Pacific Graphics 2024
R. Chen, T. Ritschel, and E. Whiting
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 7

LightUrban: Similarity Based Fine-grained Instancing for Lightweighting Complex Urban Point Clouds

Z.A. Lu[1], W.D. Xiong[2], P. Ren[1], and J.Y. Jia[3,4†]

[1]Tongji University, China [2]Shenzhen University, China [3]Jilin Animation Institute, China
[4]The Hong Kong University of Science and Technology (Guangzhou), China

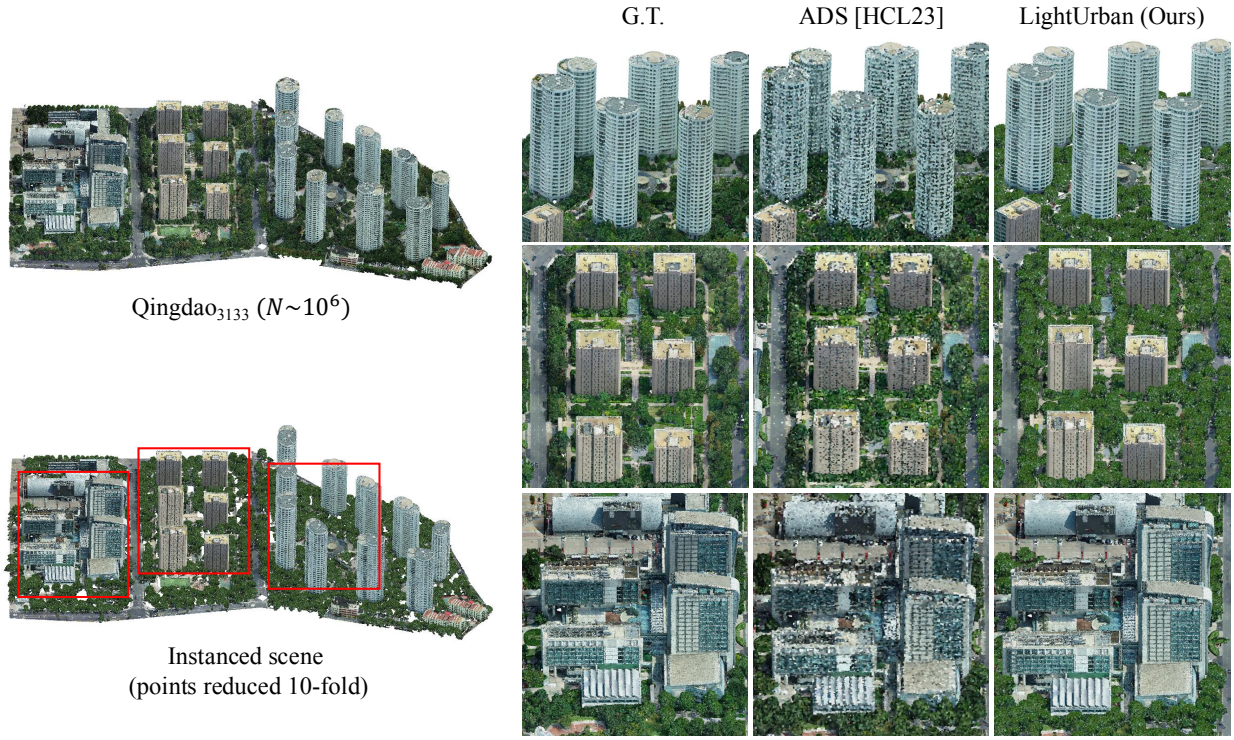


Figure 1: *Visual comparisons on the original urban scene and the lightweighted scene with our instancing method called* LightUrban.

Abstract

Large-scale urban point clouds play a vital role in various applications, while rendering and transmitting such data remains challenging due to its large volume, complicated structures, and significant redundancy. In this paper, we present LightUrban*, the first point cloud instancing framework for efficient rendering and transmission of fine-grained complex urban scenes. We first introduce a segmentation method to organize the point clouds into individual buildings and vegetation instances from coarse to fine. Next, we propose an unsupervised similarity detection approach to accurately group instances with similar shapes. Furthermore, a fast pose and size estimation component is applied to calculate the transformations between the representative instance and the corresponding similar instances in each group. By replacing individual instances with their group's representative instances, the data volume and redundancy can be dramatically reduced. Experimental results on large-scale urban scenes demonstrate the effectiveness of our algorithm. To sum up, our method not only structures the urban point clouds but also significantly reduces data volume and redundancy, filling the gap in lightweighting urban landscapes through instancing.*

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis; Point-based models;***

† Corresponding author: csjyjia@aliyun.com

1. Introduction

Urban point clouds are crucial in applications including virtual reality, digital twins, Web3D, etc. It is common to find such data with large data volumes, complex structures and considerable redundancy. These properties pose great challenges to online transmission and rendering.

For transmitting large-scale point clouds, a strategy is to compress the point cloud on the sending device and decompress it on the receiving end. While this technique can facilitate data transmission, the compressing and decompressing processes are time-consuming and do not address the issues during rendering. The lightweight rendering of large-scale urban point clouds remains a significant challenge.

To efficiently render large-scale point clouds in a reasonable time, a straightforward method is to down-sample the point clouds, therefore reducing the data volume for both transmission and rendering [WJC*21, XLM*22, HCL23]. However, such methods inevitably lead to accuracy loss in geometric shapes.

The urban point clouds mainly consist of objects in the category of vegetation, buildings, vehicles, etc. Commonly, the major components of urban point clouds are buildings and vegetation objects, which account for about 90% of the points. However, these objects often share similar visual characteristics. To achieve efficient and realistic rendering for such scenes, traditional mesh instancing techniques render multiple copies of an identical mesh object with varying attributes [Ceb04]. Inspired by the heuristics of mesh instancing, we here introduce *LightUrban*, a point cloud lightweighting method to simultaneously tackle the challenges in rendering and transmission mentioned above.

Our method mainly has two stages: instance segmentation and scene instancing. In the first stage, we perform instance segmentation from coarse to fine, obtaining fine-grained objects of buildings and vegetation. In the instancing stage, we divide the structured instances into similar instance groups based on their geometric features. For each group, an instance will be selected as its representative instance. Next, transformations of all the instances are computed according to their representative instance. As a result, the original scene is lightweighted into an instanced scene with compact data representation, semantic structure and low redundancy.

Our contributions are summarized as follows:

- The first point cloud instancing framework for effective rendering and transmission of large-scale urban scenes.
- A coarse-to-fine segmentation module to extract both fine-grained buildings and vegetation. For buildings, we introduce a novel top-down sub-building segmentation method that leverages the consistency of building components from top to bottom. For vegetation, we present a distance metric for better segmenting connected trees is encoded into a local maximum filter.
- An instancing module includes: a cost-effective and non-parametric encoder for unsupervised geometric feature extraction, a feature matching criterion for similarity detection, a graph-based feature grouping method for automatic instances clustering, and a fast pose and size estimation method for instances.

2. Related Work

2.1. Point cloud compression

Compression methods aim at reducing the data volume for feasible transmission. Traditional point cloud compression techniques like feature squeezing [HGW24] and conditional entropy coding [GZLX23] have been employed to enhance compression efficiency, gradually forming point cloud compression frameworks like G-PCC [ZCDM23]. With the development of implicit fields, recent neural scene representation approaches [RSY*24] employ networks trained to minimize both geometry and attribute distortion, surpassing G-PCC in some scenarios. Compared to compression methods, we focus on lightweighting point clouds which can benefit both transmission and rendering tasks.

2.2. Point cloud learning for instancing

Large-scale instance segmentation. 3D instance segmentation builds upon semantic segmentation, requiring the prediction of not only object classes but also different instances of the same class. Nowadays, large-scale point cloud datasets such as Semantic3D [HSL*17], STPLS3D [CHY*22], and UrbanBIS [YXZ*23] are increasingly becoming the mainstream data format for 3D segmentation tasks. Accordingly, numerous methods suitable for instance segmentation of large-scale point clouds have emerged. PointGroup [JZS*20] clusters object instances by considering two different sets of point coordinates simultaneously. HAIS [CFZ*21] utilizes spatial relationships between points and point sets for instance segmentation. However, the instances generated by existing networks are not yet sufficient for instancing. There are two reasons: i) existing networks focus on instance segmentation of buildings, performing poorly on other urban objects such as vegetation. ii) the granularity of segmented buildings is relatively coarse, with many connected buildings that could be further divided into sub-buildings. Hence, we propose a fine-grained instance segmentation module, focusing on fine-grained segmentation on connected buildings and individual tree segmentation.

Geometric feature extraction. Handcrafted 3D descriptors, like FPFH [RBB09], 3DSC [KPNK03], and SHOT [STDS14], are used to describe local features, while global descriptors often being extensions of local ones, such as VFH [RBTH10], which builds on FPFH by splitting the feature histogram into viewpoint direction components. In recent years, learning-based methods like PointNet [QSMG17], PointNet++ [QYSG17], and PointCNN [LBS*18] have emerged, enabling end-to-end feature learning without the need for manually designed descriptors. These methods, however, are supervised and require significant training costs. As a result, semi-supervised or unsupervised approaches, such as PointMAE [PWT*22] and PointContrast [XGG*20], have gained attention for their efficiency, leveraging techniques like auto-encoding and contrastive learning to handle large-scale point clouds more effectively.

LightUrban reduces redundancy by instancing objects with similar geometric features, a process that involves an unsupervised geometric extraction method inspired by PointNN [ZWG*23]. It requires no training cost and is efficient for large-scale, diverse, and complex urban landscapes.

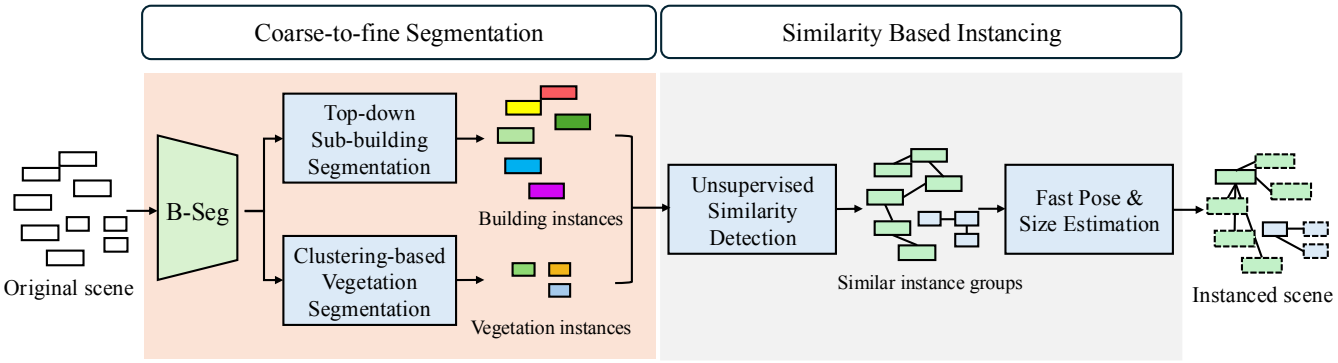


Figure 2: *The overview of our framework,* LightUrban. *Firstly, the input scene is segmented into building and vegetation instances from coarse to fine. Next, these instances are partitioned into groups based on the similarity of global features. The scene is instanced by replacing each group with a representative instance.*

Point cloud retrieval. Point cloud retrieval methods can be classified into two categories: retrieval based on projection views and 3D features. For retrieval based on projection views, 3D shapes are represented as a set of 2D projections, each described by image descriptors. Although this method can yield good results, a large number of views may reduce retrieval efficiency. For retrieval based on 3D features, the most commonly used 3D features in existing work are 3D descriptors. Shape similarities are measured using various geometric descriptors, including shape topology [BGSF08, CLCL11], shape distribution [ABDBP07, ASYS09], and spherical harmonics [FMK*03, MDTS09]. With the development of neural networks, learning-based 3D features have shown higher ability in the representation of complex data. Considering the complicated structure of urban point clouds, LightUrban learns the embedded vectors of instances through the geometric feature extraction module and retrieves the most similar instances using the dot products of their embeddings.

Point cloud pose estimation. Point cloud 6D pose includes its 3D position and 3D orientation. Pose estimation involves computing the transformation matrix between two point clouds or between a point cloud and a standard coordinate system, which can be categorized into correspondence-based method, template-based method and voting-based method. Correspondence-based methods use traditional descriptors [KPNK03] or deep learning [ZSN*17] to find 3D-to-3D correspondences. Template-based methods find correspondences between the current input and existing templates with known 6D poses [SLG*19, AGSL19]. Voting-based methods indirectly vote for key points [HSH*20] or directly vote for 6D poses from each 3D point [WXZ*19]. LightUrban supports both 6D pose and 3D size estimation, making it more suitable than existing methods for computing transformation matrices between various instances.

3. Overview

Instancing urban point clouds is a nontrivial task. First, segmenting fine-grained instances from large-scale, complex scenes demands higher accuracy compared to existing instance segmentation tasks. Second, simple and efficient feature extraction methods are urgently needed to divide numerous instances with diverse geometries into groups of similar shapes. Finally, feature matching and grouping methods should ensure similar instances are divided into the same group, followed by fast pose and size estimation for instanced rendering.

Our input is a large-scale urban point cloud $\mathcal{P}$, consisting of a set of unorganized points $\{p_i\}$. Our target is to lightweight $\mathcal{P}$ and obtain an instanced point cloud $\mathcal{P}^*$. The overview of our framework is illustrated in Figure 2.

In the segmentation stage, we extract instances in $\mathcal{P}$ from coarse to fine. At first, $\mathcal{P}$ is sent to a segmentation network called B-Seg [YXZ*23], obtaining coarse-grained instances with different semantics $\{\mathcal{P}_c\}$, each $\mathcal{P}_c$ is an instance made up of points. However, such coarse-grained instances could be inaccurate, i.e., each building instance can consist of several individual sub-buildings. Hence, we proposed a sub-building segmentation algorithm (Section 4.1) and a vegetation segmentation algorithm (Section 4.2) significantly refine $\{\mathcal{P}_c\}$ in parallel, receiving fine-grained instances $\{\mathcal{P}_f\}$.

In the instancing stage, $\{\mathcal{P}_f\}$ is first sent to an unsupervised similarity detection module. We present a non-parametric encoder to learn the global geometry features f_G of each instance $\mathcal{P}_f$ (Section 5.1). Next, those instances with similar features will be organized into an instance group (Section 5.2). A representative instance will be selected from each group. Subsequently, given the representative instance of a group, the fast pose and size estimation module is applied to each instance in this group to estimate its local coordinate system, pose and size. These parameters are then employed to compute the transformation matrix (Section 5.3).

Finally, all the transformation matrices together with the corresponding representative instances are collected for the lightweight instancing of large-scale urban point clouds.

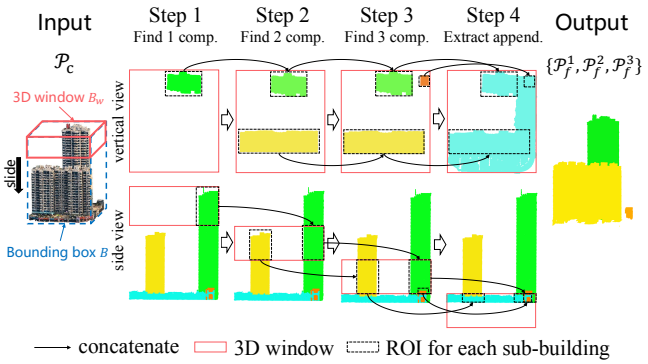


Figure 3: *Top-down sub-building segmentation. For the input connected building point cloud $\mathcal{P}_c$, the process of separating sub-buildings is observed from both side and vertical views. Finally, we obtain three sub-buildings, denoted as $\mathcal{P}_f^1$, $\mathcal{P}_f^2$, $\mathcal{P}_f^3$.*

4. Coarse-to-fine Instance Segmentation

For buildings, we proposed a top-down sub-building segmentation module for fine-grained segmentation, as a coarse-grained building may consist of multiple sub-buildings. For vegetation, each point cloud sequentially undergoes clustering-based segmentation to obtain fine-grained individual trees.

4.1. Top-down sub-building segmentation

Given a coarse-grained building point cloud $\mathcal{P}_c$ which may have multiple sub-buildings, we first compute its oriented bounding box (OBB), denoted as B. Next, we define a 3D sliding window B_w, whose length and width are aligned with the OBB. The height of B_w is adjusted according to the height of B with a predefined ratio r_w. Our algorithm slides B_w from the top of B along the vertical direction of OBB with step size h_w. At each step, points inside B_w is partitioned into several connected components, with each component corresponding to a part of each sub-building. Between steps, adjacent connected components are merged. By the time B_w reaches the base of B, only one connected component should remain. Using the regions of interest previously identified for each sub-building, points are then allocated to their respective sub-buildings, thereby completing the top-down sub-building segmentation.

Figure 3 illustrates the process of detecting sub-buildings from a coarse building point cloud with our segmentation algorithm. In the first iteration, the sliding window B_w is placed at the height of B. Only the top part of the sub-building visualized in green is within the window. We consider all these points as a single connected component (comp.). After sliding the B_w with step size h_w, there appears two disconnected components inside B_w. In iteration three, we can see three components visualized in green, yellow, and orange respectively. This process continues until B_w reaches the bottom of the building and all points inside B_w reconnect. Prior to this, we recorded the region of interest (black dashed lines) for each connected component. By obtaining the points within the region of interest for each connected component, the last division is completed. Finally, concatenate all connected components with adjacent regions of interest to extract appendages (append.) belonging to the same sub-building. Please refer to the supplemental for the pseudocode.

4.2. Clustering-based vegetation segmentation

Compared to buildings, vegetation point clouds are denser and the size of trees is smaller. The boundary between individual trees is fuzzy. Moreover, there exists significant noise in such point clouds. Existing instance segmentation networks often fail in segmenting vegetation point clouds. To address this issue, We propose a two-stage method to achieve fine-grained segmentation of individual trees, containing density-based clustering and local maximum segmentation.

Density-based clustering. We use a density-based clustering similar to DBSCAN [KRA*14], which partitions a set of points that are sufficiently close to each other into a cluster. Benefiting from the idea of DBSCAN classifying points as core points, boundary points and noise points, we can easily achieve point cloud denoising while retaining meaningful vegetation points. After the density-based clustering, we obtain several point cloud clusters, each cluster containing multiple closely adjacent trees.

Local maximum segmentation. We find that each individual tree can be approximately regarded as a convex hull, which means it has a unique highest point. Based on this conclusion, we make use of a local-maxima filter [WNG00] for individual tree detection, with each local maximum point corresponding to an individual tree. For individual tree segmentation, it is necessary to use the local maximum points as centroids and gather points around them. However, we found that traditional clustering algorithms, which use Euclidean distance metrics, are not suitable for tree segmentation. For example, when a short tree is adjacent to a tall one, the points near the ground in the tall tree are generally closer in Euclidean distance to the crown of the short tree. This leads to these points, which should belong to the tall tree, being erroneously assigned to the short tree. Therefore, we design a distance metric that is applicable to vegetation. For any point $p_i = (x_i, y_i, z_i)$ and a local maximum point $p_m = (x_m, y_m, z_m)$, our distance metric is defined as:

$$\mathrm{dist}(p_i, p_m) = \sqrt{\frac{(x_i - x_m)^2 + (y_i - y_m)^2}{(z_i \cdot z_m)^k}}, \quad k \geq 0, \tag{1}$$

where k defines the degree to which distance is influenced by vertical height. Generally, the greater the height difference between adjacent trees, the larger the value of k should be. Specifically, this distance will degrade to the Euclidean distance when $k = 0$. According to this distance metric, each point can find the nearest local maximum point, and the set of points around each local maximum point represents an individual tree.

5. Similarity Based Instancing

Similarity based instancing encompasses two main parts: unsupervised similarity detection and fast pose and size estimation. The unsupervised similarity detection further consists of non-parametric encoder as well as feature matching and grouping.

5.1. Non-parametric encoder

Referring to the PointNN [ZWG*23], we design an encoder structure to extract the global geometric feature of a given point cloud as shown in Figure 4.

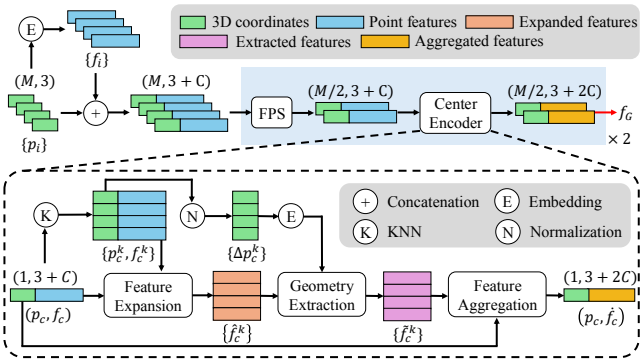


Figure 4: *The architecture of our non-parametric encoder.*

Point embedding. In large-scale urban point clouds, each point may only contain positional information. We refer to the word embedding in Transformer [VSP*17] and extend them to 3D point clouds to capture fine-grained semantic information of local 3D structures. For a point $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, expand each dimension of it to d dimensions. Define the function $\mathrm{SE} : \mathbb{R}^1 \to \mathbb{R}^d$. Taking the first dimension x_i as an example, for $j \in [0, d]$ and $k \in \mathbb{Z}$:

$$\mathrm{SE}(x_i)^{(j)} = \begin{cases} \sin(\omega_k \cdot x_i), & \text{if } j = 2k \\ \cos(\omega_k \cdot x_i), & \text{if } j = 2k+1 \end{cases}, \quad \omega_k = \alpha\beta^{-2k/d}, \tag{2}$$

where α and β control the magnitude and wavelengths, respectively. Let $C = 3d$ as the dimension of the embedded coordinate, the embedded vector of p_i is

$$\mathrm{Embed}(p_i) = \mathrm{Concat}(\mathrm{SE}(x_i), \mathrm{SE}(y_i), \mathrm{SE}(z_i)) \in \mathbb{R}^C. \tag{3}$$

By leveraging the inherent properties of trigonometric functions, the relative position between two points can be described through the dot product of their embeddings, thereby expanding the semantic features of the points.

Center encoding. After embedding the point cloud, farthest point sampling (FPS) [MD03] is conducted to reduce the quantity to half of the original. The sampled points are regarded as center points capable of aggregating local features of their neighborhoods. In the center encoder, for a given center point (p_c, f_c), k-nearest neighbors algorithm (kNN) [Pet09] is first applied to obtain its surrounding neighbor points $\{p_c^k, f_c^k\}$, which are used to expand its geometric features. Let $\mathbb{K}$ be the set of indices for all neighbor points, the expanded features of the center point are defined as:

$$\hat{f}_c^k = \mathrm{Concat}(f_c, f_c^k), \quad k \in \mathbb{K}. \tag{4}$$

Notice that the dimensions of f_c and f_c^k are different, so when concatenating, the broadcast mechanism is used to expand the dimension of f_c.

To describe the spatial distribution of K neighbors within a local neighborhood, we weigh $\hat{f}_c^k$ using relative positional encoding. The coordinates of K neighbors are normalized and denoted as $\{\Delta p_c^k\}$. After embedding via Equation 3, its dimensions remain consistent with $\hat{f}_c^k$. Inside geometry extraction, the expanded features are weighed as:

$$\tilde{f}_c^k = (\hat{f}_c^k + \mathrm{Embed}(\Delta p_c^k)) \odot \mathrm{Embed}(\Delta p_c^k), \tag{5}$$

where $\odot$ denotes element-wise multiplication.

After weighing, both max pooling and average pooling are used to aggregate the features of neighboring points, enlarging the receptive field of the center point and obtaining richer local features $\dot{f}_c$. Inside feature aggregation,

$$\dot{f}_c = \mathrm{MaxP}(\{\tilde{f}_c^k\}) + \mathrm{AvgP}(\{\tilde{f}_c^k\}), \tag{6}$$

where $\mathrm{MaxP}(\cdot)$, $\mathrm{AvgP}(\cdot)$ are max pooling and average pooling, respectively. The above pooling operations are permutation-invariant, making them suitable for unordered point clouds.

Multi-stage hierarchy. The multi-stage hierarchy consists of multiple FPS and Center Encoder components (the light blue area in Figure 4). After the point cloud passes through FPS and Center Encoder once, the number of points is halved and the features are doubled. The output of the previous stage serves as the input for the next stage, forming a cyclic structure. In our architecture, this cycle repeats twice, and the final output dimension of the last Center Encoder is $(M/4, 3+4C)$. To obtain global features f_G representing the original point cloud, the global pooling operation is applied,

$$f_G = \mathrm{MaxP}(\{\dot{f}_c\}) + \mathrm{AvgP}(\{\dot{f}_c\}). \tag{7}$$

The red arrow in Figure 4 represents the global pooling. Finally, the original point cloud obtains its global features through our non-parametric encoder.

5.2. Feature matching and grouping

For several global features of individual building point clouds, we need to divide sufficiently similar features into groups as instances. As mentioned earlier, measuring the similarity between two features can be achieved by computing the dot product of their corresponding vectors. Thus, for a query feature, it is straightforward to calculate its similarity with every other feature. However, a problem arises: how do we determine the top T features that are sufficiently similar to the query feature so that they can be divided into the same group? For an unseen point cloud, the value of T is unknown. To address this issue, we first propose a criterion for evaluating whether features are sufficiently similar, which serves as the core of feature matching. Then, we introduce a feature grouping method based on connected graphs and disjoint-set, achieving the grouping of similar features.

Figure 5 illustrates the overall workflow of the feature matching and grouping module. Each global feature vector of the building is treated as a node. After feature matching, there will be a connected edge between pairs of matched vectors, thus forming several connected graphs. The purpose of feature grouping is to obtain all nodes within the same connected graph with minimal time and

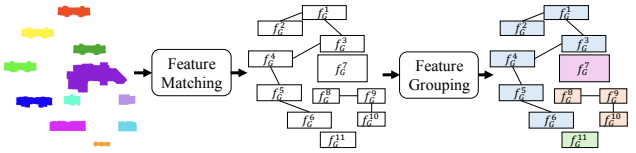


Figure 5: *The workflow of feature matching and grouping.*

space costs and select a representative node as the instanced object. Please refer to the supplemental for the flowchart.

Feature matching. The essence of feature matching is to establish an edge between two nodes that satisfy the matching criteria. At first, a similarity ranking table for all feature vectors should be calculated. For W global feature vectors, denoted as $\mathbf{F}_G \in \mathbb{R}^{1\times W}$, the similarity ranking table is calculated as follows:

$$\mathbf{M}_{similar} = \text{Argsort}(\mathbf{F}_G^{\top} \otimes \mathbf{F}_G), \tag{8}$$

where $\otimes$ denotes matrix multiplication and $\text{Argsort}(\cdot)$ returns the indices that sort the matrix by row. The i-th row of $\mathbf{M}_{similar}$ represents the ordered result list of f_G^i.

Our feature matching algorithm is as follows: for each feature vector f_G^i, get its ordered result list and select f_G^j that is most similar to it. Conversely for f_G^j, get its ordered result list and check if the position of f_G^i is within the predefined threshold. If so, f_G^i and f_G^j are deemed bidirectionally similar, leading a successful match; otherwise, they are unidirectionally similar, leading a mismatch. The threshold comes between 0 to 1, and a smaller threshold indicates a higher requirement for similarity. Figure 6 (left) illustrates two cases of successful match and mismatch with a threshold of 0.5. In case (a), f_G^1's ordered result list has f_G^3 as its first feature. Conversely, when f_G^3 is treated as the query feature, f_G^1 appears second in its ordered result list, which meets our criterion. In case (b), f_G^7's ordered result list has f_G^5 as its first feature. But f_G^7 appears last in f_G^5 ordered result list, resulting in a mismatch.

Feature grouping. The essence of feature grouping is to obtain all nodes belonging to the same connected graph. A simple method is to start from any node and traverse all connected nodes using depth-first search. For minimizing time and space costs, we implemented feature grouping using a disjoint-set. Figure 6 (c) illustrates the construction process of the disjoint-set according to the order of adding from f_G^1 to f_G^7. The solid arrows indicate that the first feature in the ordered result list acts as a child node, pointing to the parent node represented by the query feature. The dashed arrows indicate that when the parent node has another higher-level parent node, the child node directly points to the highest-level parent node. Therefore, the actual pointing relationship should be the newly added solid arrows rather than dashed arrows. Figure 6 (d) illustrates the final structure of the disjoint-set, where all nodes are colored the same, representing a similar instance group. The object corresponding to f_G^2, as the highest-level parent node (root node), is designated as the representative instance.

Through feature grouping, a representative instance can be selected from a large number of similar instances. By applying specific transformations to the representative instance, it can replace all other similar instances. In other words, this representative instance establishes a mapping relationship with the other instances. This integrates the independent and unrelated objects into an organized form, representing the complex scene with a clear graph structure.

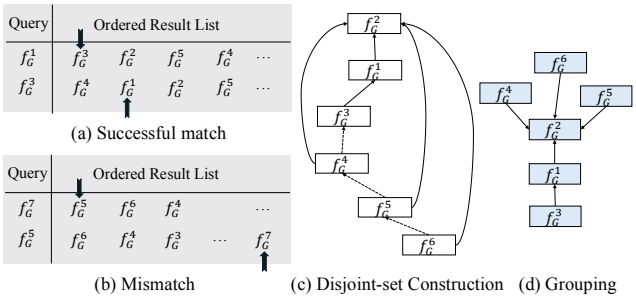


Figure 6: *(a) f_G^1 and f_G^3 exhibit bidirectional similarity, resulting in a successful match. (b) f_G^5 and f_G^7 only exhibit unidirectional similarity, leading to a mismatch. (c) A disjoint-set is constructed based on the matching relationships between features. (d) The root node is identified and the corresponding object is designated as the representative instance.*

5.3. Pose and size estimation

Each building or tree has nine degrees of freedom, including position, orientation and size. The position and orientation can be collectively referred to as the pose. Inspired by 6D pose estimation [HFZL20] in SLAM, we propose a fast pose and size estimation method applicable to building and vegetation instances, estimating the orientation, position and size from the local coordinate system, and combining them into the transformation matrices.

Local coordinate system estimation. For the orientation, we denote the local coordinate system as **X**, **Y**, and **Z**. Since most urban point cloud objects lie flat on horizontal ground, **Z** is defined as $(0,0,1)$, pointing vertically upward. **X** and **Y** are calculated by projecting all points onto the horizontal plane and finding orthogonal bases that maintain rotational invariance. For the size, the object's lengths along the three axes are calculated to ensure consistency across different orientations. The position is determined by the geometric center of the object.

We employ PCA [MR93] to project points onto orthogonal axes, maximizing variance and ensuring zero covariance. The axes produced by PCA can be used as **X** and **Y**. The object's position and size can then be accurately calculated by rotating it to align with the standard coordinate system.

Transformation matrices calculation. To comply with instancing specifications, instead of directly computing transformations between representative and similar instances, we first standardize the representative instances and then calculate the transformations from these standardized instances to the similar ones. i) Standard-

ization. For an representative object $\mathcal{P}_a$, denote its orientation (rotation) as $\mathbf{R}_{\mathcal{P}_a} \in \mathbb{R}^{3\times3}$, its size as $\mathbf{S}_{\mathcal{P}_a} \in \mathbb{R}^{3\times1}$, and its translation (position) as $\mathbf{T}_{\mathcal{P}_a} \in \mathbb{R}^{3\times1}$ in Cartesian space. In homogeneous space, the transformation matrix for standardizing is:

$$\mathbf{M}_{std} = \begin{bmatrix} \mathbf{E} & -\mathbf{T}_{\mathcal{P}_a} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{P}_a}^{\top} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathrm{diag}(1/\mathbf{S}_{\mathcal{P}_a}) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \quad (9)$$

where $\mathbf{E}$ denotes the identity matrix, $\mathbf{0}$ denotes the zero matrix, and $\mathrm{diag}(\cdot)$ denotes the function that converts a column vector into the corresponding diagonal matrix. ii) Instancing. For a similar instance $\mathcal{P}_b$, the transformation matrix for instancing in homogeneous space is:

$$\mathbf{M}_{inst} = \begin{bmatrix} \mathrm{diag}(\mathbf{S}_{\mathcal{P}_b}) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{P}_b} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{E} & \mathbf{T}_{\mathcal{P}_b} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (10)$$

Denote $\mathrm{H}(\cdot)$ as the function for converting points from Cartesian space to homogeneous space, and $\mathrm{C}(\cdot)$ as the function for the reverse process. $\mathcal{P}_b$ can be replaced with $\mathcal{P}_a$ transformed by two matrices:

$$\mathcal{P}_b = \mathrm{C}(\mathbf{M}_{inst}\mathbf{M}_{std}\mathrm{H}(\mathcal{P}_a)). \quad (11)$$

As a result, all similar instances in a large-scale scene can be derived from one representative instance using the calculated transformations. This allows each instance, originally represented by tens of thousands of points, to be replaced by a single 4×4 transformation matrix, significantly reducing the point count.

6. Experiments

We report the performance on the UrbanBIS benchmark, which is a large-scale point cloud dataset designed for practical urban-level semantic and building-level instance segmentation. Particularly, it provides semantic-level annotations on a rich set of urban objects, including buildings, vegetation, vehicles, etc. Buildings and vegetation occupy approximately 90% of the point clouds. We conduct experiments over a vast area covering 10.78 square kilometers, containing 2.5 billion points.

6.1. Fine-grained instance segmentation

The number of instances. We first compare the number of instances of buildings and vegetation in Qingdao, Wuhu, and Longhua using different segmentation methods. Among these methods, PointGroup, HAIS, SoftGroup, and B-Seg are supervised methods that build learnable neural networks. ITS-dalponte2016 (ITS) [DC16] is an unsupervised method to be used in segmenting trees. As shown in Table 1, supervised methods exhibit significantly better segmentation capabilities for buildings than for vegetation, and the results of different supervised methods are quite similar across the three scenes. This is because the UrbanBIS dataset provides more semantic and instance labels for building point clouds, while vegetation point clouds have relatively fewer labels, making learnable segmentation methods more effective for buildings than vegetation. Moreover, since the scenes contain plenty of closely connected buildings and vegetation, supervised methods face limitations when dealing with more fine-grained segmentation tasks. ITS-dalponte2016, which is specifically designed for individual tree segmentation, lacks the ability to segment buildings but can more effectively segment closely connected vegetation. Our method goes further by enhancing the existing instance segmentation networks, providing fine-grained segmentation capabilities for both buildings and vegetation, thus obtaining more instances of individual buildings and vegetation.

Table 1: *The numbers of buildings and vegetation (Veg) in Qingdao, Wuhu and Longhua processed by different methods.*

Method	Qingdao		Wuhu		Longhua	
	Building	Veg	Building	Veg	Building	Veg
PointGroup	648	49	1055	50	988	47
HAIS	651	47	1076	48	994	46
SoftGroup	653	50	1086	49	994	47
B-Seg	658	52	1101	52	996	48
ITS	-	9453	-	11526	-	13353
Ours	963	9464	1496	11492	1394	13344

Table 2: *The performance of different individual segmentation methods for buildings and vegetation.*

Category	Method	MA	MP	MR	mIoU	T(s)
Building	K-means@50	0.637	0.757	0.403	0.357	13.57
	RANSAC	0.645	0.807	0.436	0.451	17.93
	Ours	0.994	0.984	0.994	0.978	21.97
Vegetation	ITS	-	0.851	0.802	0.714	106.60
	PyCrown	-	0.876	0.857	0.764	45.51
	Ours	-	0.887	0.837	0.757	37.34

Segmentation performance. Although we designed the unified fine-grained segmentation process for both connected buildings and vegetation, the specific methods for the two are different. Therefore, we select a substantial amount of connected buildings and vegetation in Qingdao, Wuhu, and Longhua, conducting experiments on buildings and vegetation, as shown in Table 2.

For connected buildings, we implement another two segmentation algorithms using K-means [M*67] and RANSAC [FB81] as comparisons to our method. K-means@50 refers to performing K-means clustering after downsampling the original building point cloud by 50%. The RANSAC algorithm is based on plane extraction and is adapted to extract and merge adjacent building planes for segmenting sub-buildings. We evaluate performance using mean accuracy (MA), mean precision (MP), mean recall (MR), and mean intersection over union (mIoU). From the results, it can be seen that K-means@50 has the worst performance due to its significant error in estimating the number of sub-buildings, as it cannot know the number of clusters in advance. Although RANSAC can accurately segment planes, it struggles to effectively merge these planes into sub-buildings. By comparison, our methods can precisely divide different sub-buildings using a top-down sliding window, significantly improving segmentation performance.

For vegetation, we calculate evaluation metrics on each segmented individual tree. Since a single tree represents a small portion of the entire vegetation, the accuracy error is large and thus not calculated. Our method shows improvements in both time and performance compared to the ITS-dalponte2016, and it takes sig-

Table 3: *The results of obtaining similar point clouds for a given query using different methods.*

Method	Recall@10	Recall@15	Params
FPFH	0.66	0.70	0
SHOT	0.67	0.72	0
PointNet	0.93	0.96	>8M
Ours	0.92	0.95	0

Table 4: *The comparison results of our feature matching and grouping module with HDBSCAN.*

Method	Grouping efficiency	Grouping completeness
HDBSCAN	0.409	0.798
Ours	0.962	0.976

nificantly less time than PyCrown [ZDSJ18] with comparable performance.

Moreover, our distance metric is suitable for the segmentation of other urban landscapes, such as boats and vehicles. Please refer to the supplemental for more results.

6.2. Unsupervised similarity detection

Non-parametric encoder. We compare the non-parametric encoder module with unsupervised methods such as FPFH, SHOT, and the supervised method PointNet. We select scene blocks from UrbanBIS that contain similar buildings and vegetation, manually removing irrelevant point clouds and ensuring there are 10 similar buildings or vegetation samples. We define recall@10 and recall@15, representing the recall rate of the top 10/15 query results for a given query point cloud. Additionally, we compare the number of parameters and execution time for different methods. As shown in Table 3, traditional unsupervised methods have weaker geometric feature extraction capabilities, resulting in lower performance. The supervised PointNet achieves higher performance at the cost of greater parameter quantity and execution time. In contrast, our non-parametric encoder not only has low time and space complexity but also achieves performance comparable to supervised methods.

Feature matching and grouping. After extracting geometric features from each building and vegetation point cloud instance, we conduct performance comparisons between our proposed feature matching and grouping module and HDBSCAN [CMZS15], as shown in Table 4. We define two metrics, grouping efficiency and grouping completeness, to measure performance. Grouping efficiency is the ratio of the number of the most frequent similar instances to the total number of instances in that group. Grouping completeness is the ratio of the number of the most frequent similar instances to the total number of such instances in the dataset.

The results show that HDBSCAN struggles to group similar instances due to insignificant differences in global features, causing HDBSCAN to classify all such instances into the same group. In contrast, our feature matching and grouping module only focuses on the two closest global features, effectively detecting similar instances and assigning instances with different appearances to multiple groups.

Table 5: *The performance comparison of different methods for the pose and size estimation of building and vegetation instances.*

Method	Rotation		Translation		Size		CD	T(s)
	RMSE	MAE	RMSE	MAE	RMSE	MAE		
FGR+ICPv1	0.6163	0.3709	1.8067	1.2862	-	-	12.33	255.5
FGR+ICPv2	0.5154	0.3240	1.8950	1.2035	-	-	7.05	560.8
NDT	0.5714	0.2862	0.0021	0.0016	-	-	4.93	974.0
Ours	0.2543	0.1203	0.0018	0.0007	0.1460	0.0448	0.45	117.8

6.3. Fast pose and size estimation

We compare the performance of different methods for estimating instance transformations, as shown in Table 5. We manually select 20 blocks from UrbanBIS, each with 50 building or tree instances, and calculate the rotation, translation, and size transformation matrices between representative and similar instances, evaluating the errors using RMSE and MAE. Additionally, we measure the Chamfer Distance between the transformed instances and the ground truth, as well as execution time. Among the methods, FGR [ZPK16] (for initial alignment), ICPv1 (point-to-point), and ICPv2 (point-to-plane) are combined. FGR+ICPv1 performs the worst due to sensitivity to initial alignment quality. FGR+ICPv2 shows slight improvement but is still affected by normal estimation errors. NDT [KL16], which uses global information, achieves higher accuracy but is less time-efficient. None of these methods account for size transformations. In contrast, our method efficiently estimates both pose and size with the fastest execution speed.

6.4. Stage Results

We visualize the stage results of the LightUrban framework for lightweighting the $Wuhu_{4649}$ example, as illustrated in Figure 7. In the instance segmentation stage, buildings and vegetation in the original scene are segmented into individual objects, with different instances represented in distinct colors. In the instancing stage, objects with similar geometries are grouped into the same similar instance group, with instances within each group displayed in the same color.

As a result, the original scene is instanced by replacing similar objects with their representative instance. For a scene containing approximately 10^7 points, LightUrban replaces each similar instance with a 4×4 transformation matrix of its representative instance. Consequently, the scale of the instanced scene is approximately equal to the combined scale of all representative instances, containing approximately 10^5 points. Additionally, visual comparisons of the details show that the instanced scene significantly preserves the original landscape. Through intuitive visualization results, LightUrban not only reduces the data volume by several orders of magnitude but also structures complex point clouds, while maintaining visual consistency before and after instancing.

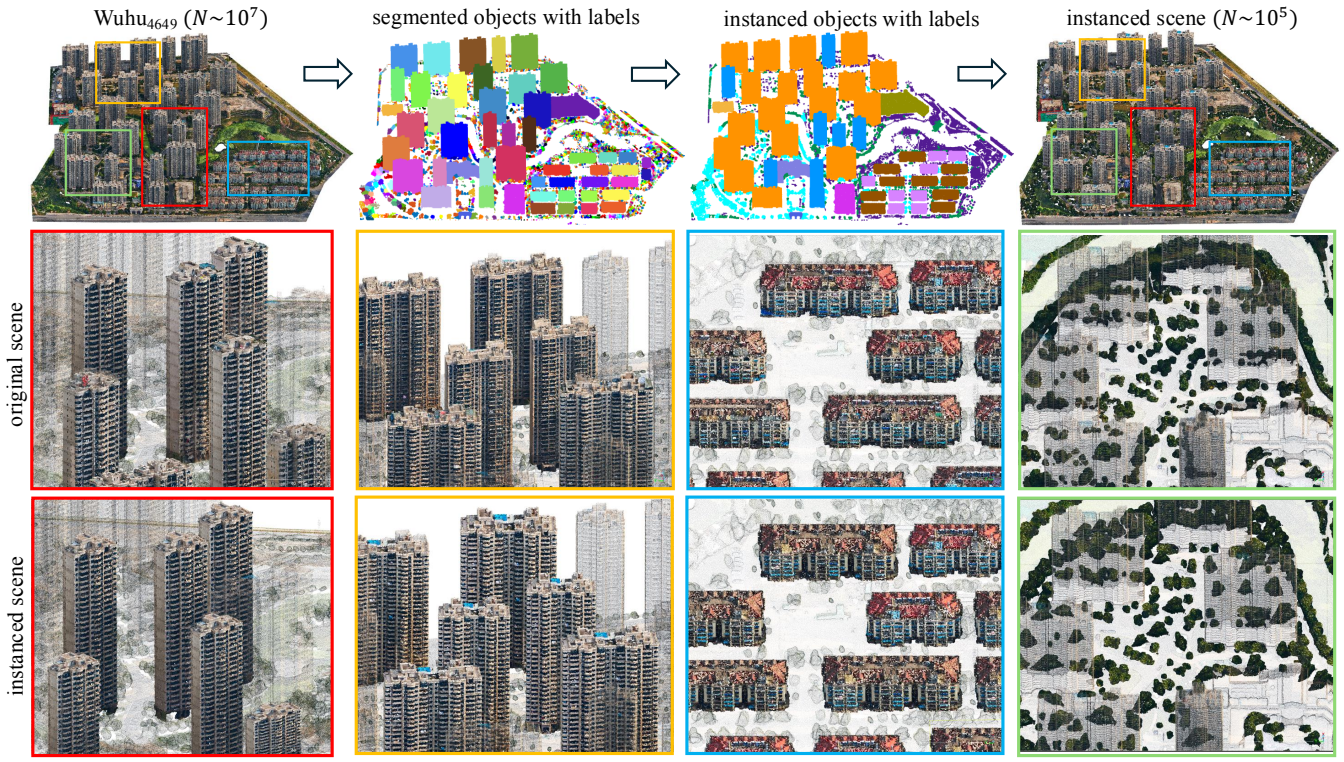


Figure 7: *The pipeline of LightUrban for an urban scene. The detailed comparisons before and after instancing are visualized as insects.*

6.5. Overall performance

Finally, we compare the point clouds generated by LightUrban and a downsampling method called ADS [HCL23] with the input scene. For fair comparisons, we let the results generated with our method and ADS share a similar number of points.

The visual comparisons are provided in Figure 8. We utilize OpenGL to render the original point cloud and the lightweighted point clouds under the same viewpoint. For quantitative evaluation, we calculate the SSIM of the rendered viewpoints of lightweighted scenes and the original scene. The statistics are shown in Table 6. Furthermore, we show the statistics on the classic compressing methods, ZIP, for the references of compression ratio. ZIP can achieve relatively small data volume but its advantage is only in the task of transmission. The user needs to decompress the data for further rendering tasks.

The statistics in quantitative evaluations show that our LightUrban achieves lightweight effects comparable to the classic ZIP algorithm. Since we focus on building and vegetation, LightUrban is especially efficient for scenes with highly repetitive building and tree instances. For the Qingdao$_{3133}$ and Wuhu$_{4649}$, the sizes of our lightweighted results are less than half of the compressed results. For rendering tasks, across all scenes, both our LightUrban and ADS require significantly less data volume compared to ZIP. Since ZIP is a lossless compression algorithm, for a fair comparison, we only compare the SSIM values of ADS with ours. The SSIM values of lightweighted scenes generated with our LightUrban are dramatically better than those using the ADS.

For qualitative comparison, the rendering results show that LightUrban can preserve effective features and offer fine-grained visualization quality than ADS. However, since there exist differences among the instances inside a group, our method can bring color discrepancies (*Qingdao*$_4$) or empty regions (*Longhua*$_7$).

7. Conclusion

Large-scale urban point clouds often contain highly repetitive instances, primarily buildings and trees, which account for 90% of the points. Considering the high redundancy, we proposed LightUrban, an automatic method to lightweight such point clouds for feasible transmission and rendering, via instancing the buildings and trees in the urban scene. Our method mainly contains two stages: coarse-to-fine instance segmentation and similarity-based instancing. Experimental results show that our method achieves a great balance between data volume and visual accuracy, making it extremely useful in applications where efficient transmission and rendering are critical.

Limitations and future work. The limitations of our method can be summarized as follows: i) The choice of threshold value in feature matching will affect the results of instance groups. Using a small value tends to group similar instances into smaller groups (Figure 9 (a)), while using a large value will possibly group not-

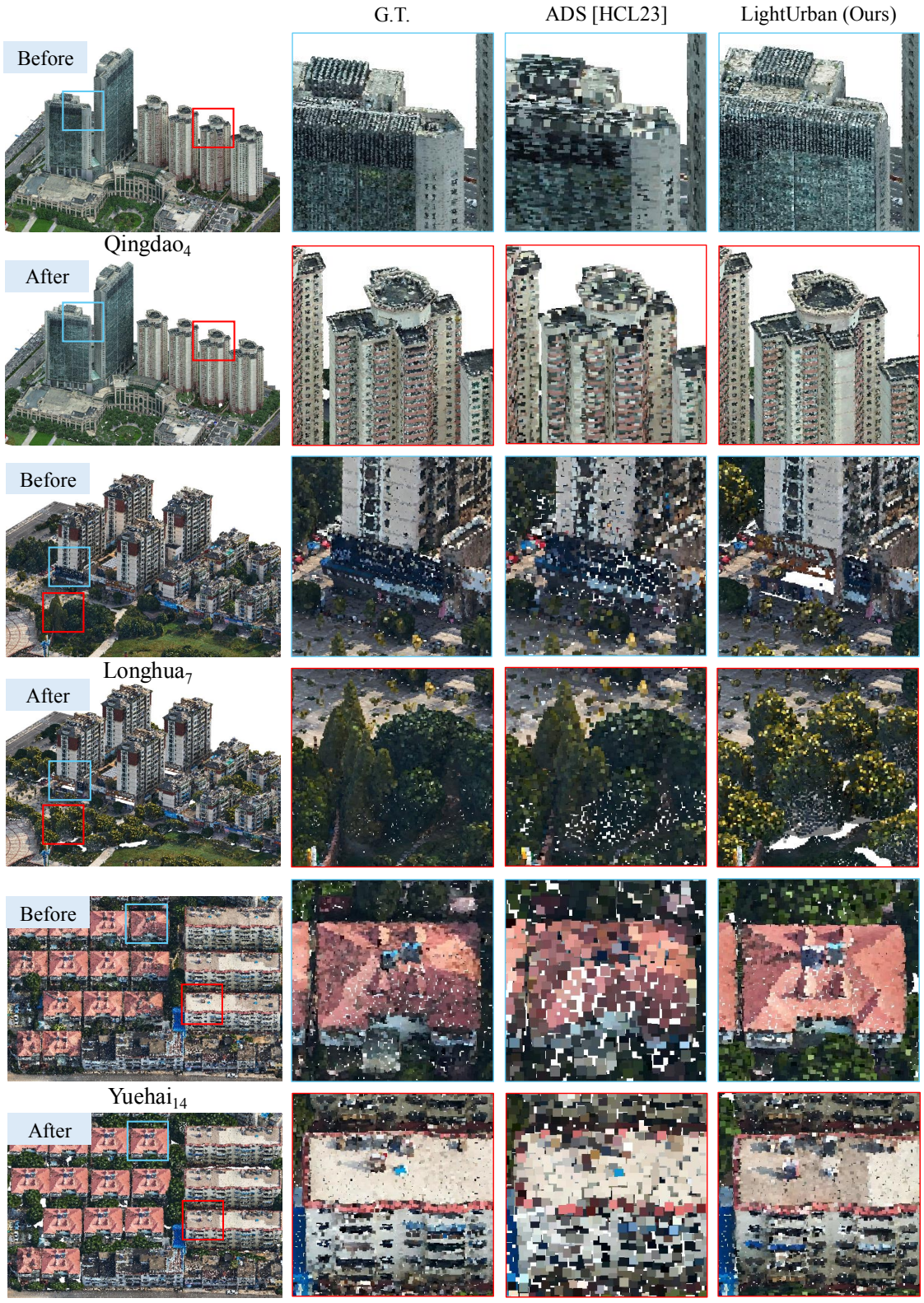
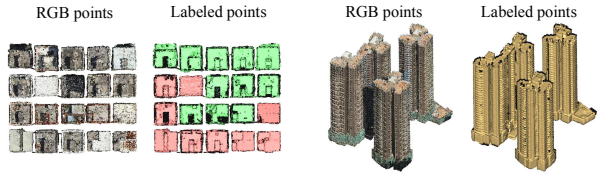


Figure 8: *Comparisons of the point clouds before and after lightweighting with ADS and ours on three scenes.*

Table 6: *The statistics of point clouds generated with lightweighting methods are shown in Figure 1, Figure 7 and Figure 8. We show the data sizes of the original scenes, as well as that involved in transmission and rendering tasks. All the data sizes are in MB.*

Scene	Size	Method	Transmit ↓	Render ↓	SSIM ↑
$Qingdao_{3133}$	326.3	ZIP	84.8	326.3	-
		ADS	37.9	37.9	0.493
		Ours	38.1	38.1	0.745
$Wuhu_{4649}$	512.6	ZIP	117.8	512.6	-
		ADS	58.5	58.5	0.388
		Ours	58.0	58.0	0.720
$Qingdao_{4}$	165.9	ZIP	41.1	165.9	-
		ADS	63.2	63.2	0.422
		Ours	62.9	62.9	0.731
$Longhua_{7}$	66.1	ZIP	16.2	66.1	-
		ADS	16.3	16.3	0.260
		Ours	16.8	16.8	0.698
$Yuehai_{14}$	59.4	ZIP	12.7	59.4	-
		ADS	14.2	14.2	0.138
		Ours	14.0	14.0	0.713



(a) Similar instances in multiple groups (b) Not-so-similar instances in a group

Figure 9: *The grouping results given (a) a small threshold value and (b) a large threshold value.*

so-similar instances (Figure 9 (b)). ii) Due to the nature of the instancing method, our method can introduce losses in the variations of appearance among instances and data distortion to some extent. To mitigate this issue, users can adjust the grouping parameters to generate instances group with higher similarity. iii) Less effective for low repetitive scenes.

In the future, we consider extending this framework to the development of large-scale urban rendering pipelines. Notably, although the input data for LightUrban is point clouds, its output is not limited to a specific data format. Any geometric representation, including meshes, voxels, and implicit fields, can be used to present the instanced landscape. Leveraging this flexibility, people can further explore the development of a unified large-scale urban rendering pipeline that supports various geometric representations.

Acknowledgement

We thank the reviewers for their constructive comments. This research is supported by Ant Group and the National Natural Science Foundation of China (No.6207071897).

References

[ABDBP07] ASSFALG J., BERTINI M., DEL BIMBO A., PALA P.: Content-based retrieval of 3-d objects using spin image signatures. *IEEE Transactions on Multimedia 9*, 3 (2007), 589–599. 3

[AGSL19] AOKI Y., GOFORTH H., SRIVATSAN R. A., LUCEY S.: Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 7163–7172. 3

[ASYS09] AKGÜL C. B., SANKUR B., YEMEZ Y., SCHMITT F.: 3d model retrieval using probability density-based shape descriptors. *IEEE transactions on pattern analysis and machine intelligence 31*, 6 (2009), 1117–1133. 3

[BGSF08] BIASOTTI S., GIORGI D., SPAGNUOLO M., FALCIDIENO B.: Size functions for comparing 3d models. *Pattern Recognition 41*, 9 (2008), 2855–2873. 3

[Ceb04] CEBENOYAN C.: Graphics pipeline performance, 2004. 2

[CFZ*21] CHEN S., FANG J., ZHANG Q., LIU W., WANG X.: Hierarchical aggregation for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 15467–15476. 2

[CHY*22] CHEN M., HU Q., YU Z., THOMAS H., FENG A., HOU Y., MCCULLOUGH K., REN F., SOIBELMAN L.: Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset. *arXiv preprint arXiv:2203.09065* (2022). 2

[CLCL11] CHAO M.-W., LIN C.-H., CHANG C.-C., LEE T.-Y.: A graph-based shape matching scheme for 3d articulated objects. *Computer Animation and Virtual Worlds 22*, 2-3 (2011), 295–305. 3

[CMZS15] CAMPELLO R. J., MOULAVI D., ZIMEK A., SANDER J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 1–51. 8

[DC16] DALPONTE M., COOMES D. A.: Tree-centric mapping of forest carbon density from airborne laser scanning and hyperspectral data. *Methods in ecology and evolution 7*, 10 (2016), 1236–1245. 7

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395. 7

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. *ACM Transactions on Graphics (TOG) 22*, 1 (2003), 83–105. 3

[GZLX23] GAO P., ZHANG L., LEI L., XIANG W.: Point cloud compression based on joint optimization of graph transform and entropy coding for efficient data broadcasting. *IEEE Transactions on Broadcasting* (2023). 2

[HCL23] HONG C.-Y., CHOU Y.-Y., LIU T.-L.: Attention discriminant sampling for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 14429–14440. 2, 9

[HFZL20] HE Z., FENG W., ZHAO X., LV Y.: 6d pose estimation of objects: Recent technologies and challenges. *Applied Sciences 11*, 1 (2020), 228. 6

[HGW24] HU Y., GONG R., WANG Y.: Bits-to-photon: End-to-end learned scalable point cloud compression for direct rendering. *arXiv preprint arXiv:2406.05915* (2024). 2

[HSH*20] HE Y., SUN W., HUANG H., LIU J., FAN H., SUN J.: Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 11632–11641. 3

[HSL*17] HACKEL T., SAVINOV N., LADICKY L., WEGNER J. D., SCHINDLER K., POLLEFEYS M.: Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847* (2017). 2

[JZS*20] JIANG L., ZHAO H., SHI S., LIU S., FU C.-W., JIA J.: Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition* (2020), pp. 4867–4876. 2

[KL16] KIM J. W., LEE B. H.: Robust and fast 3-d scan registration using normal distributions transform with supervoxel segmentation. *Robotica 34*, 7 (2016), 1630–1658. 8

[KPNK03] KÖRTGEN M., PARK G.-J., NOVOTNI M., KLEIN R.: 3d shape matching with 3d shape contexts. In *The 7th central European seminar on computer graphics* (2003), vol. 3, Budmerice Slovakia, pp. 5–17. 2, 3

[KRA*14] KHAN K., REHMAN S. U., AZIZ K., FONG S., SARASVADY S.: Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)* (2014), IEEE, pp. 232–238. 4

[LBS*18] LI Y., BU R., SUN M., WU W., DI X., CHEN B.: Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems 31* (2018). 2

[M*67] MACQUEEN J., ET AL.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297. 7

[MD03] MOENNING C., DODGSON N. A.: *Fast marching farthest point sampling*. Tech. rep., University of Cambridge, Computer Laboratory, 2003. 5

[MDTS09] MADEMLIS A., DARAS P., TZOVARAS D., STRINTZIS M. G.: Ellipsoidal harmonics for 3-d shape description and retrieval. *IEEE transactions on multimedia 11*, 8 (2009), 1422–1433. 3

[MR93] MAĆKIEWICZ A., RATAJCZAK W.: Principal components analysis (pca). *Computers & Geosciences 19*, 3 (1993), 303–342. 6

[Pet09] PETERSON L. E.: K-nearest neighbor. *Scholarpedia 4*, 2 (2009), 1883. 5

[PWT*22] PANG Y., WANG W., TAY F. E., LIU W., TIAN Y., YUAN L.: Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision* (2022), Springer, pp. 604–621. 2

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 652–660. 2

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems 30* (2017). 2

[RBB09] RUSU R. B., BLODOW N., BEETZ M.: Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation* (2009), IEEE, pp. 3212–3217. 2

[RBTH10] RUSU R. B., BRADSKI G., THIBAUX R., HSU J.: Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ international conference on intelligent robots and systems* (2010), IEEE, pp. 2155–2162. 2

[RSY*24] RUAN H., SHAO Y., YANG Q., ZHAO L., NIYATO D.: Point cloud compression with implicit neural representations: A unified framework. *arXiv preprint arXiv:2405.11493* (2024). 2

[SLG*19] SARODE V., LI X., GOFORTH H., AOKI Y., SRIVATSAN R. A., LUCEY S., CHOSET H.: Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint arXiv:1908.07906* (2019). 3

[STDS14] SALTI S., TOMBARI F., DI STEFANO L.: Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding 125* (2014), 251–264. 2

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł., POLOSUKHIN I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 5

[WJC*21] WANG X., JIN Y., CEN Y., LANG C., LI Y.: Pst-net: Point cloud sampling via point-based transformer. In *Image and Graphics: 11th International Conference, ICIG 2021, Haikou, China, August 6–8, 2021, Proceedings, Part III 11* (2021), Springer, pp. 57–69. 2

[WNG00] WULDER M., NIEMANN K. O., GOODENOUGH D. G.: Local maximum filtering for the extraction of tree locations and basal area from high spatial resolution imagery. *Remote Sensing of environment 73*, 1 (2000), 103–114. 4

[WXZ*19] WANG C., XU D., ZHU Y., MARTÍN-MARTÍN R., LU C., FEI-FEI L., SAVARESE S.: Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 3343–3352. 3

[XGG*20] XIE S., GU J., GUO D., QI C. R., GUIBAS L., LITANY O.: Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16* (2020), Springer, pp. 574–591. 2

[XLM*22] XU X., LI K., MA Y., GENG G., WANG J., ZHOU M., CAO X.: Feature-preserving simplification framework for 3d point cloud. *Scientific reports 12*, 1 (2022), 9450. 2

[YXZ*23] YANG G., XUE F., ZHANG Q., XIE K., FU C.-W., HUANG H.: Urbanbis: A large-scale benchmark for fine-grained urban building instance segmentation. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023), pp. 1–11. 2, 3

[ZCDM23] ZHANG J., CHEN T., DING D., MA Z.: G-pcc++: Enhanced geometry-based point cloud compression. In *Proceedings of the 31st ACM International Conference on Multimedia* (2023), pp. 1352–1363. 2

[ZDSJ18] ZÖRNER J., DYMOND J., SHEPHERD J., JOLLY B.: Pycrown-fast raster-based individual tree segmentation for lidar data. *Landcare Research Ltd.: Lincoln, New Zealand* (2018). 8

[ZPK16] ZHOU Q.-Y., PARK J., KOLTUN V.: Fast global registration. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14* (2016), Springer, pp. 766–782. 8

[ZSN*17] ZENG A., SONG S., NIESSNER M., FISHER M., XIAO J., FUNKHOUSER T.: 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1802–1811. 3

[ZWG*23] ZHANG R., WANG L., GUO Z., WANG Y., GAO P., LI H., SHI J.: Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. *arXiv preprint arXiv:2303.08134* (2023). 2, 5