Point2Real: Bridging the Gap between Point Cloud and Realistic Image for Open-World 3D Recognition

Hanxuan Li[1,2], **Bin Fu**[1,2], **Ruiping Wang**[1,2], **Xilin Chen**[1,2]
[1] Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences(CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{lihanxuan21s, fubin21b, wangruiping, xlchen}@ict.ac.cn

Abstract

Recognition in open-world scenarios is an important and challenging field, where Vision-Language Pre-training paradigms have greatly impacted the 2D domain. This inspires a growing interest in introducing 2D pre-trained models, such as CLIP, into the 3D domain to enhance the ability of point cloud understanding. Considering the difference between discrete 3D point clouds and real-world 2D images, reducing the domain gap is crucial. Some recent works project point clouds onto a 2D plane to enable 3D zero-shot capabilities without training. However, this simplistic approach leads to an unclear or even distorted geometric structure, limiting the potential of 2D pre-trained models in 3D. To address the domain gap, we propose Point2Real, a training-free framework based on the realistic rendering technique to automate the transformation of the 3D point cloud domain into the Vision-Language domain. Specifically, Point2Real leverages a shape recovery module that devises an iterative ball-pivoting algorithm to convert point clouds into meshes, narrowing the gap in shape at first. To simulate photo-realistic images, a set of refined textures as candidates is applied for rendering, where the CLIP confidence is utilized to select the suitable one. Moreover, to tackle the viewpoint challenge, a heuristic multi-view adapter is implemented for feature aggregation, which exploits the depth surface as an effective indicator of view-specific discriminability for recognition. We conduct experiments on ModelNet10, ModelNet40, and ScanObjectNN datasets, and the results demonstrate that Point2Real outperforms other approaches in zero-shot and few-shot tasks by a large margin.

1 Introduction

In the field of 3D point cloud understanding, the open-world setting has been receiving increasing attention(Ha and Song 2023; Zhang, Dong, and Ma 2023). Open-world scenarios are filled with categories unseen in the datasets, which poses huge challenges for the traditional method trained on close-set. Similar open-world problems also exist in the 2D domain, and these problems have been significantly mitigated with the development of Vision-Language Pre-training. Methods like CLIP (Radford et al. 2021) align visual and textual modalities to equip the model with strong zero-shot transfer capabilities, which sparks the motivation

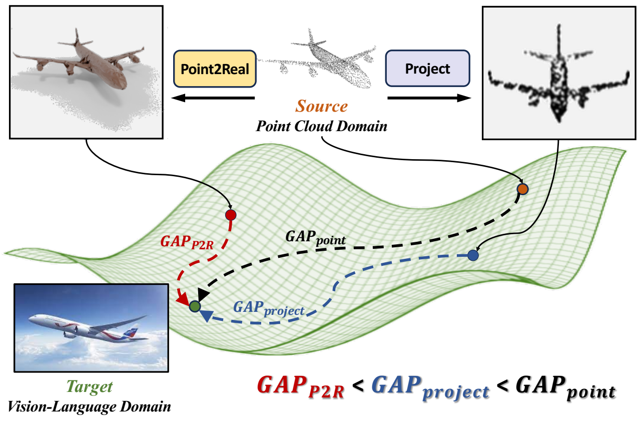

Figure 1: *The Effect of Point2Real in Reducing Domain Gap.* Assuming different types of visual data, such as the 3D Point Cloud domain and the 2D Vision-Language domain, as points in a domain manifold, the distance of the shortest geodesic path connecting them represents the domain gap. Converting point clouds into realistic rendering images, Point2Real can reduce this gap to GAP_{P2R}, which is better than projection.

to transfer similar paradigms to 3D. However, compared to the billions of data used to train 2D Vision-Language models, 3D annotated data is extremely scarce currently. Hence, to realize open-world 3D recognition, introducing pre-trained 2D models into the 3D domain is more practical than training a 3D large-scale model from scratch.

Point cloud data is unordered, sparse, and discrete, which is different from the grid-based natural images used in 2D pre-training. There is a huge domain gap between the point cloud domain and the 2D Vision-Language domain, which hinders the direct transfer from 2D pre-trained models to the 3D domain. Some efforts have been made to reduce the domain gap through the depth maps of point clouds. Pioneer work (Zhang et al. 2022) projects point clouds into depth maps to obtain CLIP-recognizable images. Calculating the similarity with depth maps and CLIP-encoded 3D category texts, this framework can achieve zero-shot transfer capability without training on 3D datasets. The following works (Zhu et al. 2023; Huang et al. 2023) aim to refine the gen-

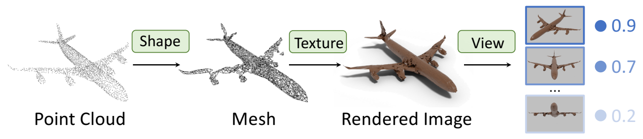

Figure 2: *Three Steps of Point2Real.* Point2Real consists of three modules, which progressively transform point clouds into realistically rendered images from the aspects of shape, texture, and viewpoint.

erator and encoder of the depth map to address the domain gap between depth maps and 2D images. However, different from natural images, the depth maps are characterized by blurry edges and contain only single-channel color information, limiting the performance of CLIP-based recognition. As shown in Figure 1, different from prior works focusing merely on the processing of depth maps, our work aims to more explicitly tackle the domain gap between point clouds and natural images, and explore another new direction to map point clouds to photo-realistic images, unleashing CLIP's full potential in the 3D domain.

We observe that the domain gap between point clouds and natural images primarily exists in three aspects: **1) Shape**. Point cloud data is discrete, vastly different from the continuous and smooth geometric structures commonly found in photo-realistic images. **2) Texture**. As the 2D pre-train model is highly sensitive to texture information, merely recovering shape without texture is insufficient. Basic point cloud data only contains spatial coordinates, lacking the crucial texture information. **3) View**. 3D point cloud has an additional dimension compared to 2D images, requiring the identification of suitable and discriminative view angles for generating images. With such observations in mind, we introduce **Point2Real**, an automatic framework for selecting suitable viewpoints to render point clouds with clear geometric structures and realistic textures.

As shown in Figure 2, Point2Real consists of three steps. First, to refine the shape of the point cloud, we propose **Shape Recovery Module** to automatically transform discrete point clouds into meshes, which approximate the real world. To ensure the framework's zero-shot capability and avoid overfitting to training data, we employ a graphics-based framework that includes iterative ball pivoting reconstruction, holes repairing, and surface smoothing. This method strengthens the geometric information of the point cloud, endowing it with smooth surfaces and distinct edges.

Second, as the reconstructed meshes still lack texture information, Point2real includes **Texture Rendering Module**, which can add texture information to meshes and perform ray tracing-based rendering, thereby obtaining rendered images of the object from various viewpoints. Considering that different categories may require various textures, we set a texture candidate list. For each object, we use all the candidate textures for rendering and the classification logits for texture selection.

Finally, among the multi-view rendered images, to select more discriminative viewpoints for recognition, we propose **Multi-view Adapter**. Intuitively, viewpoints that expose more geometric features will be recognized more easily. Building on this insight, we utilize the depth surface as an indicator to represent the exposed geometry from a given viewpoint and design a heuristic algorithm based on the surface area to select viewpoints that sufficiently expose geometric features automatically.

To quantitatively validate the zero-shot and few-shot capabilities of Point2Real, we conduct experiments on the ModelNet10 (Wu et al. 2015), ModelNet40 (Wu et al. 2015), and ScanObjectNN (Uy et al. 2019) datasets, commonly used for object classification tasks. Point2Real is shown to significantly outperform other approaches, achieving state-of-the-art results in both zero-shot and few-shot settings.

2 Related Work

2.1 Zero-shot Learning in 3D

The demand for recognizing unknown classes naturally exists in the 3D domain. Cheraghian, Rahman, and Petersson first extended zero-shot framework to 3D. Follow-up works (Cheraghian et al. 2019, 2022, 2020) make efforts to alleviate the Hubness problem (Radovanović, Nanopoulos, and Ivanović 2010) and extend the framework to more zero-shot settings. Recently, with the impressive zero-shot capabilities demonstrated by Visual-Language Models in the 2D domain, some efforts have been made to introduce CLIP into 3D for open-world recognition. Some works (Xue et al. 2023; Qi et al. 2023; Huang et al. 2023) aim to obtain a CLIP-aligned encoder by training on 3D datasets. In contrast, PointCLIP (Zhang et al. 2022), which is among the earliest works to bring CLIP into 3D, achieves zero-shot capability without training. PointCLIP achieves this by projecting 3D point clouds into 2D depth maps to obtain CLIP-recognizable images. Furthermore, PointCLIP V2 (Zhu et al. 2023) proposes an LLM-based Prompting structure to replace the simple prompt and a Realistic Shape Projection to replace the Sparse Visual Projection. Similar to these methods, Point2Real also introduces CLIP to obtain open-world recognition capability. Furthermore, for an input point cloud, Point2Real can output rendered images with texture information and smooth geometry. This endows Point2Real with enhanced zero-shot capability.

2.2 Transfer 3D Point Cloud to 2D Image

Compared to the 3D domain, the 2D domain provides much more abundant neural networks and a larger amount of data. Hence, in contrast to point-based models (Ma et al. 2022; Qi et al. 2017a,b; Wang et al. 2019; Wu, Qi, and Fuxin 2019). some works (Roveri et al. 2018; Ahmed, Liang, and Chew 2019; Feng et al. 2018; Guo et al. 2016) transform 3D point clouds into 2D images for shape classification. Therein, PointCLIP (Zhang et al. 2022) and SimpleView (Goyal et al. 2021) conduct projection using a viewpoint transformation. PointCLIP V2 (Zhu et al. 2023) applies a four-step procedure including voxelize, density, smooth, and squeeze to realize a denser distribution of points on the 2D plane. Most of the aforementioned approaches are based on projection techniques and attempt to improve the quality of

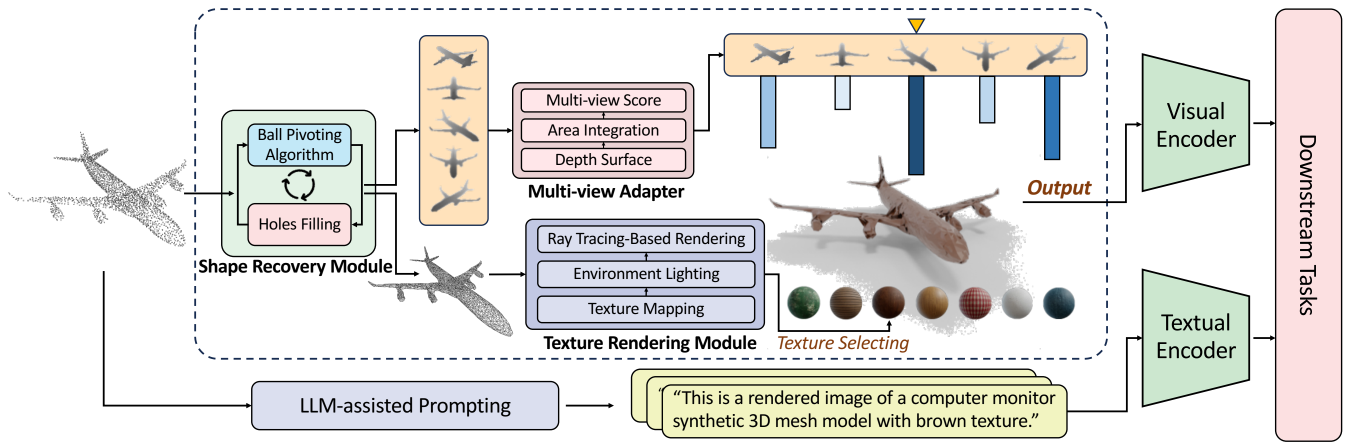

Figure 3: *The pipeline of Point2Real.* For visual encoding, we apply Shape Recovery Module, Texture Rendering Module, and Multi-view Adapter to generate realistic images of point clouds. For textural encoding, we use LLM-assisted prompting to generate class-specific prompts. Point2Real can also be used in few-shot tasks with a dual-path network introduced in Sec.3.3.

depth maps. However, the results still differ a lot from photo-realistic images. Motivated by this, we take a novel approach by directly converting discrete point clouds into meshes with clear geometry and further enhancing their realism in terms of texture and viewpoint.

2.3 Mesh Reconstruction from Point Cloud

Extracting meshes from point clouds is a fundamental task. As a classic and effective method, the Ball-Pivoting algorithm (Bernardini et al. 1999) iteratively rotates a sphere with a fixed radius on the point cloud and connects neighboring points to construct triangle patches. Another classical method is the Poisson Surface Reconstruction(PSR) algorithm (Kazhdan, Bolitho, and Hoppe 2006; Kazhdan and Hoppe 2013), which utilizes the principles of PSR and performs volume-based surface reconstruction based on the signed distance function of the input point cloud. There are also some later methods (Peng et al. 2021; Sellán and Jacobson 2022) improved based on PSR. Apart from the above graphics-based methods, numerous data-driven methods (Yu et al. 2018b,a; Groueix et al. 2018; Guerrero et al. 2018; Williams et al. 2019; Vakalopoulou et al. 2018) have emerged in recent years. these methods would employ neural networks to learn prior from 3D datasets. In the open-world scenario, there is a need for generalization beyond the training set. Therefore, we opt for graphics-based methods to avoid overfitting. Specifically, in this work, due to the high reliance of PSR-type methods on point cloud density, which is challenging to guarantee in various applications, our Shape Recovery Module is based on the Ball-Pivoting algorithm.

3 Methodology

In this section, we will introduce how Point2Real converts point clouds into realistic rendered images and how we select the better viewpoints. In the end, we will show how Point2Real works in 3D open-world tasks. The whole pipeline of Point2Real is illustrated in Figure 3.

3.1 Convert Point Cloud into Realistic Image

A two-step procedure is adopted to convert point clouds into realistic rendered images. First, Shape Recovery Module will transfer point clouds into meshes. Second, Texture Rendering Module can select a suitable texture to render images from several viewpoints.

Shape Recovery Module Point clouds are discrete, resulting in a scattered plot when projected onto a 2D plane. This representation suffers a substantial domain gap with realistic images that possess continuous and smooth geometric structures. This severely limits the recognition potential of CLIP in 3D. To improve the recognizability of geometric structures in point cloud data, we opt for the reconstruction of the input point cloud into a mesh structure. Compared to other 3D representations such as voxels, meshes can more flexibly and efficiently represent the appearance of objects, allowing for rendering results that closely approximate reality.

Numerous techniques are available for mesh reconstruction. To avoid the risk of overfitting the training data and ensure the framework's capability for zero-shot scenarios, the Ball-Pivoting algorithm (Bernardini et al. 1999) is employed for mesh reconstruction. Building upon the traditional Ball-Pivoting algorithm, we introduce an iterative variant to enhance the reconstruction quality. The whole process involves normal estimation, iterative reconstruction, holes filling, and smoothing, ultimately resulting in the conversion of the input point cloud into a mesh.

Specifically, for the point cloud denoted as $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{N}$, we can estimate the normal as $\mathcal{N} = \{\mathbf{n}_i\}_{i=1}^{N}$ with PCA method, where $\mathbf{n}_i$ represents the smallest eigenvalue of the covariance matrix $\mathbf{C}_i$ computed from the local neighborhood $\mathcal{V}_i$ of point $\mathbf{p}_i$:

$$\mathbf{C}_i = \frac{1}{k} \sum_{\mathbf{p}_j \in \mathcal{V}_i} (\mathbf{p}_j - \bar{\mathbf{p}}_i)(\mathbf{p}_j - \bar{\mathbf{p}}_i)^T. \tag{1}$$

The Ball-Pivoting algorithm, denoted as $f_{BPA}(\cdot)$, iteratively pivots balls over the point cloud surface, connecting adjacent

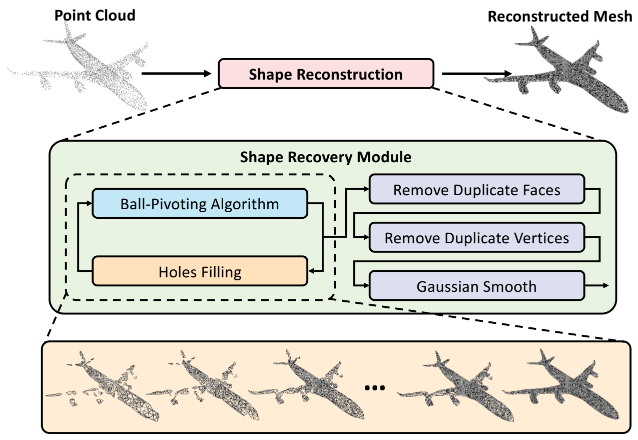

Figure 4: *The Pipeline of Shape Reconstruction.* Applying the iterative Ball-Pivoting algorithm, it is evident that the quality of the mesh improves as the number of iterations increases.

points to form triangles set

$$\mathcal{F} = f_{BPA}(\mathcal{P}, \mathcal{N}, \rho), \tag{2}$$

and the meshes can be obtained by

$$\mathcal{M} = (\mathcal{P}, \mathcal{F}), \tag{3}$$

where ρ represents a specified radius of the ball. Considering point clouds are sparse in many scenarios, the traditional Ball-Pivoting algorithm may leave many holes on the surface. By applying the Ball-Pivoting algorithm with different ρ, we can obtain a triangle set $\mathcal{F}'$ with a different grain level and generate a better 3D surface

$$\mathcal{M}' = (\mathcal{P}, \mathcal{F} \cup \mathcal{F}'). \tag{4}$$

With multiple iterations of reconstruction, point clouds can be converted into meshes with smooth surfaces and clear geometric structures, as depicted in Figure 4.

Texture Rendering Module The Texture Rendering Module employs a ray tracing-based rendering approach to achieve the generation of photo-realistic images from the reconstructed meshes. The meshes are placed at the center of the scene, illuminated by suitable environmental lighting. Following UV mapping and material shading, each mesh $\mathcal{M}$ is rendered from M different viewpoints.

Considering different categories may have varying texture requirements, the Texture Rendering Module maintains a texture candidates list $\{t_i\}_{i=1}^{T}$, encompassing textures such as brown, metallic green, and leather red. For each object, we utilize all textures to render from M viewpoints, resulting in the image set $\{I_{ij}\}_{i=1,j=1}^{(M,T)}$. Through the computation of classification logits introduced in Section 3.3, we obtain the most suitable texture used to classify each object.

3.2 Aggregate Features with Multi-view Adapter

As some works (Zhang et al. 2022; Zhu et al. 2023) noticed, aggregating features from multiple viewpoints can provide a more comprehensive understanding of 3D point clouds. However, not all viewpoints are suitable for classification. Some viewpoints can be challenging for recognition even for humans. For instance, the rear views of furniture may frequently appear as indistinguishable rectangles. When testing on the dataset, some approaches may pre-assign weights for viewpoints. However, this is unfeasible in practical applications.

To address this challenge, we propose the Multi-view Adapter, which can evaluate the recognizability of a group of viewpoints and generate weights for each view. The insight of this module is that with a uniform texture, viewpoints that emphasize more geometric features will lead to easier recognition. As shown in Figure 5, we design a heuristic algorithm based on the depth surface to automatically select viewpoints that sufficiently reveal geometric features. The depth surface represents the surface formed by the depth map in three-dimensional space, where the convex and concave regions correspond to areas closer to and farther from the camera, respectively. Intuitively, if the area of the depth surface is larger, indicating greater undulations, it implies the presence of numerous depth discontinuities in that viewpoint, signifying the complexity of the geometric features.

Specifically, for an object's depth map D_i in set $\{D_i\}_{i=1}^{M}$, its depth surface can be represented as:

$$S_i := \{\mathbf{z} - D_i(x, y) = 0\}, \tag{5}$$

where $\mathbf{z}(x, y)$ is a function of the image coordinates x and y, denoting the grey value of D_i at coordinates (x, y). Furthermore, the area of the depth surface S_i can be represented as follows:

$$\begin{aligned} A_i &= \iint_{S_i} dS_i = \iint_{D_i} \left\| \frac{\partial \mathbf{z}}{\partial x} \times \frac{\partial \mathbf{z}}{\partial y} \right\| dx\, dy \\ &= \iint_{D_i} \sqrt{\left(-\frac{\partial \mathbf{z}}{\partial x}\right)^2 + \left(-\frac{\partial \mathbf{z}}{\partial y}\right)^2 + 1}\, dx\, dy, \end{aligned} \tag{6}$$

where A_i is the area of the depth surface for the i-th viewpoint, $\frac{\partial \mathbf{z}}{\partial x}$ and $\frac{\partial \mathbf{z}}{\partial y}$ are the partial derivatives of $\mathbf{z}$ with respect to x and y, respectively. They represent how the depth value changes along the x and y directions, indicating the steepness of the depth surface. In the actual algorithm implementation, we approximate the partial derivatives as:

$$\frac{\partial \mathbf{z}}{\partial x} \approx \frac{\mathbf{z}(x + \Delta x, y) - \mathbf{z}(x, y)}{\Delta x}, \tag{7}$$

$$\frac{\partial \mathbf{z}}{\partial y} \approx \frac{\mathbf{z}(x, y + \Delta y) - \mathbf{z}(x, y)}{\Delta y}, \tag{8}$$

where Δx and Δy are the pixel sizes in the x and y directions, respectively.

By calculating the integral for every depth map in $\{D_i\}_{i=1}^{M}$, the areas of depth surfaces for M viewpoints can be denoted as $\{A_i\}_{i=1}^{M}$. With normalization, centralization, and activation, the final weight can be obtained as follows:

$$\omega_i = \sigma\left(\frac{A_i - \bar{A}}{\hat{A}}\right), \tag{9}$$

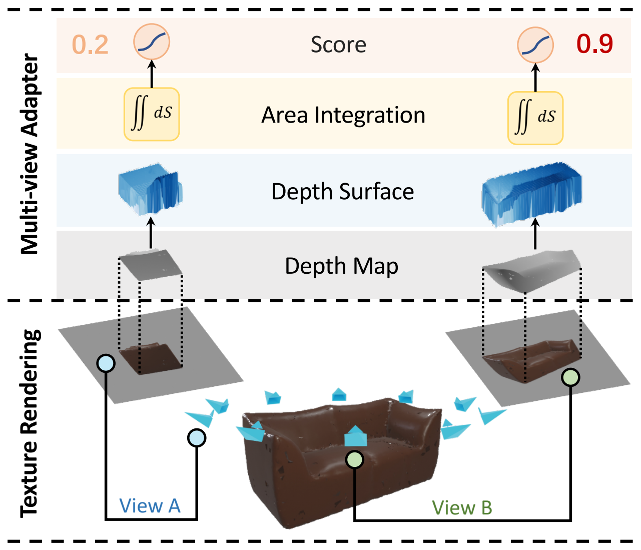

Figure 5: *Depth Surface.* Compared to View A, View B exhibits a clearer geometric structure. As a result, the depth surface of View B will appear significantly larger than that in View A. By calculating the area of the depth surface, View B will be assigned a higher weight.

where σ denotes the sigmoid function, $\bar{A}$ and $\hat{A}$ denotes the mean function and the standard deviation function. The resulting weights set $\{\omega_i\}_{i=1}^M$ represents the relative importance or recognizability of each viewpoint. Aggregating features from M viewpoints based on $\{\omega_i\}_{i=1}^M$ will contribute to zero-shot recognition in Point2Real framework under diverse real-world scenarios.

3.3 Point Cloud Understanding by Point2Real

Zero-shot classification Point2Real does not rely on 3D data and can be directly tested on existing datasets without any training. For the input point cloud $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^N$, to obtain a realistic shape with a smooth and flat surface, Shape Recovery Module converts $\mathcal{P}$ into meshes $\mathcal{M} = (\mathcal{P}, \mathcal{F})$, where $\mathcal{F}$ denotes the set of faces. To introduce appropriate texture information to the blank mesh $\mathcal{M}$, Texture Rendering Module renders images from M different viewpoints and T texture candidates $\{t_i\}_{i=1}^T$, resulting in the rendered image set $\{I_{ij}\}_{i=1,j=1}^{(M,T)}$. The rendered multi-view images are individually fed into the frozen CLIP visual encoder to obtain multi-view features denoted as $\{f_{ij}\}_{i=1,j=1}^{(M,T)}$. Meanwhile, Multi-view Adapter calculates view weights $\{\omega_i\}_{i=1}^M$ based on the depth surfaces of M viewpoints, which reflect their distinctiveness. For the textual branch, we employ LLM-assisted Prompting, which is proposed by PointCLIP V2 (Zhu et al. 2023), to generate class-specific prompts. With the CLIP textural encoder, the prompts of K classes can be encoded into C dimensions as a zero-shot classifier $W_{text} \in \mathbb{R}^{K\times C}$. For each texture t_j, the corresponding classification logits_j is computed individually:

$$\text{logits}_j = \sum_{i=1}^{M} \omega_i f_{ij} W_{text}^T, \text{ for } j = 1, \ldots, T, \tag{10}$$

the most suitable texture t_τ can be obtained by:

$$\tau = \underset{j=1\ldots,T}{\operatorname{argmax}} \max(\text{logits}_j), \tag{11}$$

and the classification logits of $\mathcal{P}$ can be obtained by:

$$\text{logits}_\mathcal{P} = \text{logits}_\tau . \tag{12}$$

The classification results can be derived from logits_P.

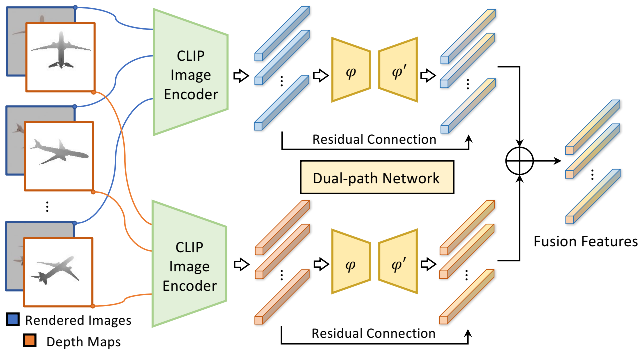

Figure 6: *Dual-path Network.* This architecture extends Point2Real to few-shot tasks. The upper branch of the network extracts features from rendered images, while the lower branch extracts features from depth maps. By fusing these two sets of features, the model gains enhanced learning capabilities.

Few-shot classification To further enhance the practical applicability of our framework, we expand Point2Real to the few-shot scenario, enabling the model to finetune the framework using a limited amount of 3D data. We adopt a dual-path network to simultaneously leverage shape and depth information. The dual-path network extracts features from both rendered images and depth maps, aggregating them to attain a holistic comprehension of point clouds. Considering the potential risk of overfitting in the presence of limited data, we opt for relatively simple network structures to implement the aforementioned modules like PointCLIP (Zhang et al. 2022).

Specifically, for the multi-view rendered images $\{I_i\}$ and their corresponding depth maps $\{D_i\}$, we utilize the CLIP visual encoder to separately extract object features $\{f_i^I\}$ and depth features $\{f_i^D\}$. Both sets of features are then combined using a dual-path three-layer Multi-layer Perceptron (MLP) with a residual connection, as shown in Figure 6. The classification results can be obtained by calculating the similarity between the fused features and textual features, similar to the zero-shot task.

4 Experiments

In this section, we will first present the implementation details of Point2Real. Subsequently, we will separately show the performance of our method under zero-shot and few-shot settings.

Method	3D Training-free	MN10	MN40	S-OBJ_ONLY	S-OBJ_BG	S-PB_T50_RS
PointCLIP (Zhang et al. 2022)	✓	30.2	23.8	21.3	19.3	15.4
Cheraghian (Cheraghian et al. 2022)	-	68.5	N/A	N/A	N/A	N/A
CLIP2Point (Huang et al. 2023)	-	66.6	49.4	35.5	30.5	23.3
ReCon (Qi et al. 2023)	-	75.6	61.7	43.7	40.4	30.5
ULIP (Xue et al. 2023)	-	N/A	60.4	N/A	N/A	**49.9**
PointCLIP V2 (Zhu et al. 2023)	✓	73.1	64.2	50.1	41.2	35.4
Point2Real (ours)	✓	**79.5**	**65.8**	**57.7**	**53.0**	**38.1**

Table 1: *Zero-shot 3D Classification (%) on ModelNet10, ModelNet40 and ScanObjectNN.* We report the zeeo-shot performance of Point2Real. "3D Training-free" refers to not requiring training on 3D datasets.

Shape	Texture	View	MN10	MN40
-	-	-	73.1	64.2
✓	-	-	75.9	60.3
✓	✓	-	78.4	65.3
✓	✓	✓	79.5	65.8

Table 2: *Ablation Study of Point2Real on ModelNet10 and ModelNet40 zero-shot classification (%).* We conduct the ablation study in three dimensions. For shape, texture, and view dimension, ✓ respectively represents Mesh Reconstruction, Realistic Rendering, and Multi-view Adapter. − indicates the above operations will be replaced with Depth Projection, Coloration with white, and Mean Function.

Reconstruction	Iteration	Zero-shot
Ball-Pivoting	1	78.6
Poisson	1	56.3
Iterative Ball Pivoting	5	79.5

Table 3: *Ablation Study of Shape Recovery Module on ModelNet40 zero-shot classification (%).* We convert the ablation study by replacing Shape Recovery Module with one-iteration Ball-Pivoting and Poisson Reconstruction.

4.1 Implementation Details

The entire framework can work end-to-end. For mesh reconstruction, we apply 5 iterations of the ball-pivoting algorithm with $\rho = 5, 4, 3, 2, 1$ for each iteration. The rendering procedure is implemented based on Blender Python API with Cycles engine (Blender Online Community 2023), and 10 viewpoints uniformly distributed around the object are selected for rendering. For rendering, the texture candidate list includes beige, metallic green, leather red, pink, brown, and dark grey, and the output image format is $224 * 224$. The kernel size of the Gaussian filter in the Multi-view Adapter is (7, 7, 5). Following PointCLIP V2's setup, we use 1024 points as input, and choose ViT-B/16 as our visual encoder. More details are listed in the supplementary material. The supplementary material and code are publicly available.[1]

4.2 Zero-shot Classification

Settings To evaluate the recognition capability of Point2Real for unknown categories, we conduct zero-shot classification experiments on three commonly used 3D datasets: ModelNet10, ModelNet40, and ScanObjectNN. ModelNet10 and ModelNet40 contain 10 and 40 classes of common synthetic objects, respectively, while ScanObjectNN includes 15 classes of real-world objects. We make use of three splits of the ScanObjectNN dataset: OBJ_ONLY, OBJ_BG, and PB_T50_RS. Under a more challenging setting, similar to PointCLIP and PointCLIP V2, Point2Real can directly test its performance on the entire dataset without training. Following PointCLIP V2's settings, we use the test sets from all the aforementioned datasets for evaluation and adopt LLM-assisted Prompting without View Weighing. We compare our method with existing related works under their respective best settings. Specifically, for CLIP2Point and PointCLIP V2, we use ViT-B/16 as the backbone. For PointCLIP, we use ResNet101, ResNet-50×4, and ViT-B/16, corresponding to ModelNet10, ModelNet40, and ScanObjectNN, respectively, to achieve its best performance.

[1]https://github.com/HanXuan-Li/Point2Real

Performance As shown in Table 1, we compare our method with existing approaches that demonstrate zero-shot results. Some of these rely on training on 3D datasets. Cheraghian divides the ModelNet10 dataset into seen and unseen parts and conducts training on the seen classes before testing on the unseen classes. CLIP2Point and ULIP focus on using 3D data to train a point cloud encoder aligned with the CLIP. In contrast, PointCLIP, PointCLIP V2, and our method can directly test on 3D datasets without training in 3D, which would bring more challenges. Among all mentioned datasets, we achieve state-of-the-art results, compared with all training-free methods and most training methods. On the ModelNet10 dataset, we achieve 79.4% accuracy, surpassing ReCon by +3.9% and PointCLIP V2 by +6.4%. For the ModelNet40 dataset, which includes outdoor object categories, we outperform PointCLIP V2 by +1.6%. Moreover, for the progressively challenging datasets S_OBJ_ONLY, S_OBJ_BG, and S_PB_T50_RS, we exceed PointCLIP V2 by +7.6%, +12.2%, and +2.7%, respectively.

Ablation Study As shown in Table 2, we conduct ablation experiments on Point2Real, involving three dimensions

	PointCLIP V2	CLIP2Point	Point2Real
Time (s)	0.0097	0.0175	2.9360† / 0.0397‡
Acc. (%)	73.1	66.6	**79.5**† / **78.1**‡

Table 4: *Computational Cost.* All tests are conducted on a single NVIDIA GeForce RTX 4090 GPU. † denotes Point2Real implemented based on Blender, ‡ denotes a faster version based on PyTorch3D.

Shots	1	2	4	8	16
PointCLIP	52.6	69.2	76.4	81.2	86.3
PointCLIP + RSP	62.3	70.7	74.2	81.5	85.2
Point2Real	67.5	71.9	77.6	82.1	86.3

Table 5: *Few-shot classification Performances(%) on ModelNet40.* RSP denotes the Realistic Shape Projection module of PointCLIP V2.

for rendering point clouds realistically: **shape**, **texture**, and **view**. If we simply use a white color mesh reconstructed by the Shape Recovery Module, the zero-shot accuracy is 75.9% in ModelNet10 and ModelNet40, reduced by -3.6% and -5.5% respectively. When we use the texture candidate list for rendering, the texture information can improve zero-shot performance by +2.5% and +5.0%, respectively. With the Multi-view Adapter added, the accuracy can return to the highest value. Consistent with intuition, we observe that texture dimensions can lead to a substantial improvement for both datasets, which reflects that CLIP is indeed sensitive to texture information. Due to the fact that more than half of the viewpoints adopted in the datasets are easily distinguishable, the multi-views adapter seems to be less important in the ablation experiment. However, in practical applications, this module will provide greater assistance when we cannot ensure the recognizability of the majority of viewpoints. In Table 3, we conduct an ablation study on the Shape Recovery Module. If we use the Ball-Pivoting algorithm only once or use poison reconstruction, the performance will decrease by -0.9% and -23.2%, respectively. Furthermore, as presented in Table 4, Blender-based Point2Real can complete a full inference within 3 seconds, with the Texture Rendering Module consuming the majority of the time consumption. For real-time application scenarios, Blender can be substituted with a more efficient rendering framework, such as PyTorch3D. A simplified version based on PyTorch3D can achieve comparable efficiency to existing methods.

4.3 Few-shot Classification

Settings In order to test the performance of Point2Real in few-shot scenarios. We conduct k-shot experiments on ModelNet40 dataset, where $k \in \{1, 2, 4, 8, 16\}$. For k-shot experiment, we randomly sample k objects of each class from the training set for training. The brown texture is used for rendering. During the training process, for the sake of performance and efficiency, we keep the CLIP encoder frozen and only train the dual-path network. We compare our method with PointCLIP under its best settings. Considering there is no open-source implementation for the Few-shot part of PointCLIP V2 currently, we use PointCLIP+RSP to approximate its performance for comparison. We compose PointCLIP+RSP from PointCLIP's inter-view adapter and PointCLIP V2's Realistic Shape Projection (RSP). Specifically, for the visual backbone, we use RN101 for PointCLIP, ViT-B/16 for PointCLIP+RSP and Point2Real, respectively. For the textual input, we use the prompts "point cloud of a big [CLASS].", "An obscure grayscale depth map of an inclined rough [CLASS] 3D model.", and "This is a rendered image of a [CLASS] synthetic 3D mesh model." for PointCLIP, PointCLIP+RSP, and Point2Real, respectively.

Performance As shown in Table 5, Point2Real consistently outperforms other methods across different shot settings. In the one-shot setting, our method achieves 67.5%, surpassing PointCLIP by +14.9%, and PointCLIP+RSP by +5.2%. We observe that as the number of shots increases, the improvement of Point2Real is gradually reducing. We believe this may be attributed to the increasing training data, which enables the model to learn how to classify based on the variations in categories within the dataset. This implies that Point2Real's advantages can be more pronounced in a data-limited environment.

Ablation Study PointCLIP+RSP employs depth maps as input to train a single-path Adapter, whereas Point2Real uses both depth maps and rendered images as input to train a dual-path adapter. Therefore, PointCLIP+RSP can be regarded as an experiment where the rendering image input component is ablated in our approach. As illustrated in Table 5, we exhibit improvements across all different settings compared to PointCLIP+RSP. This indicates that the realistic rendering image generated by Point2Real can enhance the performance in few-shot scenarios, particularly when the number of shots is low.

5 Conclusions

We propose Point2Real, a powerful framework designed for transferring 2D Vision-Language Models into 3D. By converting point clouds into realistic rendered images from shape, texture, and viewpoint dimensions, Point2Real significantly reduces the gap between the 3D point cloud domain and the 2D Vision-Language domain. Our work can function as a zero-shot and training-free plugin, enabling the application of Pre-trained Vision-Language Models to the 3D domain. Experimental results conclusively demonstrate the strong zero-shot transfer ability of Point2Real, yielding state-of-the-art performance across most of the datasets. Furthermore, Point2Real can be flexibly adapted to few-shot tasks, effectively enhancing performance. While our work has limitations, notably increased time overhead attributed to the Blender rendering method, this issue can be significantly mitigated by adopting a more efficient rendering framework like Pytorch3D. Future works will focus on leveraging limited 3D data to optimize the transformation from point clouds to realistic images and applying Point2Real in more downstream tasks.

Acknowledgments

This work is partially supported by National Key R&D Program of China No. 2021ZD0111901, and Natural Science Foundation of China under contracts Nos. U21B2025, U19B2036.

References

Ahmed, S. M.; Liang, P.; and Chew, C. M. 2019. EPN: Edge-Aware PointNet for Object Recognition from Multi-View 2.5D Point Clouds. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3445–3450.

Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; and Taubin, G. 1999. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4): 349–359.

Blender Online Community. 2023. Blender: a 3D modelling and rendering package. http://www.blender.org. Accessed: 2023-07-12.

Cheraghian, A.; Rahman, S.; Campbell, D.; and Petersson, L. 2019. Mitigating the Hubness Problem for Zero-Shot Learning of 3D Objects. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Cheraghian, A.; Rahman, S.; Campbell, D.; and Petersson, L. 2020. Transductive Zero-Shot Learning for 3D Point Cloud Classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 923–933.

Cheraghian, A.; Rahman, S.; Chowdhury, T. F.; Campbell, D.; and Petersson, L. 2022. Zero-Shot Learning on 3D Point Cloud Objects and Beyond. *International Journal of Computer Vision*, 130(10): 2364–2384.

Cheraghian, A.; Rahman, S.; and Petersson, L. 2019. Zero-shot Learning of 3D Point Cloud Objects. In *2019 16th International Conference on Machine Vision Applications (MVA)*, 1–6.

Feng, Y.; Zhang, Z.; Zhao, X.; Ji, R.; and Gao, Y. 2018. GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 264–272.

Goyal, A.; Law, H.; Liu, B.; Newell, A.; and Deng, J. 2021. Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 3809–3820.

Groueix, T.; Fisher, M.; Kim, V. G.; Russell, B. C.; and Aubry, M. 2018. A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 216–224.

Guerrero, P.; Kleiman, Y.; Ovsjanikov, M.; and Mitra, N. J. 2018. PCPNet: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum*, 37(2): 75–85.

Guo, H.; Wang, J.; Gao, Y.; Li, J.; and Lu, H. 2016. Multi-View 3D Object Retrieval With Deep Embedding Network. *IEEE Transactions on Image Processing*, 25(12): 5526–5537.

Ha, H.; and Song, S. 2023. Semantic Abstraction: Open-World 3D Scene Understanding from 2D Vision-Language Models. In *Proceedings of The 6th Conference on Robot Learning*, 643–653.

Huang, T.; Dong, B.; Yang, Y.; Huang, X.; Lau, R. W.; Ouyang, W.; and Zuo, W. 2023. CLIP2Point: Transfer CLIP to Point Cloud Classification with Image-Depth Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 22157–22167.

Kazhdan, M.; Bolitho, M.; and Hoppe, H. 2006. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, 61–70.

Kazhdan, M.; and Hoppe, H. 2013. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3): 1–13.

Ma, X.; Qin, C.; You, H.; Ran, H.; and Fu, Y. 2022. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. In *International Conference on Learning Representations (ICLR)*.

Peng, S.; Jiang, C.; Liao, Y.; Niemeyer, M.; Pollefeys, M.; and Geiger, A. 2021. Shape As Points: A Differentiable Poisson Solver. In *Advances in Neural Information Processing Systems (NIPS)*, 13032–13044.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentationn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems (NIPS)*, 5099–5108.

Qi, Z.; Dong, R.; Fan, G.; Ge, Z.; Zhang, X.; Ma, K.; and Yi, L. 2023. Contrast with Reconstruct: Contrastive 3D Representation Learning Guided by Generative Pretraining. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 28223–28243.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 8748–8763.

Radovanović, M.; Nanopoulos, A.; and Ivanović, M. 2010. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *Journal of Machine Learning Research*, 11: 2487–2531.

Roveri, R.; Rahmann, L.; Oztireli, C.; and Gross, M. 2018. A Network Architecture for Point Cloud Classification via Automatic Depth Images Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4176–4184.

Sellán, S.; and Jacobson, A. 2022. Stochastic Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 41(6): 1–12.

Uy, M. A.; Pham, Q.-H.; Hua, B.-S.; Nguyen, T.; and Yeung, S.-K. 2019. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1588–1597.

Vakalopoulou, M.; Chassagnon, G.; Bus, N.; Marini, R.; Zacharaki, E. I.; Revel, M.-P.; and Paragios, N. 2018. AtlasNet: Multi-atlas Non-linear Deep Networks for Medical Image Segmentation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 658–666.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 38(5): 1–12.

Williams, F.; Schneider, T.; Silva, C.; Zorin, D.; Bruna, J.; and Panozzo, D. 2019. Deep Geometric Prior for Surface Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10130–10139.

Wu, W.; Qi, Z.; and Fuxin, L. 2019. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9621–9630.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1912–1920.

Xue, L.; Gao, M.; Xing, C.; Martín-Martín, R.; Wu, J.; Xiong, C.; Xu, R.; Niebles, J. C.; and Savarese, S. 2023. ULIP: Learning a Unified Representation of Language, Images, and Point Clouds for 3D Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1179–1189.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018a. EC-Net: an Edge-aware Point set Consolidation Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 386–402.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018b. PU-Net: Point Cloud Upsampling Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2790–2799.

Zhang, J.; Dong, R.; and Ma, K. 2023. CLIP-FO3D: Learning Free Open-world 3D Scene Representations from 2D Dense CLIP. arXiv:2303.04748.

Zhang, R.; Guo, Z.; Zhang, W.; Li, K.; Miao, X.; Cui, B.; Qiao, Y.; Gao, P.; and Li, H. 2022. PointCLIP: Point Cloud Understanding by CLIP. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8552–8562.

Zhu, X.; Zhang, R.; He, B.; Guo, Z.; Zeng, Z.; Qin, Z.; Zhang, S.; and Gao, P. 2023. PointCLIP V2: Prompting CLIP and GPT for Powerful 3D Open-world Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2639–2650.