GRAPH-BASED POINT CLOUD DENOISING USING SHAPE-AWARE CONSISTENCY FOR FREE-VIEWPOINT VIDEO

Keisuke Nonaka [†], *Ryosuke Watanabe* [†], *Haruhisa Kato* [†], *Tatsuya Kobayashi* [†], *Eduardo Pavez* [‡], *Antonio Ortega* [‡]

[†]KDDI Research, Inc. [‡] University of Southern California

ABSTRACT

We propose a novel graph-based denoising method to correct the quantization error (step noise) arising in the process of generating the visual hull, a commonly used technique to synthesize free-viewpoint video. To reduce this step noise effectively, we propose two new notions of consistency, pixel value consistency and normal vector consistency. The resulting denoising method involves a first step of graph construction using the proposed consistency metrics, followed by graph filtering of the 3D point cloud coordinates. Our experiments show that our approach provides visually and quantitatively better performance than state-of-the-art methods.

***Index Terms*—** Free-viewpoint video, point cloud, denoising, shape refinement, shape smoothing

1. INTRODUCTION

Free-viewpoint video [1, 2, 3, 4] is one of the most promising new video viewing experiences, allowing users to watch images of a subject seen from a virtual perspective, which does not correspond to an actual physical camera. A noteworthy approach for realizing practical free-viewpoint video is the *visual hull* [3, 5, 6], which generates a subject shape as a 3D model by projecting multiple silhouettes of the subject onto a 3D space. Two main advantages of this approach are that (i) it can use a relatively small number of cameras for 3D model generation (there is only limited quality degradation even with sparse camera placement) and (ii) that it does not require a depth sensor, making it flexible enough for outdoor shooting. On the other hand, one of the challenges encountered in using the visual hull method in practice is that the shape representation is limited to a quantized voxel grid, obtained by splitting the 3D space into cubes with a pre-specified size. This causes errors in the estimation of subject shape, which impairs the quality of the synthesized image given that texture mapping is performed with incorrect coordinates.

The above 3D models are mainly represented as polygons or a point cloud, and techniques collectively referred to as point cloud denoising (PCD), which improve the quality of 3D model shapes by smoothing the 3D coordinates of the point cloud, have been proposed. Among them, graph-based PCD [7, 8, 9] and learning-based PCD [10, 11] have achieved remarkable results in recent years. Graph-based PCD techniques start by constructing a graph that takes into account the relation between non-uniform points and optimizes the graph in the graph frequency domain. However, these graph-based PCD techniques only aim to remove additive noise with a given probability distribution, such as Gaussian noise, and do not take into account the quantization errors (*step noise* on the surface). Learning-based PCD methods also assume additive noise, and are trained with data affected by type of noise. In our experiments we found that results are highly dependent on the training dataset and are not suitable for removing step noise (see Section 3). Consequently, when conventional methods are applied to correct step noise they do not provide a sufficiently good denoising quality.

In this paper, we propose a graph-based PCD for step noise, with an application to 3D point clouds of objects obtained by approaches using multiple camera images such as visual hull. While other graph-based PCD methods just use Gaussian kernel or inverse distance as edge weights of a graph, our method modifies the weights by using consistency metrics. In particular, considering the features of the generation process, we propose two notions of quality-related consistency, namely, *pixel value consistency* (PVC) and *normal vector consistency* (NVC), which capture the consistency of pixel values in the camera images mapped on each point and the consistency of normal vector directions among neighboring points, respectively. PVC and NVC update the weights of the graph to smooth out the shape errors including the step noise. In our experiments, we show that our method can suppress the step noise in the process of visual hull generation compared with the state-of-the-art (SOTA) graph-based and learning-based PCD.

A summary of the contributions of our method is as follows:

- We propose a denoising method for 3D models. A novel feature of our method is that it uses multiple raw images from the cameras used in the model acquisition process, e.g., visual hull. The potential benefits of using raw camera data, as well as the 3D model, have not been widely studied.
- We propose two new metrics, PVC and NVC, for evaluating synthesized image quality and surface smoothness of a subjective model, under the assumption that we can use the multiple raw images for model acquisition.
- We show a way to combine these two metrics with graph-based point-cloud denoising (PCD) and improve its performance, so that we can outperform state of the art PCD methods on a noise removal task.

2. PROPOSED POINT CLOUD DENOISING

The flowchart of the proposed method is shown in Fig. 1. The proposed method is inspired by graph Laplacian regularization (GLR) [7], one of the fastest graph-based PCD methods. More specifically, the 3D coordinates of the point cloud are optimized by filtering in the graph frequency domain as in the conventional methods. However, our proposed method modifies the weight matrix that defines the graph structure in [7] by using criteria based on PVC and NVC. With this modification, the strength of denoising changes point-by-point, and the denoising process is better suited to the point cloud acquired by the visual hull. In the following subsections, we describe each process in detail.

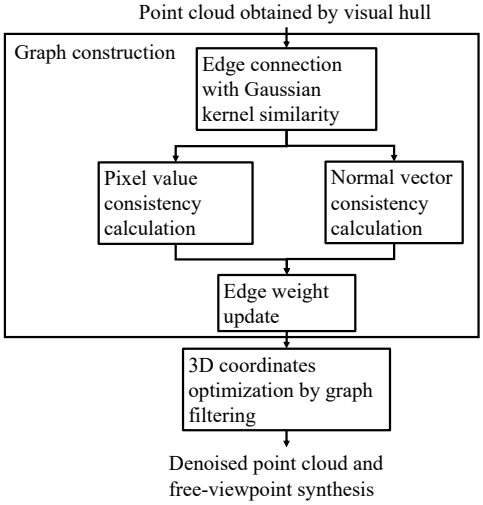


Fig. 1. Flowchart of the proposed method.

2.1. Graph notation and edge connection

We treat a point cloud as an undirected weighted graph $G = \{V, E, \mathbf{W}\}$ composed of a vertex (point $\boldsymbol{p} = [x, y, z]^{\mathrm{T}}$) set V with cardinality $|V| = n$. An edge set E with cardinality $|E| = m$ connects those vertices. The adjacency $\mathbf{W}$ is a real symmetric $n \times n$ matrix, where its element $w_{i,j} = w_{j,i}$ represents the weight of an edge between a point $\boldsymbol{p}_i$ and a point $\boldsymbol{p}_j$. To obtain the edge set E and its weights, we use the k-nearest neighbor (k-NN) method, which is the most basic edge connection method according to the existing scheme of graph signal processing. Here, k is a parameter indicating the number of neighboring points to be connected from a point. As described in Section 1, the conventional graph-based PCD methods use Gaussian kernel or inverse distance edge weights $w_{i,j}$, while in our method $w_{i,j}$ is chosen to suppress the step noise. We introduce our two techniques next. Note that we work with point clouds generated using images taken by multiple cameras (e.g., using the visual hull) *and* we assume that those raw images are available to us.

2.2. Pixel value consistency (PVC)

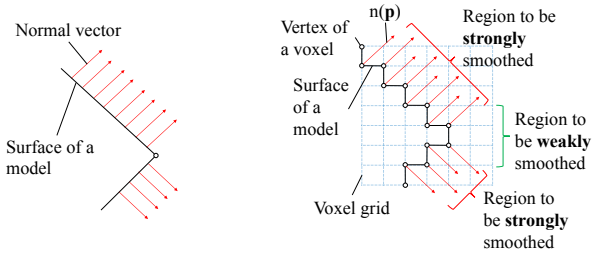
To design the PVC criterion, we focused on the fact that a shape error causes an error in the pixel coordinates to be mapped from each camera. Based on this observation, we calculate a measure of PVC, which expresses the consistency of pixel values (color or luminance) in the camera images mapped on each point. In particular, for a given point $\boldsymbol{p}$ let $I_i(\boldsymbol{p})$ ($i \in \{1, 2, \ldots, C\}$) be the pixel luminance of the pixel corresponding to that point in camera image I_i. Using this value, the PVC ϕ on $\boldsymbol{p}$ is defined as follows:

$$\phi(\boldsymbol{p}) = \mathrm{sdev}(I_i(\boldsymbol{p}))/255, \tag{1}$$

where the function $\mathrm{sdev}(\cdot)$ is the standard deviation of the pixel luminance over all camera images where $\boldsymbol{p}$ is *visible*. Thus, when calculating ϕ the pixel intensities of a camera where $\boldsymbol{p}$ is occluded are not used in the calculation of the standard deviation.

By using $\phi(\boldsymbol{p})$, the difference in PVC between point $\boldsymbol{p}_i$ and point $\boldsymbol{p}_j$, $w_{i,j}^{PVC}$, is used to define an edge weight between those two points:

$$w_{i,j}^{PVC} = \frac{|\phi(\boldsymbol{p}_i) - \phi(\boldsymbol{p}_j)|}{\mathrm{MAX}_{\mathrm{all}}(|\phi(\boldsymbol{p}_i) - \phi(\boldsymbol{p}_j)|)}, \tag{2}$$



(a) Surface of a 3DCG model on a continuous (ideal) coordinates domain

(b) Surface of a 3DCG model on a voxel grid domain

Fig. 2. Example of normal vectors in two shape representations. A plane that should be represented smoothly like (a) has step noise by quantizing with a voxel grid as (b).

where the function $\mathrm{MAX}_{\mathrm{all}}(\cdot)$ returns the maximum value among all $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ pairs and works as a normalization term, which limits all weights $w_{i,j}$ to be the interval $[0, 1]$ as is the case for most kernel functions. The larger the discrepancy between the pixel values mapped from each camera, the larger the value of $w_{i,j}^{PVC}$, so that the larger the error in the point cloud coordinates, the stronger the smoothing will be, according to the optimization described below.

2.3. Normal vector consistency (NVC)

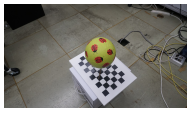
The point cloud obtained by the visual hull approach is quantized into a voxel grid. Thus, the plane often has step noise (Fig. 2 (b)). However, ideally the plane on the step noise should be a smooth plane as shown in Fig. 2 (a). Even in this case, the normal vectors of the neighboring points that make up the plane should be similar. On the other hand, the normal vectors corresponding to neighboring points of a shape edge that covers a wider area are different. Our NVC is designed based on these assumptions about the normal vectors.

The conventional method [9] incorporates smoothness of the normal vector into its objective function and achieves good results. However, since the calculation of normal vectors is non-linear mapping, it is difficult to directly optimize the point cloud coordinates. Thus, while [9] treated a target point cloud as a bipartite graph (i.e., separated a graph into two set of points) and has proposed an approximate optimization scheme by alternately optimizing the coordinates of the points belonging to each set so that the normal vectors are smooth, its convergence is not guaranteed.

Our NVC approach is also based on normal vector smoothness. However, in order to reduce computational complexity, we use normal vectors estimated from the original (noisy) point cloud to select the (static) graph edge weights. Thus the graph weights based on our proposed NVC criterion between point $\boldsymbol{p}_i$ and point $\boldsymbol{p}_j$, $w_{i,j}^{NVC}$, are given by:

$$w_{i,j}^{NVC} = cos(n(\boldsymbol{p}_i), n(\boldsymbol{p}_j)) = \frac{n(\boldsymbol{p}_i) \cdot n(\boldsymbol{p}_j)}{|n(\boldsymbol{p}_i)||n(\boldsymbol{p}_j)|}, \tag{3}$$

where $n(\boldsymbol{p}_i)$ and $n(\boldsymbol{p}_j)$ are the normal vectors which are defined as an average of the normal vectors of planes that $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ belong to, respectively. The operator $\cdot$ is an inner product of two vectors for calculation of the cosine similarity. In the following, we refer to the edges of a point cloud shape as shape edges to distinguish them from edges on a graph. NVC aims to reduce the weight of regions where the difference in normal vectors from neighboring points is large.


(a) Ball


(b) Hula


(c) Baseball

Fig. 3. Examples of input image. Each caption is scenario name.

Scenario	Ball	Hula	Baseball
# of cameras	32	9	16
camera position	hemisphere	semicircle	semicircle
# of points	158,990	37,460	126,754

Table 1. Our dataset settings

2.4. Edge weight update and optimization of point cloud coordinates

The proposed method improves the conventional method by adaptively changing the weight $w_{i,j}$ with weights derived from (2) and (3):

$$w_{i,j} = \frac{\alpha w_{i,j}^{S} + \beta w_{i,j}^{PVC} + \gamma w_{i,j}^{NVC}}{\mathrm{MAX}_{\mathrm{all}}(\alpha w_{i,j}^{S} + \beta w_{i,j}^{PVC} + \gamma w_{i,j}^{NVC})}, \tag{4}$$

where, the variables α, β, and γ adjust the degree of influence on denoising of each weight based on similarity and consistency. Then, we optimize the coordinates of $\boldsymbol{p}$ following the scheme of the conventional method, GLR [7].

3. EXPERIMENTS

In this section, we conduct comparison experiments with SOTA methods. These methods are RIMLS [12] which assumes a point cloud is a locally smooth surface and denoises it by projecting points onto the surface, and GPDNet [10] and DMR [11] as learning-based methods which apply deep neural network scheme to denoising, and GLR [7], GTV [7] and FGLR [9] as graph-based methods similar to our method. RIMLS is processed using the open software Meshlab [13]. For GPDNet and DMR, we used the author's code and pretrained model distributed on Github [14, 15]. For GLR, GTV and FGLR, because we were unable to access the original implementation, we used our own implementation.

3.1. Input dataset details

We prepared an image dataset suitable for sparse camera placement. Examples and the details of our dataset are shown in Fig. 3 and Table 1. In addition to our dataset, as an open dataset with camera parameters and image pairs, "DinoRing" and "TempleRing" shown in another paper [16] are also used.

3.2. Denoising shape results

In this section, due to lack of space we only show and discuss the result of the 3D model shape in the Baseball and Hula scenario. However, the results are similar for other scenarios (Fig. 4). For RIMLS, the default values of Meshlab are used for all parameters. For GPDNet and DMR, we used the default parameter, but since these methods discard the original surface information, we reconstruct the surface using the Ball Pivoting function implemented in Meshlab and fill the holes of the model using a conventional surface generation method [17]. For GLR, GTV and FGLR, a parameter that determines the strength of the regularization is the same value as in the proposed method. Among these, for GTV and FGLR, which perform iterative optimization, the iterations are continued until the variation in the objective energy becomes less than a threshold or until we execute 200 iterations. In addition, the parameters that determine the balance of each consistency and the number of neighboring points are set to $\alpha = 1, \beta = 1, \gamma = 0.1$, and $k = 5$ for all the experiments. The reason for adopting the small value of $\gamma = 0.1$ is to avoid the effect of NVC becoming too large because the cosine similarity with neighboring points approaches 1 in many regions of step noise and planes. As an ablation study to elucidate the effect of this small NVC, we also evaluate the proposed method without NVC ($\gamma = 0$).

As shown in the results of Baseball scenario, the original data have step noise on the surface due to the quantization caused by the voxel grid. RIMLS which approximates the surface to a plane, is less effective in smoothing, because it is mapped to flat regions contained in the step noise. The results of GPDNet and DMR show that the noise, which is the uneven surface of the objects (e.g. a baseball bat), is not properly removed, even if we exclude the fact that the holes are not completely filled due to the surface reconstruction and the quality is degraded. This is due to the lack of match between the pretrained model and our target task, indicating that the learning-based methods are highly dependent on the training data. GLR shows relatively good results, but the step noise is still present compared to the proposed method. An interesting result is that the GTV results show block-wise artifacts on its surface. This is because GTV works to decrease the ℓ^1-norm (sparseness) of the difference with neighboring points and align the points along a voxel grid. As a result, each surface faces a different direction individually. This implies that an objective function based on the ℓ^1-norm of the difference with neighboring points, such as GTV, is not effective in removing step noise, which is the objective of this paper. Similar to RIMLS, FGLR is also not effective at removing step noise. This is because the approximation of optimization with a bipartite graph does not match our target problem and we confirmed the energy of the objective function did not converge even after 200 iterations.

Compared with conventional methods, our proposed method effectively smooths out this step noise, resulting in a smooth surface. In particular, for NVC, it can be confirmed that our method reduces the step noise better than if NVC is not used. On the other hand, the thinning of the bat is an adverse effect of strong smoothing due to the small difference among the normal vectors of neighbor points on a gentle curve. This result shows the limitation of our method, which requires a rough estimation of the true shape features in advance for more precise denoising by adaptively adjusting the weight of each consistency term for each region.

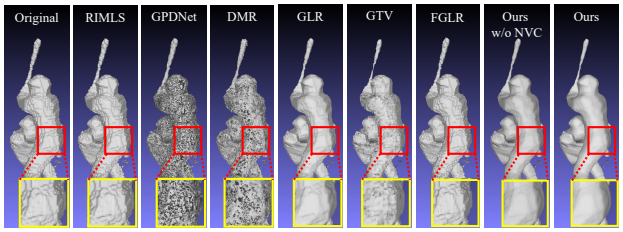
In addition, from the results of Hula scenario, we show the edge quality after denoising. As shown in the enlarged region in the center of Fig. 4 (b), we can see that in the sharp edge region of the skirt, the proposed method maintains almost the same edge quality as conventional methods, such as GLR, while reducing the step noise.

3.3. Numerical results

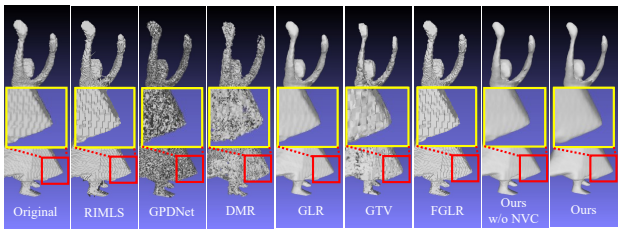
In this section, the evaluation is performed on the numerical score for each denoising quality. The point cloud model obtained by visual hull does not have a model that shows the true object shape (groundtruth). In addition, an important point of the free-viewpoint application is that the synthesized colors mapped from the input cameras are similar among all cameras. Thus, we apply the averaged PVC for all weights $w_{i,j}^{PVC}$ that can objectively evaluate the degree of misalignment of mapped pixel values. Intuitively, this score can be

	Original	RIMLS	GPDNet	DMR	GLR	GTV	FGLR	Ours without NVC	Ours
DinoRing	2.94 (2.16)	3.08 (2.30)	4.74 (3.44)	5.75 (3.02)	1.50 (1.14)	2.29 (1.83)	3.12 (2.41)	1.57 (1.48)	**1.31** (**1.07**)
TempleRing	2.81 (2.00)	2.99 (2.14)	4.35 (2.63)	4.71 (2.95)	1.75 (1.26)	2.63 (2.08)	3.06 (2.21)	1.66 (1.54)	**1.64** (**1.20**)
Ball	1.63 (1.25)	1.85 (1.46)	9.72 (4.01)	9.45 (3.94)	1.37 (1.07)	2.34 (1.81)	2.10 (1.42)	1.31 (1.02)	**1.26** (**0.98**)
Hula	3.38 (2.96)	3.41 (2.99)	8.14 (5.45)	7.48 (5.15)	3.41 (2.98)	4.57 (3.74)	4.62 (3.15)	3.37 (2.94)	**3.16** (**2.75**)
Baseball	3.91 (3.54)	4.05 (3.67)	11.2 (7.98)	12.1 (8.24)	3.29 (3.21)	4.09 (3.96)	4.98 (3.76)	3.24 (3.19)	**3.01** (**3.01**)

Table 2. Evaluation results for PVC. The mean PVC values are shown, and the numbers in parentheses represent the standard deviation. All values are shown with "$\times 10^{-2}$" omitted. Bold type indicates the smallest value among all methods.



(a) Baseball



(b) Hula

Fig. 4. Synthesized images by the conventional methods and our method (the image with the yellow frame shows an enlarged image with the red frame).

considered as percentages of points with a larger standard deviation of mapped pixel values (i.e., points with degraded image quality) compared to neighboring points.

As shown in Table 2, the PVC value of the proposed method and GLR show good results, but GLR has a slight loss of quality compared to the proposed method. In addition, our method without NVC has a lower PVC score than GLR on the DinoRing dataset. This indicates that our method has a limitation that it may be difficult to make use of PVC effect in homogeneous textured objects like DinoRing, but NVC improves the stability and accuracy of denoising even with such objects. We can also see that most of the conventional methods show a large PVC value compared to the original, which objectively suggests that there is quality degradation. The above is generally consistent with the intuitive results of Subsection 3.2.

Optimization of parameters for our method (e.g., α) is left for future work, but we have experimentally confirmed that the results of our method do not change significantly when we use a certain range of parameter settings. For example, in Baseball data, we confirmed that the value of PVC of our method with the combination of $\alpha = 1$, $\beta \in \{0.5,\ 1\}$, $\gamma \in \{0.1,\ 0.3,\ 0.5\}$ leads to results in the range from 2.8×10^{-2} to 3.1×10^{-2}.

3.4. Computational complexity

The processing time using Hula data for each method is shown in Table3. Note that GLR, GTV, FGLR, and the proposed method include the preprocessing time required for the denoising process (e.g. construction of k-NN graph). In the derivation of the GTV solution, the iteration process by non-linear optimization is terminated when the absolute difference from the previous value is less than 5.0×10^{-3}. For FGLR, we carry out the iteration process by 200 times because the objective function does not converge using our dataset. The time of GPDNet and DMR includes only inference time. In GTV, only the inverse matrix calculation in each iteration, which has extremely high computational complexity, is solved by the parallelized conjugate gradient method and runs on the CPU. GLR and our method are also run on the CPU but without parallelization. From the table, we can see that the proposed method is capable of denoising with computation time that is slightly higher than that of GLR, but much lower than those of GPDNet, DMR, GTV, and FGLR.

	RIMLS	GPDNet	DMR	GLR	GTV	FGLR	Ours
Hula	0.64	32.48	11.08	0.19	81.93	48.06	0.92

Table 3. Comparison of the processing time in seconds.

The difference in processing time between GLR and the proposed method is the overhead incurred in calculating PVC and NVC. This overhead time, assuming that the independent calculation of PVC and NVC for each point could be completely parallelized on the GPU, has the following computational complexity. For PVC, the calculation of standard deviation in Eq. 1 has complexity $O(C)$, where C is the number of cameras, and subtracting ϕ and calculating its maximum value among all edges in Eq. 2 has complexity $O(m)$ For NVC, on the other hand, the complexity of calculating both the normal vector and cosine similarity in Eq.3 is $O(1)$. In addition, the edge weight update in Eq.4 also has a complexity $O(m)$ the same as the above maximum value calculation. As a result, the total overhead time of our method has a computational complexity $O(m) + O(C)$ which can be considered as $O(m)$ because $C \ll m$ generally. This implies that our method can run faster (as fast as GLR) with GPU parallelization.

4. CONCLUSION

We proposed a method to reduce the step noise caused by the voxel grid for the visual hull by using two quality-related consistencies: PVC for evaluating pixel values mapped from multiple cameras and NVC for strongly smoothing the step noise based on the assumption of a normal vector of an ideal plane. The proposed method improves on the conventional graph-based method which is superior in terms of processing time, and we showed that the proposed method is effective in removing the step noise compared with the SOTA methods. As future work, we will study how to adjust the weights of the consistencies adaptively according to the estimated rough shape features.

5. REFERENCES

[1] Hiroshi Sankoh, Sei Naito, Keisuke Nonaka, Houari Sabirin, and Jun Chen, "Robust billboard-based, free-viewpoint video synthesis algorithm to overcome occlusions under challenging outdoor sport scenes," in *ACM International Conference on Multimedia*, 2018, pp. 1724–1732.

[2] Keisuke Nonaka, Ryosuke Watanabe, Jun Chen, Houari Sabirin, and Sei Naito, "Fast plane-based free-viewpoint synthesis for real-time live streaming," in *IEEE Visual Communications and Image Processing (VCIP)*, 2018, pp. 1–4.

[3] Jun Chen, Ryosuke Watanabe, Keisuke Nonaka, Tomoaki Konno, Hiroshi Sankoh, and Sei Naito, "Fast free-viewpoint video synthesis algorithm for sports scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3209–3215.

[4] Daniel Berjón, Pablo Carballeira, Julián Cabrera, Carlos Carmona, Daniel Corregidor, César Díaz, Francisco Morán, and Narciso García, "FVV live: Real-time, low-cost, free viewpoint video," in *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2020, pp. 1–2.

[5] Uemoto Yusuke, Keita Takahashi, and Toshiaki Fujii, "Free viewpoint video generation system using visual hull," in *International Workshop on Advanced Image Technology (IWAIT)*, 2018, pp. 1–4.

[6] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan, "Image-based visual hulls," in *the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000, pp. 369–374.

[7] Yann Schoenenberger, Johan Paratte, and Pierre Vandergheynst, "Graph-based denoising for time-varying point clouds," in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2015, pp. 1–4.

[8] Chinthaka Dinesh, Gene Cheung, Ivan V. Bajić, and Cheng Yang, "Local 3D point cloud denoising via bipartite graph approximation total variation," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2018, pp. 1–6.

[9] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajić, "Point cloud denoising via feature graph laplacian regularization," *IEEE Transactions on Image Processing*, vol. 29, pp. 4143–4158, 2020.

[10] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli, "Learning graph-convolutional representations for point cloud denoising," in *The European Conference on Computer Vision (ECCV)*, 2020.

[11] Shitong Luo and Wei Hu, "Differentiable manifold reconstruction for point cloud denoising," in *the 28th ACM International Conference on Multimedia*, 2020.

[12] Cengiz Oztireli, Gaël Guennebaud, and Markus Gross, "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression," *Computer Graphics Forum*, vol. 28, no. 2, pp. 493–501, 2009.

[13] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, 2008.

[14] Diego Valsesia, "Learning graph-convolutional representations for point cloud denoising (ECCV 2020)," https://github.com/diegovalsesia/GPDNet, Accessed on: Sept. 1, 2021 [Online].

[15] Shitong Luo, "DMRDenoise," https://github.com/luost26/DMRDenoise, Accessed on: Sept. 1, 2021 [Online].

[16] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 519–528.

[17] Jingwei Huang, Yichao Zhou, and Leonidas Guibas, "Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups," *arXiv preprint arXiv:2005.11621*, 2020.