Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

C2SPoint: A Classification-to-Saliency network for point cloud saliency detection*

Zhaoyi Jiang[a], Luyun Ding[a], Gary Tam[b], Chao Song[a], Frederick Li[c], Bailin Yang[a,d]

[a]School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou, 310018, China
[b]Department of Computer Science, Swansea University, United Kingdom
[c]Department of Computer Science, Durham University, United Kingdom
[d]School of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou, 310018, China

ARTICLE INFO

Article history:
Received July 13, 2023

Keywords:
point cloud saliency detection,
weakly-supervised learning,
learning features,
PointNet++

ABSTRACT

Point cloud saliency detection is an important technique that support downstream tasks in 3D graphics and vision, like 3D model simplification, compression, reconstruction and viewpoint selection. Existing approaches often rely on hand-crafted features and are only applicable to specific datasets. In this paper, we propose a novel weakly supervised classification network, called C2SPoint, which directly performs saliency detection on the point clouds. Unlike previous methods that require per-point saliency annotations, C2SPoint only requires category labels of the point clouds during training. The network consists of two branches: a Classification branch and a Saliency branch. The former branch is composed of two Adaptive Set Abstraction layers for feature extraction and a Saliency Transform layer for learning saliency knowledge from the classification network. The latter branch introduces a multi-scale point-cluster similarity matrix for propagating the cluster saliency to each point within it, resulting in the prediction of point-level saliency. Experimental results demonstrate the effectiveness of our method in point cloud saliency detection, with improvements of 2% in both AUC and NSS compared to state-of-the-art methods.

1. Introduction

The rapid advancement of radar scanning technology has greatly facilitated the acquisition of three-dimensional (3D) point clouds. However, due to their inherent sparsity and large number of points, preprocessing techniques are necessary to enhance computational efficiency and reduce storage overheads. In this context, point cloud saliency detection (PCSD) emerges as a promising approach that simulates the capabilities of the human visual system (HVS), providing a powerful alternative to data-driven operations such as filtering to improve efficiency. PCSD plays a crucial role in accurately identifying the most informative regions on 3D objects, leading to improved performance in downstream visual tasks [1, 2]. This capability has significant implications for enhancing human-machine interaction in saliency-based approaches [3]. PCSD also finds broad applications across various domains, including reconstruction [4, 5], compression [6, 7], simplification [8, 9], and viewpoint selection [10, 11]. These applications leverage the insights provided by PCSD to drive advancements in point cloud analysis and processing.

In recent years, several PCSD algorithms have been proposed [12, 13, 14]. However, these methods predominantly rely on hand-crafted features, which present limitations in terms of generalization and detection accuracy. To overcome these challenges, researchers have explored the use of neural networks for feature learning. They often conduct user studies to collect sub-

*Only capitalize first word and proper nouns in the title.
*Corresponding author: Tel.: +0-000-000-0000; fax: +0-000-000-0000;
e-mail: example@email.com (Corresponding Author Name)
[1]Footnote 1.

jective selection data from participants, aiming to implement supervised methods for saliency calculation. The underlying premise is that saliency reflects human visual perception of 3D objects [15]. For instance, Lavoue et al. [16] conducted an eye-tracking experiment to map eye fixations to 3D objects and investigate the impact of various factors on human attention. However, the limited availability of training datasets hinders the applicability of deep neural networks to this approach.

To address these challenges, we consider the possibility to develop a weakly supervised learning technique for 3D PCSD. We first observe that there is a strong correlation among visual recognition and saliency detection tasks, where the higher-level image features acquired through classification tasks prove beneficial for image saliency detection [17]. Neuroscience studies show that there are similarities in activated areas in the cortex during the processing of visual information [18, 19] and mutual enhancement [20] for both classification and saliency detection tasks. Visualizations of benchmark datasets [15] shown in Fig. 1 reveal that objects of the same category tend to have similar saliency distributions. This indicates that the human visual system selectively attends to informative features for rapid object identification, without scrutinizing all the details [21]. It also suggests a close relationship between visual saliency perception and object classification. Accordingly, [21] has developed a weakly supervised technique to transfer 2D multi-view object classification to mesh saliency. However, the approach cannot make use of the full 3D structure details of point cloud as it is 2D view based. On the other hand, large existing 3D classification datasets, such as ModelNet40 [22], are readily available, which offer uniqueness and certainty of 3D object categories, and provide ample data for training deep neural networks. All these motivate us to ask the question: "Can we benefit from well-trained 3D point cloud object classification tasks and use their high-level features directly to transfer learning 3D point-level saliency, through weak supervision?"

To this end, we propose C2SPoint, a Classification-to-Saliency network that leverages weakly supervised learning for 3D PCSD. Our network uses PointNet++ [23] as the backbone and consists of two branches: a classification branch and a saliency branch. The classification branch includes 1) cluster saliency extraction and 2) point-cluster similarity calculation, while the saliency branch utilizes a multi-scale point-cluster similarity matrix for saliency prediction. Training C2SPoint solely requires 3D object categories, obviating the need for tedious manual saliency annotations on every point. Our study also shows improvement against existing work. Our main contributions include:

- We propose C2SPoint, a novel neural network for 3D point cloud saliency detection. It is trained via weak supervision, leveraging only point cloud category labels without the need for point-wise saliency annotations.
- A multi-scale point-cluster similarity matrix is introduced to capture the relationships between points and clusters at multiple scales, enabling the prediction of point-level saliency from cluster saliency.
- Our experimental results demonstrate the superiority of the proposed method over state-of-the-art algorithms for point cloud saliency detection.

2. Related work

Feature extraction is a critical aspect of Point Cloud Saliency Detection. Existing techniques in PCSD can be classified into two categories based on how features are computed: hand-crafted detectors and learning-based detectors. Among these methods, transfer learning for 3D saliency detection has gained attention and has motivated our work in this paper.

2.1. Saliency detection based on hand-crafted features

Many existing point cloud saliency detection algorithms employ hand-crafted features to construct descriptors for analyzing point cloud saliency [12, 13, 14]. Prior research by Guo et al. [13] introduced a saliency detection algorithm that leverages covariance descriptors, demonstrating high discrimination and robustness. By utilizing techniques like the sigma-point method and principal component analysis (PCA), their approach significantly improved saliency detection efficiency. Other methods [12, 14] have adopted a cluster-to-point mechanism to enhance the efficiency of saliency computation. These approaches involve initially simplifying the point cloud into clusters, where each cluster is represented by a cluster head and aggregated features from local points. Subsequently, cluster saliency is computed based on cluster-level features and propagated to individual points using diverse strategies. For example, Tasse et al. [12] propose a point-cluster probability matrix to convert cluster saliency into point-wise saliency, while Ding et al. [14] incorporate a random walk ranking algorithm to assign cluster saliency to individual points. These cluster-based methods align with human visual attention patterns, which tend to focus on local regions rather than specific target points within a 3D object. Motivated by the effectiveness of these approaches, we employ a cluster-to-point saliency computation in our work to leverage these advantages.

Using hand-crafted features, these methods do not require large training data and manual saliency annotation. However, the expressiveness of the hand-crafted features is limited and tends to be susceptible to noise. In this study, we consider a weakly supervised learning approach to perform saliency detection, which is robust against noise. Our method does not require manual saliency annotation and shows superior results.

2.2. Saliency detection based on learning features

Inspired by the success of deep learning techniques in image saliency detection, recent research has shifted towards learning-based Point Cloud Saliency Detection, surpassing traditional hand-crafted algorithms. However, the sparse, irregular, and unordered nature of 3D point clouds poses challenges to deep learning architectures. These architectures are predominantly designed for processing and analyzing regular data, such as images and videos, which have inherent spatial structures. The lack of explicit spatial structures in 3D point clouds makes it difficult to directly apply deep learning techniques developed for regular data [24]. Many early techniques convert 3D point clouds into formats that are compatible with convolution neural

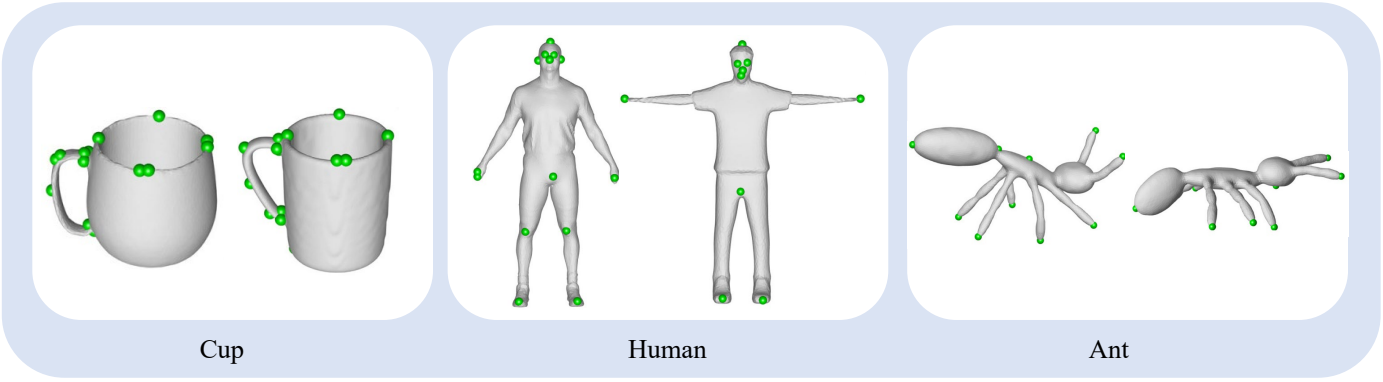

Fig. 1. The visualization of the points of interest selected by users in the datasets provided by Chen et al. [15].

networks, and have received considerable attention. One common approach encodes irregular points as feature vectors. It enables neural networks to process and analyze the point clouds effectively. For example, Atmosukarto and Shapiro [25] used a histogram of adjacent low-level features for each vertex and a classifier to label points as salient or non-salient. However, their manual annotation approach is time-consuming. Chen et al. [15] trained a regression model to predict the distribution of Schelling points from a dataset of 400 meshes. Wang et al. [3] designed a 5-layer convolutional neural network to predict human gaze on 3D objects. Limited training data would lead to underfitting in these networks. Shu et al. [26] devised a method for detecting 3D points of interest using projective neural networks. Zheng et al. [27] uses differentiable point dropout, relocating points to the center of objects to determine their importance ranking for classification. However, this approach overlooks the mechanism of human attention distribution on specific objects, focusing solely on the points' impact on the classification task. Hence, it is not appropriate to directly refer to the derived per-point importance as point saliency.

Leveraging learned features, the above methods often require large training data and manual saliency annotation. By contrast, we consider a weakly supervised approach to transfer learning saliency detection from the point cloud classification task. Our method solely requires 3D object categories, eliminating the need for tedious manual saliency annotations on every point.

2.3. Transfer Learning for 3D Saliency Detection

Traditional machine learning methods typically assume that the training and testing data belong to the same domain, sharing similar feature spaces and data distributions. However, in cross-domain and cross-distribution tasks, where acquiring training data and manual annotation are challenging, traditional learning-based approaches face significant obstacles. To address this, transfer learning methods [28, 29, 30, 31] have gained attention for leveraging datasets from other domains to learn relevant knowledge for the target task, offering robustness and flexibility.

3D saliency detection is a target task that often faces limitations in training datasets. Existing 3D saliency datasets with point-wise annotations are small, containing only a maximum of 400 objects from 20 categories [15], which is insufficient for effective network training. Song et al. [21] propose a multi-view convolutional neural network for saliency extraction, leveraging transfer learning to transform classification knowledge into mesh saliency. Inspired by this idea, we present a novel algorithm for Point Cloud Saliency Detection based on weak supervision, which only requires object-level labels (i.e., category labels) and eliminates the need for expensive point-wise annotation. Our work differs from [21] in three fundamental aspects. First, our neural network directly processes the input point clouds, while their method takes multi-view images as input and requires additional data preprocessing before training. Second, we focus on cluster saliency rather than view saliency, which allows us to capture and preserve the 3D structure information of objects. Third, we design a multi-scale point-cluster similarity matrix to describe the correlation between points and clusters and propagate the cluster saliency to each point. This matrix preserves multi-scale details, representing a reliable 3D-to-3D correspondence, while the 2D-to-3D correspondence in [21] may be less accurate in regions with low local vertex density.

3. Method

Our novel semi-supervised network, namely C2SPoint, consists of a Classification branch and a Saliency branch (see Fig. 2). The Classification branch utilizes two Adaptive Set Abstraction (ASA) layers for feature extraction and a Saliency Transform (ST) layer to learn saliency information from the classification network. The Saliency branch introduces a multi-scale point-cluster similarity matrix for point-level saliency prediction. During training, C2SPoint directly operates on the raw point cloud and learns saliency information from the classification task. During deployment, the workflow involves four main steps: 1) point cloud feature extraction; 2) cluster saliency evaluation; 3) multi-scale point-cluster similarity computation; and 4) point-wise saliency calculation.

3.1. Classification branch

The Classification branch of C2SPoint, as depicted in Fig. 2, adapts the PointNet++ backbone to promote saliency. It consists of two key components: the adaptive set abstraction layer (ASA) and the saliency transform layer (ST).

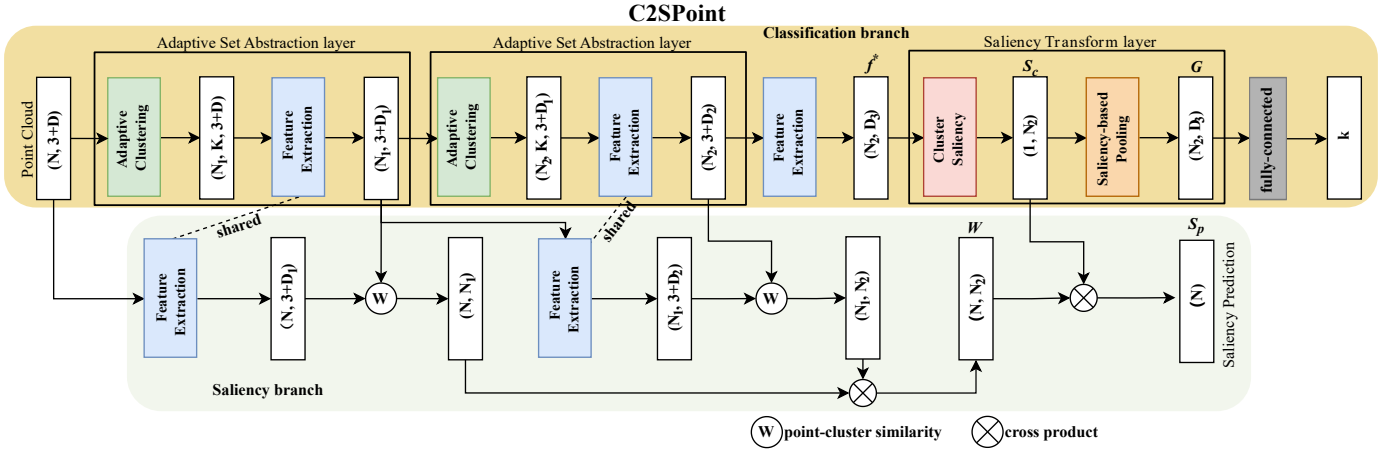

Fig. 2. The proposed method framework. The upper half illustrates the Classification branch, which consists of two Adaptive Set Abstraction (ASA) layers for feature extraction and a Saliency Transform (ST) layer for learning saliency information from the classification network. Each ASA layer comprises an Adaptive Clustering (AC) module and a Feature Extraction (FE) module. The Saliency Transform (ST) layer includes a Cluster Saliency (CS) module and a Saliency-based Pooling (SP) module. The bottom half illustrates the saliency branch, which introduces a multi-scale point-cluster similarity matrix for point-level saliency prediction. The circled W symbol represents the computation of point-cluster similarity, indicating the correlation between clusters and points.

The ASA layer extracts multi-scale point cloud features using the adaptive clustering module (AC) and the feature extraction module (FE). We employ two ASA layers to balance computation cost and the ability to adaptively extract informative features from the point cloud. Each ASA layer takes an $N \times (3+D)$ matrix as input, representing N points with 3D coordinates and D-dimensional point features. It outputs an $N' \times (3+D')$ matrix representing N' clusters with 3D coordinates and new D'-dimensional cluster features that capture local context.

The ST layer leverages knowledge from the classification network to learn saliency information. It utilizes the cluster saliency module (CS) and the saliency-based pooling module (SP). The ST layer takes the outputs of all ASA layers as input, which is a matrix of size $D' \times N'$ for a given point cloud. Each column of this matrix represents a feature descriptor of one cluster. The ST layer then produces a re-weighted $D' \times N'$ descriptor that represents the entire point cloud. It is important to note that the cluster saliency output by the CS module is further utilized in the Saliency branch to generate point-wise saliency. Finally, the Classification branch outputs a k-dimensional vector to predict the category of the given point cloud.

In the following sections, we will provide detailed explanations of these sub-modules.

3.1.1. Adaptive Set Abstraction layer

Adaptive clustering module. This module aims to assign each point to a cluster based on the proximity of points to the cluster center in both Euclidean space and descriptor space. The architecture follows the backbone of PointNet++ [23] due to its excellent performance in classification and direct point processing. However, the clustering mechanism of PointNet++, which uses Farthest Point Sampling and ball query, only considers Euclidean distance when assigning points to clusters, resulting in unstable downsampled points. For saliency detection, we believe that points within the same cluster should be close not only in their positions but also in their features. To address this, we propose an adaptive clustering (AC) module that incorporates both Euclidean and descriptor distance.

Given a 3D point cloud $P = \{P_i | i = 1, \ldots, N\} \subset \mathbb{R}^{3+D}$ as input, the output of this module is N' clusters with a data size of $N' \times K \times (3+D)$. The AC module is designed as follows:

Firstly, we employ farthest point sampling to select N' points from the point cloud as the initial sampling subset $C^{init} = \left\{c_j^{init} | j = 1, \ldots, N'\right\}$, ensuring that the selected points cover the point cloud as widely as possible. The cluster heads are initially defined as c_j^{init}.

Secondly, we select the M nearest points using the k-Nearest Neighbor algorithm to construct the local neighborhood and calculate the initial features for each cluster:

$$f^{init}\left(c_j^{init}\right) = \frac{1}{M} \sum_{p_m \in \delta\left(c_j^{init}\right)} f\left(p_m\right) \tag{1}$$

where p_m refers to any point in the local neighborhood $\delta(c_j^{init})$ of the cluster head c_j^{init}. Note that $f(\cdot)$ represents the feature descriptor of points. In the first ASA layer, $f(\cdot)$ represents the original geometric features of the input point cloud, i.e., normal vectors. In the second ASA layer, $f(\cdot)$ denotes the output features learned from the previous ASA layer.

It is worth noting that the k-nearest neighbor algorithm, which only uses Euclidean distance, is not sufficient to measure closeness in descriptor space. To capture the local geometric feature differences of the point cloud, we introduce the chi-squared distance into the distance metric, following the methods in [12, 14]. This inclusion reduces the differences in feature descriptors within a cluster, resulting in greater similarity between points in the same cluster in terms of spatial location and geometric structure. Thus, the average descriptor in Eq. 1 adequately describes the local neighborhood information of the cluster head. The distance metric $D(c, p)$ is calculated as:

$$D(c,p) = \frac{\|u(c) - u(p)\|}{\max_u} + \lambda \frac{\chi^2(f^{init}(c), f^{init}(p))}{\max_f} \quad (2)$$

where $u(\cdot)$ represents the spatial coordinate, $\max_u$ and $\max_f$ are the maximum distances from all points to the cluster head c in Euclidean and descriptor space, respectively. λ is used to assign different importance to spatial distance and chi-square distance based on their contributions to the clustering results. $\|\cdot\|$ denotes the Euclidean distance, and $\chi^2(\cdot)$ is the descriptor distance defined as:

$$\chi^2(f^{init}(c), f^{init}(p)) = \sum_{b=1}^{B} \frac{(f_b^{init}(c) - f_b^{init}(p))^2}{f_b^{init}(c) + f_b^{init}(p)} \quad (3)$$

where B represents the number of bins used for dividing the descriptor f^{init}.

Next, the existing subsets are used to calculate the cluster center c_o as follows:

$$c_o = \frac{1}{M} \sum_{m=1}^{M} p_m \quad (4)$$

Note that the cluster center may not correspond to an existing point in P. Therefore, it is necessary to re-determine the closest point to the cluster center as the new cluster head.

Finally, the final cluster set $C = \{c|j = 1, \ldots, N'\}$ and its corresponding subset $C_{\delta(c_j)} = \{p_{j_1}, p_{j_2}, \ldots, p_{j_M}\}$ are acquired by determining the neighborhood of the cluster head.

Feature extraction module. The feature extraction module aims to fuse the features of the local neighborhood around the cluster head to obtain the cluster feature f_c. It takes an $N' \times K \times (3 + D)$ tensor as input and outputs an $N' \times (3 + D')$ matrix f of N' clusters with 3-dimensional coordinates and new D'-dimensional feature vectors.

Firstly, the spatial coordinates of all points in the cluster are normalized to form a local neighborhood space with the cluster head as the origin:

$$\hat{u}(p_\delta) = u(p_\delta) - u(c) \quad (5)$$

where $p_\delta \in C_{\delta(c)}$ and $\hat{u}(\cdot)$ denotes the normalized coordinate.

Secondly, a shared Multi-Layer Perceptron (MLP) is utilized to encode the features of each point in the cluster independently. The cluster feature is formulated as:

$$f_c = \mathrm{MAX}\{MLP(\hat{u}(p_\delta) \oplus f(p_\delta))\} \quad (6)$$

where $\oplus$ represents feature concatenation, and MAX denotes the max-pooling operation.

The cluster feature f_c captures the geometric structural information of the cluster's local region and can be effectively used to calculate cluster saliency. Note that an additional feature extraction layer is introduced after the second ASA layer (Fig. 2). This additional layer's purpose is to extract a high-dimensional feature f^* that provides a better description of the clustering results obtained by C2SPoint.

3.1.2. Saliency Transform layer

Cluster saliency module. The CS module takes the outputs of all ASA layers (an $N' \times D'$ matrix representing the cluster feature) as input and outputs an N'-dimensional vector to the next layer. The concept of cluster saliency is inspired by the human visual system (HVS), which suppresses responses to frequently occurring features while maintaining sensitivity to unique and deviating features from the norm [32, 33]. Deviating features in a region are considered salient for the HVS. Therefore, the cluster saliency s_c is calculated by measuring the dissimilarity of Euclidean distances between cluster features. The saliency of a cluster is determined by the sum of dissimilarities and can be formulated as:

$$d(c_j, c_k) = \left\| f^*(c_j) - f^*(c_k) \right\| \quad (j \neq k) \quad (7)$$

$$s_{c_j} = \sum_{j \neq k} d(c_j, c_k) \quad (8)$$

where $d(c_j, c_k)$ represents the dissimilarity between the feature of cluster c_j and c_k, with $j, k \in \{1, \ldots, N'\}$. s_{c_j} denotes the cluster saliency of cluster c_j, which will be used for predicting point-level saliency in the Saliency branch.

Saliency-based pooling module. The Saliency-based pooling module integrates saliency information into the classification network to enhance its performance. Salient regions in 3D objects are typically unique and crucial for distinguishing categories [32]. Therefore, the saliency of a region should positively correlate with its impact on the classification result. Motivated by this, the cluster feature f^* is reweighted using the cluster saliency s_c as the weight to obtain the global feature G, which represents the input point cloud effectively:

$$G = s_c \cdot f^* \quad (9)$$

Finally, the global feature G is fed into a fully connected layer, enabling it to participate in the training process of the classification network. By integrating cluster saliency s_c into the classification task, the influence of salient regions on the classification results is increased. Furthermore, changes in the classification results also impact the cluster saliency during backpropagation. This bidirectional process establishes a close relationship between classification and saliency knowledge, ensuring reliable classification-to-saliency transfer learning.

3.2. Saliency branch

This branch computes point-level saliency using a multi-scale point-cluster similarity matrix. This matrix captures the correlation between points and clusters based on multi-scale features. At each ASA layer, the Saliency branch takes an $N \times (3 + D')$ matrix of point features and an $N' \times (3 + D')$ matrix of cluster features as input. It produces an $N \times N'$ similarity matrix, which quantifies the similarity between each point and each cluster. This similarity matrix is utilized to measure the similarity of a point p to each cluster. Finally, the point-level saliency is computed by combining the saliency values from all clusters through a linear combination. This aggregation of saliency values enables a comprehensive assessment of the significance of a point within the point cloud.

3.2.1. Multi-scale point-cluster similarity

Tasse et al. [12] introduced a point-cluster probability to capture the likelihood of a point belonging to a cluster, prioritizing the influence of the most probable cluster on point saliency. However, their approach overlooks local details within clusters. To address this limitation, we use multi-scale features to calculate a point-cluster similarity matrix. This matrix reflects the relationship between each point and cluster, capturing fine-grained details across different scales.

Single-scale point-cluster similarity. Inspired by Tasse et al.'s method [12], we construct an auxiliary weight matrix ξ to analyze the relationship between each point p and cluster c at each scale. In the propagation step, we aim to prioritize the influence of the most likely correlated cluster on the saliency of point p. To achieve this, we calculate the Euclidean distance between the spatial coordinates of a point and each cluster, as well as the Chi-square distance between the descriptor of a point and each cluster. This information is incorporated into the definition of the auxiliary weight matrix at scale r as follows:

$$\xi_r(p,c) = \frac{e^{-(\lambda_u \left\| \hat{u(p)} - u(c) \right\|_2 - \lambda_f \chi^2(f_r(p), f_r(c)))}}{\sigma_u \sigma_f} \tag{10}$$

Here, $\xi_r(\cdot,\cdot)$ represents the auxiliary weight matrix at scale r. The parameters λ_u and λ_f determine the importance of the spatial coordinates and geometric features, respectively, and we set $\lambda_u = 1$ and $\lambda_f = 5$ in our method. Also, σ_u represents the mean Euclidean distance of all points to the center position of the point cloud, while σ_f represents the mean chi-square distance of all points to the global descriptor of the point cloud.

By utilizing the constructed auxiliary weight matrix ξ, we define the single point-cluster similarity matrix as:

$$\xi_r'(p,c) = \frac{\xi_r(p,c)}{\sum_{q \in P} \xi_r(q,c)} \tag{11}$$

After obtaining the single point-cluster similarity matrix $\xi_r'(p,c)$, we further normalize each row to obtain a matrix with values ranging between 0 and 1. This normalization step ensures that the weights reflect the relative influence of each cluster on the saliency of a given point at the specific scale. The normalized matrix is defined as follows:

$$W_r(p,c) = \frac{\xi_r'(p,c)}{\sum_{c_k \in C} \xi_r'(p,c_k)} \tag{12}$$

Here, $W_r(p,c)$ represents the normalized point-cluster similarity weight between point p and cluster c at scale r. The denominator in the equation is the sum of all normalized similarities for point p across all clusters c_k.

Note that our algorithm takes into account the influence of neighboring clusters on points. Points that are located far away from a cluster in terms of spatial distance are unlikely to belong to that cluster. To handle this, we introduce a distance threshold, which is set as 1% of the radius of the bounding sphere of the point cloud. When the distance between point p_i and cluster c_j exceeds this threshold, the point-cluster similarity is set to 0, indicating no influence from the distant cluster, i.e., $W_r(p_i, c_j) = 0$ if distance$(p_i, c_j) >$ threshold.

By incorporating this distance-based constraint, we ensure that the point-cluster similarity accurately captures the local relationships between points and their nearby clusters, while disregarding distant clusters that are unlikely to have a significant influence on the saliency of a given point.

Multi-scale point-cluster similarity. Multi-scale analysis provides a more comprehensive understanding of the input data at different levels of detail [34], which can lead to more compact and informative representations. Based on this, a multi-scale point-cluster similarity fusion mechanism is proposed to aggregate the similarity of points p and clusters c computed by fine-grained and coarse-grained features. Specifically, a nonlinear weighted mechanism is adopted to implement the fusion of point-cluster similarity at each scale.

$$W_{ms} = \sum_r (\max(W_r) - \text{mean}(W_r))^2 W_r \tag{13}$$

where $\max(\cdot)$ is the maximum similarity and $\text{mean}(\cdot)$ is the average similarity at scale r. This ensures that the similarity at the scale with the largest peak-valley difference is given greater weight in the final multi-scale point-cluster similarity W_{ms}.

Point-cluster similarity transfer. There are multiple adaptive set abstraction (ASA) layers in the Classification branch of C2SPoint, which are implemented to continuously expand the receptive field of feature extraction and learn deeper structural information. Therefore, the similarity for each ASA layer needs to be further transferred to describe the closeness between the input point cloud P and the final clustering result C of all the ASA layers.

Let W_{ms}^l represent the similarity in the l-th adaptive set abstraction layer, where $l = 1, 2, \ldots, L$. It can be inferred that:

$$W_{ms}^1 \in \mathbb{R}^{N \times N_1}, W_{ms}^2 \in \mathbb{R}^{N_1 \times N_2}, \ldots, W_{ms}^L \in \mathbb{R}^{N_{L-1} \times N_L} \tag{14}$$

where $N > N_1 > \cdots > N_L = N'$. This means that the number of cluster centers decreases with the stacked ASA layers until the final N' cluster centers represent N' clusters. For the input point cloud P and the cluster C^L (i.e., the final cluster C) output by the L-th ASA layer (i.e., the last ASA layer), the similarity W^* can be transferred by performing matrix multiplications on the intermediate results at each ASA layer.

$$W^* = W_{ms}^{1 \to L} = W_{ms}^1 \cdot W_{ms}^2 \cdots W_{ms}^L \tag{15}$$

3.2.2. Point-level saliency computation

Firstly, the ModelNet40 dataset [22] is utilized to train the Classification branch of C2SPoint, which serves two purposes: 1) training the network for the classification task, and 2) extracting saliency knowledge from the saliency-based pooling module. Subsequently, the trained model is employed to extract saliency predictions. The saliency computation process consists of three steps:

Cluster saliency extraction. The original point clouds are fed into the network, and the final clustering result is obtained through two ASA layers. The cluster saliency s_c is then calculated in the saliency transform layer of the Classification branch. (See Sec. 3.1)

Point-cluster similarity calculation. Single-scale point-cluster similarity is computed at each ASA layer, representing a mini-propagation of saliency from the output subsampled points (clusters) to the input points. In this process, the input data of each layer serves as the original point set for computing point saliency, while the output represents the clustering results partitioned by the AC module within that layer. Notably, the input features f_{in} are further processed by a shared Multi-Layer Perceptron (MLP) to ensure a consistent number of channels between the input and output features at each layer.

$$\hat{f}_{in} = mlp^{l}(f_{in}) \tag{16}$$

Next, $\hat{f}_{in}$ is combined with the descriptor of cluster C^l generated by the l-th AC module to calculate the point-cluster similarity matrix W^l for that ASA layer.

$$W^{l} = W_{ms}(p_{in}, p_{out}) \tag{17}$$

where p_{out} refers to the descriptor of cluster C^l for better clarity.

Finally, the similarity W^* between the input point cloud P and the final clusters C is generated by multiplying the similarity matrices W computed at each ASA layer. (See Sec. 3.2)

Point-level saliency computation. Point-level saliency is computed by propagating the saliency of clusters to the saliency of points using the multi-scale point-cluster similarity matrix W^*. This can be represented as:

$$s = W^{*} s_{c} \tag{18}$$

Here, W^* represents the point-cluster similarity matrix that describes the correlation between points and clusters. By multiplying W^* with the cluster saliency vector s_c, we ensure that the saliency of a point p is most influenced by the clusters it is most likely to belong to.

4. Experiments

4.1. Datasets, Experiment Setups, and Evaluation Metrics

Training datasets. For training the proposed C2SPoint, we utilize the ModelNet40 dataset [22], which consists of 12,311 CAD models from 40 categories. We follow the official split, using 9,843 shapes for training and 2,468 shapes for testing.

Benchmark datasets. To evaluate the performance of our proposed method, we employ the benchmark dataset provided by Chen et al. [15]. This dataset comprises saliency distribution maps and human-selected interest points for 400 3D objects across 20 categories.

Implementation details. We implement C2SPoint using PyTorch. The Adam optimizer is utilized with a momentum of 0.9 and weight decay set to 0.0001. The training process spans 200 epochs, with a batch size of 12. We set the initial learning rate to 0.001. The configuration of clusters, radius, and MLPs follows that of PointNet++ [23]. Specifically, the channel of the last MLP in C2SPoint is set to (256, 512, 1024).

Evaluation Metrics. The accuracy of the proposed algorithm is evaluated using commonly used metrics in saliency detection, namely AUC, NSS, and LCC [35]. AUC is an evaluation metric specifically designed for saliency detection tasks. It focuses on the ranking of saliency values while disregarding non-salient regions. The metric is calculated by determining the area under the ROC curve. NSS, on the other hand, is a widely employed measure for evaluating saliency detection algorithms. It quantifies the saliency value of fixation points along the eye-scanning path of each user. LCC, which stands for Linear Correlation Coefficient, is utilized to assess the strength of the linear relationship between the saliency values generated by the computational model and the ground truth.

Table 1. Classification accuracy on ModelNet40 [22].

Network	Accuracy%
PointNet [36]	89.2
PointNet++ [23]	91.9
CfS-CNN [21]	88.3
C2SPoint(ours)	91.0

4.2. The transferability from classification to saliency

Classification results. Since C2SPoint is essentially a network for classification, it is necessary to evaluate the effectiveness on a classification task. In Table 1, we compared our method with three commonly used networks for 3D classification, namely [21, 23, 36]. These networks were selected for two main reasons: First, we utilized PointNet++ as the backbone for our network, and comparing with these networks allows us to assess the impact of incorporating saliency into the classification process. Second, CfS-CNN is another network trained in a weakly supervised manner, but it is based on multi-view images and work on meshes only. We try to compare with [21] on a classification task to show the effect of preserving the inherent 3D characteristics in our method.

When compared to PointNet [36] and CfS-CNN [21], our work, C2SPoint, demonstrates improved classification accuracy by 0.8% and 2.7%, respectively. The superior performance of C2SPoint can be attributed to its multi-scale mechanism, which captures detailed local structures and surpasses PointNet [36] in classification. Additionally, by utilizing the original point clouds as input instead of multi-view 2D images, C2SPoint preserves the inherent 3D characteristics of the points, leading to more accurate classification results compared to CfS-CNN [21]. However, in comparison to PointNet++ [23], C2SPoint exhibits a slight decrease in classification accuracy. Although integrating cluster saliency into C2SPoint enhances the influence of salient regions on classification results, it does not completely eliminate the impact of low-salient regions. Nevertheless, C2SPoint demonstrates excellent overall performance in classification, showcasing the reliability of the network.

High-Drop test. The High-Drop test is conducted to examine the influence of saliency on classification results. In this experiment, we utilize ModelNet40 [22] to perform the test. The proposed algorithm is employed to compute the point-wise saliency, and subsequently, the salient points are systematically removed to assess their impact on classification.

Fig. 3 presents the classification accuracy before and after the High-Drop test for all categories in ModelNet40 [22]. The results reveal that the removal of these salient points leads to a

decrease in classification accuracy exceeding 70%. Additionally, prior to the point removal, the accuracy of 20 categories remains above 80%, while after the removal, less than half of the models maintain an accuracy above 80%. These findings underscore the significant influence of the salient points identified by the proposed algorithm on the classification results.

In Fig. 4, the visual illustration demonstrates the impact of removing salient points (highlighted in red) on the classification results. It is evident that the removal of these salient points leads to incorrect classification outcomes, which contradicts human visual perception. These results highlight the significance of salient regions in distinguishing objects, thereby affirming the feasibility of leveraging saliency information from classification knowledge.

4.3. Quantitative evaluation

Here, we further compared our method with several existing approaches, namely handcrafted based methods [12, 13, 14] and learning based approach [26].

Table 2 illustrates the superior performance of our algorithm compared to state-of-the-art methods [12, 13, 14, 26], achieving the highest AUC value in 65% of the categories. Notably, learning-based methods, including our algorithm, demonstrate better performance than all other methods, exhibiting higher AUC scores across most object categories. This finding underscores the effectiveness of deep learning-based saliency detection across diverse categories. Additionally, our algorithm showcases significant improvements of 22%, 18%, and 13% respectively on the Fish, Spring, and Teddy models when compared to the algorithm proposed by Shu et al. [26].

Also, Table 3 presents the quantitative enhancements of our algorithm over Shu et al.'s approach [26], with a 2% increase in AUC value and a 2% higher NSS value. When compared to handcrafted based methods [12, 13, 14], our algorithm demonstrates superior performance.These results clearly show that our algorithm is more accurate in 3D point cloud saliency detection.

4.4. Qualitative evaluation

Comparison with point-based saliency algorithms. In Fig. 5, our proposed algorithm is compared to point-based algorithms. The algorithm presented in [12] fails to correctly detect the palm of the hand. This can be attributed to the palm being widely distributed with low global rarity, making it less salient to the human visual system. The algorithm proposed by Guo et al. [13] identifies the contour of the teddy as salient. However, human eyes tend to focus more on the face and limbs of the teddy. The method by Ding et al. [14] does not perform well on cylindrical shapes, such as the octopus feet and chair legs. In most 3D objects, our algorithm achieves saliency detection results that are more consistent with the ground truth [15], such as accurately identifying the feet of the octopus and chair, as well as the limbs of the teddy. However, our algorithm exhibits weaker performance on the egea and vase models. It is possible that the classification accuracy of C2SPoint does not reach 100%, resulting in learning features that are not entirely accurate for these particular 3D objects and subsequently leading to less satisfactory saliency detection results.

Comparison with mesh-based saliency algorithms. As shown in Fig. 6, our algorithm successfully detects the salient regions corresponding to the eyes and ears of the bunny, which aligns with human visual perception. Similarly, our algorithm accurately identifies the horns and claws of the dragon as salient regions. In contrast, the mesh-based saliency algorithms exhibit limitations in their performance. For instance, the algorithm by Wu et al. [37] heavily relies on parameter settings and fails to capture certain salient regions, such as the eyes of the dragon model. The algorithm proposed by Leifman et al. [38] generates excessive salient regions, while the algorithm introduced in [39] produces unreasonable salient regions in the dragon model due to its dependency on pre-segmentation. In comparison, our proposed method achieves accurate detection results without relying on topological structures, outperforming the aforementioned mesh-based algorithms

Gallery of saliency detection results. In Fig. 7, we can observe that the salient regions detected by our algorithm exhibit strong consistency with the pseudo ground truth [15]. Furthermore, our algorithm demonstrates robust performance in saliency detection, even when applied to testing datasets that were not included in the network training. This outcome highlights the effectiveness of utilizing learning-based features, which significantly enhances the adaptability of our algorithm to objects across diverse categories

4.5. Ablation study

To evaluate the effectiveness of the Cluster Saliency module in C2SPoint, we conducted an ablation study by replacing the Saliency Transform layer with max pooling. The cluster saliency was set to 1, while all other conditions remained the same. This ablated version calculated point saliency solely based on classification knowledge. Depicted in Table 4, our algorithm with the Cluster Saliency module achieved a higher cluster saliency of 0.157 in AUC, 0.056 in NSS, and 0.026 in LCC compared to its ablated version. They show that the Cluster Saliency module effectively extracts saliency information.

Furthermore, we validated the effectiveness of the point-cluster similarity matrix by combining the cluster saliency with the clustering results to generate saliency scores for all points. Specifically, the saliency scores of points within a cluster were set to the cluster saliency (i.e., $s_p = s_c(p \in c)$). Quantitative evaluations in Table 4 demonstrate that without using point-cluster similarity propagation, the AUC decreased by 0.392, NSS decreased by 0.344, and LCC decreased by 0.045. These findings indicate that the point-cluster similarity matrix effectively improves the saliency detection performance.

4.6. Runtime analysis

Table 5 illustrates the runtime required for calculating the saliency of various 3D objects discussed in this study. The runtime measurements were conducted on a workstation equipped with an Intel 2.6GHz CPU and a TITAN XP GPU with 12GB of memory. Interestingly, as the number of points in the objects increases, the corresponding computation time shows a relatively slow growth. For example, despite the Armadillo having approximately 54 times more points than the Cow, the saliency

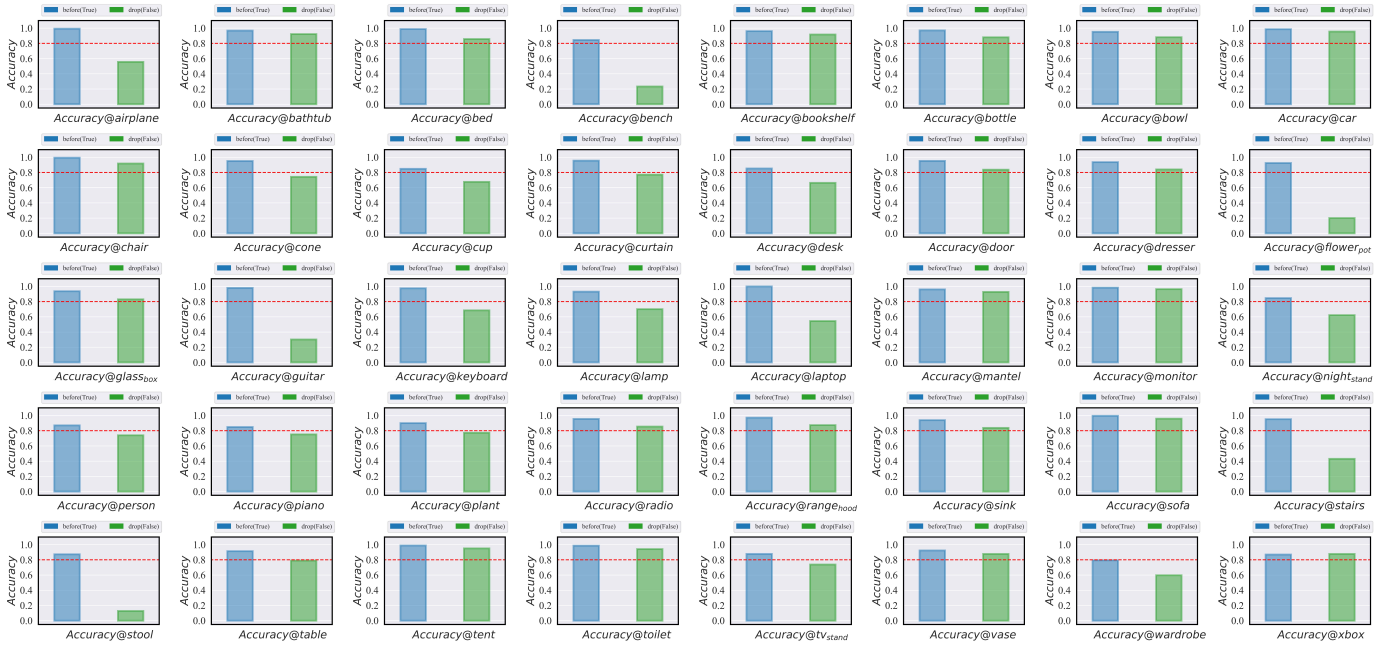

Fig. 3. High-Drop experiment on ModelNet40 dataset [22]. Each subfigure shows the classification accuracy results. The blue bars represent the accuracy of C2SPoint, the green bars represent the accuracy after the High-Drop experiment, and the red dashed lines indicate the 80% accuracy benchmark.

Table 2. Comparison of AUC scores in each category between our method and other state-of-the-art methods. Here, [12, 13, 14] are hand-craft feature based methods and [26] is learning based method. Bold text means the best performing technique.

	Human	Cup	Glasses	Airplane	Ant	Chair	Octopus	Table	Teddy	Hand	Plier	Fish	Bird	Spring	Armadillo	Buste	Mechanic	Bearing	Vase	Four-leg
Ours	**0.68**	**0.68**	0.48	**0.77**	**0.76**	0.67	**0.71**	**0.69**	0.68	**0.66**	0.50	**0.85**	**0.72**	**0.80**	**0.75**	0.51	**0.71**	0.69	0.69	**0.83**
Shu [26]	0.65	0.65	**0.65**	0.70	0.67	**0.70**	0.62	**0.69**	**0.74**	**0.66**	**0.63**	0.63	0.67	0.62	0.69	**0.67**	0.68	**0.71**	0.68	0.70
Ding [14]	0.65	0.68	0.56	0.73	0.68	0.56	0.52	0.65	0.65	0.64	0.56	0.79	**0.72**	0.55	0.74	0.60	0.68	0.69	**0.72**	0.80
Guo [13]	0.59	0.64	0.53	0.60	0.59	0.59	0.56	0.65	0.57	0.61	0.59	0.66	0.59	0.55	0.66	0.63	0.70	0.62	0.63	0.61
Tasse [12]	0.57	0.60	0.52	0.62	0.60	0.57	0.54	0.63	0.56	0.61	0.59	0.66	0.60	0.55	0.65	0.62	0.68	0.60	0.62	0.60

Table 3. Performance on the benchmark dataset in terms of area under ROC curve (AUC), Normalized Scanpath Saliency (NSS), and Linear Correlation Coefficient (LCC).

	Ours	Shu [26]	Ding [14]	Guo [13]	Tasse [12]
AUC↑	**0.69**	0.67	0.65	0.61	0.60
NSS↑	**0.85**	0.83	0.83	0.80	0.80
LCC↑	0.40	**0.61**	0.45	0.35	0.38

Table 4. Evaluation of the saliency transform layer and point-cluster similarity matrix.

Metrics	AUC↑	NSS↑	LCC↑
w/o ST	0.536	0.791	0.375
w/o Similarity	0.301	0.503	0.356
Ours	**0.693**	**0.847**	**0.401**

calculation time is only around 5 times longer than that of the Cow. This observation indicates that the proposed method is highly effective and exhibits insensitivity to the number of object points. This behavior can be attributed to the cluster-based nature of our algorithm, which efficiently processes the points in a grouped manner. Overall, these results demonstrate the efficiency and scalability of our method, highlighting its ability to handle varying object complexities without significantly compromising the computation time.

5. Conclusion

We propose a novel point cloud saliency detection algorithm based on a weakly supervised training approach using our classification network C2SPoint. By transferring classification knowledge to saliency knowledge, our algorithm achieves superior performance in computing point cloud saliency compared to state-of-the-art 3D saliency detection algorithms based on benchmark datasets. Furthermore, our method has diverse applications. For example, saliency-driven approaches enhance computational efficiency in point cloud compression by prioritizing salient regions for higher bit allocation. In point cloud simplification, saliency serves as a dynamic clustering radius,

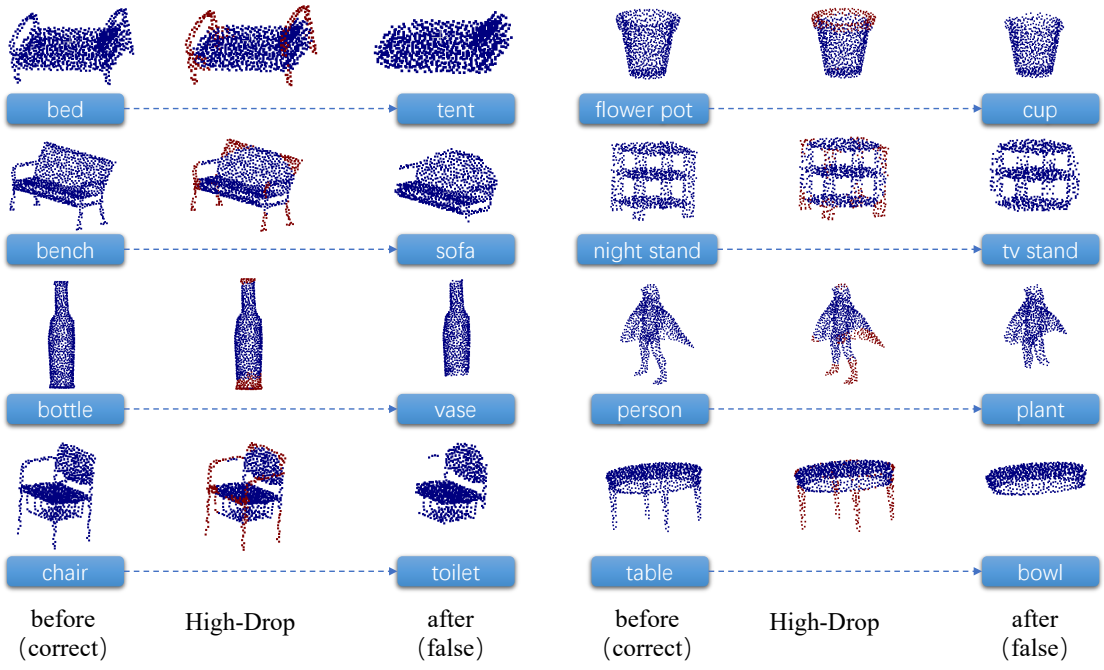

Fig. 4. Visualization of the High-Drop test. The red points represent salient points computed by the proposed method. From left to right: the original correct prediction, the visualization of dropped points, and the incorrect prediction after dropping points.

Table 5. Computation runtimes for calculating the saliency of various 3D objects mentioned in the paper.

Objects	# Points (K)	Time (s)
Cow	3.2	9.90
Airplane	5.9	9.82
Pig	13.2	11.15
Cup	15.0	11.34
Bunny	35.2	18.91
Armadillo	172.9	49.92

guiding the simplification process. For viewpoint selection, summing the saliency values of visible points at each view helps identify the most informative or least informative view.

However, there are some limitations in our method. The use of FPS for selecting initial cluster centers, although efficient, introduces variability in the cluster heads, leading to potentially less robust clustering results. Future work will focus on enhancing the clustering algorithm, such as point cloud segmentation, to improve the stability of the results. On the other hand, our point-cluster similarity matrix is designed in hand-crafted way and this can be improved in a learning-based way in the future work.

6. Acknowledgments

This research was partially supported by Zhejiang Province Natural Science Foundation No. LY21F020013, LY22F020013, the National Natural Science Foundation of China No. 62172366. Gary Tam is supported by the Royal Society grant IEC/NSFC/211159. For the purpose of Open Access the author has applied a CC BY copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Ni, B, Xu, M, Nguyen, TV, Wang, M, Lang, C, Huang, Z, et al. Touch saliency: Characteristics and prediction. IEEE Transactions on Multimedia 2014;16(6):1779–1791.

[2] Frintrop, S, Rome, E, Christensen, HI. Computational visual attention systems and their cognitive foundations: A survey. ACM Transactions on Applied Perception (TAP) 2010;7(1):1–39.

[3] Wang, X, Koch, S, Holmqvist, K, Alexa, M. Tracking the gaze on objects in 3d: How do people really look at the bunny? ACM Transactions on Graphics (TOG) 2018;37(6):1–18.

[4] Sharma, R, Schwandt, T, Kunert, C, Urban, S, Broll, W. Point cloud upsampling and normal estimation using deep learning for robust surface reconstruction. arXiv preprint arXiv:210213391 2021;.

[5] Yang, J, Ahn, P, Kim, D, Lee, H, Kim, J. Progressive seed generation auto-encoder for unsupervised point cloud learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021, p. 6413–6422.

[6] Sheng, X, Li, L, Liu, D, Xiong, Z. Attribute artifacts removal for geometry-based point cloud compression. IEEE Transactions on Image Processing 2022;31:3399–3413.

[7] Que, Z, Lu, G, Xu, D. Voxelcontext-net: An octree based framework for point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, p. 6042–6051.

[8] Lv, C, Lin, W, Zhao, B. Approximate intrinsic voxel structure for point cloud simplification. IEEE Transactions on Image Processing 2021;30:7241–7255.

[9] Sun, Y, Zhang, S, Wang, T, Lou, F, Ma, J, Wang, C, et al. An improved spatial point cloud simplification algorithm. Neural Computing and Applications 2021;:1–15.

[10] Goel, S, Kanazawa, A, Malik, J. Shape and viewpoint without keypoints. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16. Springer; 2020, p. 88–104.

[11] Mitchell, JC. Sampling rotation groups by successive orthogonal images. SIAM Journal on Scientific Computing 2008;30(1):525–547.

Fig. 5. Comparisons with point-based algorithms. Each row represents the saliency visualization of our algorithm, the algorithm by Ding et al. [14], the algorithm by Guo et al. [13], the algorithm by Tasse et al. [12], and the ground truth provided by Chen et al. [15], respectively. From left to right, the models shown are egea, kettle, chair, octopus, human, vase, pig, teddy, and hand. In each model, the areas that are closer to red indicate higher saliency.

[12] Tasse, FP, Kosinka, J, Dodgson, N. Cluster-based point set saliency. In: Proceedings of the IEEE international conference on computer vision. 2015, p. 163–171.

[13] Guo, Y, Wang, F, Xin, J. Point-wise saliency detection on 3d point clouds via covariance descriptors. The Visual Computer 2018;34:1325–1338.

[14] Ding, X, Lin, W, Chen, Z, Zhang, X. Point cloud saliency detection by local and global feature fusion. IEEE Transactions on Image Processing 2019;28(11):5379–5393.

[15] Chen, X, Saparov, A, Pang, B, Funkhouser, T. Schelling points on 3d surface meshes. ACM Transactions on Graphics (TOG) 2012;31(4):1–12.

[16] Lavoué, G, Cordier, F, Seo, H, Larabi, MC. Visual attention for rendered 3d shapes. In: Computer Graphics Forum; vol. 37. Wiley Online Library; 2018, p. 191–203.

[17] Grill-Spector, K, Kanwisher, N. Visual recognition: As soon as you know it is there, you know what it is. Psychological Science 2005;16(2):152–160.

[18] Treue, S. Visual attention: the where, what, how and why of saliency. Current Opinion in Neurobiology 2003;13(4):428–432. URL: https://www.sciencedirect.com/science/article/pii/S0959438803001053. doi:https://doi.org/10.1016/S0959-4388(03)00105-3.

[19] Deng, L, Wang, Y, Liu, B, Liu, W, Qi, Y. Biological modeling of human visual system for object recognition using glop filters and sparse coding on multi-manifolds. Machine Vision and Applications 2018;29:965–977.

[20] Devi, K, Gomathi, R. Brain tumour classification using saliency driven nonlinear diffusion and deep learning with convolutional neural networks (cnn). Journal of Ambient Intelligence and Humanized Computing 2021;12. doi:10.1007/s12652-020-02200-x.

[21] Song, R, Liu, Y, Rosin, PL. Mesh saliency via weakly supervised classification-for-saliency cnn. IEEE transactions on visualization and computer graphics 2019;27(1):151–164.

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

[23] Qi, CR, Yi, L, Su, H, Guibas, LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems 2017;30.

[24] Guo, Y, Wang, H, Hu, Q, Liu, H, Liu, L, Bennamoun, M. Deep learning for 3d point clouds: A survey. IEEE transactions on pattern analysis and machine intelligence 2020;43(12):4338–4364.

[25] Atmosukarto, I, Shapiro, LG. A salient-point signature for 3d object retrieval. In: Proceedings of the 1st ACM international conference on Multimedia information retrieval. 2008, p. 208–215.

[26] Shu, Z, Yang, S, Xin, S, Pang, C, Jin, X, Kavan, L, et al. Detecting 3d points of interest using projective neural networks. IEEE Transactions on Multimedia 2021;24:1637–1650.

[27] Zheng, T, Chen, C, Yuan, J, Li, B, Ren, K. Pointcloud saliency maps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, p. 1598–1606.

[28] Horache, S, Deschaud, JE, Goulette, F. 3d point cloud registration with multi-scale architecture and unsupervised transfer learning. In: 2021 International Conference on 3D Vision (3DV). IEEE; 2021, p. 1351–1361.

[29] Xiao, A, Huang, J, Guan, D, Zhan, F, Lu, S. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence; vol. 36. 2022, p. 2795–2803.

[30] Zhang, Z, Da, F, Yu, Y. Learning directly from synthetic point clouds for "in-the-wild" 3d face recognition. Pattern Recognit 2022;123:108394. URL: https://doi.org/10.1016/j.patcog.2021.108394. doi:10.1016/j.patcog.2021.108394.

[31] Zhang, R, Guo, Z, Zhang, W, Li, K, Miao, X, Cui, B, et al. Pointclip: Point cloud understanding by CLIP. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. IEEE; 2022, p. 8542–

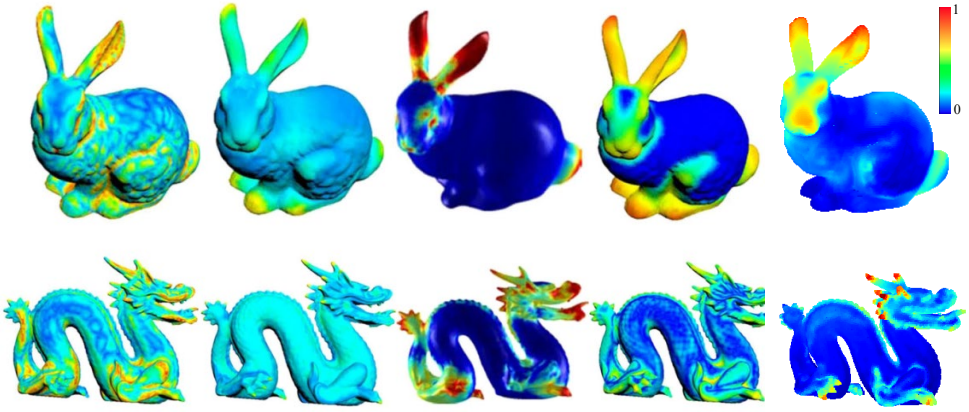

Fig. 6. Comparisons with mesh-based algorithms. The upper row shows the bunny model, while the bottom row shows the dragon model. From left to right, each column displays the saliency detection results of the algorithm by Wu et al. [37], the algorithm by Leifman et al. [38], the algorithm by Tao et al. [39], and our proposed algorithm.

8552. URL: https://doi.org/10.1109/CVPR52688.2022.00836. doi:10.1109/CVPR52688.2022.00836.

[32] Koch, C, Poggio, T. Predicting the visual world: silence is golden. nature neuroscience 1999;2(1):9–10.

[33] Wolfe, JM. Guided search 2.0 a revised model of visual search. Psychonomic bulletin & review 1994;1:202–238.

[34] Lee, CH, Varshney, A, Jacobs, DW. Mesh saliency. In: ACM SIGGRAPH 2005 Papers. 2005, p. 659–666.

[35] Tasse, FP, Kosinka, J, Dodgson, NA. Quantitative analysis of saliency models. In: SIGGRAPH ASIA 2016 Technical Briefs. 2016, p. 1–4.

[36] Qi, CR, Su, H, Mo, K, Guibas, LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–660.

[37] Wu, J, Shen, X, Zhu, W, Liu, L. Mesh saliency with global rarity. Graphical Models 2013;75(5):255–264.

[38] Leifman, G, Shtrom, E, Tal, A. Surface regions of interest for viewpoint selection. IEEE transactions on pattern analysis and machine intelligence 2016;38(12):2544–2556.

[39] Tao, P, Cao, J, Li, S, Liu, X, Liu, L. Mesh saliency via ranking unsalient patches in a descriptor space. Computers & Graphics 2015;46:264–274.

Fig. 7. Gallery of saliency maps generated by our algorithm compared to the pseudo ground truth [15]. The top half shows saliency detection results using our proposed algorithm, while the bottom half shows saliency detection results using the pseudo-ground truth.