2023

Generative and Implicit Methods for 3D Point Cloud Processing

Mohammad Samiul Arshad

GENERATIVE AND IMPLICIT METHODS FOR 3D POINT CLOUD PROCESSING

by

MOHAMMAD SAMIUL ARSHAD

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2023

To the wellspring of my perseverance - my unwavering parents, my loving wife, and my precious daughters. Excellentia est clavis ad successum.

ACKNOWLEDGMENTS

First and foremost, thanks to the Almighty!

I want to express my heartfelt gratitude to my supervising professor, Dr. William J. Beksi, for his unwavering support and encouragement throughout my doctoral journey. His constant motivation and invaluable guidance have been instrumental in shaping the success of my research. In addition, I would like to extend my thanks to my esteemed academic advisors, Dr. Vassilis Athitsos, Dr. Gautam Das, and Dr. Manfred Huber. Their keen interest in my research and willingness to serve on my dissertation committee have been pivotal in enriching the quality of my work. I am truly fortunate to have had the opportunity to work under the guidance of such accomplished individuals, and I am sincerely grateful for their contributions to my academic and research endeavors.

I would like to express my gratitude to the Texas Advanced Computing Center for their invaluable support in providing computational resources for my doctoral studies. Furthermore, I want to extend my thanks to my labmates for their collaborative spirit and engaging discussions. Their input and feedback have been immensely valuable in shaping the direction of my research and refining the ideas presented in my work.

I want to extend my heartfelt gratitude to the dedicated teachers, both in Bangladesh and in the United States who have played a significant role in shaping my education and academic journey. I am truly grateful to my friends for their inspiration and encouragement. Their unwavering support and guidance have been a source of strength and motivation, propelling me forward in pursuit of my educational aspirations.

Finally, I am deeply grateful to my parents, whose love, unwavering support, and invaluable guidance have been pivotal in shaping the person I have become today. Their constant encouragement and unwavering belief in my abilities have been a wellspring of strength and inspiration throughout every phase of my life. The sacrifices they have made on my behalf are immeasurable,

and no amount of appreciation could truly capture the depth of my gratitude. Last but not least, I want to express my heartfelt thanks to my wife, Umma Salma, whose unwavering love, support, and assistance have been instrumental in helping me achieve my goals along this challenging journey. Her presence and encouragement have been a constant source of motivation, pushing me forward even during the toughest times.

August 03, 2023

ABSTRACT

GENERATIVE AND IMPLICIT METHODS FOR 3D POINT CLOUD PROCESSING

Mohammad Samiul Arshad, Ph.D.

The University of Texas at Arlington, 2023

Supervising Professor: William J. Beksi

3D point clouds are a popular form of data representation with many applications in computer vision, computer graphics, and robotics. As the output of range sensing devices, point clouds have gained popularity with the current interest in self-driving vehicles. More formally, point clouds are an unordered set of irregular points collected from the surface of an object. Each point consists of a Cartesian coordinate, along with additional information such as an RGB color value and surface normal estimate. However, deep learning methods fall short in the processing of 3D point clouds due to the irregular and permutation-invariant nature of the data.

In this dissertation, we design novel types of neural networks that leverage raw 3D point clouds for data creation and reconstruction. First, we investigate dense colored point cloud generation and present an understanding of shape color correlation with a progressive conditional generative adversarial network (PCGAN). PCGAN learns to create a 3D data distribution by producing colored point clouds with subtle details at a range of resolutions. Next, we reconstruct open surfaces with inner details by extracting surface points from an unsigned distance field with an implicit point voxel network (IPVNet). In IPVNet, we show that by combining features from different 3D representations such as point clouds and voxels, deep learning models can reduce both inaccuracies and the number of outliers in the reconstruction. Finally, we discuss reconstructing

a 3D surface from a single image by learning an implicit function through a spatial transformer (LIST). Within the LIST framework, we introduce an innovative spatial transformer that creates the ability to accurately retrieve intricate details from a single image without the need for any additional rendering information.

Overall, we provide a comprehensive investigation of generative and implicit point cloud processing techniques. We establish novel deep-learning frameworks to facilitate the 3D reconstruction and generation tasks. Additionally, we make our source code and other resources publicly available for the benefit of the research community.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

The processing of 3D point clouds has witnessed a remarkable upswing in research activity in recent years. This surge in interest can be attributed to the growing number of applications where point clouds are valuable. For example, they are used in robot navigation and autonomous vehicles to understand the environment, augmented reality to enhance virtual objects with real-world context, as well as in healthcare for tasks such as medical imaging. In this context, a 3D point cloud can be conceptualized as an unordered collection of irregularly distributed points sampled from the surface of an object. Each point in the cloud is defined by its Cartesian coordinate and may also contain additional information such as surface normal estimates and RGB color values. Point clouds are typically generated by range-sensing devices such as structured light, time-of-flight, or light detection and ranging (LiDAR).

Processing 3D point clouds presents several significant challenges compared to other data modalities. The raw output from sensors often results in sparse points, irregular noise, and the presence of artifacts. Special attention and dedicated techniques are required to address these issues and effectively process the raw sensor data. Computer vision algorithms, which have shown great success in other domains, encounter difficulties when it comes to processing 3D point clouds. These methods often rely on the assumption of spatial or temporal regularity in the data, which is not inherently present in the irregular and unordered nature of 3D point clouds. As a result, traditional deep learning architectures struggle to effectively capture and utilize the spatial relationships and patterns within point cloud data. Therefore, specialized computer vision algorithms are required to fully utilize the representational power of point clouds for 3D computer vision tasks.

Motivated by these observations, this dissertation explores new computer vision algorithms and data representations to facilitate raw point cloud learning for 3D computer vision tasks. Specifically, we investigate automated point cloud generation and reconstruction. Generation and reconstruction are fundamental tasks in computer vision that play a crucial role in capturing and understanding the three-dimensional world. The importance of generation lies in the ability to create synthetic 3D data, including point clouds or volumetric representations, which can be used for training and evaluation purposes. Synthetic data generation allows researchers and practitioners to overcome limitations in real-world data collection, such as scarcity, cost, or privacy concerns. By generating diverse and realistic 3D data, it becomes possible to train and fine-tune algorithms and models for various tasks, including object recognition, scene understanding, and pose estimation.

On the other hand, reconstruction focuses on the process of capturing and reconstructing the 3D structure and geometry of real-world objects or scenes from sensor data. This task is essential for applications such as 3D modeling, augmented reality, robotics, and virtual reality. Accurate and detailed reconstruction enables the creation of realistic virtual environments, precise object recognition and tracking, and the ability to manipulate and interact with 3D objects. Reconstruction techniques also contribute to depth estimation, surface reconstruction, and motion analysis, enabling a deeper understanding of the 3D world. Overall, generation and reconstruction in 3D computer vision are critical for advancing the field and enabling a wide range of applications. They provide the necessary tools and data to train and evaluate algorithms, create immersive experiences, and enhance the perception and interaction capabilities of computer vision systems in three-dimensional space.

This research introduces several significant advancements in the field of raw 3D point cloud generation and reconstruction. First, we address the challenge of generating high-resolution point clouds by introducing PCGAN. Next, we propose IPVNet, a method for reconstructing target surfaces at any desired resolution using sparse scanning data. Finally, we introduce LIST, which enables the recovery of 3D surfaces from a single 2D image.

All of these works provide open-source licenses, granting access to the source code, processing scripts, training data, and experimental results. The resources are made available online and offered to the public at no cost for their use and benefit. Below, we provide a summary of each of these contributions.

Chapter 2: A Progressive Conditional Generative Adversarial Network for Generating Dense and Colored 3D Point Clouds, *International Conference on 3D Vision, 2020*.

We introduce a novel conditional generative adversarial network that creates dense 3D point clouds, with color, for assorted classes of objects in an unsupervised manner. To overcome the difficulty of capturing intricate details at high resolutions, we propose a point transformer that progressively grows the network through the use of graph convolutions. The network is composed of a leaf output layer and an initial set of branches. Every training iteration evolves a point vector into a point cloud of increasing resolution. After a fixed number of iterations, the number of branches is increased by replicating the last branch. Experimental results show that our network is capable of learning and mimicking a 3D data distribution, and produces colored point clouds with fine details at multiple resolutions.

Chapter 3: IPVNet: Learning Implicit Point-Voxel Features for Open-Surface 3D Reconstruction, *Journal of Visual Communication and Image Representation, 2023*.

Reconstruction of 3D open surfaces (e.g., non-watertight meshes) is an underexplored area of computer vision. Recent learning-based implicit techniques have removed previous barriers by enabling reconstruction in arbitrary resolutions. Yet, such approaches often rely on distinguishing between the *inside* and *outside* of a surface in order to extract a zero level set when reconstructing the target. In the case of open surfaces, this distinction often leads to artifacts such as the artificial closing of surface gaps. However, real-world data may contain intricate details defined by salient surface gaps. Implicit functions that regress an unsigned distance field have shown promise in reconstructing such open surfaces. Nonetheless, current unsigned implicit methods rely on a *discretized representation* of the raw data. This not only bounds the learning process to the rep-

resentation's resolution, but it also introduces outliers in the reconstruction. To enable accurate reconstruction of open surfaces without introducing outliers, we propose a learning-based implicit point-voxel model (IPVNet). IPVNet predicts the unsigned distance between a surface and a query point in 3D space by leveraging both raw point cloud data and its discretized voxel counterpart. Experiments on synthetic and real-world public datasets demonstrates that IPVNet *outperforms* the state of the art while producing *far fewer* outliers in the resulting reconstruction.

Chapter 4: LIST: Learning Implicitly from Spatial Transformers for Single-View 3D Reconstruction, *International Conference on Computer Vision, 2023*.

Accurate reconstruction of both the geometric and topological details of a 3D object from a single 2D image embodies a fundamental challenge in computer vision. Existing explicit/implicit solutions to this problem struggle to recover self-occluded geometry and/or faithfully reconstruct topological shape structures. To resolve this dilemma, we introduce LIST, a novel neural architecture that leverages local and global image features to accurately reconstruct the geometric and topological structure of a 3D object from a single image. We utilize global 2D features to predict a coarse shape of the target object and then use it as a base for higher-resolution reconstruction. By leveraging both local 2D features from the image and 3D features from the coarse prediction, we can predict the signed distance between an arbitrary point and the target surface via an implicit predictor with great accuracy. Furthermore, our model does not require camera estimation or pixel alignment. It provides an uninfluenced reconstruction from the input-view direction. Through qualitative and quantitative analysis, we show the superiority of our model in reconstructing 3D objects from both synthetic and real-world images against the state of the art.

CHAPTER 2

Preliminaries

There are several common metrics used to measure the similarity between two individual point clouds. In this chapter, we define the metrics employed to evaluate our models.

Chamfer Distance (CD): The chamfer distance (CD) between two meshes is defined as

$$\text{CD}(y_{\text{GT}}, y_{\text{pred}}) = \sum_{a \in y_{pred}} \min_{b \in y_{gt}} ||a - b|| + \sum_{b \in y_{gt}} \min_{b \in y_{pred}} ||b - a||, \tag{2.1}$$

where, y_{GT} and y_{pred} are two point clouds extracted from the surface of the ground-truth and reconstructed object, respectively.

Intersection over Union (IoU): The volumetric intersection over union (IoU) is defined as the quotient of the volume of the intersection of two meshes and the volume of their union,

$$\text{IoU}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) = \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{GT}}|}{|\mathcal{M}_{\text{pred}} \cup \mathcal{M}_{\text{GT}}|}. \tag{2.2}$$

F-score: The F-score, proposed in [11] as a comprehensive scoring metric for single-view reconstruction, combines precision and recall to quantify the overall reconstruction quality. Concretely, the F-score at a distance threshold d is given by

$$F(d) = \frac{2 \cdot P(d) \cdot R(d)}{P(d) + R(d)},$$

where $P(\cdot)$ and $R(\cdot)$ represents the precision and recall, respectively. Precision quantifies the accuracy while recall assesses the completeness of the reconstruction. For the ground-truth y_{gt} and reconstructed point cloud y_{pred}, the precision of an outcome at d can be calculated as

$$P(d) = \sum_{i \in y_{\text{pred}}} [\min_{j \in y_{\text{GT}}} ||i - j|| < d].$$

Similarly, the recall for a given d may be computed as

$$R(d) = \sum_{j \in y_{\text{GT}}} [\min_{i \in y_{\text{pred}}} ||j - i|| < d].$$

Earth Movers Distance (EMD): The earth mover's distance (EMD) [12] is the solution of a transportation problem which attempts to transform one set to the other. For two equally sized subsets $S1 \subseteq \mathbb{R}^3, S2 \subseteq \mathbb{R}^3$, their EMD is defined by

$$d_{\text{EMD}}(S1, S2) = \min_{\phi: S1 \to S2} \sum_{x \in S1} \|x - \phi(x)\|^2,$$

where ϕ is a bijection.

Jensen-Shannon Divergence (JSD): The Jensen-Shannon divergence (JSD) quantifies the dissimilarity between marginal distributions defined in three-dimensional Euclidean space. Given an axis-aligned point cloud data in a canonical voxel grid within the ambient space, one can assess the extent to which point clouds of set A tend to occupy similar positions as those of set B. This is achieved by tallying the points within each voxel across all point clouds of set A and similarly for set B. The JSD between the resulting empirical distributions (P_A, P_B) is calculated as

$$JSD(P_A, P_B) = \frac{1}{2} D(P_A \| M) + \frac{1}{2} D(P_B \| M),$$

where $M = \frac{1}{2}(P_A + P_B)$ and $D(\cdot)$ represents the Kullback-Leibler divergence [13] between the two distributions.

Coverage: Coverage is a measure used to assess the extent to which instances (point clouds) from set A are matched with instances from set B. It is defined as

$$\text{Coverage}(A, B) = \frac{\text{Number of matched instances in } A}{\text{Total number of instances in } B},$$

where the number of matched instances in set A is determined by the matching process with set B. This metric indicates the proportion of instances from set B that have corresponding matches within set A. The distance metrics CD and EMD can be used to identify the matched instances.

Minimum Matching Distance (MMD): The minimum matching distance (MMD) is defined as

$$\text{MMD}(B, A) = \frac{1}{|B|} \sum_{\mathbf{b} \in B} \min_{\mathbf{a} \in A} \text{dist}(\mathbf{b}, \mathbf{a}),$$

where A and B are two sets of point clouds, and $|B|$ is the cardinality of set B, and $\text{dist}(\mathbf{b}, \mathbf{a})$ represents the distance between point clouds $\mathbf{b}$ and $\mathbf{a}$ using the chosen distance metric (e.g., CD, EMD, etc.). Please refer to [8] for a discussion between coverage and the MMD.

CHAPTER 3

A Progressive Conditional Generative Adversarial Network for Generating Dense and Colored 3D Point Clouds

3.1 Introduction

In recent years, research on processing 3D point clouds has gained momentum due to an increasing number of relevant applications. From robot navigation [14, 15] to autonomous vehicles [16, 17, 18], augmented reality [19, 16] to health care [20, 21, 22], the challenges of working with 3D datasets are being realized. Among miscellaneous data modalities, raw point clouds are becoming popular as a compact homogeneous representation that has the ability to capture intricate details of the environment. Intuitively, a 3D point cloud can be thought of as an unordered set of irregular points collected from the surface of an object. Each point consists of a Cartesian coordinate, along with other additional information such as a surface normal estimate and RGB color value. Although 3D point clouds are the product of range sensing devices (e.g., structured light, time-of-flight, light detection and ranging, etc.), the application of conventional machine learning techniques on the direct sensor output is nontrivial. In particular, deep learning methods fall short in the processing of 3D point clouds due to the irregular and permutation invariant nature of the data.

Generating synthetic 3D point cloud data is an open area of research with the intention of facilitating the learning of non-Euclidean point representations. In three dimensions, synthetic data may take the form of meshes, voxels, or raw point clouds in order to learn a representation that aids the solution of computer vision tasks such as classification [23, 24, 25, 26, 27], segmentation [23, 24, 28, 29, 30, 31, 32, 33], and reconstruction [34, 35, 36, 37, 38, 39]. Currently, researchers make use of point clouds sampled from the mesh of manually designed objects as synthetic data

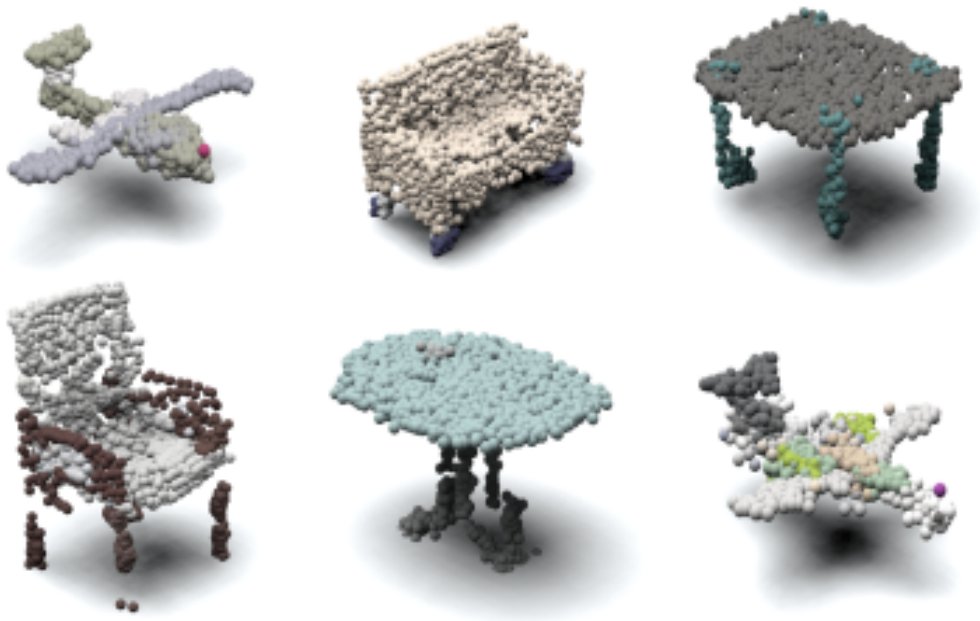


Figure 3.1: Examples of 3D point clouds synthesized by our progressive conditional generative adversarial network (PCGAN) for an assortment of classes. PCGAN generates both geometry and color for point clouds, without supervision, through a coarse to fine training process.

for training deep learning models [23, 24, 9, 40]. However, the geometry and texture of these point clouds is bounded by the resolution of the modeled objects. Moreover, due to the complexity of the design process, the number of composed objects can fail to satisfy the enormous data needs of deep learning research. Automatically synthesizing 3D point clouds can solve this problem by providing a source of potentially infinite amounts of diverse data.

Although color and geometry are among the defining features of 3D objects in the physical world, current point cloud generation methods either create the geometry [9, 41, 10, 8, 42, 43] or the color [40] of the point cloud, but not both. We believe that 3D point cloud generators should have the ability to synthesize dense point clouds with complex details that mimic real-world objects in both *geometry* and *color*. To this end, we propose a progressive conditional generative adversarial network (PCGAN) that faithfully generates dense point clouds with color using tree

structured graph convolutions. To the best of our knowledge, our work is the first attempt to generate both geometry and color for point clouds in an unsupervised fashion with progressive, i.e., coarse to fine training. Figure 3.1 shows examples of 3D point clouds generated by our network.

Generating high-resolution data is a difficult and computationally expensive task as the number of features and the level of details increases with resolution. In traditional generative methods [44, 45, 46], the generator tries to optimize both global structure and local features simultaneously which can overwhelm an unsupervised network. To reduce the learning complexity, a progressively growing generative adversarial network [47] may be used to learn features in a coarse to fine manner. Progressive growing has also been shown to improve training time [47] since the generator first optimizes low-resolution global structures which helps the optimization of local details at higher resolutions. In terms of 3D point clouds, boosting the resolution makes the dataset denser by the expansion of the number of points. Consequently, this adds more complexity to the generation procedure and amplifies the computational cost. This directly proportional relationship between point cloud resolution and complexity/computational cost has inspired us to use progressive growing in PCGAN for dense point cloud generation. Our network is end-to-end trainable and it learns the distribution of the data as well as the mapping from label to data to generate samples of numerous classes with a single network.

3.2 Contributions

The key contributions of our work are threefold.

- We introduce a progressive generative network that creates both geometry and color for 3D point clouds in the absence of supervision.
- We include both a qualitative and quantitative analysis of the objects synthesized by our network.
- We present the Frèchet dynamic distance metric to evaluate colored dense point clouds.

To allow other researchers to use our software, reproduce the results, and improve on them, we have released PCGAN under an open-source license. The source code and detailed installation instructions are available online [48].

The remainder of this chapter is organized as follows. We give a summary of related research in Section 3.3. In Section 3.4, we define the problem mathematically and provide the essential background information in Section 3.5. The details of our model are provided in Section 3.6, and we present the experimental setup and results in Section 3.7. A conclusion of this work is given in Section 3.8.

3.3 Related Work

In this section, we summarize pertinent work on the generation of 3D point clouds. Interested readers are encouraged to read [49, 50, 51] for a comprehensive survey of deep learning research on 3D point cloud datasets.

3D Point Cloud Generation. The first generative model capable of producing raw point clouds comes from the work of Achlioptas et al. [8]. Using PointNet [23] as the backbone, Achlioptas et al. introduced an autoencoder and two variants of a generative adversarial network to generate point clouds. Prior to [8], Qi et al. introduced PointNet [23], the first neural network to operate directly on point cloud data. Eckart et al. [52] used hierarchical Gaussian mixture models (hGMMs) to process point clouds, Zaheer et al. [53] utilized deep networks to analyze point clouds as sets, and Li et al. [54] made use of self-organizing maps and hierarchical feature extraction to discriminate point clouds.

Following [8], Li et al. [55] used two generative networks to learn a latent distribution and generated points based on learned features. Yang et al. [56] improved upon [8] by incorporating graph-based enhancement on top of PointNet and 2D grid deformation. Valsesia et al. [10] used graph convolutions and exploited pairwise distances between features to construct a generator. Similar to [55], Yang et al. [41] generated point clouds by learning two hierarchical distributions

and through the use of continuous normalizing flow. Mo et al. [57] mapped part hierarchies of an object shape as a tree and implemented an encoder-decoder network to generate new shapes via interpolation. Gadelha et al. [58] presented a tree network for 3D shape understanding and generation tasks by operating on 1D-ordered point lists obtained from a k-d tree space partitioning.

Ramasinghe et al. [43] generated high-resolution point clouds by operating on the spatial domain, and Hertz et al. [42] used hGMMs to generate shapes in different resolutions. However, both [43] and [42] fail to generate fine shape details. Xie et al. [59] proposed an energy-based generative PointNet [23], and Sun et al. [60] implemented auto-regressive learning with self-attention and context awareness for the generation and completion of point clouds. Cao et al. used adversarial training to generate color for point cloud geometry in [40]. In follow up work, they used a style transfer approach to transform the geometry and color of a candidate point cloud according to a target point cloud or image [61].

Compared to the focus of the aforementioned works on solely generating the geometry or color of point clouds, our model can produce both color and geometry in an unsupervised manner while maintaining exceptional details at high resolutions. The generator of our model is inspired by the tree structured graph convolution generator of Shu et al. [9]. However, the generator proposed by Shu et al. does not generate point clouds in color nor does it incorporate the advantages of progressive training. The multiclass generation model of Shu et al. is class agnostic thus making the generation process of a specific class of objects uncontrollable. Conversely, we incorporate conditional generation to control the creation process and use progressive training to build high-resolution point clouds with color. Although, Tchapmi et al. [39] also introduced a tree graph-based decoder, the focal point of their work was the completion of point cloud geometry with supervision.

Graph Convolutions. Point clouds can be portrayed as graphs where points represent nodes and the co-relation among neighboring points represent edges. Applying the notion of graph convolution operations to process point clouds is a relatively new area of research. In [28], Qi et al.

proposed the hierarchical application of PointNet to learn point cloud features as a graph embedding. However, their work did not incorporate the co-relation of neighboring points and consequently disregards local features. To account for local features, Atzmon et al. [62] used a Gaussian kernel applied to the pairwise distances between points. Wang et al. [24] introduced DGCNN which uses aggregated point features and pairwise distances among k points to dynamically construct a graph. The discriminator of our model was motivated by DGCNN. Nevertheless, DGCNN was designed for the classification of point cloud geometry while our discriminator aims to act as a critic of point cloud geometry and color to distinguish between real and synthetic data given a class label.

Progressive Training. Progressive training has been shown to improve the quality of 2D image generation [47, 63, 64]. Our work is the first attempt to use progressive training in an unsupervised 3D generative network. In previous research, Valsesia et al. [10] used upsampling layers based on k neighbors of the adjacency graph to increase the feature size between each graph convolution. However, their proposed architecture learns without progressive growing and suffers the same computational complexity of regular generative adversarial networks. The work of Yifan et al. [38] is the only example of coarse-to-fine generation in 3D. Yet, Yifan et al. used a supervised patch-based approach to upsample point clouds where the overall structure of the data is provided as a prior. In contrast, our network progressively learns the global shape of the data through the underlying distribution of the point cloud geometry and color of a class with no supervision or priors given.

3.4 Problem Statement

Consider a set of classes, $C = \{c_1, \ldots, c_n\}$, each representing mixed objects as 3D colored point clouds, $x \in \mathbb{R}^{N \times 6}$, where N is the number of points. Given a class $c \in C$, we seek to learn the underlying features that constitute c and generate a realistic point cloud $\hat{x} \in \mathbb{R}^{N \times 6}$.

3.5 Background

In the following subsections we lay out the necessary background knowledge upon which our work is grounded.

3.5.1 Wasserstein/Kantorovich-Rubinstein Distance

Given two distributions P_r and P_g in a metric space $\mathcal{M}$, the Wasserstein/Kantorovich-Rubinstein distance calculates the minimal cost to transform P_r into P_g or vice-versa [65]. A distance of order p can be expressed as

$$W_p(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\|x - y\|\big],$$

where $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(x, y)$ with marginals P_r and P_g.

3.5.2 Generative Adversarial Networks

Introduced by Goodfellow et al. [44], a generative adversarial network (GAN) is a special type of neural network that focuses on learning the underlying distribution, P_r, of a dataset to generate new samples. A GAN consists of a generator G and a discriminator D that compete against each other in a minimax game. The generator tries to manipulate a random vector z, drawn from a fixed distribution P_g, into synthetic samples $\hat{x}$ that are indistinguishable from the real data x. The discriminator seeks to differentiate between $\hat{x}$ and x. More formally, the objective of a GAN can be written as

$$\begin{aligned}\min_G \max_D (G, D) = \mathbb{E}_{x\sim P_r}[\log D(x)] \\ + \mathbb{E}_{z\sim P_g}[\log(1 - D(G(z)))].\end{aligned}$$

3.5.3 Wasserstein GAN with Gradient Penalty

To optimize the convergence of a GAN, Arjovsky et al. [45] introduced the Wasserstein GAN (WGAN) whose goal is to minimize the distance between the real data distribution P_r and the generated data distribution P_g using the Wasserstein metric. To make the goal of inter-distribution distance minimization tractable, Arjovsky et al. used the dual form of the Wasserstein distance, i.e., the Wasserstein-1 [65] with the GAN objective

$$\min_G \max_{D \in \mathcal{D}} (G, D) = \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{z \sim P_g}[D(G(z))],$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions. To ensure continuity in space, the discriminator of the WGAN must be 1-Lipschitz which is achieved through weight clipping. However, since weight clipping penalizes the norm of the gradient the stability of network can be compromised.

Gulrajani et al. [46] improved the WGAN by using a gradient penalty (WGAN-GP) instead of weight clipping. To enforce the 1-Lipschitz condition, the WGAN-GP constrains the norm of the gradient to be at most 1. This is achieved by a penalty term applied to the gradient of the discriminator,

$$\begin{aligned}\min_G \max_{D \in \mathcal{D}} (G, D) = \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{z \sim P_g}[D(G(z))] \\ + \lambda \, \mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}}\big[(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1)^2\big],\end{aligned}$$

where $\tilde{x} \sim P_{\tilde{x}}$ are the points uniformly sampled along the straight line between pairs of points from the real data distribution P_r and generated data distribution P_g.

3.6 Model Architecture

Our model has the following two main components: a generator G and a discriminator D. Unless stated otherwise, we refer to a point cloud x as a 6-dimensional matrix with N points, i.e.,

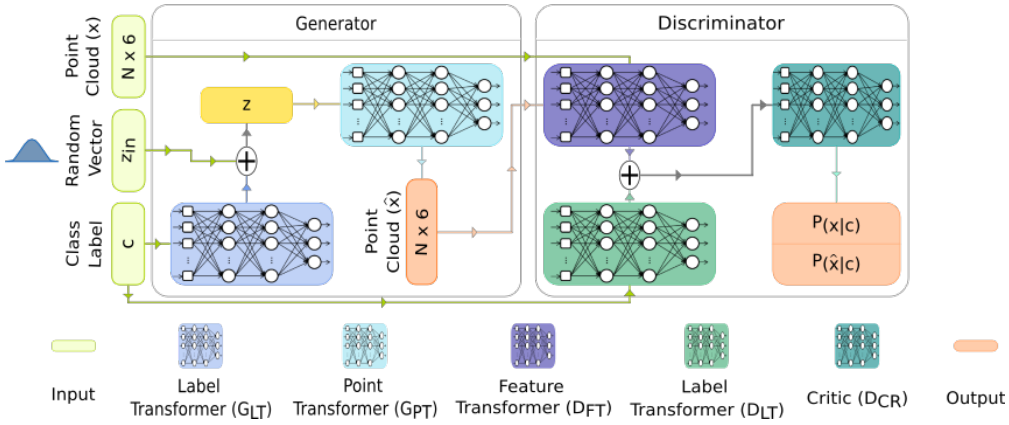


Figure 3.2: Given a random vector z_{in} and class label c, the generator produces a colored point cloud $\hat{x}$. The real point cloud x, class label c, and the generated point cloud $\hat{x}$ are given to the discriminator, which then tries to predict the probability of the data being real or synthetic.

$x \in \mathbb{R}^{N\times 6}$ where each point represents a Cartesian coordinate and RGB color. For multiclass generation, both the generator and the discriminator are conditioned on a class label, $c \in C$, which is randomly chosen from a set of n classes $C = \{c_1, \ldots, c_n\}$. To optimize the generator and the discriminator we make use of the WGAN-GP techniques discussed in Section 3.5.3. Figure 3.2 shows the overall architecture of PCGAN.

3.6.1 Generator

The generator G takes as input a random vector, $z_{in} \in \mathcal{N}(0, 1)$, along with a class label $c \in C$ represented as one hot vector. The class label controls the generation of colored point clouds of a desired class. The generator is comprised of two sub-networks: a label transformer G_{LT} and a point transformer G_{PT}.

The class label goes through G_{LT}, a shallow two-layer perceptron, and a class vector $g_c \in \mathbb{R}^{64}$ is constructed. The input vector z_{in} is then concatenated with g_c to produce a point vector z which is given to G_{PT} as input, i.e.,

$$z = (z_{in}, g_c).$$

Following [9], the point transformer incorporates tree structured graph convolutions (TreeGCN). As the name suggests, information in TreeGCN is passed from the root node to a leaf node instead of all neighbors, i.e., the i-th node at layer l is generated by aggregating information from its ancestors $A = \{a_i^{l-1}, a_i^{l-2}, \ldots, a_i^1, a_i^0\}$. However, we have empirically found that the information from up to three immediate ancestors $A = \{a_i^{l-1}, a_i^{l-2}, a_i^{l-3}\}$ is sufficient for realistic point cloud generation. This observation improves the overall computational cost of our network as G_{PT} is not bounded by the entire depth of the tree as in [9] (additional details and an analysis are included in Section 4.5). Therefore, the output of $(l+1)$-layer of G_{PT} is a first order approximation of the Chebyshev expansion

$$p_i^{l+1} = \sigma\left(\mathbf{F}_m^l(p_i^l) + \sum_{q_j \in A(p_i^l)} W_j^l q_j^l + b^l\right),$$

where

$$\mathbf{F}_m^l(p_i^l) = \sum_{j=1}^{m} S_j p_i^l + r_j,$$

$\sigma(\cdot)$ is an activation function, p_i^l is the i-th node of the graph at the l-layer, and q_j^l is the j-th ancestor of p_i^l from the set of three immediate ancestors of p_i^l. W, b, S, r are learnable parameters and $\mathbf{F}_m$ is a sub-network with m support.

Figure 3.3 presents an overview of the progressive growing of PCGAN. We have sub-divided the point transformer G_{PT} into two parts, a branch B and a leaf L, where a leaf acts as the output layer of G_{PT} for increasing replications R. Each branch incorporates a tree graph of expanding depth, $H = (h_1, \ldots, h_T)$, where T is the total number of branches. The leaf also incorporates a tree graph of depth h_L. First, we optimize the generator to produce a point cloud $\hat{x}$ at a predefined base resolution $(N_{R_1} \times 6)$ where $R_1 = h_L \Pi_{i=1}^{T} h_i$. After a fixed number of iterations, we introduce a new branch B_{T+1} through the replication of the branch B_T and we increase the depth $H = (h_1, \ldots, h_{T+1})$ of the incorporated tree graph. The replication of an already optimized branch facilitates the generation of higher resolution point clouds without starting the learning process

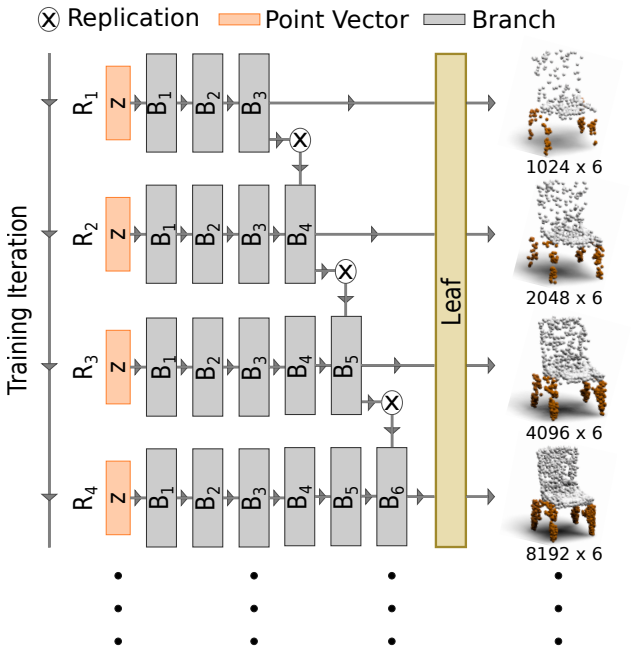


Figure 3.3: The progressive growing of the point transformer G_{PT}. The network starts with a leaf output layer and branches $(B_1 \ldots, B_T)$, each incorporating a tree graph, and converts a point vector z into a low-resolution point cloud. After a predefined number of iterations, the number of branches is increased through the replication R of the last branch B_T. In this figure, T was kept small for visualization purposes. The process is continued until the generation of a point cloud at the desired resolution is achieved.

from scratch. We continue this procedure until the desired resolution of the point cloud is realized, i.e., $R_d = h_L \Pi_{i=1}^{T+d} h_i$.

3.6.2 Discriminator

Given a point cloud x, the discriminator D tries to predict the probability of x being real or synthesized. The discriminator is comprised of the following three sub-networks: a feature transformer D_{FT}, a label transformer D_{LT}, and a critic D_{CR}. The feature transformer network was inspired by [24] to collect local and global features through a dynamic graph construction. Nonetheless, we expand the feature size in every layer to account for the color of the point clouds.

Given an input point vector, each layer of the point transformer network constructs a dynamic k-NN graph, $\mathcal{G}^l = (\mathcal{V}^l, \mathcal{E}^l)$, with self loops. Each point of the point cloud represents a vertex in the

graph and the vertex of each subsequent layer depends on the output of the preceding layers. For a graph $\mathcal{G}^l$ at layer l, the edges e_i^l between a vertex v_i^l and its k nearest neighbors are defined as a nonlinear function $\Theta = [\theta_1, \ldots, \theta_m]$ with learnable parameter θ,

$$e^l_{i,(1,\ldots,k)} = \Theta_{j\in(1,\ldots,k)}(v_i^l, v_j^l).$$

Although there are three choices available for Θ, we use the function defined in [24] to capture both local and global features. Concretely, for two vertices v_i and v_j we have

$$\Theta(v_i, v_j) = \theta(v_i, v_i - v_j).$$

Lastly, the output of the feature transform network is a feature vector f_c collected from the channel-wise maximum operation on the edge features from all the edges of each vertex,

$$f_c = \max_{(i,j)\in\mathcal{E}} \Theta(v_i, v_j).$$

The label transformer of the discriminator functions in an analogous way to the label transformer of the generator and it provides a label vector d_c. Note that even though D_{LT} and G_{LT} are similar in architecture, their objectives are distinct. Hence, we use two different networks for this reason. The feature vector f_c together with the label vector d_c are fed to the critic, a fully-connected three layer sub-network that aspires to predict the probability of its input vector being real or synthetic given that it was collected from object class c, i.e.,

$$D_{CR}(f_c, d_c) = [P(\text{real} \,|\, c), P(\text{generated} \,|\, c)].$$

3.7 Experiments

In this section, we describe our experimental setup and provide an analysis of the results.

Class Label	Number of Samples
Airplane	4029
Chair	4064
Table	8474
Sofa	3149
Motorcycle	333

Table 3.1: The class labels and the number of samples in the training data.

Dataset. We used the synthetic dataset ShapeNetCore [4] to conduct our experiments. ShapeNetCore is a collection of CAD models of various object classes among which we chose Chair, Table, Sofa, Airplane, and Motorcycle for our experiments. We have selected these classes for the diversity of their shapes and we keep the number of classes to five to reduce the training time. In addition, we have eliminated any CAD model that does not have material/color information. The training data was prepared by collecting point clouds of desired resolution from the surface of the CAD models. We normalized each point cloud such that the object is centralized in a unit cube and the RGB colors are in the range $[-0.5, +0.5]$. Table 3.1 shows the number of sample objects in each class.

Implementation Details. The generator of PCGAN learns to produce colored point clouds from low to high resolutions. We start the generation process with $N_{R_1} = 1024$ points and double the number of points progressively. To save memory and reduce computation time, we fix $N_{R_d} = 8192$ points as the highest resolution for the point clouds. The input to the generator is a vector, $z_{in} \in \mathbb{R}^{64}$, sampled from a normal distribution $z_{in} \in \mathcal{N}(0, 1)$, and the labels of the chosen classes $c \in C$.

For the point transformer, the number of branches was set to $T = 5$ with depth increments $H = [1, 2, 2, 2, 2]$. The leaf increments were set to $h_L = 64$. The depth increment hyperparameter for the branches and the leaves is similar to the branching of [9]. The feature dimension for the layers of G_{PT} was set to $[128, 128, 256, 256, 128, 128, 6]$. We used the Xavier initialization [66] for G_{PT} and the support q for each branch was set to 10 as in [9].

The feature dimension for the layers of the feature transform sub-network were set to $[6, 64, 128, 256, 512, 1024]$. To comply with the constraints of WGAN-GP, we did not use any batch normalization or dropout in D_{FT}. For D_{FT}, $k = 20$ was used to construct the k-NN graph and k was increased by 10 with each increment in resolution.

For both the generator and the discriminator, LeakyReLU nonlinearity with a negative slope of 0.2 was employed. The learning rate α was set to 10^{-4} and the Adam optimizer [67] with coefficients $\beta_1 = 0.0$ and $\beta_2 = 0.95$ were used. The gradient penalty coefficient λ for WGAN-GP was set to 10.

Metrics. To quantitatively evaluate generated samples in 2D, the Frèchet inception distance [68] is the most commonly used metric. We propose a similar metric called the Frèchet dynamic distance (FDD) to evaluate the generated point clouds where DGCNN [24] is used as a feature extractor. Although similar metrics exist [9, 60] where PointNet is used to extract features, the color of the point clouds is not considered and they suffer from the limitations of PointNet.

We use DGCNN because it collects both local and global information over the feature space and it also performs better than PointNet in point cloud classification [24]. To implement FDD, we trained DGCNN until a 98% test accuracy per class was achieved on the task of classification. Then, we extracted a 512-dimensional feature vector from the average pooling layer of DGCNN to calculate the mean vector and covariance matrix. For real point clouds x and synthetic point clouds $\hat{x}$, the FDD calculates the 2-Wasserstein distance,

$$\text{FDD}(x, \hat{x}) = \|\mu_x - \mu_{\hat{x}}\| + \text{tr}(\Sigma_x + \Sigma_{\hat{x}} - 2(\Sigma_x \Sigma_{\hat{x}})^{1/2}),$$

where μ and Σ represent the mean vector and the covariance matrix, respectively. The matrix trace is denoted by $\text{tr}()$. Additionally, we have used the matrices from Achlioptas et al. [8] for point cloud evaluation and we have compared the results with [8, 10, 9].

3.7.1 Results

A set of synthesized objects along with their real counterparts is shown in Figure 3.4. As is evident from the samples, our model first learns the basic structure of an object in low resolutions and gradually builds up to higher resolutions. PCGAN also learns the relationship between object parts and color. For example, the legs of the chair have equal colors while the seat/arms are a different color, the legs of the table have matching colors while the top is a contrasting color, and the airplane wings/engine have the same color while the body/tail has a dissimilar color.

Quantitative Analysis. We generated 5000 random samples for each class and performed an evaluation using the matrices from [8]. Table 3.2 presents our findings along with comparisons to previous studies [8, 9, 10]. Note that although our model is capable of generating higher resolutions and colors, we only used point clouds with $N = 2048$ points in order to be comparable with other methods. Also, separate models were trained in [8, 9, 10] to generate point clouds of different classes while we have used the *same* model to generate point clouds for all five classes. Even though we have achieved comparable results in Table 3.2, the main focus of our work is dense colored point cloud generation and point clouds with a resolution of $N = 2048$ is an intermediate result of our network. We have also evaluated colored dense point clouds ($N = 8192$) using the proposed FDD metric with the results presented in Table 3.3.

Computational Complexity. Tree structured graph convolutions are used in each layer of the point transformer sub-network. Since every subsequent node in the tree is originally dependent upon all of its ancestors, the time complexity of the graph convolutions is $\sum_{l=1}^{L} B \times n_i^l \times A_i^l \times V_i^l$, where B is the batch size of the training data, L is the total number of layers, n_i^l is the height of the tree graph at the i-th node, A_i is the induced number of ancestors preceding i-th node, and V_i^l is the induced vertex number of the i-th node [9]. However, we restrict G_{PT} to aggregate information from at most three levels of ancestors in each layer. Thus, $n_i^l \leq 3$ and the effective time complexity is $\sum_{l=1}^{L} B \times 3 \times A_i^l \times V_i^l$.

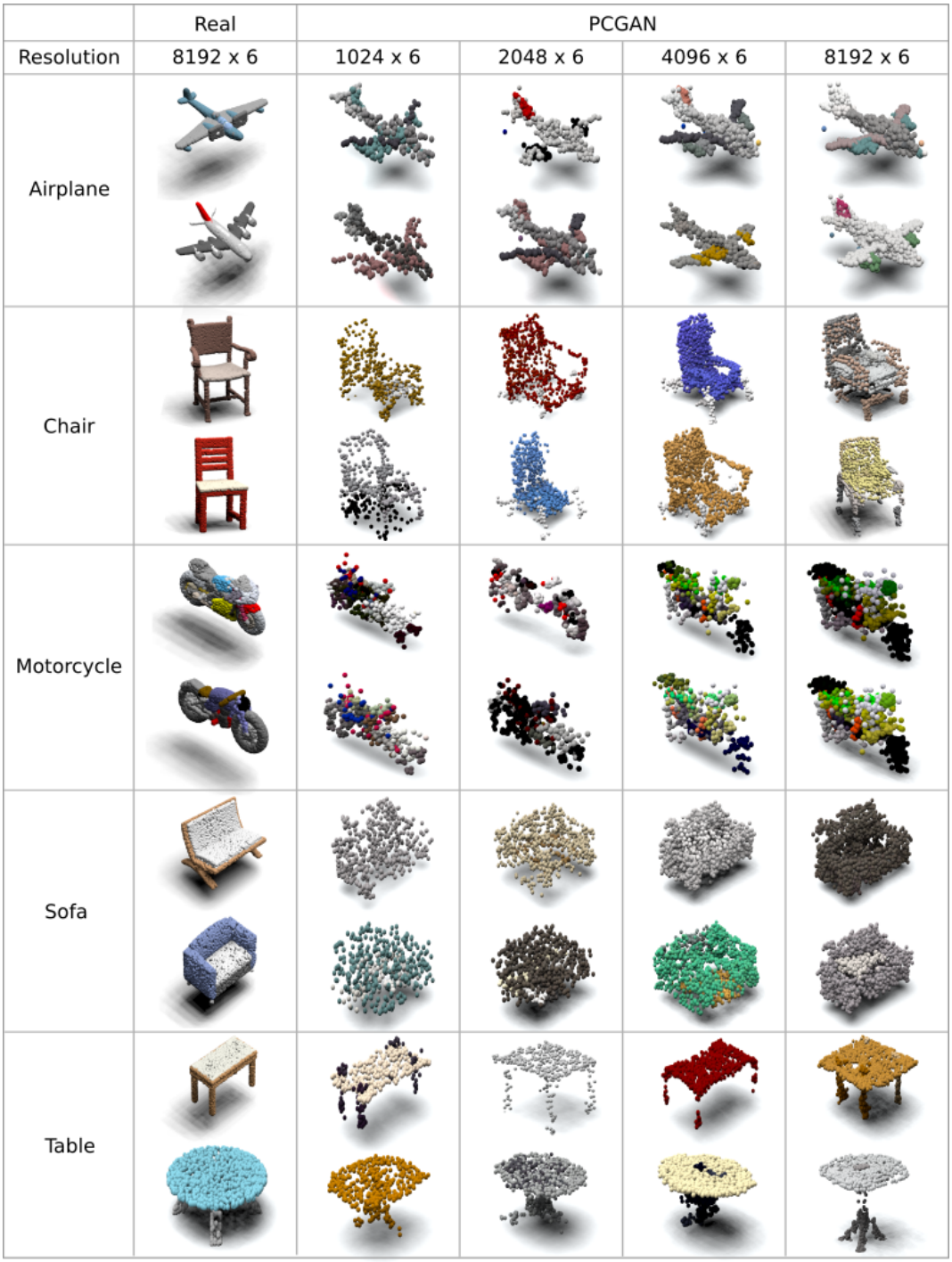


Figure 3.4: The results of generated samples produced by PCGAN. Our model first learns the basic structure of an object at low resolutions and gradually builds up towards high-level details. The relationship between the object parts and their colors (e.g., the legs of the chair/table are the same color while seat/top are contrasting) is also learned by the network. Mitsuba 2 [1] was used to render the point clouds.

Progressive Experiments. We have experimented with different strategies of progressive growing such as branching $H = [2, 4, 4, 4], [1, 2, 4, 8], [1, 2, 4, 2]$ and leaf $h_L = [4, 16, 64]$ increments. Although branching and leaf combinations may generate more discernible point cloud geometries and colors for individual classes, we have found that $H = [1, 2, 2, 2]$ and $h_L = 64$ work best in our experiments. Additionally, instead of creating new branches we have experimented with introducing new leaves for progressive growing. However, this seems to destabilize the network and results in the generation of inferior samples.

Limitations and Future Work. The main drawback of our model is the computational complexity. With ≈ 20000 point clouds from five classes, our model takes roughly 15 minutes per iteration (MPI) on four Nvidia GTX 1080 GPUs to generate point clouds $\hat{x} \in \mathbb{R}^{1024\times 6}$. The MPI rises with every increase in resolution. For future work, we will attempt to reduce the computational complexity of our network. Our model also struggles with generating objects that have fewer examples in the training data. The generated point clouds of the Motorcycle class in Figure 3.4 is an example of this (ShapeNetCore has only 333 CAD models of the Motorcycle). In the future, we will try to improve the generalization ability of our model and we will work on the addition of surface normal estimates.

3.8 Conclusion

This chapter introduces PCGAN, the first conditional generative adversarial network to generate dense colored point clouds in an unsupervised mode. To reduce the complexity of the generation process, we train our network in a coarse to fine way with progressive growing and we condition our network on class labels for multiclass point cloud creation. We evaluate both point cloud geometry and color using our new FDD metric. In addition, we provide comparisons on point cloud geometry with recent generation techniques using available metrics. The evaluation results show that our model is capable of synthesizing high-quality point clouds for a disparate array of object classes.

Class	Model	JSD ↓	MMD-CD ↓	MMD-EMD ↓	COV-CD ↑	COV-EMD ↑
Airplane	r-GAN (dense)	0.182	0.0009	0.094	31	9
	r-GAN (conv)	0.350	0.0008	0.101	26	7
	Valsesia et al. (no up.)	0.164	0.0010	0.102	24	13
	Valsesia et al. (up.)	0.083	0.0008	0.071	31	14
	tree-GAN	0.097	0.0004	0.068	61	20
	PCGAN (ours)	0.085	0.0010	0.070	37	29
Chair	r-GAN (dense)	0.238	0.0029	0.136	33	13
	r-GAN (conv)	0.517	0.0030	0.223	23	4
	Valsesia et al. (no up.)	0.119	0.0033	0.104	26	20
	Valsesia et al. (up.)	0.100	0.0029	0.097	30	26
	tree-GAN	0.119	0.0016	0.101	58	30
	PCGAN (ours)	0.089	0.0027	0.093	30	33
Sofa	r-GAN (dense)	0.221	0.0020	0.146	32	12
	r-GAN(conv)	0.293	0.0025	0.110	21	12
	Valsesia et al. (no up.)	0.095	0.0024	0.094	25	19
	Valsesia et al. (up.)	0.063	0.0020	0.083	39	24
	PCGAN (ours)	0.16	0.0027	0.093	24	27
Motorcycle	**PCGAN** (ours)	0.25	0.0016	0.097	10	9
Table	**PCGAN** (ours)	0.093	0.0035	0.089	45	43

Table 3.2: A qualitative evaluation of the Jensen-Shannon divergence (JSD), the minimum matching distance (MMD), coverage (COV) with the Earth mover's distance (EMD), and the pseudo-chamfer distance (CD). The results of previous studies are from [9, 10]. The magenta and cyan values denote the best and the second best results, respectively. The resolution of the evaluated point clouds was 2048×3.

Class	Real Data	Generated Samples
Airplane	1.12×10^{-5}	4.58
Chair	2.07×10^{-9}	3.07
Motorcycle	1.04×10^{-4}	13.25
Sofa	5.88×10^{-6}	3.14
Table	4.37×10^{-7}	2.02

Table 3.3: The FDD score for point cloud samples generated by PCGAN. Notice that the scores for real point clouds are almost zero. The point clouds were evaluated at a resolution of 8192×6.

CHAPTER 4

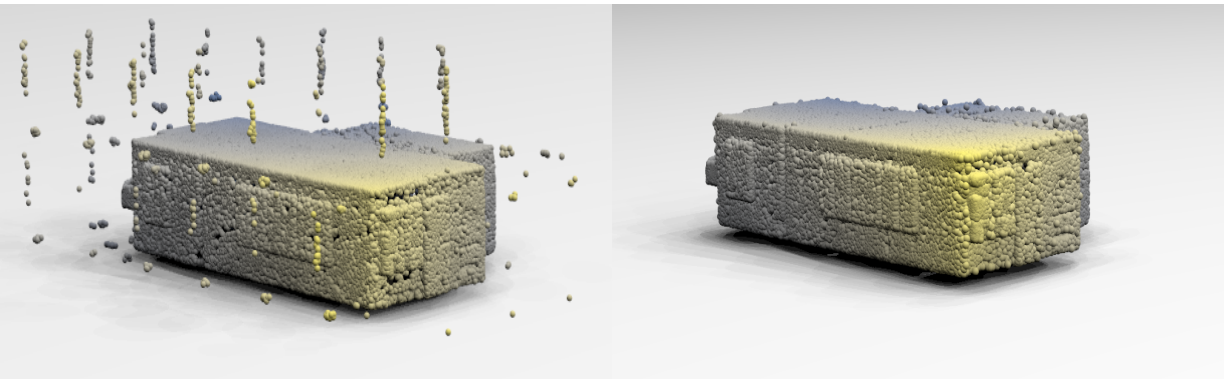
IPVNet: Learning Implicit Point-Voxel Features for Open-Surface 3D Reconstruction

4.1 Introduction

3D computer vision for generating (e.g., [69, 70]) and reconstructing (e.g., [71, 72]) point clouds has gained momentum due to applications such as robotics, autonomous driving, and virtual reality. Capturing detailed point cloud data from the real world is a difficult and expensive task. Moreover, due to the limitations of 3D sensor technologies (e.g., LiDAR, RGBD, etc.), data can be sparse (i.e., missing details) and incomplete (i.e., noisy with holes and outliers). The 3D reconstruction of missing parts and reintroduction of details is not a trivial task. Researchers have looked into a myriad of ways to complete 3D data. Learning-based implicit functions have become popular among 3D reconstruction techniques due to their demonstrated superiority in capturing details and the ability to generate data in arbitrary resolutions.

Implicit functions operate by first converting raw data into an occupancy grid and then learning a voxel occupancy or a distance field that classifies a query point as either inside or outside of the surface. In low resolutions, occupancy grids lose information during voxelization since multiple points within the boundary of a grid are merged together. To preserve fine details in the input data, a high-resolution representation is required. However, the computational costs and memory requirements increase *cubically* with voxel resolution. For example, Chibane et al. [3] require 8.86 GB of memory to train with a single input (batch size 1) at a resolution of 256^3. This large memory footprint makes it impractical to scale beyond the aforesaid resolution.

Instead of relying on the voxels, researchers have also tried to use raw point clouds with a learned signed distance field (SDF) on the surface. Nevertheless, implicit functions that learn an SDF via extraction of a zero level set must distinguish between the inside/outside of the surface.



(a) (b)

Figure 4.1: An outside view of a dense 3D reconstructed scene from the Gibson Environment [2] dataset using (a) the state of the art [3], and (b) our proposed approach. Note that our method produces significantly less outliers.

As a result, the reconstruction is produced as a closed surface even if the target shape includes surface gaps. However, real-world data may consist of salient open surfaces. Closing the surface of such data often leads to the introduction of outliers and lost details.

To reconstruct accurate geometry without introducing outliers, we propose IPVNet, an implicit model that learns an unsigned distance field (UDF) by jointly accumulating features from raw point clouds and voxel grids to reconstruct open surfaces. As shown in Figure 4.1, our approach produces significantly less outliers compared to the state of art [3]. Note that by reconstructing a surface, we refer to the construction of dense point clouds that lie on the surface, which is a key part of the reconstruction process. Although one could extract and render the surface mesh from such point clouds, we present our results in the form of *raw point clouds*. For completeness, we record additional results as rendered meshes in our experimental evaluation.

The technique of learning features from both point clouds and voxels has been shown to achieve superior performance in classification and segmentation, detection, and generation. However, to the best of our knowledge, our work is the first approach to combine point-voxel features to learn a UDF for open-surface reconstruction. Such improved features allow our model to recon-

struct *richer detail* even with *low-resolution* voxel grids. Moreover, the involvement of raw point features allows us to use a more sophisticated inference module that produces *significantly fewer* outliers in the reconstructed output. Our key contributions are summarized as follows.

- We introduce IPVNet, a novel approach for implicitly learning from raw point cloud and voxel features to 3D reconstruct complex open surfaces.
- We develop an inference module that extracts a zero level set from a UDF and drastically lowers the amount of outliers in the reconstruction.
- We show that IPVNet outperforms the state of the art on both synthetic and real-world public datasets, and we provide an ablation analysis to understand the importance of point-voxel fusion.

To reproduce and improve upon our results, the project source code is publicly available to the research community [73].

The remainder of this chapter is organized as follows. We provide an overview of related research in Section 4.2. In Section 4.3, a concise summary of implicit functions is provided. The details of IPVNet are presented in Section 4.4, and the experimental evaluation and results are described in Section 4.5. We discuss limitations and future directions of our work in Section 4.7 and a conclusion is given in Section 4.8.

4.2 Related Work

3D reconstruction is a well researched area with a number of different approaches and algorithms. In this section, we review and compare our work with learning-based implicit approaches. For a more comprehensive review, we refer the reader to a contemporary survey on 3D reconstruction [74].

4.2.1 Implicit Function Learning

Instead of explicitly predicting a surface, implicit feature learning methods try to either predict if a particular point in 3D space is inside or outside of a target surface (occupancy), or determine how far the point is from the target surface (distance). To reconstruct 3D data in arbitrary resolutions and learn a continuous 3D mapping, Mescheder et al. [75] presented a network that predicts voxel occupancy. Peng et al. [76] improved the occupancy network by incorporating 2D and 3D convolutions. An encoder-decoder architecture was used by Chen et al. [77] to learn voxel occupancy. Michalkiewicz et al. [78] estimated an oriented level set to extract a 3D surface. Littwin and Wolf [79] used encoded feature vectors as the network weights to predict voxel occupancy. Park et al. [80] introduced DeepSDF, an encoder-decoder architecture that predicts a signed distance to the surface instead of voxel occupancy. Genova et al. [81] divided an object's surface into a set of shape elements and utilized an encoder-decoder to learn occupancy. Sitzmann et al. [82] introduced SIREN to implicitly learn complex signals for various downstream tasks including 3D shape representations via a signed distance function.

Bhatnagar et al. [83] combined implicit functions with parametric modeling to jointly reconstruct body shape under clothing by predicting occupancy. To retain richer details in the reconstruction, Chibane et al. [84] used 3D feature tensors to predict voxel occupancy. Rather than transforming point clouds into a occupancy grid, Atzmon and Lipman made use of raw point clouds to learn and predict an SDF to the target surface [85], and later incorporated derivatives in a regression loss to further improve the reconstruction accuracy [86]. Gropp et al. [87] used geometric regularization to learn directly from raw point clouds. To make the reconstruction process more scalable, Mi et al. [88] introduced SSRNet to construct local geometry-aware features for octree vertices. By leveraging gradient-based meta-learning algorithms, Sitzmann et al. [89] developed MetaSDF to improve the generalization ability of implicit learning.

With a similar aim of improving generalization, Tretschk et al. [90] created a patch-based representation to learn an SDF. A local implicit grid to learn an SDF and reconstruct 3D data was used by Jiang et al. [91]. Liu et al. [92] implemented deep implicit least squares to regress an SDF for 3D reconstruction. Deep implicit fusion to estimate an SDF for online 3D reconstruction was presented by Huang et al. [93]. Duggal et al. [94] used signed distance regression via neural implicit modeling for 3D vehicle reconstruction from partial/noisy data. Sign agnostic learning to estimate a signed implicit field of a local surface for 3D reconstruction was proposed by Zhao et al. [95].

All of the aforementioned works either predict a voxel occupancy or a signed distance value for a given query point, which is inadequate to reconstruct open surfaces. To alleviate this inadequacy, Chibane et al. [3] predicted a UDF from an input voxel occupancy. A similar technique to learn a UDF for single-view garment reconstruction was used by Zhao et al. [96]. Venkatesh et al. [97] proposed a closest surface point representation to reconstruct both open and closed surfaces. A new NULL sign combined with conventional in and out labels to reconstruct a non-watertight arbitrary topology was proposed by Chen et al. [98]. Ye et al. [5] leveraged the relationship between every two points, instead of points and surfaces, to improve the reconstruction quality of non-watertight 3D shapes. The aforementioned works only utilize the discretized voxel representation while we make use of raw point clouds *jointly* with voxel occupancy. This enables us to accumulate improved features and reconstruct finer details with less outliers.

4.2.2 Learning from Points and Voxels

Due to the convenience of using volumetric convolutions, many works have explored voxel-based representations (e.g., [99, 100, 101]). However, voxel grids grow *cubically* with resolution and their memory intensiveness imposes an upper bound to the highest resolution possible. Point clouds are memory efficient, yet it is non-trivial to extract features from them due to their sparsity

and permutation invariant nature. Recently, researchers started combining these two representations to get the best out of them both.

Liu et al. [102] introduced PVCNN to perform classification and segmentation by extracting features from both point clouds and voxel grids via voxelization and de-voxelization. Fusion between voxel and point features for 3D classification was used by Li et al. [103]. Shi et al. [104] gathered multi-scale voxel features and combined them with point cloud keypoint features for object detection. They further improved their results by incorporating local vector pooling [105]. Point-voxel fusion to detect 3D objects was used by Cui et al. [106] and Tang et al. [107] learned a 3D model via sparse point-voxel convolution.

Noh et al. [108] accumulated point-voxel features in a single representation for 3D object detection. PVT, a transformer-based architecture that learns from point-voxel features for point cloud segmentation was introduced by Zhang et al. [109]. Wei et al. [110] used point-voxel correlation for scene flow estimation and Li et al. [111] utilized point-voxel convolution for 3D object detection. Xu et al. [112] introduced RPVNet for point cloud segmentation via point-voxel fusion. Cherenkova et al. [113] utilized point-voxel deconvolution for point cloud encoding/decoding. In contrast to the preceding works, we use a point-voxel representation to learn an implicit function for open-surface reconstruction. To restore lost details during voxelization, we propose a novel aggregation strategy that accumulates features from both point clouds and voxel grids.

4.3 Background

Implicit functions rely on one of three output choices for surface reconstruction. Concretely, given a latent representation $z \in \mathcal{Z}$ of a point cloud object $x \in \mathcal{X} \subset \mathbb{R}^{N\times 3}$ and a random query point $p \in \mathbb{R}^3$, an implicit function f aims to predict the following.

(i) Occupancy, i.e., if p lies inside or outside the object,

$$f(z, p) : \mathcal{Z} \times \mathbb{R}^3 \rightarrow [0, 1]. \tag{4.1}$$

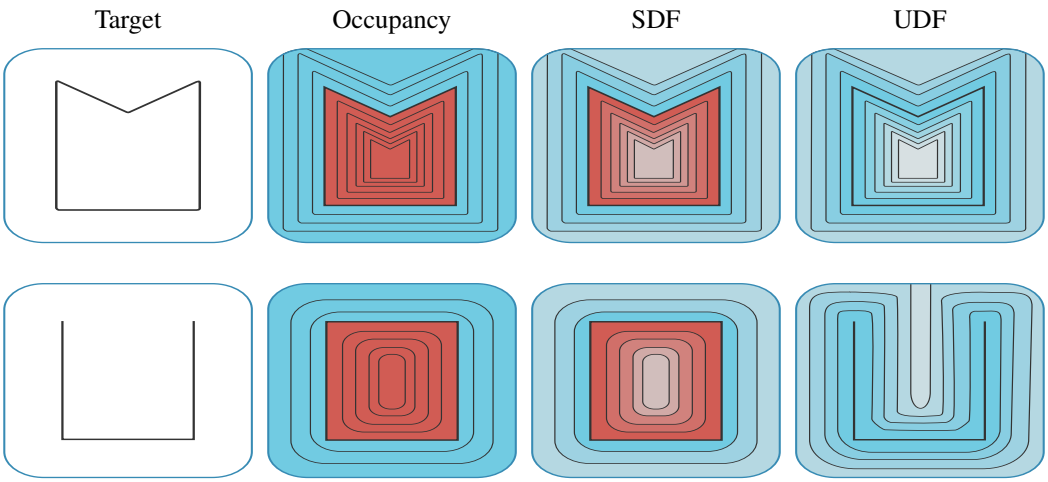


Figure 4.2: A 2D representation of a closed (top row) and an open (bottom row) surface reconstruction via occupancy, signed distance field (SDF), and unsigned distance field (UDF). Note that the occupancy and SDF reconstructions of the open surface closes the gap by producing an artifact while a UDF can preserve the surface opening. The color intensity represents the distance from the surface where blue represents a positive value, and red represents a zero (occupancy) or negative value (SDF).

(ii) Signed distance, i.e., the distance from p to the inside or outside surface of the object,

$$f(z, p) : \mathcal{Z} \times \mathbb{R}^3 \to \mathbb{R}. \tag{4.2}$$

(iii) Unsigned distance, i.e., the absolute distance from p to any surface on the object,

$$f(z, p) : \mathcal{Z} \times \mathbb{R}^3 \to \mathbb{R}_+. \tag{4.3}$$

In (4.1) - (4.3), $\mathcal{X}$ denotes the input space, $\mathcal{Z}$ is the latent space, and $N \in \mathbb{N}$ is the density/resolution of the point cloud. After learning the implicit function f, it may be queried multiple times to find the decision boundary (occupancy) or the zero level set (signed and unsigned distance) thus implicitly reconstructing the surface of the desired object. Figure 4.2 provides an overview of open/closed surface reconstruction via different implicit techniques in 2D.

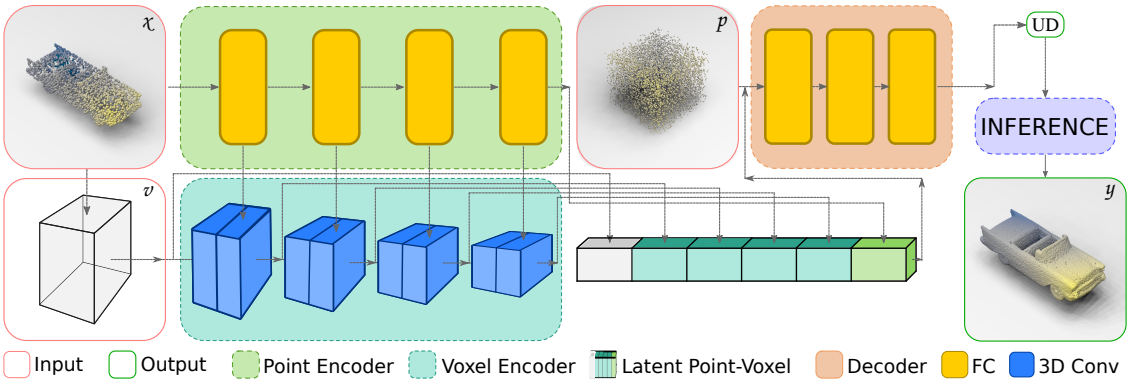


Figure 4.3: Given a sparse point cloud $x \in \mathcal{X}$ of an object, we use a novel encoding scheme to extract and aggregate point-voxel features from both the raw point cloud (x) and the voxel occupancy (v). From the accumulated features, a decoder module regresses the unsigned distance $UD(p, S)$ from query point p to the surface S. By querying the decoder multiple times, the inference submodule can reconstruct the surface of any target shape.

4.4 Implicit Learning with Point-Voxel Features

An overview of our network is presented in Figure 4.3. We have chosen unsigned distances as the output representation of IPVNet due to their precedence in representing different surfaces. Given a sparse point cloud $x \in \mathcal{X} \subset \mathbb{R}^{N\times 3}$ of an object, we use a novel encoding scheme to extract and aggregate point-voxel features from both the raw point cloud (x) and the voxel occupancy (v). From the accumulated features, a decoder module regresses the unsigned distance $UD(p, S)$ via any query point p to the surface S. In the following subsections we describe the elements of our approach.

4.4.1 Point-Voxel Features

To extract a set of multi-level features from a point cloud x, we define a neural function

$$\Theta(x) := (z_x^1, \ldots, z_x^j) \mid \Theta \colon \mathbb{R}^3 \to \mathcal{Z}, \tag{4.4}$$

where $z_x \in \mathcal{Z} \subset \mathbb{R}$ corresponds to the extracted feature vector from the raw point cloud x, and j is the total number of layers in Θ. During the early stages (i.e., when $j = 1$) the encoder is more focused on local details, whereas at the later stages the focus is shifted towards the global structure. ReLu [114] nonlinearity is used for all layers except the output layer of the point encoder.

Instead of limiting the encoded features to a single dimensional vector, a voxel representation allows for the construction of a multi-dimensional latent matrix. However, such an encoding scheme requires the input point cloud x to be discretized into a voxel grid v, i.e., $x \approx v \colon \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{M\times M\times M}$ where $M \in \mathbb{N}$ is the grid resolution. Due to the discretization process, voxel grids lose information since multiple points may lie within the same voxel. To reintroduce lost details, we combine voxel features with point features z_x. Although, a different fusion strategy can be applied to combine these features, we empirically found that a simple concatenation strategy works best.

Let $\Phi \colon \mathbb{R}^{M\times M\times M} \rightarrow \mathcal{Z}^{M\times M\times M}$ be a neural function that encodes the combined point-voxel features into a set of multi-dimensional feature grids z_{xv} of monotonically decreasing dimension. Then,

$$\Phi(v \odot \Theta(x)) := (z_{xv}^{k\times k\times k}, \ldots, z_{xv}^{l\times l\times l}), \tag{4.5}$$

where $k, l \in \mathbb{N}$ represents the dimensional upper and lower bound of the feature grid $(M > k \gg l > 1)$. The subscript xv denotes the dependency on both points and voxels. Similar to its point counterpart, the voxel encoder is more directed towards local details at the early stages. However, as the dimensionality is reduced and the receptive field grows larger, the aim shifts to the global structure. ReLu is utilized to ensure nonlinearity and batch normalization [115] provides stability while training. The latent point (z_x^j) from the point encoder, along with multi-dimensional features (z_{xv}) from the point-voxel encoder and the discretized voxel grid (v), are then used to construct the latent point-voxel:

$$z = \{z_x^j, \Phi(v \odot \Theta(x)), v\}. \tag{4.6}$$

4.4.2 Implicit Decoding

Given a query point $p \in \mathbb{R}^3$, a set of deep features F_p is sampled from the latent point-voxel features z via a sampling function Ω [116]. Specifically,

$$\Omega(z, p) := (F_p^1 \times \cdots \times F_p^n), \tag{4.7}$$

where $n = |z|$. We extract features from a neighborhood of distance $d \in \mathbb{R}$ along the Cartesian axes centered at p to obtain rich features. More formally,

$$p := \{p + q \cdot c_i \cdot d\} \in \mathbb{R}^3 \mid q \in \{1, 0, -1\}, i \in \{1, 2, 3\}, \tag{4.8}$$

where $c_i \in \mathbb{R}^3$ is the ith Cartesian axis unit vector. We define a neural function Ψ that regresses the unsigned distance to the surface S of x from the deep features (F_p). Concretely,

$$\Psi(F_p^1, \ldots, F_p^n) \approxeq UD(p, S) \mid \Psi \colon \mathcal{Z} \to \mathbb{R}_+, \tag{4.9}$$

where $UD(\cdot)$ is a function that returns the unsigned distance from p to the ground-truth surface S for any $p \in \mathbb{R}^3$. Hence, the implicit decoder to regress the unsigned distance at a given query point p is defined as

$$f_x(z, p) := (\Omega \circ \Psi)(p) \mid f_x \colon \mathcal{Z} \times \mathbb{R}^3 \to \mathbb{R}_+. \tag{4.10}$$

4.4.3 Training

IPVNet requires a pair $\{X_i, S_i\}_{i=1}^T$ associated with input X_i and corresponding ground-truth surface S_i for implicit learning. Parameterized by the neural parameter w, the point-encoder, voxel-encoder, and decoder are jointly trained with a mini-batch loss,

$$\mathcal{L}_\mathcal{B} := \Sigma_{x \in B} \Sigma_{p \in \mathcal{P}} |\min(f_x^w(p), \delta) - \min(UD(p, S_x), \delta)|, \tag{4.11}$$

where $\mathcal{B}$ is a mini-batch of input and $\mathcal{P} \in \mathbb{R}^3$ is a set of query points within distance δ of S_i. We use a clamped distance $0 < \delta < 10$ (cm) to improve the capacity of the model to represent the vicinity of the surface accurately.

4.4.4 Surface Inference

We use an iterative strategy to extract surface points from f_x. More specifically, given a perfect approximator $f_x(p)$ of the true unsigned distance $UD(p, S_i)$, the projection of p onto the surface S_i can be obtained by

$$q := p - f_x(p) \cdot \nabla_p f_x(p), \quad q \in S_i \subset \mathbb{R}^d, \forall p \in \mathbb{R}^d / C. \tag{4.12}$$

In (4.12), C is the cut locus [117], i.e., a set of points that are equidistant to at least two surface points. The negative gradient indicates the direction of the fastest decrease in distance. In addition, we can move a distance of $f_x(p)$ to reach q if the norm of the gradient is one. By projecting a point multiple times via (4.12), the inaccuracies due to $f_x(p)$ being an imperfect approximator can be reduced. Furthermore, filtering the projected points to a maximum distance threshold (max_thresh) and re-projecting them onto the surface after displacement by $d \sim \mathcal{N}(0, \delta/3)$ can ensure higher point density within a maximum distance (δ).

Instead of uniformly sampling query points within the bounding box of S_i, we use the input points $X_i \in \mathbb{R}^3$ as guidance for the query points. In particular, we apply a random uniform jitter $\mathcal{J}_a^b \in \mathbb{R}^3$ within bounds a and b to displace the input points X_i. Due to the inclusion of point features in learning, this procedure allows our model to infer more accurate surface points while restricting the number of outliers. Note that without the use of point features, this perturbation of the input points fails to restrict the number of outliers (see Section 4.6). The details of the inference procedure are provided in Algorithm 1.

Algorithm 1 Surface Point Inference

procedure INFERENCE($\mathcal{X}$)
  $\mathcal{J} \leftarrow m$ points from $U(a, b)$
  $\mathcal{P}_{init} \leftarrow \{x + j\}, \forall x \in \mathcal{X}, \forall j \in \mathcal{J}$
  for $i = 1$ to $num\_projections$ **do**
    $p \leftarrow p - f_x(p) \cdot \frac{\nabla_p f_x(p)}{||\nabla_p f_x(p)||}, \forall p \in \mathcal{P}_{init}$
  end for
  $\mathcal{P}_{filtered} \leftarrow \{p \in \mathcal{P}_{init} \,|\, f_x(p) < max\_thresh\}$
  $\mathcal{P}_{filtered}$: draw out_res number of points with replacement
  $\mathcal{P}_{filtered} \leftarrow \{p + d\} \,|\, p \in \mathcal{P}_{filtered}, d \sim \mathcal{N}(0, \delta/3)$
  for $i = 1$ to $num\_projections$ **do**
    $p \leftarrow p - f_x(p) \cdot \frac{\nabla_p f_x(p)}{||\nabla_p f_x(p)||}, \forall p \in \mathcal{P}_{filtered}$
  end for
  return $\{p \in \mathcal{P}_{filtered} \,|\, f_x(p) < max\_dist\}$
end procedure

4.4.5 Implementation Details

IPVNet was implemented using PyTorch. To extract point cloud and voxel features, we utilize multilayer perceptrons (MLPs) and 3D convolutional neural networks (CNNs), respectively. Specifically, we employ a 7-layer fully-connected MLP as the point encoder and 6-layer CNN blocks as the voxel encoder. Point features are obtained from each of the hidden layers of the point encoder and are combined with voxel features derived from the initial layer of each convolution block. We use max pooling with the fully-connected layers to make them permutation invariant.

Figure 4.4 shows the details of the different neural modules of IPVNet. To train the model, we used a learning rate of 10^{-6} and the Adam [67] optimizer. With a voxel resolution of 256^3 and a batch size of 4, it takes around 3.2 seconds to perform a forward pass using 4 Nvidia GeForce GTX 1080 Ti GPUs. To infer a single target surface with a dense point cloud consisting of 1 million points, IPVNet takes approximately 120 seconds.

4.5 Experiments

In this section, we validate the performance of IPVNet on the task of 3D object and scene reconstruction from sparse point clouds.

4.5.1 Baselines and Metrics

To compare the reconstruction quality of IPVNet, we utilized the open-source implementations of NDF [3] and GIFS [5] as baseline methods. For an unbiased comparison, we trained an NDF following the directions from [3] on our train-test split until a minimum validation accuracy was achieved. To quantitatively measure the reconstruction quality, we used the *chamfer-*L_2 distance (CD) and F-score to measure the accuracy and completeness of the surface.

4.5.2 Object Reconstruction

Due to the abundance of surface openings, we chose the "Cars" subset of the ShapeNet [4] dataset for our object reconstruction experiment. We used a random split of 70%-10%-20% for training, validation, and testing, respectively. To prepare the ground truth and input points we followed the data preparation procedure outlined in [3]. Additionally, we fixed the output point density to $\mathcal{O} = 1$ million to extract a smooth mesh from the point cloud using a naive algorithm (e.g., [118]).

To understand the effects of sparse input on the reconstruction quality, we evaluated IPVNet and the baseline using an input density of $N \sim \{300, 3000, 10000\}$ points while fixing the voxel resolution to $M = 256$. In contrast to the baseline, IPVNet can reconstruct thin structures more accurately and preserve small gaps (Figure 4.5 inset images) while quantitatively outperforming the reconstruction with different input densities (Table 4.1). Furthermore, we investigated IPVNet's ability to perform closed-surface reconstruction on preprocessed watertight meshes using 13 subsets of ShapeNet for training. The reconstruction results are shown in Figure 4.6.

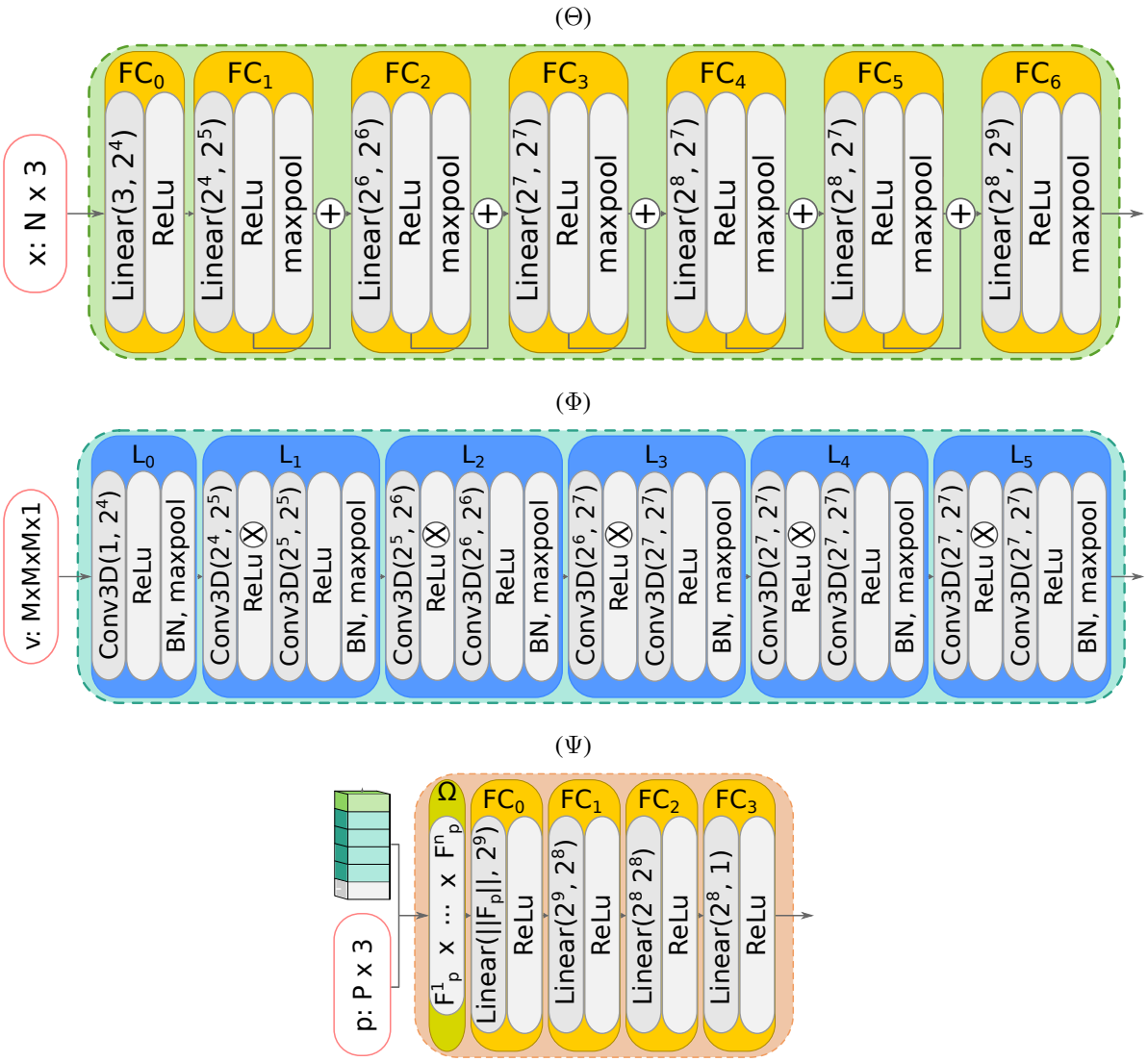


Figure 4.4: A visual depiction of the different neural architectures of IPVNet. $\bigoplus$ in (Θ) represents concatenation, and $\bigotimes$ in (Φ) indicates the fusion of point features with voxel features.

	Chamfer-L_2 ↓			*F-score* ↑	
	$N = 300$	$N = 3000$	$N = 10000$	$d = 0.1\%$	$d = 0.05\%$
NDF	1.550	0.324	0.092	0.711	0.460
IPVNet	**1.217**	**0.119**	**0.068**	**0.785**	**0.542**

Table 4.1: A quantitative comparison between IPVNet and NDF [3] on the ShapeNet Cars [4] dataset for object reconstruction from different input densities. IPVNet outperforms NDF on all input densities. The chamfer-L_2 results are of order $\times 10^{-4}$ and the reconstruction results using an input density of $N = 10000$ were used to calculate the F-score.

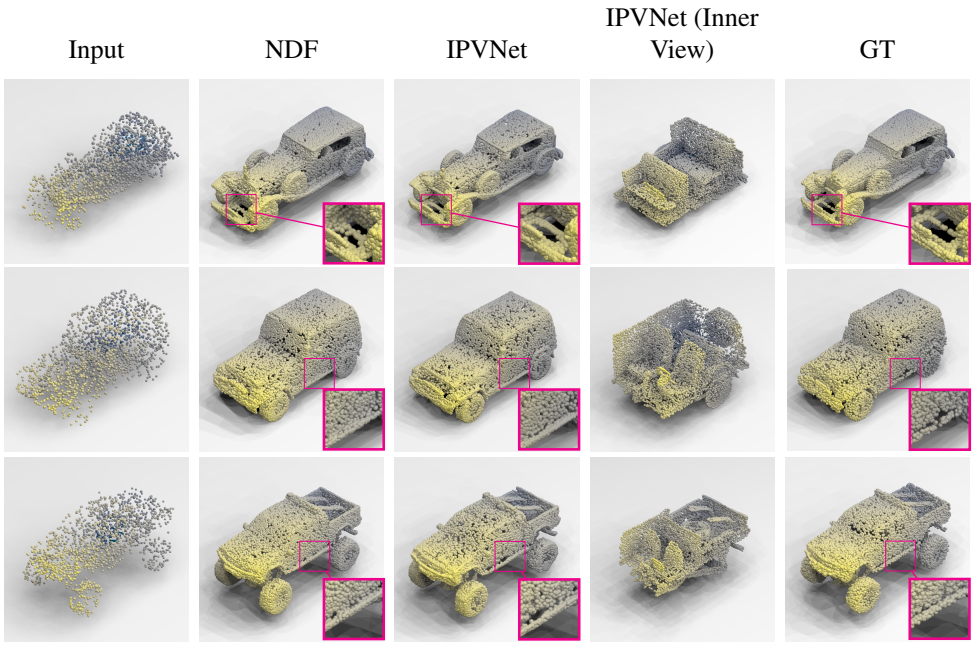


Figure 4.5: Object reconstruction using NDF [3], IPVNet, and the ground truth (GT) from the ShapeNet Cars [4] test set. IPVNet performs better on reconstructing thin structures and preserving small gaps (inset images).

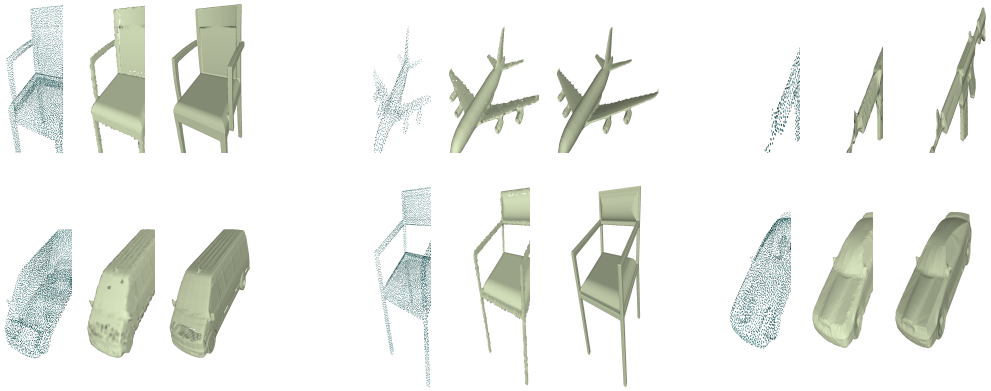


Figure 4.6: Reconstruction of closed surfaces from the ShapeNet [4] dataset. From left to right, each triplet represents the input, reconstruction, and ground truth, respectively.

	Chamfer-$L_2 \downarrow$			F-*score*$_{0.05} \uparrow$		
	64^3	128^3	256^3	64^3	128^3	256^3
NDF	1.549	0.266	0.029	0.289	0.591	0.994
GIFS	5.245	1.210	0.141	0.240	0.510	0.891
IPVNet	**1.441**	**0.162**	**0.023**	**0.335**	**0.803**	**0.995**

Table 4.2: A quantitative comparison between NDF [3], GIFS [5], and IPVNet on the Garments [6] dataset for object reconstruction at different voxel resolutions. IPVNet outperforms the baselines by significant margin in lower resolutions. The point density was fixed to $N = 3K$ for this experiment. The chamfer-L_2 results are of order $\times 10^{-4}$.

4.5.3 Real-World Scene Reconstruction

We evaluated the reconstruction of complex real-world scenes through the use of the Gibson Environment dataset [2]. The dataset consists of RGBD scans of indoor spaces. A subset of 35 and 100 scenes were prepared following the procedure from [3] for training and testing, respectively. We utilized a sliding window scheme and reconstructed the surface bounded by each window. Since the sliding window may frequently consist of a very small area of the scene with only few points, we used an output density five times as large the input density (i.e., $\mathcal{O} = 5 \times N$) to save time. The grid resolutions were kept fixed at $M = 256$ for both IPVNet and the baseline. The reconstruction results are highlighted in Figure 4.7. In addition to improving the preservation of structural details, IPVNet produces significantly fewer outliers than the baseline due to the use of point features during training and inference.

Lastly, we tested IPVNet on the challenging complex surfaces of the Garments [6] dataset. Figure 4.8 and Table 4.2 show the qualitative and quantitative results, respectively. It can be observed from Table 4.2 that IPVNet exhibits superior performance compared to the baselines, particularly at low grid resolutions. The point-voxel fusion technique utilized by our model is able to effectively recover lost details from the resulting discretization.

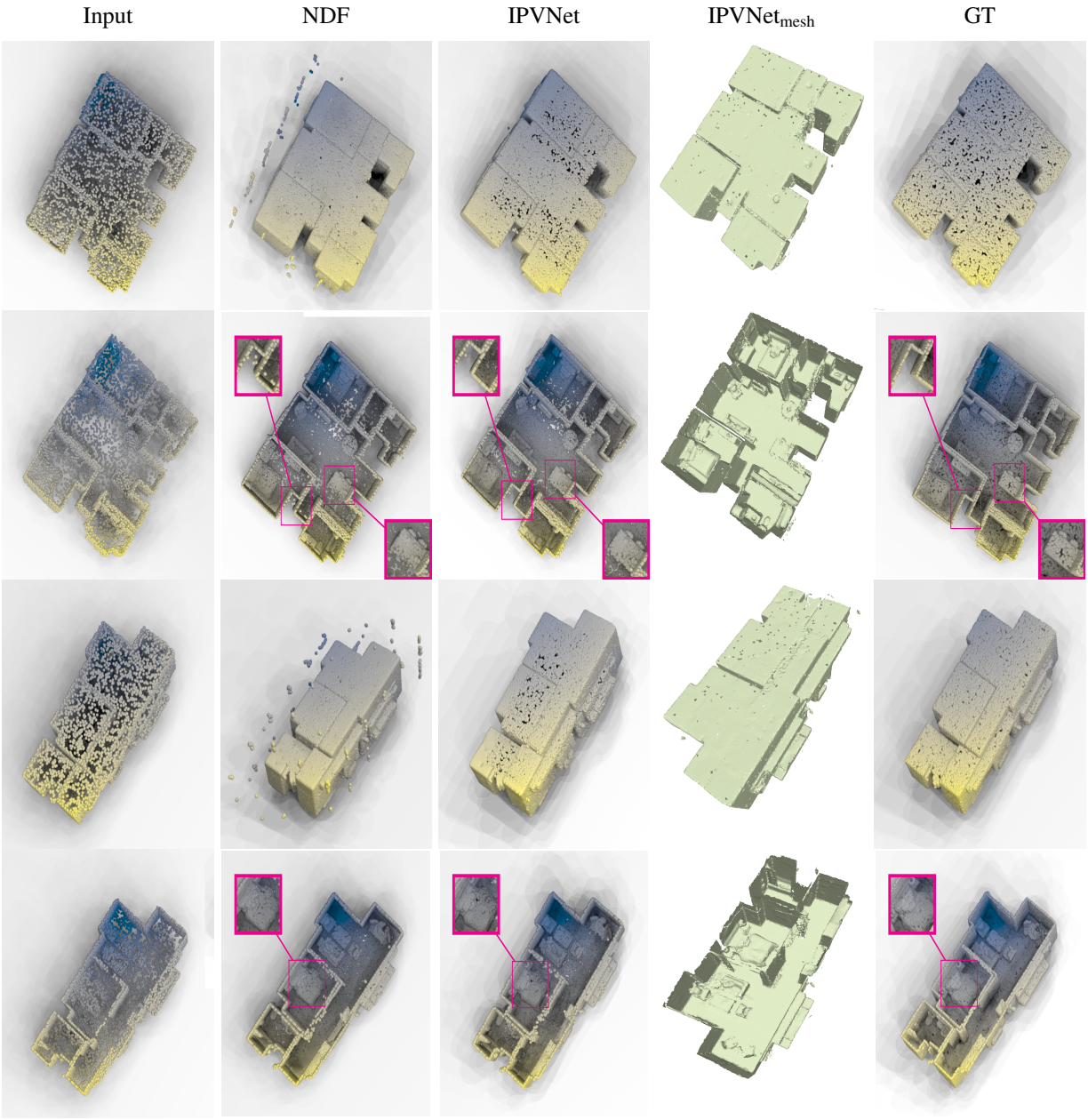


Figure 4.7: Scene reconstruction on the test set of the Gibson Environment [2] dataset using NDF [3], IPVNet, and the respective ground truth (GT). Each odd row represents an outside view of a scene while the even rows depict inside views. In contrast to the baseline, IPVNet produces significantly less outliers (outside view) and improves the preservation of geometric features (inset images).

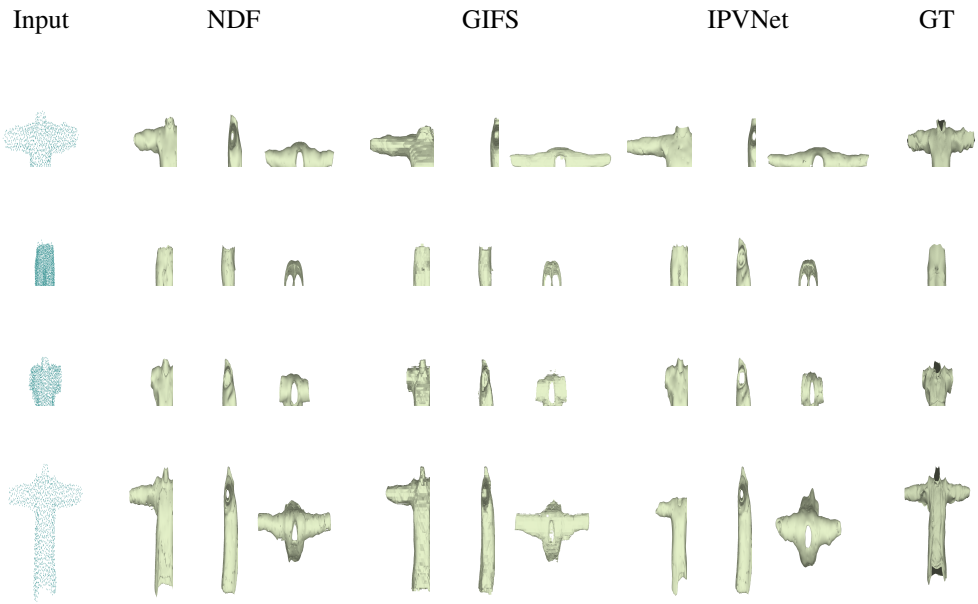


Figure 4.8: A qualitative comparison between NDF [3], GIFS [5], and IPVNet on the Garments [6] dataset.

4.6 Ablation Study

In this section, we study the effect of different design choices and how they influence the performance of IPVNet on the task of 3D reconstruction.

4.6.1 Effect of Point Features on Object Reconstruction

Since multiple points within the boundary of a grid are merged together in low resolutions, we test the effect of this information loss on object reconstruction. To understand if the point features are helpful in recovering missing information, we trained a version of IPVNet named IPVNet$_{\text{wp}}$, which has the same neural functions except for the point encoder and point-feature aggregation. Both IPVNet and IPVNet$_{\text{wp}}$ were trained with differing grid resolutions, $M \sim \{32, 64, 128, 256\}$, while using a fixed input point density of $N = 10000$. Our findings on the reconstruction of the ShapeNet Cars dataset are illustrated in Table 4.3. At lower resolutions,

Grid Resolution	% of Lost Points	*Chamfer-*$L_2 \downarrow$	
		IPVNet$_{wp}$	IPVNet
32	82%	9.587	**4.307**
64	45%	0.961	**0.543**
128	16%	0.395	**0.257**
256	4%	0.092	**0.068**

Table 4.3: The object reconstruction accuracy for different grid resolutions on the ShapeNet Cars [4] dataset using only voxel features (IPVNet$_{wp}$) and point-voxel features (IPVNet). The second column represents the percentage of raw points lost during the voxelization process due to multiple points overlapping in the same grid. In low-resolution grids, IPVNet significantly outperforms IPVNet$_{wp}$. The chamfer-L_2 results are of order $\times 10^{-4}$.

where a significant percentage of the raw points are lost due to voxelization, IPVNet outperforms IPVNet$_{wp}$ by a notable margin thus indicating the usefulness of point features.

4.6.2 Effect of Point Features on Scene Reconstruction

To test the effectiveness of point features on scene reconstruction, we used Algorithm 1 to infer the surface for both IPVNet and IPVNet$_{wp}$. The reconstruction results are displayed in Figure 4.9. Compared to the baseline, Algorithm 1 by itself can reduce the number of outliers without point features (Figure 4.9b). However, when point features are included during training the reconstruction results (Figure 4.9c) are closer to the ground truth (Figure 4.9d), and therefore more accurate details with less outliers are realized.

4.6.3 Post-Processing Outlier Removal

To provide a comparison of IPVNet against a naive post-processing step, we filter the baseline reconstruction using the coordinate range of the input point cloud as the distance threshold. The qualitative results of this experiment are recorded in Figure 4.10. It is critical to note that naive post-processing cannot remove all the outliers due to their existence near areas of surface curvature.

4.7 Limitations and Future Directions

Despite the fact that the UDF function is capable of reconstructing multiple complex surfaces, the requirement of projecting the query points several times makes the surface inference time long. Our point-voxel formulation may also be beneficial for other implicit techniques such as occupancy and signed distance prediction. We aim to investigate these directions in future work.

4.8 Conclusion

In this chapter we introduced IPVNet, a novel approach that implicitly learns from raw point and voxel features to reconstruct complex open surfaces. To improve the reconstruction quality, we make use of raw point cloud data jointly with voxels to learn local and global features. Not only have we showed that IPVNet outperforms the state of the art on both synthetic and real-world data, but we also demonstrated the effectiveness of point features on 3D reconstruction through ablation studies. Furthermore, we developed an inference module that extracts a zero level set from a UDF and drastically reduces the amount of outliers in the reconstruction. We believe IPVNet is an important step towards reconstructing open surfaces without losing details and introducing outliers, and we hope that our work will inspire more research in this area.

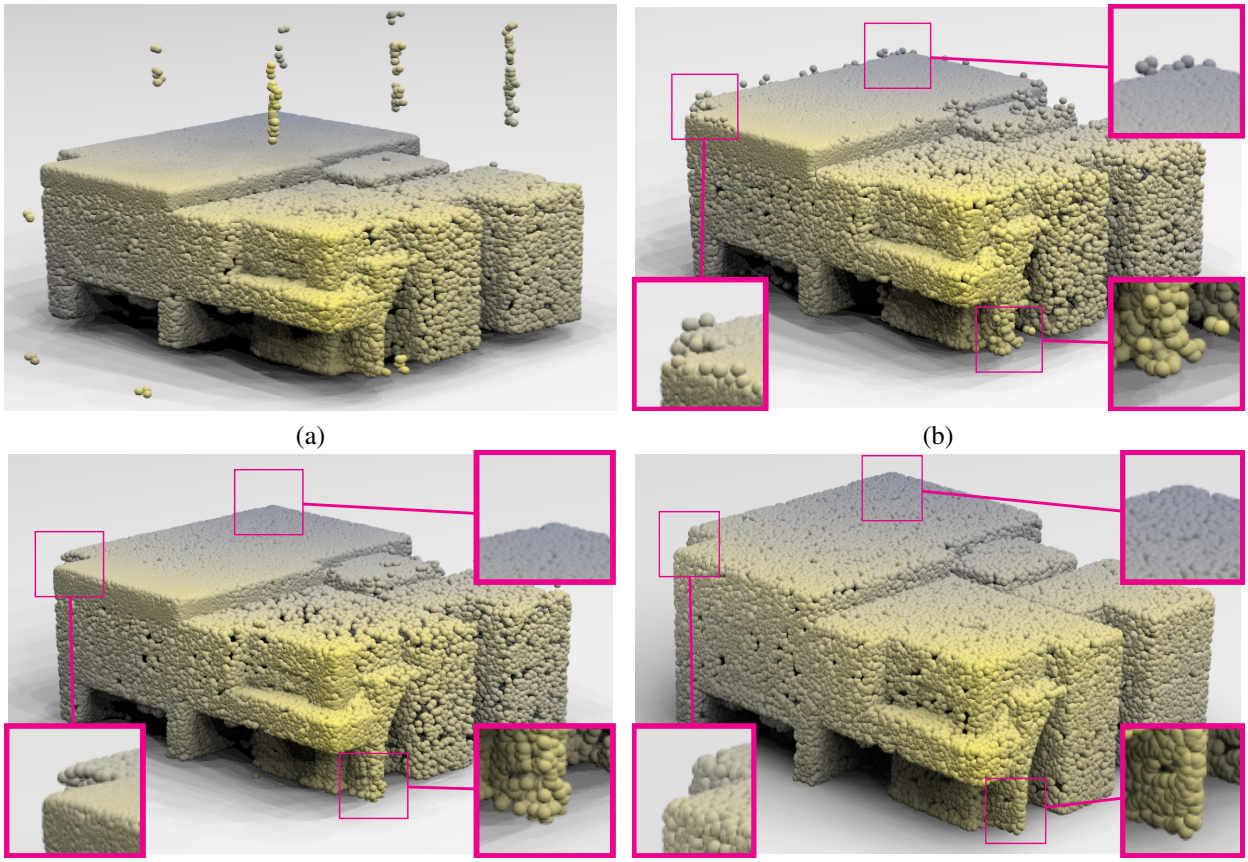


Figure 4.9: An ablation study showing the effectiveness of point features during training. To reconstruct a scene from the Gibson Environment [2] dataset, we used (a) the NDF [3] baseline and (b) $\text{IPVNet}_{\text{wp}}$ with our inference algorithm (Algorithm 1). The IPVNet reconstruction results are shown in (c) and the ground truth is displayed in (d). Notice that Algorithm 1 by itself can reduce the number of outliers. However, when point features are included during training, our reconstruction results (c) are closer to the ground truth (d) and achieve more accurate details with far fewer outliers.

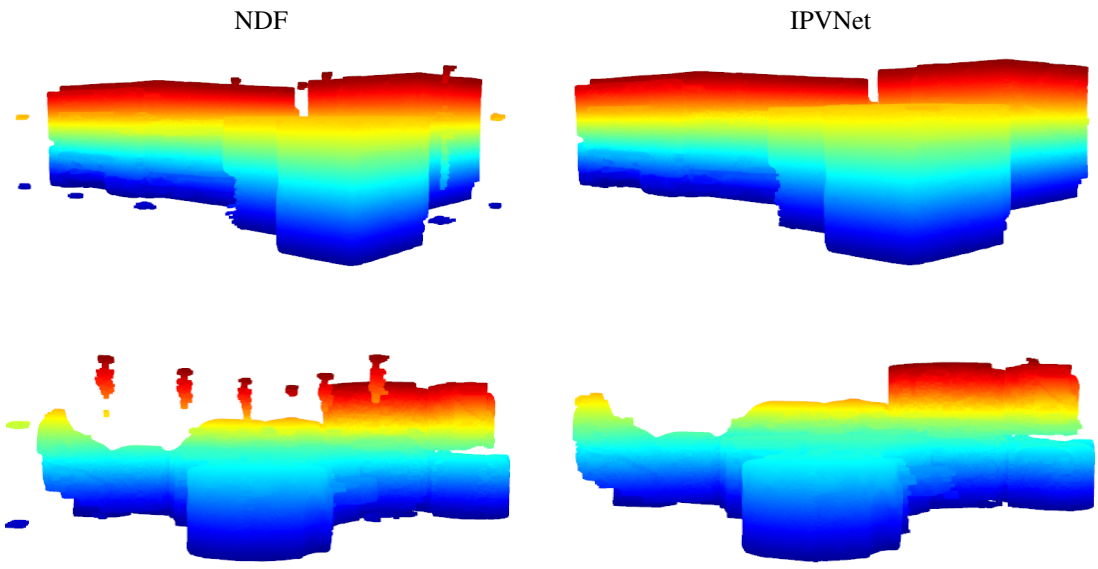


Figure 4.10: Reconstruction results after the NDF [3] baseline has been filtered using the input coordinate range as the distance threshold, and IPVNet without any filtering. NDF still includes outliers due to the surface curvature whereas the IPVNet reconstruction consists of *significantly less* outliers without any filtering.

CHAPTER 5

LIST: Learning Implicitly from Spatial Transformers for Single-View 3D Reconstruction

5.1 Introduction

Constructing a truthful portrayal of the 3D world from a single 2D image is a basic problem for many applications including robot manipulation and navigation, scene understanding, view synthesis, virtual reality, and more. Following the work of Erwin Kruppa [119] in camera motion estimation and the recovery of 3D points, researchers have attempted to solve the 3D reconstruction issue using structure from motion [120, 121, 122], and visual simultaneous localization and mapping [123, 124]. However, the main limitation of such approaches is that they require multiple observations of the desired object or scene from distinct viewpoints with shared features. Such a multi-view formulation allows for integrating information from numerous images to compensate for occluded geometry.

Reconstructing a 3D object from a single image is a more difficult task since a sole image does not contain the whole topology of the target shape due to self-occlusions. Researchers have tried both explicit and implicit techniques to reconstruct a target object with self-occluded parts. Explicit methods attempt to infer the target shape directly from the input image. Nevertheless, a major drawback of such approaches is that the output resolution needs to be defined in advance, which constrains these techniques from achieving high-quality results. Recent advances in implicit learning offer a solution to reconstruct the target shape in an arbitrary resolution by indirectly inferring the desired surface through a distance/occupancy field. Then, the target surface is reconstructed by extracting a zero level set from the distance/occupancy field.

Implicit 3D reconstruction from a single view is an active area of research where one faction of techniques [75, 77] encode global image features into a latent representation and learn an im-

Figure 5.1: Five unique views of objects reconstructed by LIST from a single RGB image. Not only does our model accurately recover occluded geometry, but also the reconstructed surfaces are *not influenced* by the input-view direction.

plicit function to reconstruct the target. Yet, these approaches can be easily outperformed by simple retrieval baselines [11]. Therefore, global features alone are not sufficient for a faithful reconstruction. Another faction leverages both local and global features to learn the target implicit field from pixel-aligned query points. However, such methods rely on ground-truth/estimated camera parameters for training/inference [125, 126], or they assume weak perspective projection [127, 128].

To address these shortcomings we propose LIST, a novel deep learning framework that can reliably reconstruct the topological and geometric structure of a 3D object from a single RGB image. Our method *does not depend on weak perspective projection, nor does it require any*

camera parameters during training or inference. Moreover, we leverage both local and global image features to generate highly-accurate topological and geometric details. To recover self-occluded geometry and aid the implicit learning process, we first predict a coarse shape of the target object from the global image features. Then, we utilize the local image features and the predicted coarse shape to learn a signed distance function (SDF).

Due to the scarcity of real-world 2D-3D pairs, we train our model on synthetic data. However, we use both synthetic and-real world images to test the reconstruction ability of LIST. Through qualitative analysis we highlight our model's *superiority in reconstructing high-fidelity geometric and topological structure*. Via a quantitative analysis using traditional evaluation metrics, *we show that the reconstruction quality of LIST surpasses existing works*. Furthermore, *we design a new metric to investigate the reconstruction quality of self-occluded geometry*. Finally, we provide an ablation study to validate the design choices of LIST in achieving high-quality single-view 3D reconstruction. Our source code is publicly available to the research community [129].

The remainder of this chapter is organized as follows. We give a summary of related research in Section 5.2. The details of our 3D reconstruction model are described in Section 5.3. Our experimental evaluation, a discussion of the results, and an ablation study are presented in Section 5.4. We conclude in Section 5.5.

5.2 Related Work

In this section we summarize pertinent work on the reconstruction of 3D objects from a single RGB image via implicit learning. Interested readers are encouraged to consult [130] for a comprehensive survey on 3D reconstruction from 2D images. Contrary to explicit representations, implicit ones allow for the recovery of the target shape at an arbitrary resolution. This benefit has attracted interest among researchers to develop novel implicit techniques for different applications. Dai *et al.* [131] used a voxel-based implicit representation for shape completion. DeepSDF,

introduced by Park *et al.* [80], is an auto-decoder that learns to estimate signed distance fields. However, DeepSDF requires test-time optimization, which limits its efficiency and capability.

To further improve 3D object reconstruction quality, Littwin and Wolf [79] utilized encoded image features as the network weights of a multilayer perceptron. Wu *et al.* [132] explored sequential part assembly by predicting the SDFs for structural parts separately and then combining them together. For self-supervised learning, Liu *et al.* [133] proposed a ray-based field probing technique to render the implicit surfaces as 2D silhouettes. Niemeyer *et al.* [134] used supervision from RGB, depth, and normal images to reconstruct rich geometry and texture. Chen and Zhang [77] proposed generative models for implicit representations and leveraged global image features for single-view reconstruction. For multiple 3D vision tasks, Mescheder *et al.* [75] developed OccNet, a network that learns to predict the probability of a volumetric grid cell being occupied.

Pixel-aligned approaches [127, 135, 128, 136] have employed local query feature extraction from image pixels to improve 3D human reconstruction. Xu *et al.* [125] incorporated similar ideas for 3D object reconstruction. To enhance the reconstruction quality of surface details, Li and Zhang [126] utilized normal images and a Laplacian loss in addition to aligned features. Zhao *et al.* [96] exploited coarse prediction and unsigned distance fields to reconstruct garments from a single view. Duggal and Pathak [137] proposed category specific reconstruction by learning a topology aware deformation field. Mittal *et al.* introduced AutoSDF [138], a model that encodes local shape regions separately via patch-wise encoding. However, these prior works rely on weak perspective projection and the rendering of metadata to align query points to image pixels. In contrast, LIST does not require any alignment or rendering data, and it recovers more accurate topological structure and geometric details.

5.3 Implicit Function Learning from Unaligned Pixel Features

Given a single RGB image of an object, our goal is to reconstruct the object in 3D with highly-accurate topological structure and self-occluded geometry. We model the target shape as

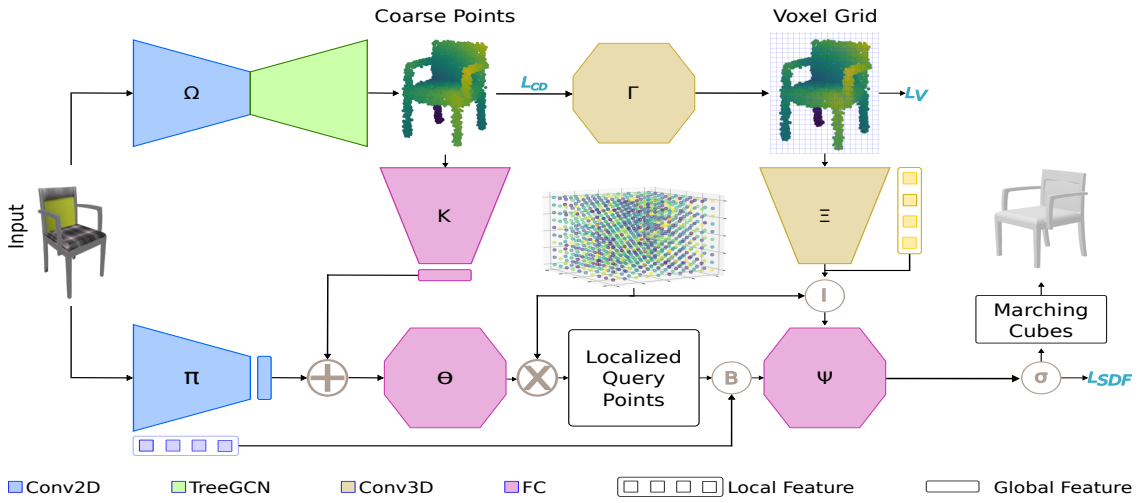


Figure 5.2: To reconstruct the target object from a single RGB image, LIST first predicts the coarse topology from the global image features. Simultaneously, local image features are used to extract local geometry at the given query locations. Finally, an SDF predictor (Ψ) estimates the signed distance field (σ) to reconstruct the target shape. Note that images and colors are for visualization purposes only.

an SDF and extract the underlying surface from the zero level set of the SDF during inference. To train our model we employ an image and query point pair (x_i, Q_i), where Q_i is a set of 3D coordinates (query points) in close vicinity to the surface of the object with a measured signed distance and x_i is a rendering of the object from a random viewpoint. An overview of the our framework is presented in Figure 5.2. The details of each component are provided in the following subsections.

5.3.1 Query Features From Coarse Predictions

Consider an RGB image $x_i \subset X \in \mathbb{R}^{H \times W \times 3}$ of height H and width W. We propose a convolutional neural encoder-decoder Ω_ω, parameterized by weights ω, to extract latent features from the image and predict a coarse estimation $\dot{y}_i^{x_i}$ of the target object. Concretely,

$$\Omega_\omega(x_i) \coloneqq \dot{y}_i^{x_i} \mid \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{N \times 3}, \tag{5.1}$$

where $\dot{y}_i^{x_i}$ is a point cloud representation of the target and N is the resolution of the point cloud. Note that the subscript i indicates i-th sample and the superscript x_i designates the source variable. For high-performance point cloud generation, we utilize tree structured graph convolutions (TreeGCN) [9] to decode the image features.

We use the coarse prediction $\dot{y}_i$ as a guideline for the topological structure of the target shape in a canonical space. To extract query features from this coarse prediction, first we discretize the point cloud in an occupancy grid $\dot{u}_i^{\dot{y}_i} \in 1^{M \times M \times M}$ of resolution M. However, the coarse prediction may contain gaps and noisy points that may impair the reconstruction quality. To resolve this, we employ a shallow convolutional network $\Gamma_{\ddot{o}}$ parameterized by weights $\ddot{o}$ to generate a probabilistic occupancy grid from $\dot{u}_i^{\dot{y}_i}$,

$$\dot{v}_i^{\dot{u}_i} \coloneqq \Gamma_{\ddot{o}}(\dot{u}_i^{\dot{y}_i}) \colon 1^{M \times M \times M} \to [0,1]^{M \times M \times M}. \tag{5.2}$$

Specifically, our aim is to find the neighboring points of $\dot{y}_i$ with a high chance of being a surface point of the target shape.

Although it is possible to regress the voxel representation directly from the global image features [139, 140, 128], learning a high-resolution voxel occupancy prediction requires a *significant* amount of computational resources [128]. Moreover, we empirically found that point cloud prediction followed by voxel discretization achieves better accuracy on diverse shapes rather than predicting the voxels directly.

Next, a neural network Ξ_ξ, parameterized by weights ξ, maps the probabilistic occupancy grid (5.2) to a high-dimensional latent matrix through convolutional operations. Then, our multi-scale trilinear interpolation scheme I extracts relevant query features f_C at each query location q_i from the mapped features. More formally,

$$f_C \coloneqq I(\Xi_\xi(\dot{v}_i^{\dot{u}_i}), Q_i). \tag{5.3}$$

In addition to q_i, we also consider the neighboring points at a distance d from q_i along the Cartesian axes to capture rich 3D features, i.e.,

$$q_j = q_j + k \cdot \hat{n}_j \cdot d, \tag{5.4}$$

where $k \in \{1, 0, -1\}$, $j \in \{1, 2, 3\}$, and $\hat{n}_j \in \mathbb{R}^3$ is the j-th Cartesian axis unit vector.

5.3.2 Localized Query Features

The coarse prediction and query features f_C can aid the recovery of the topological structure of the target shape. Nevertheless, relevant local features are also required to recover fine geometric details. To achieve this, prior arts assume weak perspective projection [127, 128] or align the query points to the image pixel locations through the ground-truth/estimated camera parameters [125, 126]. Predicting the camera parameters is analogous to predicting the object pose from a single image, which is itself a hard problem in computer vision. It involves a high chance of error and a computationally expensive training procedure. Furthermore, the error in the pose/camera estimation may lead to the loss of geometric details in the reconstruction.

To overcome these limitations, we obtain insight from spatial transformers [116] and leverage the spatial relationship between the input image and the coarse prediction. Via the coarse prediction, which portrays an object from a standard viewpoint and the query points that delineate the coarse predictions, it is possible to localize the query points to the local image features. This is done by predicting a spatial transformation with the aid of global features from the input image and the coarse prediction as follows.

First, we define a convolutional neural encoder Π_π, parameterized by weights π, to encode the input image into local $(l_\pi^{x_i})$ and global $(z_\pi^{x_i})$ features. Concretely,

$$\Pi_\pi(x_i) \coloneqq \{l_\pi^{x_i}, z_\pi^{x_i}\}. \tag{5.5}$$

Concurrently, a neural module K_κ encodes the coarse prediction $\dot{y}_i^{x_i}$ into global point features. Using global features from both the image and the coarse prediction, the spatial transformer Θ estimates a transformation to localize the query points in the image feature space. Then, localized query points $\tilde{Q}_i$ are generated by applying the predicted transformation to Q_i,

$$\Theta_\theta(z_\pi^{x_i}, K_\kappa(\dot{y}_i^{x_i}), Q_i) \coloneqq \tilde{Q}_i \mid \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times 2}. \tag{5.6}$$

Finally, a bi-linear interpolation scheme $\mathcal{B}$ extracts the local query features f_L from the local image features $l_\pi^{x_i}$,

$$f_L \coloneqq \mathcal{B}(l_\pi^{x_i}, \tilde{Q}_i). \tag{5.7}$$

Note that the point encoder K_κ and the localization network Θ are designated to ensure an accurate SDF prediction. Therefore, we do not use any camera parameters during training and we optimize these neural modules directly with the SDF prediction objective. This has the following benefits:

(i) *additional modules or training to predict the projection matrix and object pose from a single image are not required;*

(ii) *reconstructions are free from any pose estimation error, which boosts reconstruction accuracy*.

5.3.3 Signed Distance Function Prediction

To estimate the final signed distance Δ_i, we combine the coarse features f_C with the localized query features f_L and utilize a multilayer neural function defined as

$$\Psi_\psi(f_C, f_L) \coloneqq \begin{cases} \mathbb{R}^-, & \text{if } q_i \text{ is inside the target surface} \\ \mathbb{R}^+, & \text{otherwise.} \end{cases} \tag{5.8}$$

5.3.4 Loss Functions

We incorporate the chamfer distance (CD) loss and optimize the weights ω to accurately estimate the coarse shape of the target. More specifically,

$$\mathcal{L}_{\text{CD}}(y_i, \dot{y}_i) = \sum_{a \in \dot{y}_i} \min_{b \in y_i} ||a - b||^2 + \sum_{b \in y_i} \min_{b \in \dot{y}_i} ||b - a||^2, \tag{5.9}$$

where $y_i \in \mathbb{R}^{N \times 3}$ is a set of 3D coordinates collected from the surface of the object and $\dot{y}_i \in \mathbb{R}^{N \times 3}$ is the estimated coarse shape. To supervise the probabilistic occupancy grid prediction, we discretize y_i to generate the ground-truth occupancy $v_i^{y_i} \in 1^{M \times M \times M}$. The neural weight $\ddot{o}$ is then optimized by the binary cross-entropy loss,

$$\mathcal{L}_V(v_i, \dot{v}_i) = -\frac{1}{|v_i|} \Sigma(\gamma v_i \log \dot{v}_i + (1 - \gamma)(1 - v_i) \log(1 - \dot{v}_i)), \tag{5.10}$$

where γ is a hyperparameter to control the influence of the occupied/non-occupied grid points. To optimize the SDF prediction, we collect a set of query points Q_i within distance δ of the target surface and measure their signed distance σ_i. The estimated signed distance is then guided by optimizing the neural weights ξ, π, θ, and ψ through

$$\mathcal{L}_{\text{SDF}} = \frac{1}{|Q_i|} \Sigma(\sigma_i - \Delta_i)^2. \tag{5.11}$$

5.3.5 Training Details

We incorporate a two-stage procedure to train LIST. In the first stage, we only focus on the coarse prediction from the input image x_i and optimize the weights ω through $\mathcal{L}_{\text{CD}}$. Then, we freeze ω after convergence to a minimum validation accuracy and start the second stage for the SDF prediction. During the second stage, we jointly optimize $\ddot{o}$, ξ, π, κ, θ, and ψ through the combined loss $\mathcal{L} = \mathcal{L}_V + \mathcal{L}_{\text{SDF}}$. LIST can also be trained end-to-end by jointly minimizing $\mathcal{L}_{CD}$

with $\mathcal{L}_V$ and $\mathcal{L}_{\text{SDF}}$. However, we found the two-stage training procedure easier to evaluate and quicker to converge during experimental evaluation. To reconstruct an object at test time, we first densely sample a fixed 3D grid of query points and predict the signed distance for each point. Then, we use the marching cubes [141] algorithm to extract the target surface from the grid.

5.4 Experimental Evaluation

In this section, we describe the details of our experimental setup and results.

5.4.1 Implementation Overview

LIST was implemented using the PyTorch [142] library. To optimize the model, the Adam [67] optimizer was used with coefficients $(0.9, 0.99)$, learning rate 10^{-4}, and weight decay 10^{-5}. A pretrained ResNet [143] was employed as the image encoder in Ω and Π. We closely followed the generator in [9] to implement the coarse predictor in Ω with tree-structured convolutions. However, we empirically found that the degree values $(2, 2, 2, 2, 2, 2, 64)$ provided a better coarse estimation in our settings. We set the coarse point cloud density to $N = 4000$, and the occupancy grid resolution to $M = 128$. To generate a probabilistic occupancy with the same grid, we utilized a shallow convolutional network Γ.

We define Ξ as a convolutional neural network to map the probabilistic occupancy grid into a high-dimensional latent space. To extract the global query features and localize the query points, we used a fully-connected neural network Θ. The global image features are fused with the global query features on the 3rd layer of Θ. During training, we augment the images with random color jitter, and normalize the values to $[0, 1]$. To improve the estimation accuracy, we scale the ground-truth and predicted SDF values by 10.0. Following [77], we disentangled the query points by scaling with 2.0 and swapping the 1st and 3rd axis to extract query features from the coarse prediction. At test time, we extract the query points from a grid in the range $[-0.5, 0.5]$ with resolution 128^3.

5.4.2 Training and Inference Time

To train LIST it takes ≈ 1 second to make a forward pass on an Intel i7 machine with an NVIDIA GeForce GTX 1080Ti GPU. To fully pass through the Pix3D and ShapeNet datasets, it takes approximately 35 and 50 minutes, respectively. Our training process involved using 4 1080Ti GPUs for 100 epochs with a batch size of 8. To reconstruct the mesh of a single object from a corresponding RGB image, it takes $\approx$ 7s on average at a grid resolution of 128^3.

5.4.3 Datasets

Similar to [126] and [138], we utilized the 13-class subset of the ShapeNet [4] dataset to train LIST. The renderings and processed meshes from [125] were used as the input view and target shape. We trained a single model on all 13 categories. Additionally, we employed the Pix3D [7] dataset to test LIST on real-world scenarios. The train/test split from [144] was used to evaluate on all 9 categories of Pix3D. Following [144], we preprocessed the Pix3D target shapes to be watertight for training.

To prepare the ground truth, first the target shape was normalized into a unit cube and 50k points were sampled from the surface of the object. The query points were prepared by adding random Gaussian noise (n) to the surface points. Specifically,

$$Q_j = Q_S + n \mid n \in \mathcal{N}(0, P), \tag{5.12}$$

where Q_S are the sampled points and $P \in \mathbb{R}^{3\times3}$ is a diagonal covariance matrix with entries $P_{i,i} = \rho$. We empirically found that 45% of the points at $\rho = 0.003$, 44% of the points at $\rho = 0.01$, and 10% of the points at $\rho = 0.07$ achieved the best results. To supervise the coarse prediction and probabilistic occupancy grid estimation, we sub-sampled 4k points from the surface via farthest point sampling.

5.4.4 Baseline Models

For single-view reconstruction via synthetic images, we compared against the following prior arts: IMNET [77], and D^{2}IM-Net [126]. IMNET does not require pose estimation. However, the reconstruction only unitizes global features from an image. D^{2}IM-Net extracts local features by aligning the query points to image pixels through rendering metadata and it uses a pose estimation module during inference.

For single-view reconstruction from real-world images, we evaluated against TMN [145], MGN [146], and IM3D [144]. TMN deforms a template mesh to reconstruct the target object. MGN and IM3D perform reconstruction through the following steps: (i) identify objects in a scene, (ii) estimate their poses, and (iii) reconstruct each object separately.

5.4.5 Metrics

We computed commonly used metrics (e.g., CD, intersection over union (IoU), and F-score), to evaluate the performance of LIST. To evaluate the reconstructions between LIST and the baselines we used $d = 1\%$.

These traditional metrics *do not* differentiate between visible/occluded surfaces since they evaluate the reconstruction as a whole. To investigate the reconstruction quality of occluded surfaces, we propose to isolate visible/occluded surfaces based on the viewpoint of the camera and evaluate them separately using the traditional metrics. A visual depiction of this new strategy is presented in Figure 5.3.

To measure the reconstruction quality of occluded surfaces, we first align the predicted/ground-truth meshes to their projection in the input image using the rendering metadata. Then, we assume the camera location as a single source of light and cast rays onto the mesh surface by ray casting [147]. Next, we identify the visible/occluded faces through the ray-mesh intersection and subdivide the identified faces to separate them. Note that the rendering metadata is only used to evaluate

Figure 5.3: To evaluate the reconstruction quality of occluded surfaces, we first align the reconstructed shape (b) with the input image (a) and cast rays onto the surface (c). Next, we identify the (red) faces that intersect with the rays via ray-mesh intersection and separate the reconstructed mesh into (d) visible and (e) occluded areas.

the predictions. Finally, we sample 100k points from the separated occluded faces to compute the CD_{os}, and voxelize the sampled points to compute the IoU_{os} and $\text{F-Score}_{\text{os}}$.

In our implementation, we set the canvas resolution to 4096×4096 pixels and generated one ray per pixel from the camera location. It is important to note that ray casting and computing ray-mesh intersections are computationally demanding tasks. Therefore, to manage time and resources, we chose five sub-classes (chair, car, plane, sofa, table) to evaluate occluded surface reconstruction.

5.4.6 Single-View 3D Reconstruction Evaluation

		plane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	Mean
CD↓	IMNET	18.95	17.34	15.17	10.86	14.72	16.77	83.64	33.41	10.33	13.35	19.32	9.16	15.24	21.40
	D^2IM-Net	13.25	**12.51**	9.47	7.83	11.31	15.33	**34.08**	17.62	8.55	12.34	14.26	8.11	15.73	13.87
	LIST	**12.13**	13.49	**7.45**	**1.04**	**9.20**	**13.65**	47.31	**16.75**	**7.32**	**9.92**	**11.14**	**7.91**	**15.78**	**13.31**
IoU↑	IMNET	39.43	44.65	49.25	55.75	51.22	53.34	29.26	50.66	46.43	51.12	41.63	52.79	49.61	47.31
	D^2IM-Net	45.44	48.45	48.60	53.58	53.13	52.72	**32.45**	51.75	50.76	53.35	45.17	53.06	52.89	49.33
	LIST	**49.03**	47.57	**56.29**	**65.57**	**52.70**	**57.34**	24.80	**55.34**	**52.42**	**56.79**	**47.90**	**58.98**	**54.35**	**52.23**
F-score↑	IMNET	48.87	31.78	**44.34**	48.78	41.45	48.32	21.23	48.29	52.92	44.12	45.21	51.52	52.31	44.54
	D^2IM-Net	51.37	**36.76**	43.49	51.77	45.56	50.82	**29.57**	51.93	56.25	48.34	47.23	54.84	52.73	47.74
	LIST	**52.46**	36.39	42.51	**53.12**	**46.62**	**51.78**	22.88	**52.67**	**58.24**	**50.52**	**49.62**	**56.89**	**53.58**	**48.25**

Table 5.1: Quantitative results using the ShapeNet [4] dataset for various models. The metrics reported are the following: chamfer distance (CD), intersection over union (IoU), and F-score. The CD values are scaled by 10^{-3}.

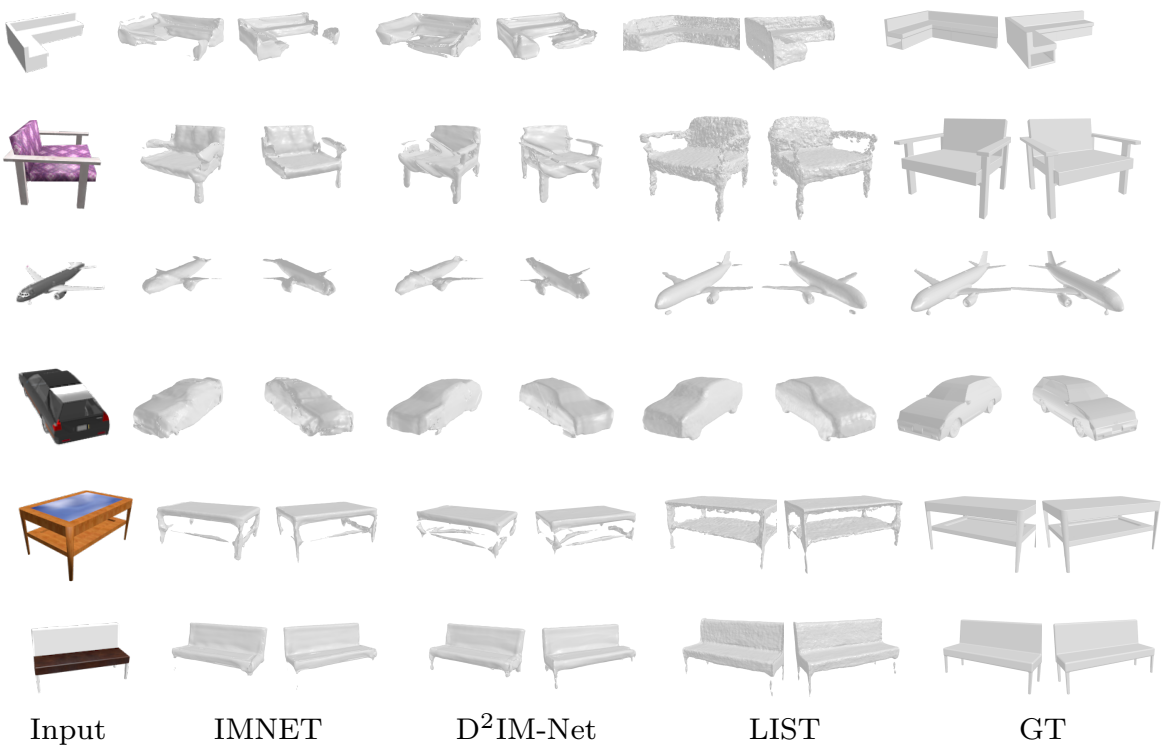


Figure 5.4: A qualitative comparison between LIST and the baseline models using the ShapeNet [4] dataset. Our model recovers *significantly better* topological and geometric structure, and the reconstruction is not tainted by the input-view direction. GT denotes the ground-truth objects.

5.4.6.1 Single-View 3D Reconstruction from Renderings of Synthetic Objects

In this experiment we performed single-view 3D reconstruction on the test set of the ShapeNet dataset. The qualitative and quantitative results are displayed in Figure. 5.4 and Table 5.1, respectively. In comparison to the baselines, the topological structure and occluded geometry recovered by LIST are considerably better. For example, in row 3 all of the baselines struggle to reconstruct the tail of the airplane and they fail to estimate the full length of the wings. In row 5, none of the baselines were able to recover the occluded part of the table. In contrast, LIST not only recovers the structure, but it also maintains the gap in between. Moreover, notice that in row 2 D^{2}IM-Net fails to resolve the directional view ambiguity and imprints an arm shaped silhouette on the seat

		plane	car	chair	sofa	table	Mean
CD_{os} ↓	IMNET	24.11	13.34	15.47	24.34	26.86	20.82
	D^2IM-Net	26.23	13.44	13.59	20.45	23.45	19.43
	LIST	**18.93**	**6.57**	**12.66**	**18.44**	**21.76**	**15.67**
IoU_{os} ↑	IMNET	45.63	46.87	38.32	45.87	39.02	43.14
	D^2IM-Net	48.44	50.33	49.43	50.32	42.22	48.14
	LIST	**53.15**	**55.37**	**51.25**	**55.22**	**43.17**	**51.63**
F_{os}-score↑	IMNET	40.93	46.94	44.43	46.84	45.64	44.95
	D^2IM-Net	47.21	50.73	48.89	49.15	47.72	48.73
	LIST	**50.33**	**52.55**	**49.34**	**51.02**	**48.11**	**50.27**

Table 5.2: A quantitative evaluation of the occluded surfaces of reconstructed synthetic objects via our evaluation strategy. The metrics reported are the following: chamfer distance (CD_{os}), intersection over union (IoU_{os}), and F_{os}-score. The CD_{os} values are scaled by 10^{-3}.

rather than reconstructing the arm. This indicates a strong influence of the input-view direction in the reconstructed surface. Conversely, LIST can resolve view-directional ambiguity and provide a reconstruction that is uninfluenced by the input-view direction. As shown in Table 5.1, LIST outperforms all the other baseline models. Additional qualitative comparison between LIST and the baselines can be found in Figure 5.9. Moreover, we have compared LIST against the baselines on reconstructing from distinct views of the same object. The qualitative results are displayed in Figure 5.10, 5.11, and 5.12.

We also evaluated LIST against the baselines on occluded surface recovery by partitioning the reconstructions using our proposed metric. The results are recorded in Table 5.2 and Figure 5.6 respectively. LIST outperformed all the baselines hence showcasing the superiority of our approach in reconstructing occluded geometry. Furthermore, LIST provides a stable reconstruction across different views of the same object as shown in Figure 5.5. However, the use of ground-truth rendering data instead of the estimated data improved the reconstruction quality. This indicates the source of the problem to be the sub-optimal prediction of the camera pose. Nonetheless, LIST is free from any such complication as our framework does not require any explicit pose estimation.

		bed	bookcase	chair	desk	sofa	table	tool	wardrobe	misc	Mean
CD↓	TMN	7.78	5.93	6.86	7.08	4.25	17.42	4.13	4.09	23.68	9.03
	MGN	5.99	6.56	**5.32**	5.93	3.36	14.19	3.12	3.83	26.93	8.36
	IM3D	**4.11**	3.96	5.45	7.85	5.61	11.73	2.39	4.31	24.65	6.72
	LIST	5.81	**1.74**	6.11	**3.87**	**2.08**	**1.68**	**1.99**	**0.80**	**5.16**	**4.36**
IoU↑	LIST	45.61	39.54	41.15	59.68	67.34	49.12	27.82	43.87	34.72	46.77
F-score↑	LIST	58.18	67.22	60.01	78.34	70.14	69.19	46.48	75.70	39.14	65.66

Table 5.3: A quantitative evaluation of the occluded surfaces of reconstructed objects via our evaluation strategy. The metrics reported are the following: chamfer distance (CD_{os}), intersection over union (IoU_{os}), and F_{os}-score. The CD_{os} values are scaled by 10^{-3}

	Base	OL	1E	2D	EC	Final
CD↓	11.35	9.64	10.72	8.48	7.89	**7.32**
IoU↑	51.34	53.95	51.40	55.23	55.10	**56.83**
F-score↑	43.11	48.06	45.92	51.37	51.33	**52.75**

Table 5.4: Quantitative results obtained from the ablation study using different network settings.

5.4.6.2 Single-View 3D Reconstruction from Real Images

In this experiment we evaluated single-view 3D reconstruction on the test set of the Pix3D dataset. The qualitative and quantitative results are provided in Figure 5.7 and Table 5.3, respectively. The baseline results were obtained from the respective papers. Compared to other methods our approach generates the most precise 3D shapes, which results in the lowest average CD and F-score. Notice that in Figure 5.7, rows 3 and 4, only LIST can accurately recover the back and legs of the chair. Additionally, LIST reconstructions provide a smooth surface, precise topology, and fine geometric details.

5.4.7 Ablation Study

5.4.7.1 Setup

To investigate the impact of each individual component in our single-view 3D reconstruction model, we performed an ablation study with the following network options.

- *Base:* A version of LIST that predicts the signed distance utilizing only global image features and coarse predictions.
- *OL:* An improved *Base* version that uses the probabilistic occupancy from the coarse prediction and occupancy loss.
- *1E:* A version of LIST where local and global image features from the same encoder are used for both coarse prediction and localized query feature extraction.
- *2D:* LIST with two separate decoders to estimate the signed distance from local and global query features. The final prediction is obtained by adding both estimations.
- *EC:* We train LIST without the localization module and use a separate pose estimation module similar to [126] to predict the camera parameters. The estimated camera parameters were used to transform the query points during inference.

To maximize limited computational resources, we focused on the most diverse five sub-classes (chair, car, plane, sofa, table) of the ShapeNet dataset for this ablation study. The qualitative and quantitative results of the experiments are recorded in Figure 5.8 and Table 5.4 respectively.

5.4.7.2 Discussion

In the ablation experiments the *Base* version was able to recover global topology, but it lacked local geometry. As shown in Fig 5.8, the probabilistic occupancy and optimization loss helped recover some details in the *OL* version. Conversely, the performance decreased slightly after the inclusion of local details in the single-encoder version (*1E*). We hypothesize that the task of query point localization, while estimating the coarse prediction, overloads the encoder and hinders meaningful feature extraction for the signed distance prediction. To overcome this issue, we used a separate encoder for the coarse prediction and query point localization. The dual-decoder version (*2D*), performed similar to the final model. Nonetheless, we found that the geometric details had a thicker reconstruction than the target during qualitative evaluation. This motivated the fusion of features rather than predictions in the final version.

We also ablated the localization module using estimated camera parameters during training and inference. As shown in Table 5.4, the final version of LIST outscores the version employing estimated camera (*EC*) parameters. This indicates that our localization module with an SDF prediction objective is more suitable for single-view reconstruction compared to a camera pose estimation sub-module. More importantly, this removes the requirement for pixel-wise alignment through camera parameters for local feature extraction. Note that the *EC* reconstruction appears qualitatively similar to the others and was therefore omitted in Figure 5.8.

5.4.8 Limitations and Future Directions

Although LIST achieves state-of-the-art performance on single-view 3D reconstruction, there are some limitations. For example, the model may struggle with very small structures. We speculate that this is due to the coarse predictor failing to provide a good estimation of such structures. Please see the 5.13 for examples of failed reconstruction results. Another shortcoming is the need for a clear image background. LIST can reconstruct targets from real-world images, yet it requires an uncluttered background to do this. In the future, we will work towards resolving these issues.

5.5 Conclusion

In this chapter, we introduced LIST, a network that implicitly learns how to reconstruct a 3D object from a single image. Our approach does not assume weak perspective projection, nor does it require pose estimation or rendering data. We achieved state-of-the-art performance on single-view reconstruction from renderings of synthetic objects. Furthermore, we demonstrated domain transferability of our model by recovering 3D surfaces from images of real-world objects. We believe our approach could be beneficial for other problems such as object pose estimation and novel view synthesis.

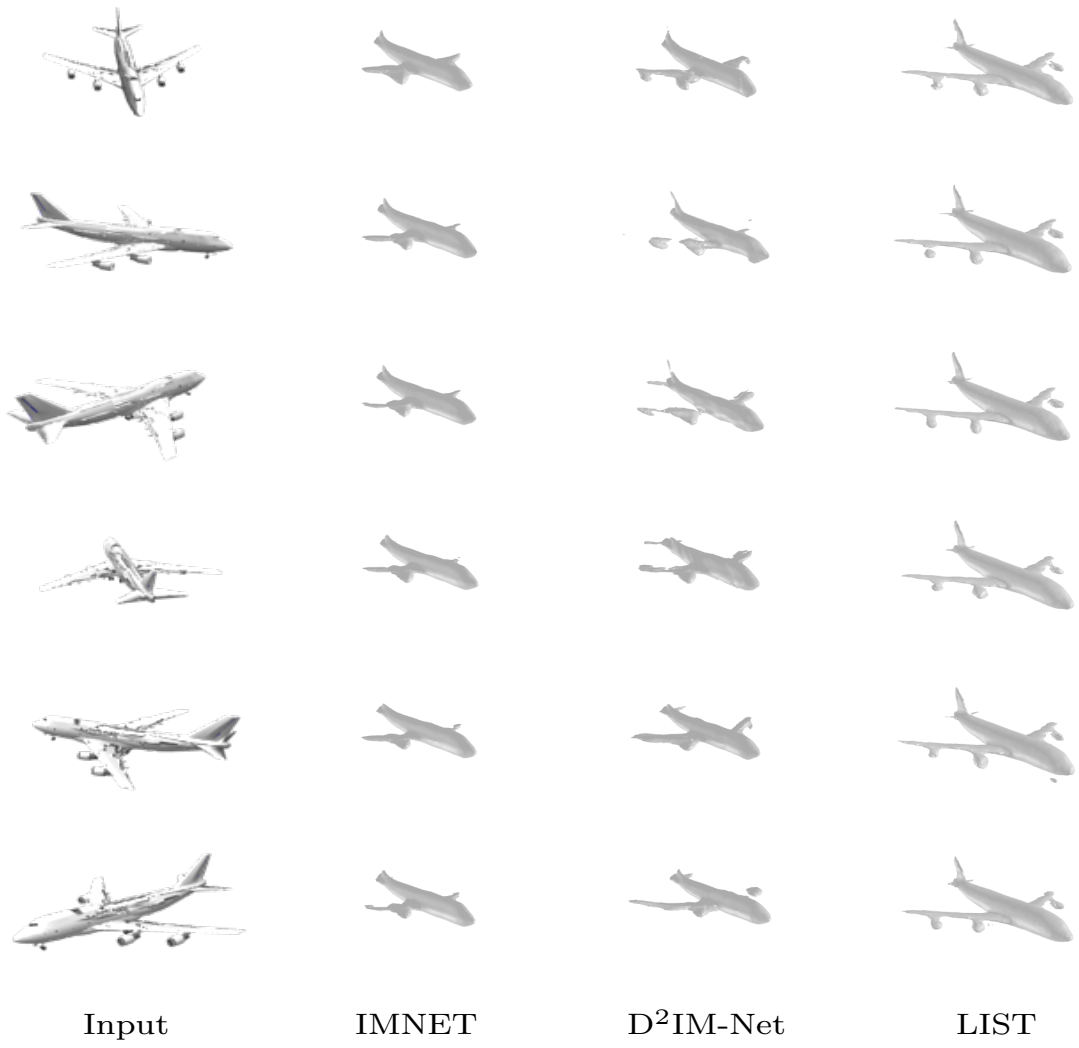


Figure 5.5: A qualitative comparison between LIST and the baseline models using distinct views of the same object. Not only can our model both maintain better topological structure and geometric details, but it also provides a reconstruction that is stable across different views of the object.

Figure 5.6: A qualitative comparison between LIST and the baseline models on occluded surface reconstruction using the ShapeNet dataset. GT denotes the ground-truth objects.

Figure 5.7: Single-view reconstruction using real-world images from the Pix3D [7] test set (best viewed zoomed in).

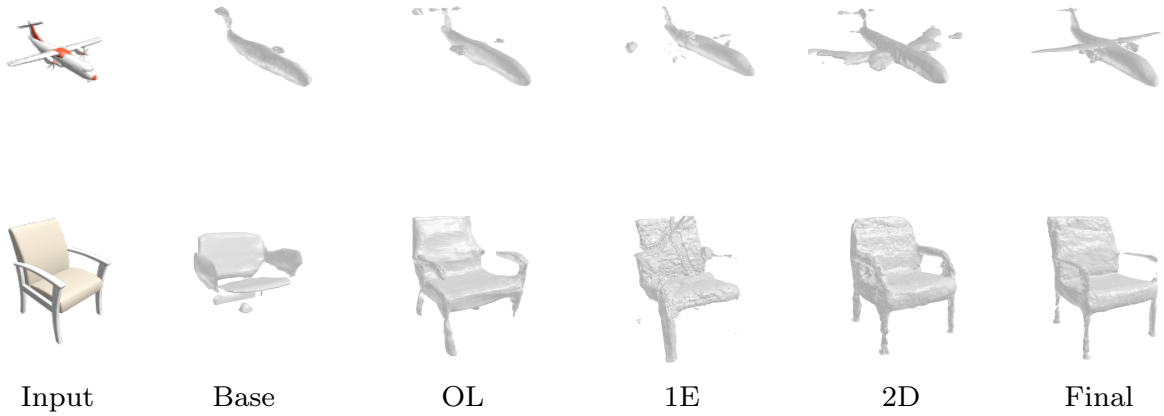


Figure 5.8: Qualitative results obtained from the ablation study using different network settings.

Figure 5.9: A qualitative comparison between LIST and the baseline models using the ShapeNet dataset. Our model recovers *significantly better* topological and geometric structure, and the reconstruction is not tainted by the input-view direction. GT denotes the ground-truth objects.

Figure 5.10: Qualitative results of LIST reconstructions using distinct views of the same object. Odd rows represent the input and even rows represent the reconstructions.

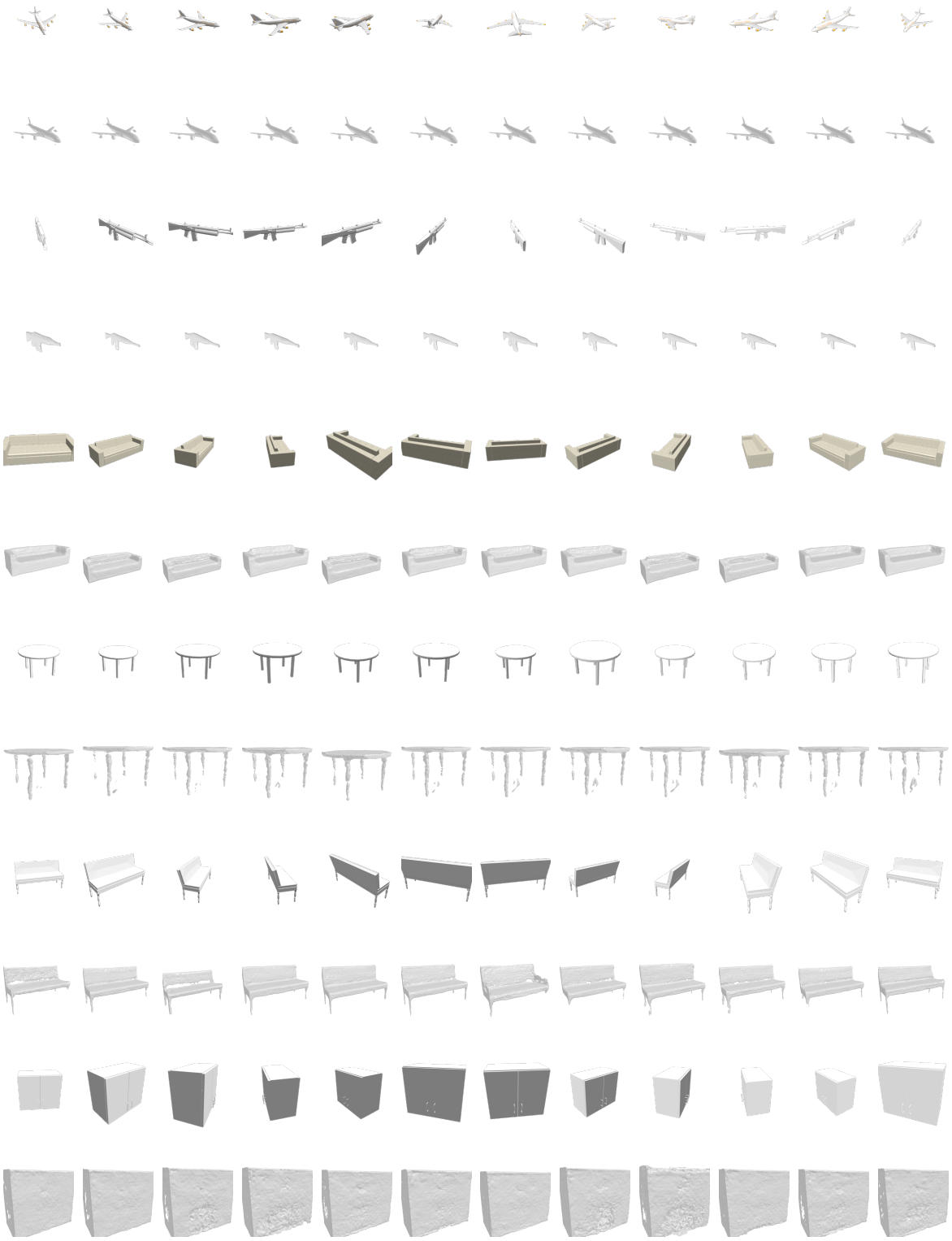


Figure 5.11: Qualitative results of LIST reconstructions using distinct views of the same object. Odd rows represent the input and even rows represent the reconstructions.

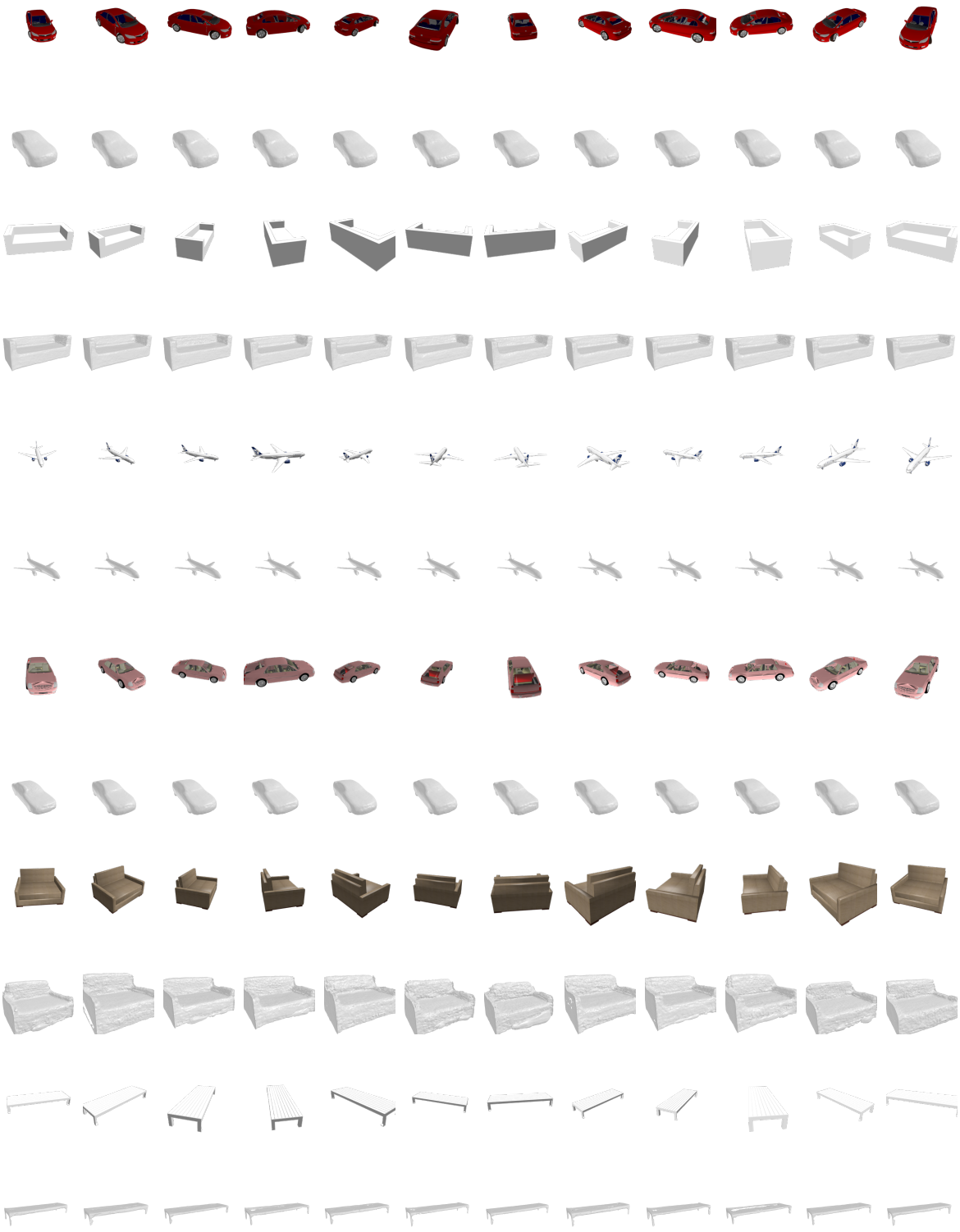


Figure 5.12: Qualitative results of LIST reconstructions using distinct views of the same object. Odd rows represent the input and even rows represent the reconstructions.

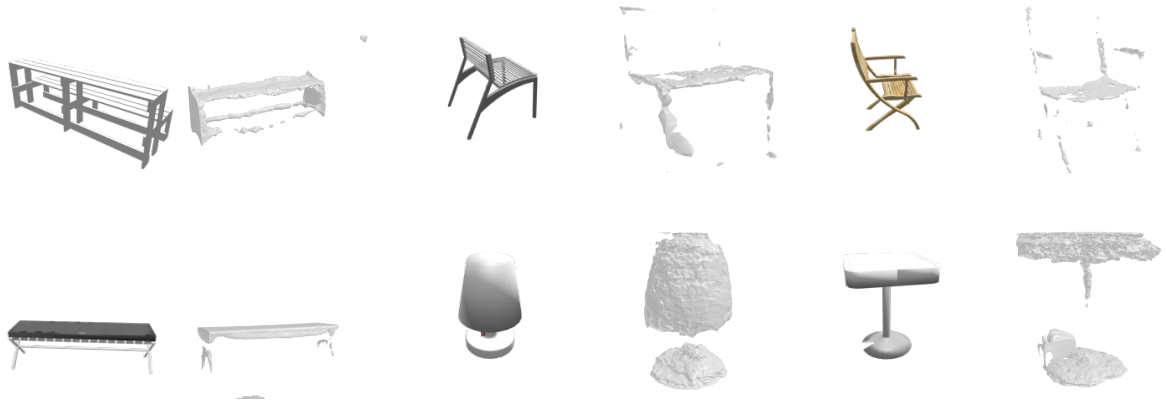


Figure 5.13: Examples of failed LIST reconstructions.

CHAPTER 6

Conclusion

In summary, this research represents a significant step forward in the field of raw 3D point cloud generation and reconstruction by introducing novel contributions that address various challenges. The first major advancement is the introduction of PCGAN, a progressive conditional generative adversarial network designed to generate dense and colored 3D point clouds for different object classes without the need for supervision. The key innovation of PCGAN lies in its point transformer, which allows the network to progressively grow in resolution using graph convolutions. This approach enables the capture of intricate details at high resolutions, resulting in colored point clouds with fine details at multiple scales. The experimental results demonstrate the network's capability to learn and replicate a 3D data distribution effectively.

The second contribution is IPVNet, a learning-based implicit point-voxel model specifically tailored for reconstructing 3D open surfaces, such as non-watertight meshes. Previous techniques struggle with surface gaps, leading to artifacts such as the artificial closing of gaps. IPVNet overcomes this challenge by predicting the unsigned distance between a surface and a query point in 3D space, leveraging both raw point cloud data and its discretized voxel representation. This method outperforms state-of-the-art techniques, delivering more accurate reconstructions while significantly reducing the number of outliers in the results. The utilization of both raw data and voxel representation allows for a more precise and robust reconstruction process.

The third major contribution of this research is the introduction of LIST, a novel neural architecture aimed at accurately reconstructing both the geometric and topological details of 3D objects from a single 2D image. Existing explicit and implicit methods grapple with self-occluded geometry and fail to provide faithful topological shape reconstruction. LIST resolves this challenge

by leveraging both global and local image features. It predicts a coarse shape of the target object using global 2D features and then refines the reconstruction at higher resolutions by incorporating local 2D features and 3D features from the coarse prediction. One key advantage of LIST is that it does not require camera estimation or pixel alignment, thus providing uninfluenced reconstructions from the input-view direction. The qualitative and quantitative analysis demonstrates the superiority of LIST compared to state-of-the-art approaches in reconstructing 3D objects from synthetic and real-world images.

Lastly, we provide open-source licenses for our source code and make them freely available to facilitate further research and development in this field. In conclusion, this research not only pushes the boundaries of 3D point cloud generation and reconstruction, but it also enriches the broader research community by providing valuable resources to fuel future exploration and innovation.

REFERENCES

[1] Realistic Graphics Lab, EPFL, "Mitsuba 2 renderer," 2020, http://www.mitsuba-renderer.org.

[2] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[3] J. Chibane, A. Mir, and G. Pons-Moll, "Neural unsigned distance fields for implicit function learning," *arXiv preprint arXiv:2010.13938*, 2020.

[4] A. X. Chang, T. Funkhouser, L. 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 preprint arXiv:1512.03012*, 2015.

[5] J. Ye, Y. Chen, N. Wang, and X. Wang, "Gifs: Neural implicit function for general shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 829–12 839.

[6] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll, "Multi-garment net: Learning to dress 3d people from images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5420–5430.

[7] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2974–2983.

[8] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 40–49.

[9] D. W. Shu, S. W. Park, and J. Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3859–3868.

[10] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3d point clouds via graph convolution," in *Proceedings of the International Conference on Learning Representations*, 2018.

[11] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3405–3414.

[12] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[13] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[14] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 1697–1702.

[15] R. C. Luo, V. W. Ee, and C.-K. Hsieh, "3d point cloud based indoor mobile robot in 6-dof pose localization using fast scene recognition and alignment approach," in *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2016, pp. 470–475.

[16] A. Pfrunder, P. V. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2601–2608.

[17] M. Whitty, S. Cossell, K. S. Dang, J. Guivant, and J. Katupitiya, "Autonomous navigation using a real-time 3d point cloud," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2010, pp. 1–3.

[18] Q. Zhu, L. Chen, Q. Li, M. Li, A. Nüchter, and J. Wang, "3d lidar point cloud based intersection recognition for autonomous driving," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 456–461.

[19] Y. Wang, S. Zhang, B. Wan, W. He, and X. Bai, "Point cloud and visual feature-based tracking method for an augmented reality-aided mechanical assembly system," *The International Journal of Advanced Manufacturing Technology*, vol. 99, no. 9-12, pp. 2341–2352, 2018.

[20] B. Liu, M. Guo, E. Chou, R. Mehra, S. Yeung, N. L. Downing, F. Salipur, J. Jopling, B. Campbell, K. Deru, W. Beninati, A. Milstein, and F.-F. Li, "3d point cloud-based visual prediction of icu mobility care activities," in *Proceedings of the Machine Learning for Healthcare Conference*, 2018, pp. 17–29.

[21] R. Bostelman, P. Russo, J. Albus, T. Hong, and R. Madhavan, "Applications of a 3d range camera towards healthcare mobility aids," in *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, 2006, pp. 416–421.

[22] S. T. Pöhlmann, E. F. Harkness, C. J. Taylor, and S. M. Astley, "Evaluation of kinect 3d sensor for healthcare imaging," *Journal of Medical and Biological Engineering*, vol. 36, no. 6, pp. 857–870, 2016.

[23] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 652–660.

[24] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions On Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[25] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

[26] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[27] Z. Cao, Q. Huang, and R. Karthik, "3d object classification via spherical projections," in *Proceedings of the International Conference on 3D Vision*. IEEE, 2017, pp. 566–574.

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[29] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2282–2290.

[30] J. Hou, A. Dai, and M. Nießner, "3d-sis: 3d semantic instance segmentation of rgb-d scans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4421–4430.

[31] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "Gspn: Generative shape proposal network for 3d instance segmentation in point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3947–3956.

[32] G. Sharma, E. Kalogerakis, and S. Maji, "Learning point embeddings from shape repositories for few-shot segmentation," in *Proceedings of the International Conference on 3D Vision*. IEEE, 2019, pp. 67–75.

[33] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[34] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum, "Learning shape priors for single-view 3d completion and reconstruction," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 646–662.

[35] Y. Wang, D. J. Tan, N. Navab, and F. Tombari, "Forknet: Multi-branch volumetric semantic completion from a single depth image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8608–8617.

[36] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1939–1948.

[37] M. Sarmad, H. J. Lee, and Y. M. Kim, "Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5898–5907.

[38] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *Proceedings of the International Conference on 3D Vision*. IEEE, 2018, pp. 728–737.

[39] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 383–392.

[40] X. Cao and K. Nagao, "Point cloud colorization based on densely annotated 3d shape dataset," in *Proceedings of the International Conference on Multimedia Modeling*. Springer, 2019, pp. 436–446.

[41] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4541–4550.

[42] A. Hertz, R. Hanocka, R. Giryes, and D. Cohen-Or, "Pointgmm: a neural gmm network for point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 054–12 063.

[43] S. Ramasinghe, S. Khan, N. Barnes, and S. Gould, "Spectral-gans for high-resolution 3d point-cloud generation," *arXiv preprint arXiv:1912.01800*, 2019.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[46] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[47] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[48] https://github.com/robotic-vision-lab/Progressive-Conditional-Generative-Adversarial-Network.

[49] S. Chaudhuri, D. Ritchie, K. Xu, and H. Zhang, "Learning generative models of 3d structures," *Eurographics Tutorial*, 2019.

[50] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "A survey on deep learning advances on different 3d data representations," *arXiv preprint arXiv:1808.01462*, 2018.

[51] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–38, 2017.

[52] B. Eckart, K. Kim, A. Troccoli, A. Kelly, and J. Kautz, "Accelerated generative models for 3d point cloud data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5497–5505.

[53] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 3391–3401.

[54] J. Li, B. M. Chen, and G. Hee Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9397–9406.

[55] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov, "Point cloud gan," *arXiv preprint arXiv:1810.05795*, 2018.

[56] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

[57] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, and L. J. Guibas, "Structurenet: Hierarchical graph networks for 3d shape generation," *arXiv preprint arXiv:1908.00575*, 2019.

[58] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3d point cloud processing," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 103–118.

[59] J. Xie, Y. Xu, Z. Zheng, S.-C. Zhu, and Y. Nian Wu, "Generative pointnet: Energy-based learning on unordered point sets for 3d generation, reconstruction and classification," *arXiv*, pp. arXiv–2004, 2020.

[60] Y. Sun, Y. Wang, Z. Liu, J. Siegel, and S. Sarma, "Pointgrow: Autoregressively learned point cloud generation with self-attention," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 61–70.

[61] X. Cao, W. Wang, K. Nagao, and R. Nakamura, "Psnet: A style transfer network for point cloud stylization on geometry and color," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 3337–3345.

[62] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[63] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[64] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[65] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[66] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[69] M. S. Arshad and W. J. Beksi, "A progressive conditional generative adversarial network for generating dense and colored 3d point clouds," in *Proceedings of the International Conference on 3D Vision*. IEEE, 2020, pp. 712–722.

[70] H. Son and Y. M. Kim, "Progressive growing of points with tree-structured generators," in *Proceedings of the British Machine Vision Conference*, 2021, pp. 1–13.

[71] M. Pepe, V. S. Alfio, D. Costantino, and D. Scaringi, "Data for 3d reconstruction and point cloud classification using machine learning in cultural heritage environment," *Data in Brief*, vol. 42, p. 108250, 2022.

[72] J. Kim, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Pointinverter: Point cloud reconstruction and editing via a generative model with shape priors," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 592–601.

[73] https://github.com/robotic-vision-lab/Implicit-Point-Voxel-Features-Network.

[74] C. C. You, S. P. Lim, S. C. Lim, J. San Tan, C. K. Lee, and Y. M. J. Khaw, "A survey on surface reconstruction techniques for structured and unstructured data," in *Proceedings of the IEEE Conference on Open Systems*, 2020, pp. 37–42.

[75] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[76] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 523–540.

[77] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.

[78] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Deep level sets: Implicit surface representations for 3d shape inference," *arXiv preprint arXiv:1901.06802*, 2019.

[79] G. Littwin and L. Wolf, "Deep meta functionals for shape representation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1824–1833.

[80] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[81] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866.

[82] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 7462–7473.

[83] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, "Combining implicit function learning and parametric models for 3d human reconstruction," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 311–329.

[84] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3d shape reconstruction and completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6970–6981.

[85] M. Atzmon and Y. Lipman, "Sal: Sign agnostic learning of shapes from raw data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2565–2574.

[86] ——, "Sald: Sign agnostic learning with derivatives," *arXiv preprint arXiv:2006.05400*, 2020.

[87] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.

[88] Z. Mi, Y. Luo, and W. Tao, "Ssrnet: scalable 3d surface reconstruction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 970–979.

[89] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein, "Metasdf: Meta-learning signed distance functions," *arXiv preprint arXiv:2006.09662*, 2020.

[90] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "Patchnets: Patch-based generalizable deep implicit 3d shape representations," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 293–309.

[91] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.

[92] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu, "Deep implicit moving least-squares functions for 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1788–1797.

[93] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, "Di-fusion: Online implicit 3d reconstruction with deep priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8932–8941.

[94] S. Duggal, Z. Wang, W.-C. Ma, S. Manivasagam, J. Liang, S. Wang, and R. Urtasun, "Secrets of 3d implicit object shape reconstruction in the wild," *arXiv preprint arXiv:2101.06860*, 2021.

[95] W. Zhao, J. Lei, Y. Wen, J. Zhang, and K. Jia, "Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 256–10 265.

[96] F. Zhao, W. Wang, S. Liao, and L. Shao, "Learning anchored unsigned distance functions with gradient direction alignment for single-view garment reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 674–12 683.

[97] R. Venkatesh, T. Karmali, S. Sharma, A. Ghosh, R. V. Babu, L. A. Jeni, and M. Singh, "Deep implicit surface point prediction networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 653–12 662.

[98] W. Chen, C. Lin, W. Li, and B. Yang, "3psdf: Three-pole signed distance function for learning surfaces with arbitrary topologies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 522–18 531.

[99] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Learning descriptor networks for 3d shape synthesis and analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8629–8638.

[100] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9204–9214.

[101] W. Huang, B. Lai, W. Xu, and Z. Tu, "3d volumetric modeling with introspective neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8481–8488.

[102] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," *arXiv preprint arXiv:1907.03739*, 2019.

[103] Y. Li, G. Tong, X. Li, L. Zhang, and H. Peng, "Mvf-cnn: Fusion of multilevel features for large-scale point cloud classification," *IEEE Access*, vol. 7, pp. 46 522–46 537, 2019.

[104] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[105] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection," *arXiv preprint arXiv:2102.00463*, 2021.

[106] Z. Cui and Z. Zhang, "Pvf-net: Point & voxel fusion 3d object detection framework for point cloud," in *Proceedings of the Conference on Computer and Robot Vision*. IEEE, 2020, pp. 125–133.

[107] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 685–702.

[108] J. Noh, S. Lee, and B. Ham, "Hvpr: Hybrid voxel-point representation for single-stage 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 605–14 614.

[109] C. Zhang, H. Wan, S. Liu, X. Shen, and Z. Wu, "Pvt: Point-voxel transformer for 3d deep learning," *arXiv preprint arXiv:2108.06076*, 2021.

[110] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "Pv-raft: point-voxel correlation fields for scene flow estimation of point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6954–6963.

[111] Y. Li, S. Yang, Y. Zheng, and H. Lu, "Improved point-voxel region convolutional neural network: 3d object detectors for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[112] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 024–16 033.

[113] K. Cherenkova, D. Aouada, and G. Gusev, "Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d," in *Proceedings of the IEEE International Conference on Image Processing*, 2020, pp. 2741–2745.

[114] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning*, 2010.

[115] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 448–456.

[116] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 28, 2015.

[117] F.-E. Wolter, "Cut locus and medial axis in global shape interrogation and representation, design laboratory memorandum 92-2," 1993.

[118] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[119] E. Kruppa, *Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung*. Hölder, 1913.

[120] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[121] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[122] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.

[123] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[124] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Computing Surveys*, vol. 51, no. 2, pp. 1–36, 2018.

[125] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32, 2019.

[126] M. Li and H. Zhang, "D2im-net: Learning detail disentangled implicit fields from single images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 246–10 255.

[127] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2304–2314.

[128] T. He, J. Collomosse, H. Jin, and S. Soatto, "Geo-pifu: Geometry and pixel aligned implicit functions for single-view human reconstruction," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9276–9287.

[129] https://github.com/robotic-vision-lab/Learning-Implicitly-From-Spatial-Transformers-Network.

[130] K. Fu, J. Peng, Q. He, and H. Zhang, "Single image 3d object reconstruction based on deep learning: A review," *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 463–498, 2021.

[131] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[132] R. Wu, Y. Zhuang, K. Xu, H. Zhang, and B. Chen, "Pq-net: A generative part seq2seq network for 3d shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 829–838.

[133] S. Liu, S. Saito, W. Chen, and H. Li, "Learning to infer implicit surfaces without 3d supervision," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32, 2019.

[134] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3504–3515.

[135] S. Saito, T. Simon, J. Saragih, and H. Joo, "Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 84–93.

[136] Y. Cao, G. Chen, K. Han, W. Yang, and K.-Y. K. Wong, "Jiff: Jointly-aligned implicit face function for high quality single view clothed human reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2729–2739.

[137] S. Duggal and D. Pathak, "Topologically-aware deformation fields for single-view 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1536–1546.

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

[139] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 628–644.

[140] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2437–2446.

[141] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM Siggraph Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[142] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 8024–8035.

[143] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[144] C. Zhang, Z. Cui, Y. Zhang, B. Zeng, M. Pollefeys, and S. Liu, "Holistic 3d scene understanding from a single image with implicit representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8833–8842.

[145] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia, "Deep mesh reconstruction from single rgb images via topology modification networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9964–9973.

[146] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang, "Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 55–64.

[147] S. D. Roth, "Ray casting for modeling solids," *Computer Graphics and Image Processing*, vol. 18, no. 2, pp. 109–144, 1982.

BIOGRAPHICAL STATEMENT

Mohammad Samiul Arshad was born in Barisal, Bangladesh, in 1991. He received his B.S. degree in 2014 from Sylhet Engineering College, affiliated with Shahjalal University of Science and Technology, Sylhet, Bangladesh. He earned his Ph.D. degree from The University of Texas at Arlington, 2023, in Computer Science and Engineering. From 2014 to 2016, he worked as a senior software engineer with TwinBit Ltd. During his Ph.D. studies, he worked in the Robotic Vision Laboratory on 3D point cloud generation and reconstruction. His current research interests are in the area of 3D vision and artificial intelligence. He is a member of several IEEE societies.