

# Non-Photorealistic Rendering of 3D Point Clouds Using Segment-Specific Image-Space Effects

Ole Wegen<sup>1</sup><sup>a</sup>, Josafat-Mattias Burmeister<sup>1</sup><sup>b</sup>, Max Reimann<sup>2</sup><sup>c</sup>, Rico Richter<sup>1</sup><sup>d</sup>  
and Jürgen Döllner<sup>2</sup><sup>e</sup>

<sup>1</sup>University of Potsdam, Germany

<sup>2</sup>Hasso-Plattner-Institute, University of Potsdam, Germany


**Keywords:** 3D Point Clouds, Non-Photorealistic Rendering, Segmentation, Image-Based Artistic Rendering.


**Abstract:** 3D point clouds are a widely used representation for surfaces and object geometries. However, their visualization can be challenging due to point sparsity and acquisition inaccuracies, leading to visual complexity and ambiguity. Non-photorealistic rendering (NPR) addresses these challenges by using stylization techniques to abstract from certain details or emphasize specific areas of a scene. Although NPR effectively reduces visual complexity, existing approaches often apply uniform styles across entire point clouds, leading to a loss of detail or saliency in certain areas. To address this, we present a novel segment-based NPR approach for point cloud visualization. Utilizing prior point cloud segmentation, our method applies distinct rendering styles to different segments, enhancing scene understanding and directing the viewer's attention. Our emphasis lies in integrating aesthetic and expressive elements through image-based artistic rendering, such as watercolor or cartoon filtering. To combine the per-segment images into a consistent final image, we propose a user-controllable depth inpainting algorithm. This algorithm estimates depth values for pixels that lacked depth information during point cloud rendering but received coloration during image-based stylization. Our approach supports real-time rendering of large point clouds, allowing users to interactively explore various artistic styles.


## 1 INTRODUCTION


3D point clouds are unstructured sets of points in 3D space that often lack consistent density or distribution. The term “cloud” metaphorically reflects their abstract, shapeless nature, akin to atmospheric clouds. Point clouds can efficiently represent diverse 3D entities, capturing various shapes, topologies, and scales. Recent advances in remote sensing technologies, particularly LiDAR (Horaud et al., 2016) and photogrammetry (Westoby et al., 2012), have made point cloud acquisition more accessible and efficient. Consequently, point clouds have become an integral part of spatial computational models and digital twins, serving various sectors such as autonomous driving (Li et al., 2021) or infrastructure management (Mirzaei et al., 2022).


However, despite their widespread adoption, visualizing point clouds remains a challenge due to inherent issues such as incompleteness, point sparsity, and inaccuracies arising from the acquisition process. These issues often result in point cloud renderings that suffer from visual clutter and ambiguity (Xu et al., 2004). Non-photorealistic rendering (NPR) offers a way to address these challenges by using techniques that do not aim for photorealism, but deliberately employ abstraction to enhance scene comprehension and guide the viewer's focus to relevant scene elements (Döllner, 2008; DeCarlo and Santella, 2002) (Figure 1). Different NPR approaches for point clouds have been designed in the last years, using both object-space and image-space effects (Awano et al., 2010; Xu et al., 2004; Wagner et al., 2022). However, many of these approaches uniformly apply the same visual style to the entire point cloud. While this helps to reduce visual clutter, it often leads to foreground objects blending into the background. Only a few methods have adopted a segment-based approach, allowing distinct rendering styles to be ap-

<sup>a</sup> <https://orcid.org/0000-0002-6571-5897>

<sup>b</sup> <https://orcid.org/0000-0003-1890-844X>

<sup>c</sup> <https://orcid.org/0000-0003-2146-4229>

<sup>d</sup> <https://orcid.org/0000-0001-5523-3694>

<sup>e</sup> <https://orcid.org/0000-0002-8981-8583>



(a) Rendering of a point cloud via rasterization of point primitives. The only available attribute of reflectance intensity is interpreted as a single-channel color.



(b) Example result of a NPR of the same point cloud based on edge enhancement, class-specific coloring, ambient occlusion, and a watercolor postprocessing operation.

Figure 1: Comparison of conventional point-based rendering with reflectance intensity information (a) and NPR that supports expressive visualization of point clouds (b), e.g., by highlighting specific objects and scene parts.

plied to different parts of a point cloud, e.g., individual stylization of different semantic classes (Richter et al., 2015; Wegen et al., 2022). Such rendering approaches are especially suited for complex scenes, where the same degree of abstraction may not be appropriate for every object or semantic class, or where it is necessary to highlight certain areas. While prior works on segment-based point cloud rendering use simple NPR techniques as a tool for focus+context visualization, the use of image-based artistic rendering (IB-AR) has rarely been explored. As (Gooch et al., 2010) note, traditional illustration and drawing styles can effectively convey information and allow for engaging depictions that capture and maintain the viewer’s interest. Therefore, in the image stylization domain, a large variety of IB-AR filters have been developed (Kyprianidis et al., 2013) that enable a wide range of visual effects.

In this work, we introduce an approach for segment-based stylization of point clouds, using different artistic styles implemented through IB-AR filters. Several challenges must be addressed to achieve this goal. First, rendering of large point clouds as well as subsequent artistic filtering and compositing of per-segment images must be implemented in a real-time manner to create an interactive experience. To this end, we propose a pipeline approach for scene-graph-based rendering and compositing of point cloud segments. The approach enables the interactive exploration of large point clouds and allows users to experiment with a variety of artistic styles.

Second, the compositing of different segments must adhere to the depth order of points. However, as image-based filtering can shift segment boundaries, naive composition based on depth testing can lead to

visual artifacts at these boundaries. To address this issue, we employ a compositing method based on layered depth images and propose a depth inpainting step to correct the segments’ depth values in areas altered by image filtering. To summarize, our contributions are:

1. A pipeline-based approach for segment-based real-time NPR of point clouds that integrates a variety of image-based stylization techniques.
2. A user-controllable segment compositing approach based on depth-buffer inpainting that eliminates artifacts at segment borders.

The remainder of this work is structured as follows: In Section 2, we review related work in the areas of point cloud segmentation, rendering, and NPR. In Section 3, we present our approach for segment-based real-time NPR of point clouds. Implementation details are given in Section 4. In Section 5, we showcase exemplary results for different application domains, demonstrate the effectiveness of our segment compositing method, and discuss limitations of our approach. Finally, Section 6 concludes the paper and outlines possible directions for future work.

## 2 BACKGROUND AND RELATED WORK

This section provides an overview of the research areas related to our work. After providing a definition of 3D point clouds, we present previous work on point cloud rendering and segmentation, before we subsequently review related work in the area of NPR, focusing IB-AR and NPR for point clouds.

## 2.1 3D Point Clouds

A 3D point cloud is a set of points in space, which is permutation invariant. Each point is defined at least by its 3D coordinates, but may have additional attributes, such as reflectance intensity or color. In this paper, we focus on surface point clouds, where each point represents a discrete sample of a continuous surface. Due to imperfections in the data acquisition process, 3D point clouds are often incomplete, have an irregular point distribution, and contain measurement inaccuracies and noise. During data preprocessing, some of these issues can be mitigated (e.g., by filtering outliers) and additional per-point attributes, such as surface normals, can be computed.

## 2.2 Point Cloud Rendering

For point cloud rendering, different approaches have been developed. A straightforward approach is to use point primitives supported by graphics APIs to render individual points with fixed size in screen space. Alternatively, splatting provides a linear approximation to the underlying surface by rendering primitives (e.g., disks) of a certain world-space size and blending overlapping regions to create the appearance of smooth surfaces (Zwicker et al., 2001). Recently, deep learning (DL) approaches have been proposed to generate high-resolution renderings from low-density point clouds (Bui et al., 2018; Aliev et al., 2020). However, these methods often require RGB images, are time-consuming to train, and the rendering performance is not yet comparable with standard rendering approaches. The described rendering techniques can be implemented with level-of-detail (LoD) data structures to enable out-of-core rendering of massive point clouds, enhance rendering performance, and reduce visual clutter (Scheiblauer, 2014). For this, point clouds are hierarchically organized in spatial data structures (e.g., kd-trees or octrees), which are used during rendering to select a suitable LoD, depending on the virtual view position. Our approach is compatible with any point cloud rendering technique, provided it can generate a depth image during the rendering process. For the sake of clarity in our demonstrations, we opt for a straightforward approach using point primitive rasterization and do not use any LoD data structure.

## 2.3 Segmentation of Point Clouds

Although different subsets of a point cloud may represent distinct parts of a scene, there is no inherent order in a point cloud that reflects this. Basic struc-

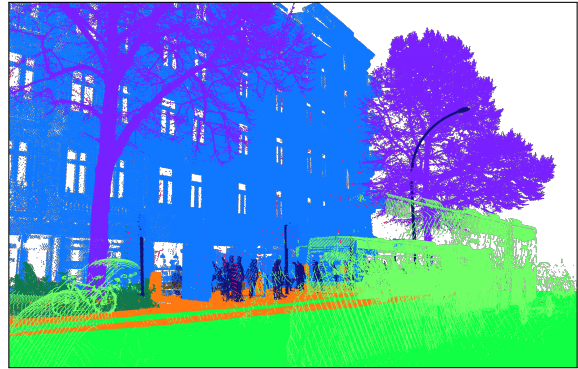
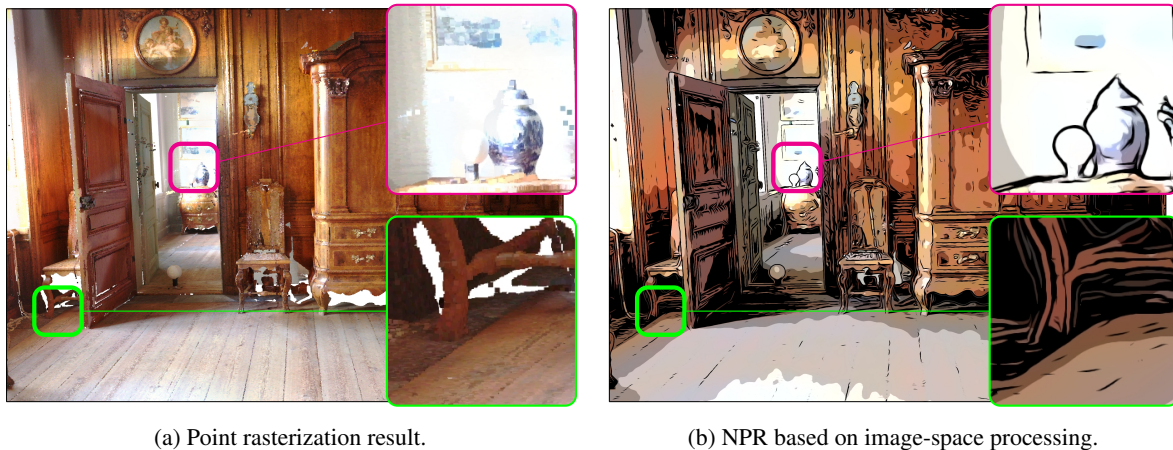


Figure 2: Result of a deep-learning-based semantic segmentation of the point cloud shown in Figure 1.

tural information can be obtained from unsupervised segmentation algorithms that divide point clouds into non-overlapping segments. Xie et al. provide an overview of these methods, of which region growing, clustering, and model fitting are the most common ones (Xie et al., 2020). Further information can be added by semantic segmentation, where each point is assigned a semantic label (Figure 2), and instance segmentation, where points with the same semantic label are grouped into individual objects. This paper does not focus on the segmentation process itself, as our segment-based stylization approach is agnostic to the segmentation method used. Nevertheless, we provide a brief overview of approaches below.

**Semantic Segmentation:** can be performed both with unsupervised, rule-based approaches as well as with supervised machine learning (ML) approaches. In rule-based approaches, additional attributes are computed for each point and subsequently used to distinguish between different semantic classes. This can involve deriving geometric descriptors (e.g., linearity, planarity, scattering) from the distribution of neighboring points (Weinmann et al., 2015). Often, rule-based approaches are based on a prior point cloud segmentation with unsupervised algorithms, allowing classification rules to be based on segment shape (Hao et al., 2022). Supervised ML models for semantic segmentation can be trained if annotated datasets are available. In recent years, deep neural networks have become popular for this task, as they often outperform statistical ML models, such as random forests (Weinmann et al., 2015). A number of DL architectures have been developed to process point clouds directly, offering performance advantages over DL architectures that are based on image- or voxel-based intermediate representations (Bello et al., 2020).

**Instance Segmentation:** approaches often rely on special shape characteristics of the object category to be segmented. Many instance segmentation methods



(a) Point rasterization result.

(b) NPR based on image-space processing.

Figure 3: Compared to simple point primitive rasterization, NPR mitigates the problem of holes due to missing points and difficulty of perceiving object boundaries.

are based on unsupervised model fitting, region growing, or clustering algorithms, such as DBSCAN (Ester et al., 1996). Some authors have also proposed deep learning approaches (Wang et al., 2018; Luo et al., 2021).

## 2.4 Non-Photorealistic Rendering

NPR uses visual abstraction and focus techniques to simplify images and direct attention, creating aesthetically pleasing and easily understandable illustrations. IB-AR is a sub-area of NPR that transforms an input image into an artistically stylized rendition using image filtering techniques. A variety of IB-AR techniques have been proposed that mimic artistic painting methods, such as oil painting (Semmo et al., 2016), watercolor (Bousseau et al., 2006), cartoon filtering (Winnemöller et al., 2006), stroke-based painterly rendering (Hertzmann, 1998), and many others (Kyprianidis et al., 2013). Further, with the advancements in DL, example-based techniques such as neural style transfer (NST) (Jing et al., 2020) have emerged, which apply an artistic style extracted from an input style image to a new content image. In our approach, we implement the previously named techniques as part of an image processor responsible for stylizing point cloud segments.

### 2.4.1 NPR of Point Clouds

The main strengths of NPR also apply to point cloud rendering: Scene understanding is enhanced and visually appealing images can be created (Figure 3). NPR of point clouds can be categorized into object-space and image-space approaches. **Object-space** approaches control the rendering result via the selection

of geometric primitives (e.g., points, splats, strokes, or 3D glyphs) and their orientation (e.g., surface-aligned or view-aligned) and scaling. **Image-space** approaches postprocess the image resulting from a prior point cloud rasterization step. In theory, any IB-AR technique could be employed during postprocessing. To date, some NST approaches have been adapted to point clouds (Cao et al., 2020), and eye-dome lighting has been developed as a technique to enhance object perception in point cloud renderings (Ribes and Boucheny, 2011). However, generic approaches for integrating point cloud rendering with arbitrary IB-AR methods have rarely been investigated.

The use of point cloud segmentation techniques to stylize segments in different ways has been explored in the past, e.g., by (Richter et al., 2015) and (Wegen et al., 2022). Both employ semantic-class-dependent NPR to enhance recognition of objects. While their approaches are focused on specific application domains and mostly employ object-space techniques, we propose a more general approach to segment-based point cloud stylization. In particular, our approach enables per-segment stylization using a multitude of NPR techniques, including especially IB-AR methods (Figure 4). The segments can be derived from semantic classes, object instances, point clusters, or application-specific attributes.

### 2.4.2 Depth-Aware IB-AR Techniques

Typically, IB-AR approaches do not make assumptions about the geometry an image depicts, which often leads to a flattening effect in the output. However, for point clouds, maintaining depth perception is important. Several image-based approaches incorporate depth information (either predicted or captured),



(a) Direct rendering of a raw point cloud.

(b) Segment-based NPR of the same point cloud.

Figure 4: A segmentation of the point cloud enables separate processing and subsequent blending of the rendering results.

to enhance depth perception and improve separation of foreground and background. For example, NST was extended to incorporate depth information in the 2D (Liu et al., 2017) and 3D (Höllein et al., 2022) domain. (Shekhar et al., 2021) enhance cartoon stylization effects by producing salient edges around depth discontinuities. Similar to our approach, depth-based segment stitching has been explored in the form of layered depth images in the context of 3D photo stylization. (Bath et al., 2022) segment an image based on predicted depth and apply different stylizations that are then stitched together using a simple form of depth-inpainting. (Mu et al., 2022) extract a point cloud from a depth image and use it to condition view-consistent style transfer features. We also use depth-aware composition of stylized segments, but instead of generating new content (as done in 3D photo stylization), we employ depth inpainting to improve blending between stylized point cloud segments.

### 3 APPROACH

In the following, we present our approach for combining IB-AR with segment-based NPR of point clouds. Figure 5 illustrates the overall process of obtaining a stylized image from point cloud data. Initially, each point cloud is preprocessed, e.g., using outlier filtering or computing additional point attributes. An integral part of the preprocessing is the segmentation of the point cloud, i.e., creating segments that represent point clusters, semantic classes, or object instances. Subsequently, on a per-frame basis, each point cloud segment is rendered separately, resulting in at least a color and a depth image. A rendering pipeline  $P$  for a single segment comprises different processing steps in a fixed order:

1. The segment is rasterized using point primitives of a user-defined size. The results are raster images containing color and depth information for further processing in subsequent pipeline steps.
2. An arbitrary number of image-based postprocessing steps can be applied to the result of the rasterization step. Examples include smoothing, color grading, or IB-AR filters.
3. A depth inpainting step is used to counter border artifacts that can occur when compositing the per-segment images.

To enable fine-grained control of the rendering process, each processing step can be parameterized. After all segments are rendered, the resulting per-segment images are composited into a final image.

#### 3.1 Countering Compositing Artifacts

Since each segment is rendered separately, the question arises of how to combine the per-segment results into a cohesive final image. Merely overlaying the results of each segment fails to accurately represent scenarios where segments are partially obscured or interleaved, as is the case with the lantern pole and the tree in Figure 1. To ensure proper visibility, depth testing has to be employed. However, image-based postprocessing can modify color values at pixels where no fragment was recorded during rasterization, e.g., when adding outlines. These newly colored pixels would have no valid depth value, leading to artifacts when compositing the rendered segments. A potential solution could be to resolve visibility before applying image filters: for each segment, the occluded areas are computed and masked by setting the alpha channel to zero before applying image-based postprocessing. While this partially mitigates the problem, it requires

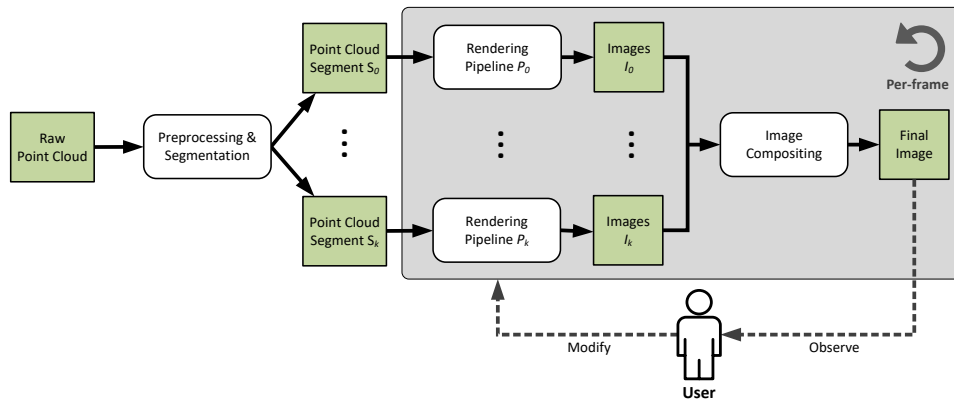


Figure 5: Illustration of the overall rendering approach. First, the point cloud is preprocessed, e.g., using outlier filtering. Additionally, it is segmented according to point clusters, semantic classes, or object instances. Then, each point cloud segment is rendered separately and the results are combined into a final image. The user can interactively control the processing pipelines for each point cloud.

the definition of a strict order of segments, implying that one segment consistently overlaps the other in areas lacking depth data. Additionally, by introducing transparency early in the process, IB-AR techniques might not produce correct results or overwrite the alpha channel as part of their processing pipeline.

Therefore, we propose to instead use a depth inpainting pass on the per-segment depth buffers. This inpainting pass is guided by the differences between the rasterized and the postprocessed color images, as depicted in Figure 6. First, for each segment, the rasterized and the postprocessed images are converted to grayscale and their difference is computed. Subsequently, this difference image is converted into a binary inpainting mask using a user-defined threshold, encoding areas significantly altered by postprocessing. Then, the depth values of all masked pixels are inpainted. For this, all pixels in a circular neighborhood, whose depth values do not correspond to the far plane depth, are averaged. The size of the averaging kernel is user-controllable. Pseudo-code for the depth inpainting pass is provided in Section 4.

## 4 IMPLEMENTATION

This sections provides implementation aspects of the proposed approach. After describing the segmentation methods we use as part of point cloud preprocessing, we detail our implementation of the described point cloud stylization approach.

### 4.1 Point Cloud Segmentation

As described, we perform point cloud segmentation independently of the rendering in a preprocessing

step. Depending on the application scenario, we employed different segmentation approaches:

**Connected Component Analysis.** We use the algorithm implemented in the open source software CloudCompare.<sup>1</sup> This algorithm uses an octree and identifies connected components based on two criteria: The octree level, which defines the minimum size of the gap between two components, and the minimum number of points per component, which causes components with fewer points to be discarded.

**Semantic Segmentation.** We present examples that were created using a rule-based approach, as well as examples based on a DL approach. Specifically, we use the rule-based approach of (Richter et al., 2015), which segments point clouds into five disjoint partitions representing buildings, vegetation, terrain, infrastructure, and water. For this purpose, the point clouds are first segmented using the algorithm of (Rabbani et al., 2006), after which the label of each segment is determined in several rule-based classification steps. As an example of a DL method, the pipeline of (Burmeister et al., 2023) is used. It is based on DL architectures that can directly process point clouds, namely KP-FCNN (Thomas et al., 2019).

**Instance Segmentation.** As an example of an instance segmentation task, we consider single tree delineation. To segment vegetation point clouds into individual trees, we use the marker-controlled watershed algorithm (Kornilov and Safonov, 2018), with the local maxima of the canopy height serving as markers.

<sup>1</sup>[www.cloudcompare.org/doc/wiki/index.php/Label.Connected.Components](http://www.cloudcompare.org/doc/wiki/index.php/Label.Connected.Components)

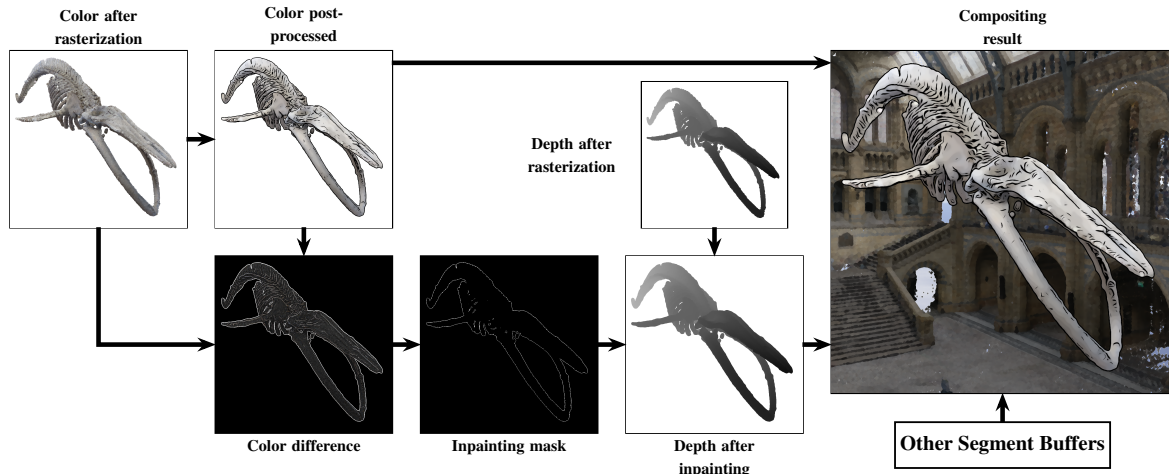


Figure 6: A flow chart depicting our depth inpainting approach to counter compositing artifacts.

## 4.2 Point Cloud NPR

The proposed segment-based point cloud stylization approach was implemented using C++ and OpenGL 4.3.

**Point Rendering.** We structure the point cloud to render as a tree. Each leaf node of the tree comprises a point cloud segment  $S$  with its own rendering pipeline  $P$  that defines the different stylization steps, as well as their parameterization. Inner nodes of the tree represent user-defined groupings of multiple segments, facilitating their joint manipulation. Changes made to

```

1 float computeDepth(ivec2 coords, float maskThreshold
  ↪ , int kernelSize){
2     float originalDepth = readDepthAt(coords);

4     if(colorsAreSimilar(coords, maskThreshold))
5         return originalDepth;

7     // inpaint depth based on neighbor depth
8     float counter = 0.0, finalDepth = 0.0;
9     for (each pixel in a circle around coords with
  ↪ radius kernelSize){
10        float neighborDepth = readDepthAt(pixel);

12        // skip depth values on the far plane
13        float skipFactor = 1.0 - step(1.0,
  ↪ neighborDepth);

15        finalDepth += neighborDepth * skipFactor;
16        counter += skipFactor;
17    }
18    return finalDepth / counter;
19 }
    
```

Listing 1: GLSL-inspired pseudocode of the function that estimates the depth of a pixel in the depth inpainting pass. The *colorsAreSimilar* corresponds to the calculation of the inpainting mask (Figure 6), i.e., it retrieves the color after rasterization and the color after postprocessing at the given coordinates, computes their intensity difference, and thresholds the result according to the *maskThreshold*.

inner nodes, e.g., addition of stylization steps or adjustment of processing parameters, are propagated to all child nodes. However, child nodes may override the pipeline configuration inherited from their parent. This hierarchical structure enables intuitive and effective control of the rendering. General settings can be applied to multiple segments by specifying them in higher-level nodes, while still allowing for individual parameterization at lower levels.

**Image Filtering.** For postprocessing after rasterization, we integrated several IB-AR effects that can be parameterized by the user. Examples include a cartoon effect (Winnemöller et al., 2006), a watercolor filter (Bousseau et al., 2006), an oilpaint filter (Semmo et al., 2016), image warping, and color grading. To define the effects to be applied to each segment, we use the description format for image and video processing operations introduced by (Dürschmid et al., 2017). This format provides a consistent description of the effect parameters and their presets and allows multiple IB-AR effects to be combined in a pipeline.

**Maintaining Interactivity.** To facilitate interactive navigation through the scenes during rendering, we employ two approaches: First, the image stylization per segment is performed asynchronous to the rest of the rendering in a separate thread. This enables smooth navigation, even with computationally intensive postprocessing pipelines. However, if the processing times for certain segments differ significantly, image artifacts can occur during camera movement. This is due to the fact that during postprocessing of a segment, the virtual viewpoint might change, which then already influences the rendering result of other segments. As a result, segments rendered from slightly different view points might be combined in the compositing stage. Second, we provide the option

Table 1: The point clouds used in the paper.

Figures	Reference
1,2,12	Street scene in the city of Hamburg, scanned with mobile mapping vehicle. Provided by AllTerra Deutschland GmbH.
3	"Tottieska malmgården, Faro Pointcloud, Decimated" ( <a href="https://skfb.ly/6RTvX">https://skfb.ly/6RTvX</a> ) by HagaeusBygghantverk. Licensed under Creative Commons Attribution.
4,6,11,12	"Hintze Hall, NHM London [point cloud]" ( <a href="https://skfb.ly/6sXWG">https://skfb.ly/6sXWG</a> ) by Thomas Flynn. Licensed under Creative Commons Attribution-NonCommercial.
7	"Stone Griffin, Downing College, Cambridge" ( <a href="https://skfb.ly/OVZx">https://skfb.ly/OVZx</a> ) by Thomas Flynn. Licensed under Creative Commons Attribution.
8	From "Hessigheim 3D" dataset (Kölle et al., 2021).
10	Street scene in the city of Essen, scanned with mobile mapping vehicle. Provided by the Department for Geoinformation, Surveying and Cadastre of the City of Essen.
9, 12	Forest area scanned with aerial LiDAR, obtained from OpenGeo-DataNRW ( <a href="http://www.opengeodata.nrw.de">www.opengeodata.nrw.de</a> ). Licensed under Data licence Germany - Zero - Version 2.0

to automatically switch to lower resolution buffers during camera movement, to maintain interactivity in scene navigation on systems with less computational power. If the virtual camera is not moved for 500 milliseconds, high-resolution buffers are used again for rendering.

**Depth Inpainting.** The implementation of the depth inpainting pass is shown in Listing 1. Although the generation of the difference image, the inpainting mask, and the processed depth buffer are conceptually separate steps, we implemented them as a single shader for performance reasons.

## 5 RESULTS

The proposed rendering process can be customized on different granularity levels: each point cloud segment is rendered by an individual processing pipeline and each pipeline step can be controlled interactively by a set of parameters. This makes the approach capable of producing diverse results. In the following, we showcase example renderings for point clouds with up to 82 million points, employing different pipeline configurations (see Table 1 for the point cloud sources). Please also refer to the supplementary video.

While our prototype mainly showcases our approach's feasibility and is not fully optimized for performance, basic runtime and scene statistics are shown in Table 2 for reference. All images were rendered with a resolution of  $1920 \times 1080$  pixels on a machine equipped with an AMD Ryzen 7 3700-X processor, 32GB RAM, and an NVIDIA RTX 3090 graphics card. The rendering performance of a given scene is influenced by numerous factors, including number of points, viewpoint, the rendering pipelines and their parameterization, the number of point cloud segments and depth inpainting kernel size.

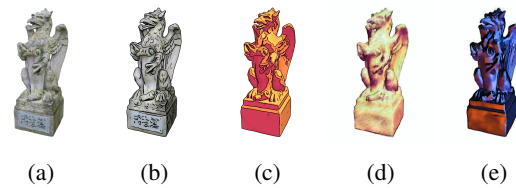


Figure 7: Different IB-AR filters applied to point cloud (a): toon (b), posterization (c), oil painting (d,e).

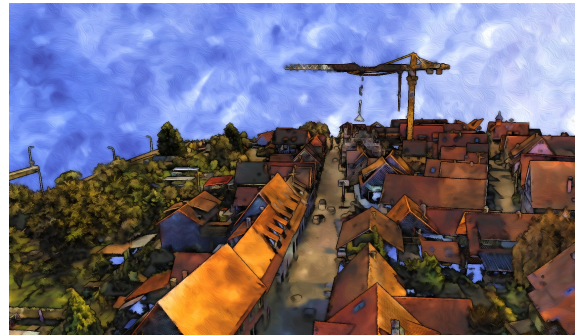
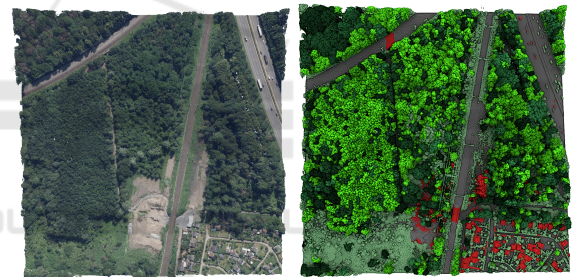


Figure 8: An edge enhancement and oilpaint IB-AR filter applied to a point cloud obtained by a UAV.



(a) Original point cloud (b) Stylized result

Figure 9: NPR based on semantic segmentation and tree instance segmentation.

### 5.1 Exemplary Results

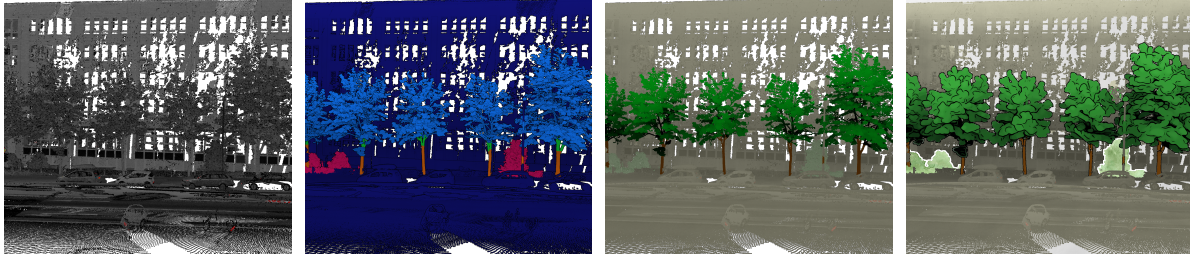
Application of IB-AR to non-segmented point clouds enables aesthetic and diverse rendering results, as demonstrated in Figure 3, Figure 7, and Figure 8. When integrated with segmentation, the approach amplifies the strengths of NPR, such as highlighting key areas, resulting in engaging visualizations. In the following, we present a few application examples of this approach.

**NPR in Vegetation Mapping.** Figure 9 shows a point cloud of a forest area. The NPR is parameterized by segmentation results with respect to coloring and edge thickness to facilitate the simultaneous distinction of semantic classes and individual trees. Figure 10 shows results for a mobile mapping point cloud. The point cloud is segmented into five classes (low veg-



Table 2: Rendering performance for the scenes depicted in this paper (capped at 60 FPS). Please note that for the measurements with an additional number in brackets, an unoptimized CPU-side pencilhatching IB-AR filter is used on some of the segments. The number in brackets reports the performance for the case that this filter is exchanged for a GPU-accelerated cartoon filter. We further state the used segmentation methods, if applicable: SSDL = deep learning semantic segmentation, SSR = rule-based semantic segmentation, CC = connected components analysis, ISA = algorithmic instance segmentation. Please also refer to Section 4 for more details on the segmentation methods.

Figure	1(b)	3(b)	4(b)	7(b)-(e)	8	9(b)	10(d)	11	12(d) upper	12(d) middle	12(d) lower
Number of Points	18.3M	4.8M	2.5M	3M	82M	12M	64.8M	2.5M	2.5M	18.3M	12M
Segmentation	SSDL	-	CC	-	-	SSR+ISA	SSDL	CC	CC	SSDL	SSR
FPS	53	60	60	60	48	60	32	45 (60)	12 (60)	22 (45)	12 (60)



(a) Reflectance intensity (b) Semantic segmentation (c) Class-based coloring (d) Stylized vegetation

Figure 10: Application example in the area of urban vegetation mapping.

etation, tree trunks, tree branches, tree crowns, non-vegetation) using a DL approach (Figure 10b). Then, coloring (Figure 10c), as well as cartoon and watercolor filtering (Figure 10d) are applied to the vegetation classes, using black outlines for trees and white outlines for low vegetation.

**NPR for Urban Visualization.** The point cloud shown in Figure 1 was obtained via mobile mapping. Preprocessing involved the exclusion of points assigned to the transportation category (e.g., buses, bicycles). Vegetation elements were recolored to green, pedestrians to red, and architectural structures to beige. To conceal gaps within the point cloud and augment the overall aesthetic quality, a watercolor effect was then applied to the entire image. By smoothing larger surfaces (e.g., building facades) and emphasizing smaller objects (e.g., pedestrians), the configuration of the urban landscape can be observed more easily.

**Animations via Warp Filtering.** In Figure 11, we employed a warping-based geometric transformation to the segmented representation of a whale skeleton, with temporal parameterization to simulate the motion of swimming (see the supplementary video). Our depth inpainting approach ensures visual coherence by maintaining the correct depth ordering for the transformed image pixels

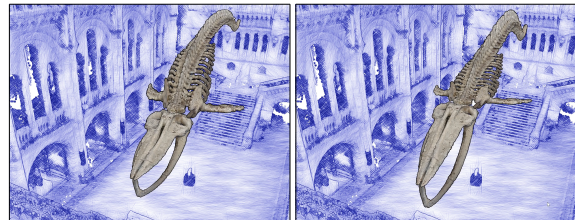
## 5.2 Comparison of Compositing Methods

Figure 12 illustrates our compositing technique and compares it to alternative approaches:

**Layering without Depth Test.** As demonstrated in Figure 12b, a naive layering of per-segment results is inadequate if segments overlap. This approach fails to resolve occlusions correctly, leading to visual errors.

**Layering with Per-Fragment Depth Tests.** Incorporating per-fragment depth tests (Figure 12c) improves the compositing of layers by ensuring that segments obstruct each other appropriately. However, this method produces visual artifacts at segment boundaries, especially when a cartoon filter is applied. Boundaries may be added at pixels without depth information, resulting in a jittery appearance as the filter’s overdraw effect is not integrated seamlessly.

**Our Depth Inpainting Approach.** Our depth inpainting approach (Figure 12d) addresses these shortcomings, leading to smooth outlines and, thus, a more uniform and aesthetically pleasing representation. As the missing depth information at segment boundaries is effectively estimated, a cohesive and visually stable output is ensured.



(a) Timestamp 1 (b) Timestamp 2

Figure 11: A geometric warp transformation that is parameterized over time is applied to one of the segments.

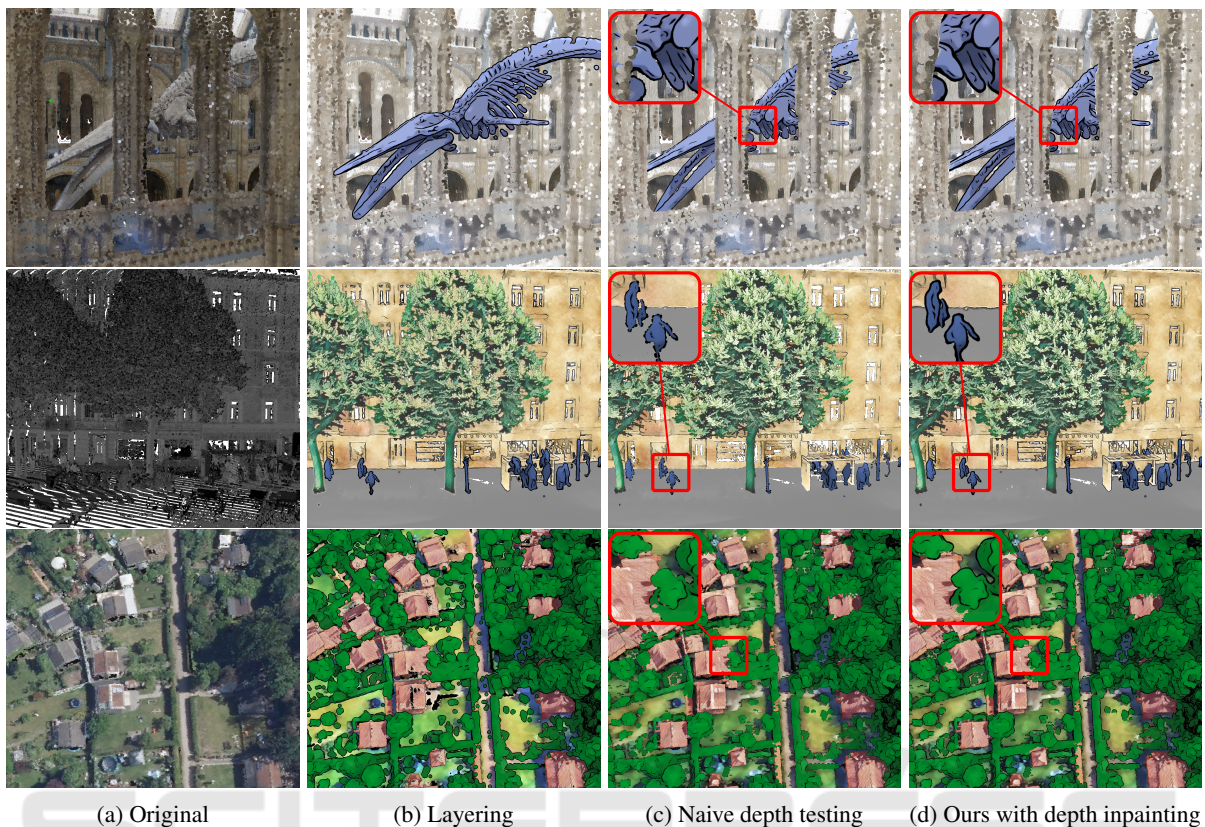


Figure 12: Comparison of our approach to other compositing approaches.

### 5.3 Discussion

One notable advantage of using image-space filtering methods for segment-based NPR of point clouds is their independence from specialized data structures, often required for organizing point clouds in object-space approaches, as well as their generality - any image filtering method can be applied without needing adaption to point cloud geometry. However, combining image-space NPR with segment-based point cloud rendering raises the following issues:

**Validity of Pixel Information.** Image-based NPR approaches often assume that each pixel contains valid data. However, when rendering sparse point clouds, pixels may represent background values rather than meaningful information, which can cause problems with image-based processing steps. For example, the sparsity of point clouds can cause edge filters to place an excessive number of edges around single points. To mitigate this problem, gap-filling techniques can be used, such as increasing the point size or using splatting. Alternatively, image-based smoothing can be combined with our depth inpainting approach.

**Handling Intersecting Segments and Edge Zones.** In segment-based point cloud rendering, dealing with

overlapping segments and ambiguous boundary zones is challenging and often requires manual adjustment of the parameters used to compose the per-segment images. However, the proposed parameters of our depth inpainting approach allow for manual adjustment and provide a degree of control over these complex areas.

**Depth Precision Constraints.** In scenarios where the far plane is very far away from the camera, depth precision may be insufficient for distant parts of the scene. This can lead to inaccuracies during depth testing, resulting in visual artifacts when compositing the per-segment images.

Regardless of whether object- or image-space methods are used, the following additional issues arise with segment-based NPR of point clouds:

**Inter-Frame Consistency.** Achieving consistent rendering results for different camera viewpoints is challenging, especially when the user can interactively control the camera. Because camera movements can lead to variations in the screen-space point density, visual artifacts such as flickering and popping can occur.

**Segmentation Quality Dependence.** As the rendering techniques and their parameters are selected based on the preceding segmentation, the quality of the final image is greatly influenced by the segmentation quality. In case of erroneous segmentation, the visual quality of the result can be significantly degraded, potentially even communicating incorrect information (e.g., in Figure 1, the lantern pole has been classified as a pedestrian).

In summary, the successful implementation of segment-based image-space NPR for point clouds currently requires high-quality data and segmentation results, as well as careful configuration of user parameters. If these constraints are met, it is an effective approach to reduce noise or highlight certain objects in point cloud renderings, as our results show.

## 6 CONCLUSIONS AND FUTURE WORK

The integration of point clouds with NPR techniques has significant potential for various applications. NPR of point clouds improves the clarity of their depiction, reduces visual complexity, and enables a wide range of expressive graphical representations. We demonstrated that in this context the integration of IB-AR provides a high degree of artistic freedom and enables diverse stylization results. Further, we have shown that point cloud segmentation allows to enhance and combine existing NPR techniques, improving object perception and enabling the highlighting of specific objects or semantic classes. To counter artifacts that can arise during composition of per-segment rendering results, we proposed a depth inpainting approach that effectively deals with complex scenes comprising overlapping and intersecting segments. As our overall approach strongly depends on the quality of the point cloud segmentation, more accurate segmentation techniques are required to further enhance the stylization results in the future. The development of foundational models through suitable pretraining methods seems to be a promising research direction. Further, in our current approach, the stylization and depth inpainting parameters need to be fine-tuned manually. In the future, automatic deduction of these parameters from point cloud properties (e.g., density or segment overlap) could be investigated. Overall, our work shows that point clouds are not only useful for analysis tasks, which is perceived as their main application in current scientific literature, but also for artistic stylization and the creation of helpful and engaging visualizations.

## ACKNOWLEDGEMENTS

We thank AllTerra Deutschland GmbH and the Department for Geoinformation, Surveying and Cadastre of the City of Essen for providing mobile mapping data. This work was partially funded by the Federal Ministry of Education and Research, Germany through grant 01IS22062 ("AI research group FFS-AI") and grant 033L305 ("TreeDigitalTwins").

## REFERENCES

- Aliev, K., Sevastopolsky, A., Kolos, M., Ulyanov, D., and Lempitsky, V. S. (2020). Neural point-based graphics. In *Proc. European Conf. Computer Vision, ECCV*, pages 696–712. Springer.
- Awano, N., Nishio, K., and Kobori, K. (2010). Interactive stipple rendering for point clouds. In *Proc. 18th Int. Conf. Central Europe Computer Graphics, Visualization and Computer Vision, WSCG*, pages 193–198. Václav Skala-UNION Agency.
- Bath, U., Shekhar, S., Tjabben, H., Semmo, A., Döllner, J., and Trapp, M. (2022). Trios: A framework for interactive 3D photo stylization on mobile devices. In *Proc. Int. Conf. Graphics and Interaction, ICGI*. IEEE.
- Bello, S. A., Yu, S., Wang, C., Adam, J. M., and Li, J. (2020). Review: Deep learning on 3D point clouds. *Remote Sens.*, 12(11):1729.
- Bousseau, A., Kaplan, M., Thollot, J., and Sillion, F. X. (2006). Interactive watercolor rendering with temporal coherence and abstraction. In *Proc. 4th Int. Symp. Non-Photorealistic Animation and Rendering, NPAR*, pages 141–149. ACM.
- Bui, G., Le, T., Morago, B., and Duan, Y. (2018). Point-based rendering enhancement via deep learning. *Vis. Comput.*, 34(6-8):829–841.
- Burmeister, J.-M., Richter, R., and Döllner, J. (2023). Concepts and techniques for large-scale mapping of urban vegetation using mobile mapping point clouds and deep learning. In *Proc. Digital Landscape Architecture Conf.*, pages 451–462. Wichmann.
- Cao, X., Wang, W., Nagao, K., and Nakamura, R. (2020). Psnet: A style transfer network for point cloud stylization on geometry and color. In *Proc. IEEE Winter Conference Applications of Computer Vision, WACV*, pages 3326–3334.
- DeCarlo, D. and Santella, A. (2002). Stylization and abstraction of photographs. *ACM Trans. Graph.*, 21(3):769–776.
- Döllner, J. (2008). Visualization, photorealistic and non-photorealistic. In *Encyclopedia of GIS*, pages 1223–1228. Springer.
- Dürschmid, T., Söchting, M., Semmo, A., Trapp, M., and Döllner, J. (2017). ProsumerFX: Mobile design of image stylization components. In *Proc. SIGGRAPH Asia Mobile Graphics & Interactive Applications*, pages 1:1–1:8. ACM.
- Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int.*

- Conf. Knowledge Discovery and Data Mining, KDD*, pages 226–231. AAAI Press.
- Gooch, A. A., Long, J., Ji, L., Estey, A., and Gooch, B. (2010). Viewing progress in non-photorealistic rendering through heinlein’s lens. In *Proc. 8th Int. Symp. Non-Photorealistic Animation and Rendering, NPAR*, pages 165–171. ACM.
- Hao, W., Zuo, Z., and Liang, W. (2022). Structure-based street tree extraction from mobile laser scanning point clouds. In *Proc. 5th Int. Conf. Image and Graphics Processing, ICIGP*, pages 373–379. ACM.
- Hertzmann, A. (1998). Painterly rendering with curved brush strokes of multiple sizes. In *Proc. 25th Annu. Conf. Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 453–460. ACM.
- Höllein, L., Johnson, J., and Nießner, M. (2022). StyleMesh: Style transfer for indoor 3D scene reconstructions. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, CVPR*, pages 6198–6208.
- Horand, R., Hansard, M. E., Evangelidis, G. D., and Ménier, C. (2016). An overview of depth cameras and range scanners based on time-of-flight technologies. *Mach. Vis. Appl.*, 27(7):1005–1020.
- Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., and Song, M. (2020). Neural style transfer: A review. *IEEE Trans. Vis. Comput. Graph.*, 26(11):3365–3385.
- Kornilov, A. S. and Safonov, I. V. (2018). An overview of watershed algorithm implementations in open source libraries. *J. Imaging*, 4(10):123.
- Kyprianidis, J. E., Collomosse, J., Wang, T., and Isenberg, T. (2013). State of the “art”: A taxonomy of artistic stylization techniques for images and video. *IEEE Trans. Vis. Comput. Graph.*, 19(5):866–885.
- Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rotensteiner, F., Wegner, J. D., and Ledoux, H. (2021). The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and multi-view-stereo. *ISPRS J. Photogramm. Remote Sens.*, 1:11.
- Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., and Li, J. (2021). Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Trans. Neural Networks Learn. Syst.*, 32(8):3412–3432.
- Liu, X.-C., Cheng, M.-M., Lai, Y.-K., and Rosin, P. L. (2017). Depth-aware neural style transfer. In *Proc. 15th Int. Symp. Non-Photorealistic Animation and Rendering, NPAR*, pages 4:1–4:10. ACM.
- Luo, H., Khoshelham, K., Chen, C., and He, H. (2021). Individual tree extraction from urban mobile laser scanning point clouds using deep pointwise direction embedding. *ISPRS J. Photogramm. Remote Sens.*, 175:326–339.
- Mirzaei, K., Arashpour, M., Asadi, E., Masoumi, H., Bai, Y., and Behnood, A. (2022). 3D point cloud data processing with machine learning for construction and infrastructure applications: A comprehensive review. *Adv. Eng. Informatics*, 51:101501.
- Mu, F., Wang, J., Wu, Y., and Li, Y. (2022). 3D photo stylization: Learning to generate stylized novel views from a single image. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, CVPR*, pages 16273–16282.
- Rabbani, T., van den Heuvel, F., and Vosselman, G. (2006). Segmentation of point clouds using smoothness constraints. In *Proc. ISPRS Commission V Symposium*, pages 248–253.
- Ribes, A. and Boucheny, C. (2011). Eye-dome lighting: A non-photorealistic shading technique. *Kitware Source Quarterly Magazine*, 7.
- Richter, R., Discher, S., and Döllner, J. (2015). Out-of-core visualization of classified 3D point clouds. In *3D Geoinformation Science: Selected Papers of the 3D GeoInfo 2014*, pages 227–242. Springer.
- Scheiblauer, C. (2014). *Interactions with gigantic point clouds*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology.
- Semmo, A., Limberger, D., Kyprianidis, J. E., and Döllner, J. (2016). Image stylization by interactive oil paint filtering. *Comput. Graph.*, 55:157–171.
- Shekhar, S., Reimann, M., Mayer, M., Semmo, A., Paseswaldt, S., Döllner, J., and Trapp, M. (2021). Interactive photo editing on smartphones via intrinsic decomposition. *Comput. Graph. Forum*, 40(2):497–510.
- Thomas, H., Qi, C. R., Deschaut, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). KPConv: Flexible and deformable convolution for point clouds. In *Proc. IEEE/CVF Int. Conf. Computer Vision ICCV*, pages 6410–6419.
- Wagner, R., Wegen, O., Limberger, D., Döllner, J., and Trapp, M. (2022). A non-photorealistic rendering technique for art-directed hatching of 3D point clouds. In *Proc. 17th Int. Joint Conf. Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP*, pages 220–227. SCITEPRESS.
- Wang, W., Yu, R., Huang, Q., and Neumann, U. (2018). SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, CVPR*, pages 2569–2578.
- Wegen, O., Döllner, J., Wagner, R., Limberger, D., Richter, R., and Trapp, M. (2022). Non-photorealistic rendering of 3D point clouds for cartographic visualization. *Abstr. Int. Cartogr. Assoc.*, 5:161.
- Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J. Photogramm. Remote Sens.*, 105:286–304.
- Westoby, M. J., Brasington, J., Glasser, N. F., Hambrey, M. J., and Reynolds, J. M. (2012). ‘Structure-from-Motion’ photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314.
- Winnemöller, H., Olsen, S. C., and Gooch, B. (2006). Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226.
- Xie, Y., Tian, J., and Zhu, X. X. (2020). Linking points with labels in 3D: A review of point cloud semantic segmentation. *IEEE Geosci. Remote Sens. Mag.*, 8(4):38–59.
- Xu, H., Gossett, N., and Chen, B. (2004). PointWorks: Abstraction and rendering of sparsely scanned outdoor environments. In *Proc. 15th EG Workshop on Rendering Techniques, EGWR*, pages 45–52. Eurographics Association.
- Zwicker, M., Pfister, H., van Baar, J., and Gross, M. H. (2001). Surface splatting. In *Proc. 28th Annu. Conf. Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 371–378. ACM.