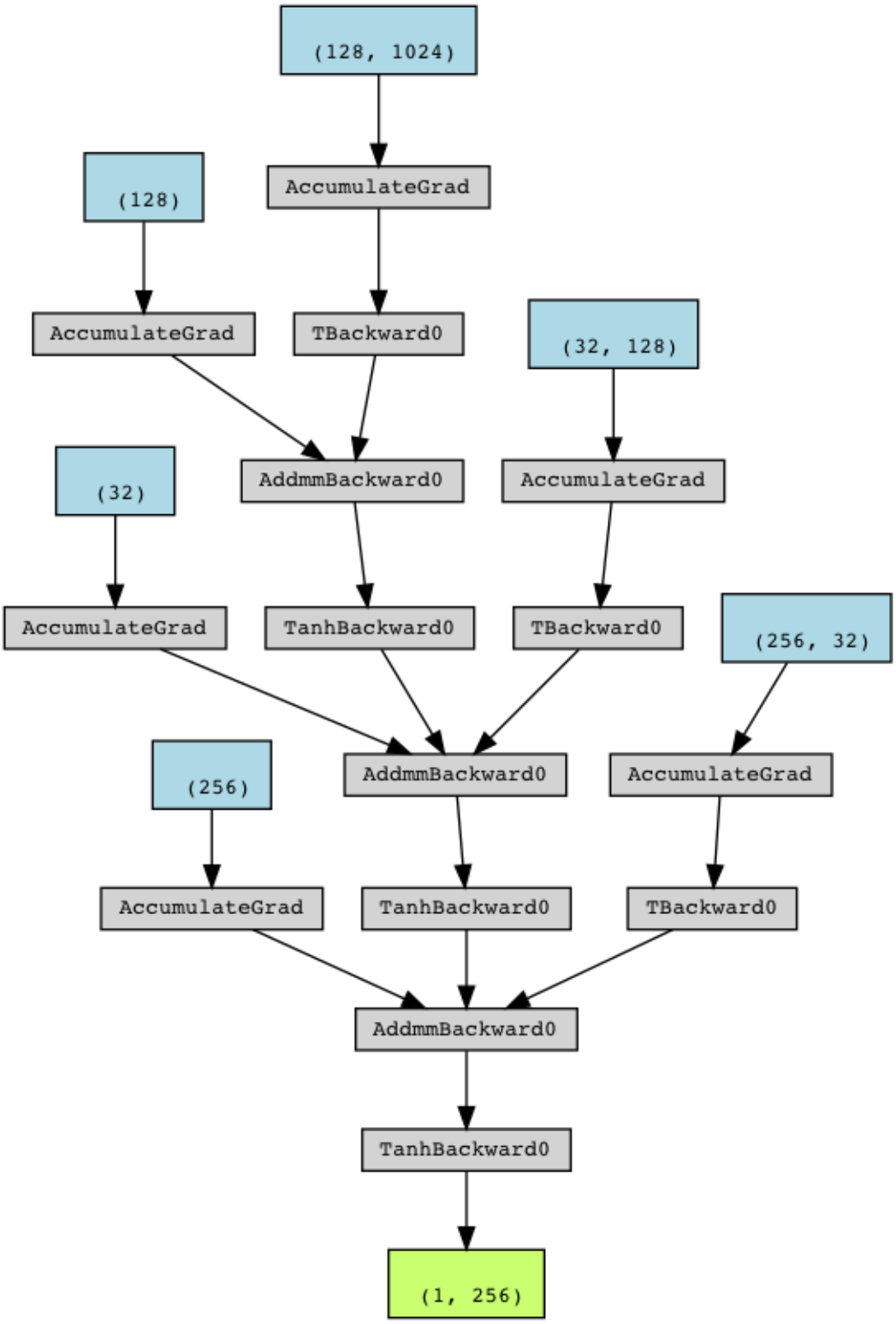

Figure A.2. A diagram of our Latent Mapper network for SVR: the model consists of 3 dense layers followed by hyberbolic tangent activations.

APPENDIX B: REAL-TO-SYNTHETIC IMAGE TRANSLATION

B.1. Sentence Templates for Prompt Engineering

We leverage CLIP to filter out irrelevant images in our real image datasets. Our method uses 60 sentence templates to compute the average CLIP distance between the target category and the content of each image. The list of templates we use for augmentation can be found in Table B.1.

Table B.1. List of templates that our method uses for augmentation. The input text prompt is added to the end of each sentence template.

‘a bad photo of a’	‘a sculpture of a’
‘a photo of the hard to see’	‘a low resolution photo of the’
‘a rendering of a’	‘graffiti of a’
‘a bad photo of the’	‘a cropped photo of the’
‘a photo of a hard to see’	‘a bright photo of a’
‘a photo of a clean’	‘a photo of a dirty’
‘a dark photo of the’	‘a drawing of a’
‘a photo of my’	‘the plastic’
‘a photo of the cool’	‘a close-up photo of a’
‘a painting of the’	‘a painting of a’
‘a pixelated photo of the’	‘a sculpture of the’
‘a bright photo of the’	‘a cropped photo of a’
‘a plastic’	‘a photo of the dirty’
‘a blurry photo of the’	‘a photo of the’
‘a good photo of the’	‘a rendering of the’
‘a doodle of a’	‘a close-up photo of the’
‘a photo of a’	‘the in a video game.’
‘a doodle of the’	‘a low resolution photo of a’
‘the toy’	‘a rendition of the’
‘a photo of the clean’	‘a photo of a large’
‘a rendition of a’	‘a photo of a nice’
‘a photo of a weird’	‘a blurry photo of a’
‘a sketch of the’	‘a pixelated photo of a’
‘itap of the’	‘a good photo of a’
‘a photo of the small’	‘a photo of the weird’
‘a drawing of the’	‘a photo of the large’
‘itap of a’	‘graffiti of the’
‘a photo of a cool’	‘a photo of a small’
‘a 3d object of the’	‘a 3d object of a’