UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Quality Evaluation, Denoising and Inpainting for Point Clouds Compressed by V-PCC

Permalink

https://escholarship.org/uc/item/0h77z501

Author

Cao, Keming

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA SAN DIEGO

Quality Evaluation, Denoising and Inpainting for Point Clouds Compressed by V-PCC

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Signal and Image Processing)

by

Keming Cao

Committee in charge:

Professor Pamela C. Cosman, Chair
Professor William S. Hodgkiss
Professor Troung Nguyen
Professor Ravi Ramamoorthi
Professor Nuno M. Vasconcelos

2021

Copyright
Keming Cao, 2021
All rights reserved.

The dissertation of Keming Cao is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2021

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I am greatly indebted to my advisor, Professor Pamela Cosman, for her support and guidance during the whole journey of my research. I deeply thank her for her patience and encouragement when I stepped into the new research field of point cloud compression, on which this dissertation is developed. I would also like to thank her for advice and for always believing in me, which helped me to overcome difficulties and inspired me to pursue higher goals in my research. She also taught me how to analyze and think as a researcher and that benefits me a lot and will keep its influence on me in my future career.

I would like to thank Prof. Truong Nguyen, Prof. Nuno Vasconcelos, Prof. William Hodgkiss, and Prof. Ravi Mamamoorthi for serving on my committee. Their feedback and lectures provided me considerable guidance during my Ph.D. progress. Also, I would like to thank Profs. Bhaskar Rao, Robert Lugannani, Gert Lanckriet, Paul Siegel, Mohan M. Trivedi, Manuela Vasconcelos, Ragesh Jaiswal and Joseph Ford from whom I took classes. What I learned in those courses laid the foundation for my research and will keep benefiting me in my future life.

I would like to express my thanks to my lab mates, Jerry Peng and Bentao Zhang. We discussed many things and I derived many inspirations and insights from them. I am also thankful to other former and current colleagues in the lab, Qing Song, Kanke Wu, Peizhi Zhu, Ziyu Ye, Rana Hegazy, Pranav Venuprasad, Yeohee Im, Taiyu Wang, Xiaoya Qian, Fan Li, Li Su, Anna Peng, Li Li, Saygin Artiran and Andrew Gilman. They built up a wonderful atmosphere and made the whole lab a warm place in which to work.

During my Ph.D. study, I was fortunate to be supported by Owlii, Inc., by Samsung Research America, and by the National Science Foundation under grant SCH-1522125.

I am grateful to my supervisors, Dr. Geert Van Der Auwera, and Dr. Adarsh Krishnan Ramasubramonian during my internship at Qualcomm. I would also like to thank Dr. Madhukar Budagavi and Dr. Esmaeil Faramarzi for supervising me during my internship at Samsung Research America. I am thankful to my supervisors and other colleagues for all their

help.

I would like to thank my friends, who have made my seven years in San Diego great experience, including Kunyao Chen, Wenchuan Wei, Sijie Ding, Xinyuan Wang, Lutong Wang, Xueshi Hou, Zhenheng Yang, Xirui Chen, Bo Liu, Yunsheng Li, Pengfei Huang. I would also like to give special thanks to my roommates, Yi Liu and Jun Wang, with whom I share daily stories.

Finally, I want to thank my mother, Jianqin Xing, and my father, Yong Cao, for their endless love and unconditional support. I also want to thank my beloved girlfriend, Yu Xin, for her love and company. This dissertation is dedicated to them.

Chapter 2, in part, is a reprint of the material as it appears in the paper: K. Cao, Y. Xu and P. Cosman, "Patch-aware Averaging Filter For Scaling in Point Cloud Compression," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in the papers: K. Cao, Y. Xu and P. Cosman, "Visual Quality of Compressed Mesh and Point Cloud Sequences," In *IEEE Access*, 8 (2020): 171203-171217. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in the paper: K. Cao and P. Cosman, "Denoising and Inpainting for Point Clouds Compressed by V-PCC," submitted to *IEEE Transactions on Circuits and Systems for Video Technology*. The dissertation author was the primary investigator and author of this paper.

VITA

2014	Bachelor of Engineering, Tsinghua University, China
2017	Master of Engineering, University of California San Diego
2021	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

K. Cao and P. Cosman, "Denoising and Inpainting for Point Clouds Compressed by V-PCC," *IEEE Transactions on Circuits and Systems for Video Technology*, In peer review.

K. Cao, Y. Xu and P. Cosman, "Visual Quality of Compressed Mesh and Point Cloud Sequences," *IEEE Access*, 8 (2020): 171203-171217.

K. Cao, Y. Xu and P. Cosman, "Patch-aware Averaging Filter For Scaling in Point Cloud Compression," *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2018.

K. Cao, Y.T. Peng and P. Cosman, "Underwater Image Restoration using Deep Networks to Estimate Background Light and Scene Depth," *2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, IEEE, 2018.

Y.T. Peng, K. Cao and P. Cosman, "Generalization of the Dark Channel Prior for Single Image Restoration," *IEEE Transactions on Image Processing*, 27.6 (2018): 2856-2868

K. Cao, G. Swamy, S. Chaudhuri, and P. Cosman, "Repeatability and Steadiness of Fingertip Force using Depth Feedback," *IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, IEEE, 2016.

ABSTRACT OF THE DISSERTATION

Quality Evaluation, Denoising and Inpainting for Point Clouds Compressed by V-PCC

by

Keming Cao

Doctor of Philosophy in Electrical Engineering
(Signal and Image Processing)

University of California San Diego, 2021

Professor Pamela C. Cosman, Chair

Augmented reality (AR), virtual reality (VR) and immersive video, as emerging types of multimedia, have recently gained more and more attention. With the development of devices that can capture 3D content and a rapid increase of related applications, the delivery and storage of 3D content have become an important research area. MPEG hosted a Call for Proposals to collect ideas to efficiently compress 3D content in three categories: static point clouds, dynamic point cloud sequences and dynamic acquisition. Among the proposals, Video-based Point Cloud Compression (V-PCC) achieves the highest quality for the second category, dynamic point cloud sequences, under a bit rate constraint.

However, the V-PCC framework is not spatially scalable. In this research, interpolation components are proposed for the V-PCC framework to make it suitable for flexible spatial resolution. As outliers might be brought in by the interpolation, a patch-aware averaging filter is applied to eliminate most outliers. Experimental results show that the interpolation component performs well both on objective evaluation and subjective visual quality.

Point cloud and mesh are the two main representations for 3D content. While compression methods for them are actively studied, there are few studies of their perceptual compression quality and none that consider observation distance. We studied the perceptual quality of compressed 3D sequences, for both a point cloud compression method (V-PCC) and a mesh-based compression method (Triangle FAN (TFAN)). Two main factors that could impact perceptual quality are considered, bit rate and observation distance. Evaluation of perceptual quality is carried out both by collecting viewer opinion scores of the compressed sequences separately, and with a side-by-side comparison. A functional model for mesh and point cloud compression quality is estimated to predict Mean Opinion Score (MOS) which yields high Pearson correlation and rank correlation scores with measured MOS.

Although V-PCC achieves the highest quality among methods proposed to MPEG, outliers and various other artifacts can degrade the V-PCC quality especially when high quantization parameter (QP) is set. After examining the causes and types of V-PCC artifacts that occur, we propose a framework to remove the highly noticeable outlier and crack artifacts caused by V-PCC so as to improve compressed point cloud visual quality. A subjective experiment showed that our approach significantly improves visual quality, and the improvement becomes more obvious with increasing QP values.

Chapter 1

Introduction

Immersive video, as one rapidly growing multimedia form in daily life, is an important future direction of interaction with content and the real world, due to the richer experience it provides compared to traditional 2D media content, including innovative navigation and interactive functionalities [1,2]. In immersive video, viewers can see details without constraint on the viewpoint, since immersion is guaranteed through 6 degrees of freedom. The advantages of immersive video lead to multimedia applications in many areas, such as teleconferencing [3,4], sports [5], and education [6]. As shown in Fig. 1.1, taken from [7] and [8], a person could sit on their couch at home but feel like they are in a real stadium while watching sports games streamed in 3D. Students in medical school could practice surgery with the help of a realistic 3D model.

Many representation formats have been proposed to represent 3D models. Parametric surfaces are one way in which a surface in 3D is defined by a parametric equation with two parameters. Constructive solid geometry divides a 3D model into a few primitive shapes, such as cubes, tubes and cones, and uses a combination of primitive shapes to represent a 3D model. A mesh represents 3D content with faces (represented by edges and vertices) that define the shape, and texture information that defines the color across the surface of each face. Point clouds, as another representation format, have become more and more popular. Point clouds represent 3D

Figure 1.1: Example applications for 3D content: Sports viewing and medical education

content with a collection of points in 3D space; each point is associated with attributes such as color information. Because a point cloud is an unordered set of points, it is easy to concatenate different point clouds together without considering the topology issues that arise with meshes. The voxel representation, as a special case of a point cloud, has a similar concept to pixels in 2D space. In this representation, the whole 3D space is divided into small cubes, called voxels, with integer indexes as their coordinates.

The development of depth sensors, such as Kinect from Microsoft and RealSense from Intel, has contributed to the acquisition of high quality 3D information, which leads to an increase of 3D data processing applications. Also, research into autonomous vehicles has boosted the requirement to process 3D information to understand the surrounding world. However, along with the advantages over traditional 2D video, huge amounts of 3D data lead to storage problems and the need for compression.

1.1 Point Cloud Compression

The compression of point clouds (PCs), one of the main representation formats for 3D data, has been studied for a long time. For PC compression, since the points are not connected as they are in a mesh, the spatial organization of points should be built so as to encode the points

efficiently. Octrees were applied in [9, 10] to progressively compress PCs. Octrees partition three dimensional space by dividing it into octants recursively like a tree structure with eight children nodes for each parent node. In [11], octrees and graph-based transforms are combined to compress geometry information in static PCs. Hierarchical clustering of points can generate the Level of Detail (LoD) in [12], which describes different levels of complexity for a PC, and the LoD is progressively compressed. A hierarchical sub-band transform that resembles an adaptive variation of a Haar wavelet is applied in [13] for PC color compression. A motion-compensated approach to encoding dynamic PC sequences is proposed in [14]. An encoding scheme for building a 3D model is based on a set of low-frequency spherical harmonic basis functions in [15] and PC sequences are encoded with a motion compensation strategy in [16].

MPEG hosted a call for proposals [17] and picked out three methods [18] as winners for three different categories: static models, dynamic sequences and dynamic acquisition. Test Model Category 2 (TMC2), which was later renamed as V-PCC, from Apple Inc. [19] achieves the best subjective and objective quality under given target bit rates for the dynamic sequence category. Its core idea is to project points, both their geometry coordinates and attributes, to 2D and convert the 3D sequence to a 2D sequence. Then any existing video codec, such as FFmpeg or HM (Test Model for HEVC) could be used to compress the 2D sequence.

The encoding process and decoding process of V-PCC are shown in Fig. 1.2 and Fig. 1.4. Here, the color information assigned for each point, called texture information, represents an attribute of the point. Any other kind of attribute could be processed in the same way. The framework of the encoder is divided into three parts: segmentation, packing and compression.

1.1.1 Segmentation

Given an input frame of a point cloud, before segmenting the points, V-PCC generates the normal direction for each point based on the underlying surface. Then, according to normal direction clusters, segmentation is applied to divide the point cloud into patches. The total number

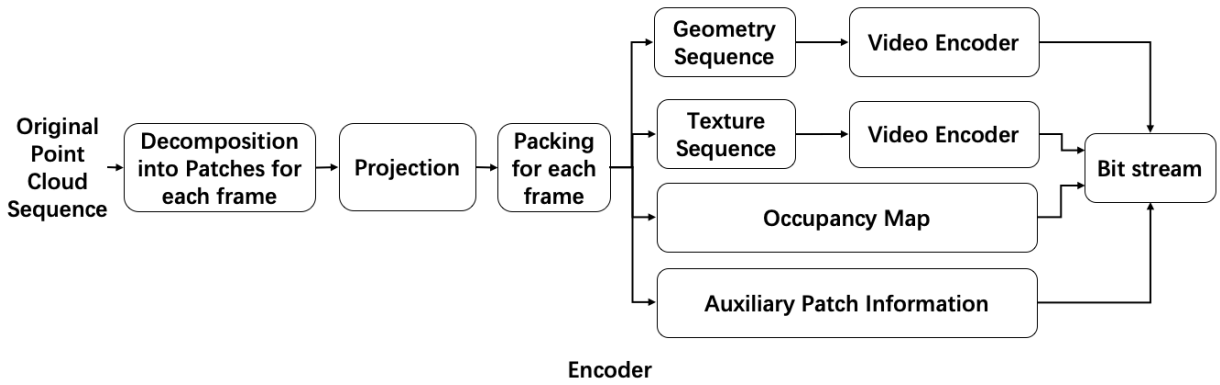

Figure 1.2: Compression framework: Encoder

of patches is constrained within 256. Fig. 1.3 from [19] shows an example.

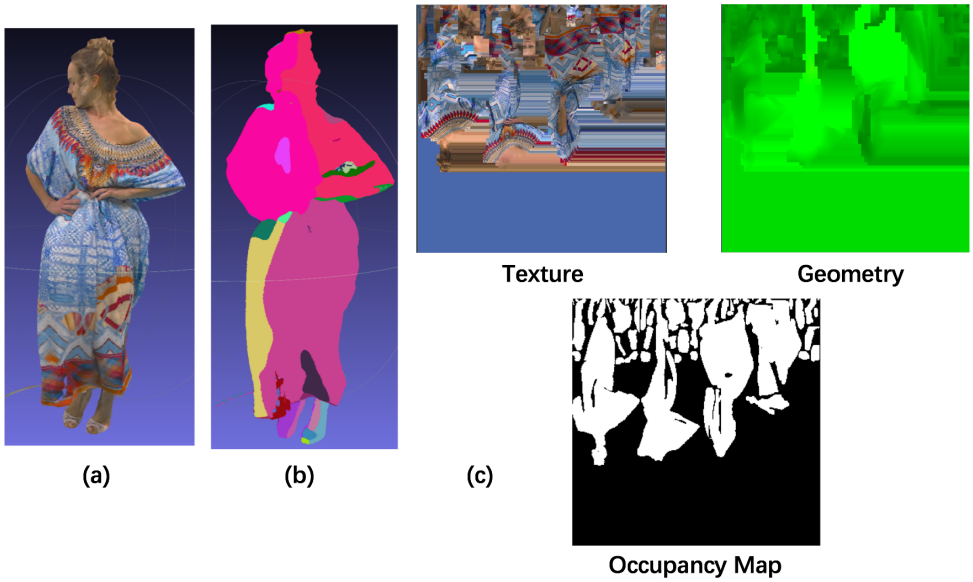

Figure 1.3: (a) Original point cloud. (b) Segmentation result with different colors representing different patches. (c) Examples for texture image, geometry image and occupancy map.

For each patch $\vec{p}_i$, $i \in \{1, 2, ..., N\}$ where N is the number of patches, one of 6 main directions ($\pm\vec{x}$, $\pm\vec{y}$, $\pm\vec{z}$) is assigned as the principal direction based on the main clustered normal direction.

1.1.2 Packing

After segmentation, projection is performed to convert each 3D patch into a 2D patch. For each point $\vec{v}$ in the 3D patch $\vec{p_i}$, the coordinate of $\vec{v}$ which corresponds to the principal direction of this point is projected onto the 2D patch $p_i^{geometry}$, while the color information is projected onto $p_i^{texture}$. The projection direction is determined by the principal direction of this 3D patch. The same location of a pixel in $p_i^{geometry}$ and $p_i^{texture}$ denotes the same 3D point. After projection, packing fits all 2D texture patches into an image $I_{texture}$, and all geometry patches into an image $I_{geometry}$ with the same pre-fixed size l_x by l_y. Packing images from all 3D frames will form two 2D video sequences, one for geometry and the other for texture. Geometry and texture sequences are in YUV format but only the Y channel is occupied for the geometry sequence because for 3D point (x, y, z), z is stored in Y channel at pixel position (x, y).

Apart from geometry and texture, occupancy maps are also generated for each frame where 0 denotes not occupied and 1 denotes occupied. Fig. 1.3 (c) shows image examples for texture, geometry and occupancy map.

1.1.3 Compression and Decompression

Geometry and texture sequences could be encoded with any 2D video codec, such as ffmpeg or HM. Occupancy maps could also be encoded with a 2D video codec or with arithmetic coding. Combining the compressed geometry sequence, texture sequence and occupancy maps, together with patch information which will help to reconstruct the 3D patch at the decoder, we get the final compressed bit stream. The decoder shown in Fig. 1.4 is a reversed encoder. First, the bit stream is separated into the geometry, texture, occupancy map and patch information. Then, we decode the geometry and texture bit streams, and project 2D patches back to 3D according to patch information and occupancy maps to reconstruct the PC frame.

In the V-PCC framework, after the resolution of the 2D sequence is fixed as an initial pa-

rameter, any scale-up or scale-down of the 2D sequence will create outlier points when projected back to 3D space. As will be discussed in Chapter 2, to add spatial scalability to V-PCC, this research proposed to add a patch-aware averaging filter to remove outliers [20].

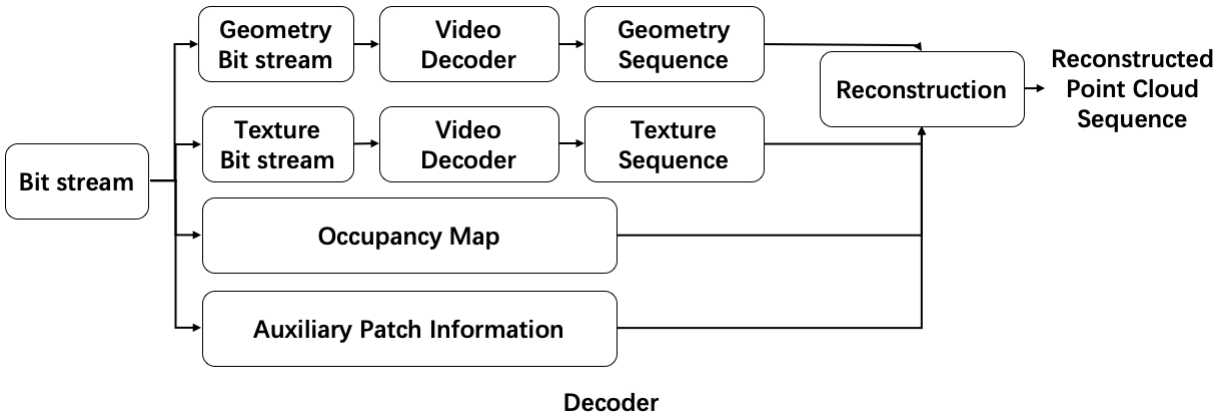

Figure 1.4: Compression framework: Decoder

1.2 Dissertation Overview

The remainder of this dissertation is organized as follows. In Chapter 2, we discuss adding a spatial scaling module to the V-PCC framework to make it spatially scalable and to improve compression flexibility. After adding the scaling module, some outlier points ten to be created in the reconstructed PC. A patch-aware averaging filter is proposed to remove most outlier points caused by scaling. The performance of different kinds of filters is evaluated. The averaging filter generates slightly lower Mean Squared Error (MSE) than the median filter and a weighted filter. The experimental results on 2D PSNR and 3D MSE show that our method performs well both on objective evaluation and on subjective visual quality. Different combinations of interpolation methods for scaling are also evaluated. Among the four different interpolation choices, nearest neighbor interpolation at both the encoder side and decoder side achieves the best performance in terms of MSE.

In Chapter 3, we compare the quality of compressed PCs and compressed meshes under the same bit rate constraints. We apply the most popular compression methods for PCs and meshes on the same 3D content but with different representation formats. A subjective study is carried out where subjects evaluate the quality of 2D videos rendered from compressed 3D content under different bit rates and observation distances. From the experimental result, we conclude that PC compression is better for low bit rates, whereas mesh compression is preferred when the observation distance is close and the bit rate is not low. When the bit rate is high, there is little difference between the two representations. Based on the collected opinion scores, two models to estimate subjective scores for PC compression and mesh compression separately are proposed which fit the experimental data well under cross-validation. The models can be used to choose a better representation for compression based on observation distance and target bit rate.

In Chapter 4, we consider how to eliminate the outliers and cracks that are created by V-PCC especially when a high QP value is applied. The causes for outliers and cracks are examined specifically for V-PCC to locate the regions where outliers and cracks tend to occur. A surface-fitting-based method is proposed to remove outliers and fill cracks. A loop structure can fill large cracks gradually. We conducted a subjective test to evaluate our proposed method; the results show that our artifact removal algorithm provides significant visual quality improvement for PCs reconstructed after V-PCC compression. The artifact removal process tends to achieve more obvious improvement of visual quality with increasing QP. The improvements were more modest but still significant for cases where complex texture or fast object motion could mask the noise, and the improvements were strongest for cases where smooth texture (such as human skin), slow motion, and simple shapes made the artifacts more obvious.

In Chapter 5, conclusions are drawn and directions for future work are discussed.

Chapter 2

Scalability for V-PCC and Patch-aware Averaging Filter

V-PCC brings high efficiency for storage of PC sequences. However, the framework of V-PCC lacks flexibility. After projecting 3D points onto 2D images to form an image sequence, the size of the image is fixed all through the compression. A scaling module which could control the size of the image should provide more flexibility, making V-PCC suitable for scalability. One reason for the fixed size might be that any scale-up or scale-down of the 2D sequence tends to create outlier points when projected back to 3D space.

In this chapter, we add interpolation modules which makes V-PCC suitable for flexible resolution. A patch-aware averaging filter is proposed to eliminate the outlier points which result from scaling-down and scaling-up. For the scaling module, different interpolation methods are discussed and evaluated, and different filters are compared.

This chapter is structured as follows. Sec. 2.1 introduces the details of our proposed method. Sec. 2.2 shows experimental results and conclusions are in Sec. 2.3.

2.1 Proposed scaling method

In the original V-PCC framework, the image size, l_x by l_y, is fixed all through the compression, which leads to less flexibility when compressing the dynamic sequence, especially when the target bit rate is constrained to be quite low. Although the geometry and texture sequences could be scaled down at at encoder side and be scaled up at decoder side, interpolation for scaling tends to bring in noise at the edges of patches, especially for the geometry sequence. An example of noise in the geometry image after scaling is shown in Fig. 2.1. The video content of the geometry sequence is not like normal 2D videos, which have strong spatial correlation, because the geometry images are packed from different patches with different depths. It is also different from the content of the texture sequence because the noise of the color information is not that obvious compared to geometry noise.

Figure 2.1: Scaling noise is the middle strip in the right image

Geometry noise can easily turn into a large outlier when back-projected to 3D space. In Fig. 2.2, we show an example of those outliers. Fig. 2.2(a) shows the original uncompressed PC. With compression and no scaling in V-PCC, the reconstructed PC looks good as shown in (b). However, if a scaling module is added without any other processing, the compressed PC suffers from outliers, shown circled in Fig. 2.2(c). So our proposed method aims to process the geometry image so as to reduce the outliers.

Three kinds of pixels are defined for a 2D patch: edge1 pixels, edge2 pixels and interior pixels. Edge1 pixels have distance 1 to the outside of a patch while edge2 pixels have distance

Figure 2.2: (a) Original uncompressed point cloud. (b) Compressed point cloud with no scaling. (c) Compressed point cloud with scaling. Outliers are circled.

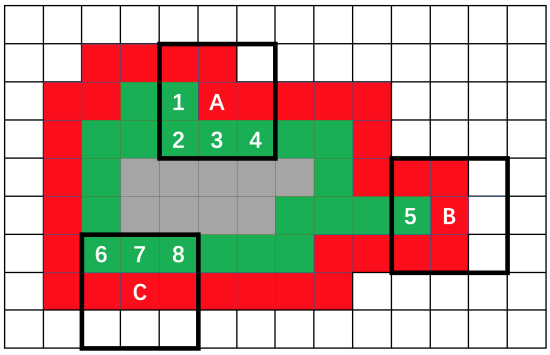

Figure 2.3: Examples for edge1 pixels (red), edge2 pixels (green) and interior pixels (gray) of a 2D patch. For pixels A, B, and C, the patch-aware averaging filter uses their numbered neighbors shown in green.

2 to the outside. Here the distance measure uses chessboard distance, in which a diagonal step counts the same as a vertical or horizontal step. The remaining pixels of a patch are called interior pixels. Examples are shown in Fig. 2.3. Three sets are formed from pixels in the 2D geometry patch $p_i^{geometry}$. Set S_{n1}^i has all edge1 pixels in patch $p_i^{geometry}$, set S_{n2}^i has all edge2 pixels, and all interior pixels go to set S_b^i. Pixel values in set S_{n1}^i are modified by a patch-aware averaging filter:

$$I'_{up}(x,y) = \begin{cases} \dfrac{1}{M}\displaystyle\sum_{m\in Q} I_{up}(x_m, y_m), & Q \neq \emptyset \\ I_{up}(x,y), & Q = \emptyset \end{cases} \tag{2.1}$$

where Q is the set of points that belong to set S_{n2}^i and have chessboard distance 1 to (x, y) and M is the size of Q. As shown in Fig. 2.3, the filtered value for pixel A is the average of pixels 1, 2, 3 and 4, for pixel B it is pixel 5, and it is the average of pixels 6, 7, 8 for pixel C.

Here, an averaging filter is chosen for simplicity. Other simple filters could also be applied, such as a median filter or weighted filter. Different filters are evaluated in Sec. 2.2.

A down-scale module is added to the encoder side of V-PCC, while an up-scale module and patch-aware filter are added to the decoder side. After inserting our proposed modules, the flowcharts of encoder and decoder are shown in Fig. 2.4 and Fig. 2.5 with added modules highlighted with light red. Here, the down-scale module is a normal scaling component for any interpolation $method$, $I_{down} = \mathbf{Scale}(I, 1/r, method)$, where I is the input image, $r \geq 1$ is the scale factor and I_{down} is the down-scaled image I. The up-scale module is a normal scaling component for any interpolation $method$, $I_{up} = \mathbf{Scale}(I_{down}, r, method)$, and the size of I_{up} is equal to that of I. Two interpolation methods are evaluated in Sec. 2.2.

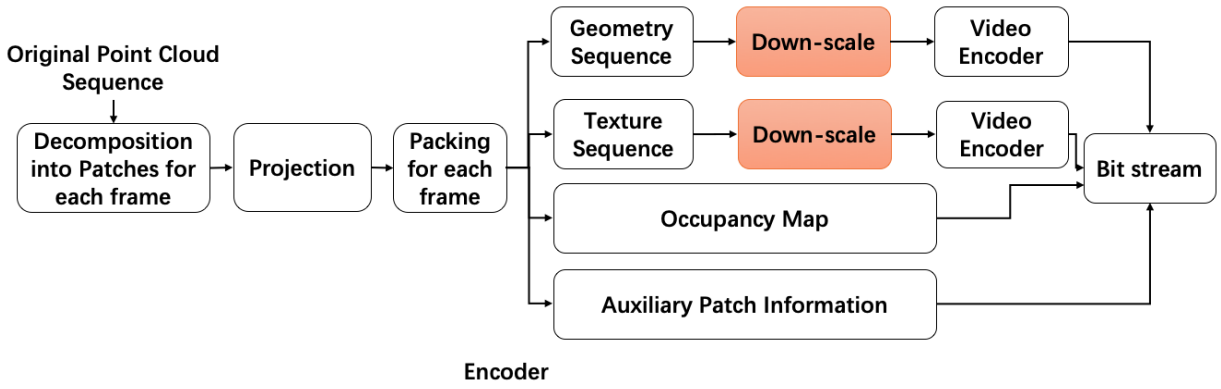

Figure 2.4: Flowchart of encoder after inserting down-scale module

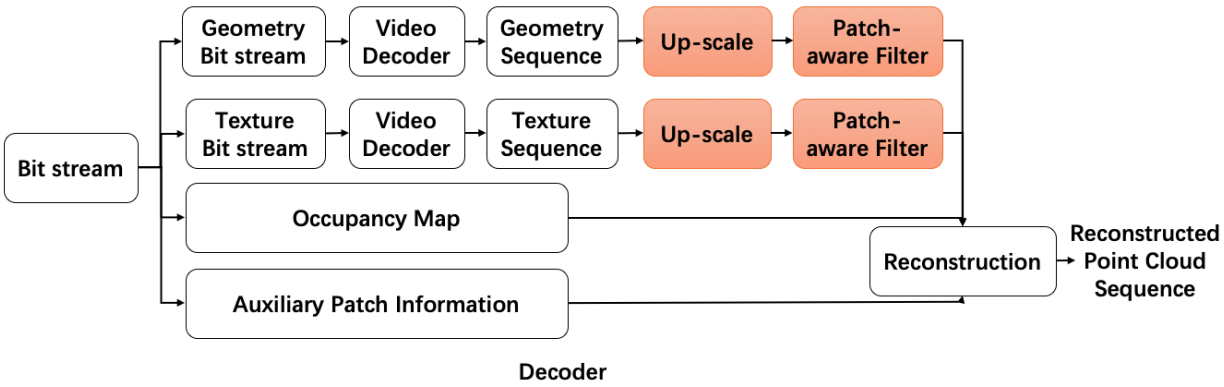

Figure 2.5: Flowchart of decoder after inserting up-scale and filter module

2.2 Experiment

2.2.1 Dataset

We use four 3D dynamic sequences from Owlii Inc. as our dataset: *basketball*, *dancer*, *model* and *exercise*. In Fig. 2.6, we show example frames to illustrate sequence content. *Exercise* contains a man wearing solid-color clothing; he does exercises slowly, with a large movement range. In *Dancer*, a man in solid-color clothing does an urban dance with fast movement and large movement range. The *Model* sequence shows a woman wearing a patterned dress with a swirly skirt; this sequence contains more complex texture than the others. Lastly, *Basketball* involves two objects, a basketball and a man; he plays with the basketball with a moderate motion speed. Both the ball and the man's shirt have some color variation. We use those sequences in PC format and each sequence has 300 frames with around 2.5 million points per frame with range 1024 for x, y and z directions.

2.2.2 Evaluation Metrics

We use both 2D and 3D evaluation for validation. For 2D, we use the PSNR of the occupied pixels denoted in the occupancy map as our evaluation metric. To evaluate the performance of the patch-aware averaging filter, as shown in Fig. 2.7, $\mathrm{PSNR}(A, B)$ does not include the filter and $\mathrm{PSNR}(A, C)$ includes the filter. The comparison between $\mathrm{PSNR}(A, B)$ and $\mathrm{PSNR}(A, C)$ shows the performance of the proposed filter.

MSE rather than PSNR is chosen for 3D because it is hard to define a fixed maximum possible value for different 3D sequences. Given input reference point cloud A and processed point cloud B, first we determine an error vector $\vec{e}_{AB}(i, j)$ from point p_i in A to the closest point p_j in B. Then we compute $\mathrm{MSE}_{AB} = \frac{1}{N_A} \sum_{p_i \in A} |\vec{e}_{AB}(i, j)|^2$ where N_A is the number of points in A. The error vector from B to A, $\vec{e}_{BA}(j, i)$, is generated in a similar way which leads to $\mathrm{MSE}_{BA} = \frac{1}{N_B} \sum_{p_j \in B} |\vec{e}_{BA}(j, i)|^2$ where N_B is the total number of points in B. The final

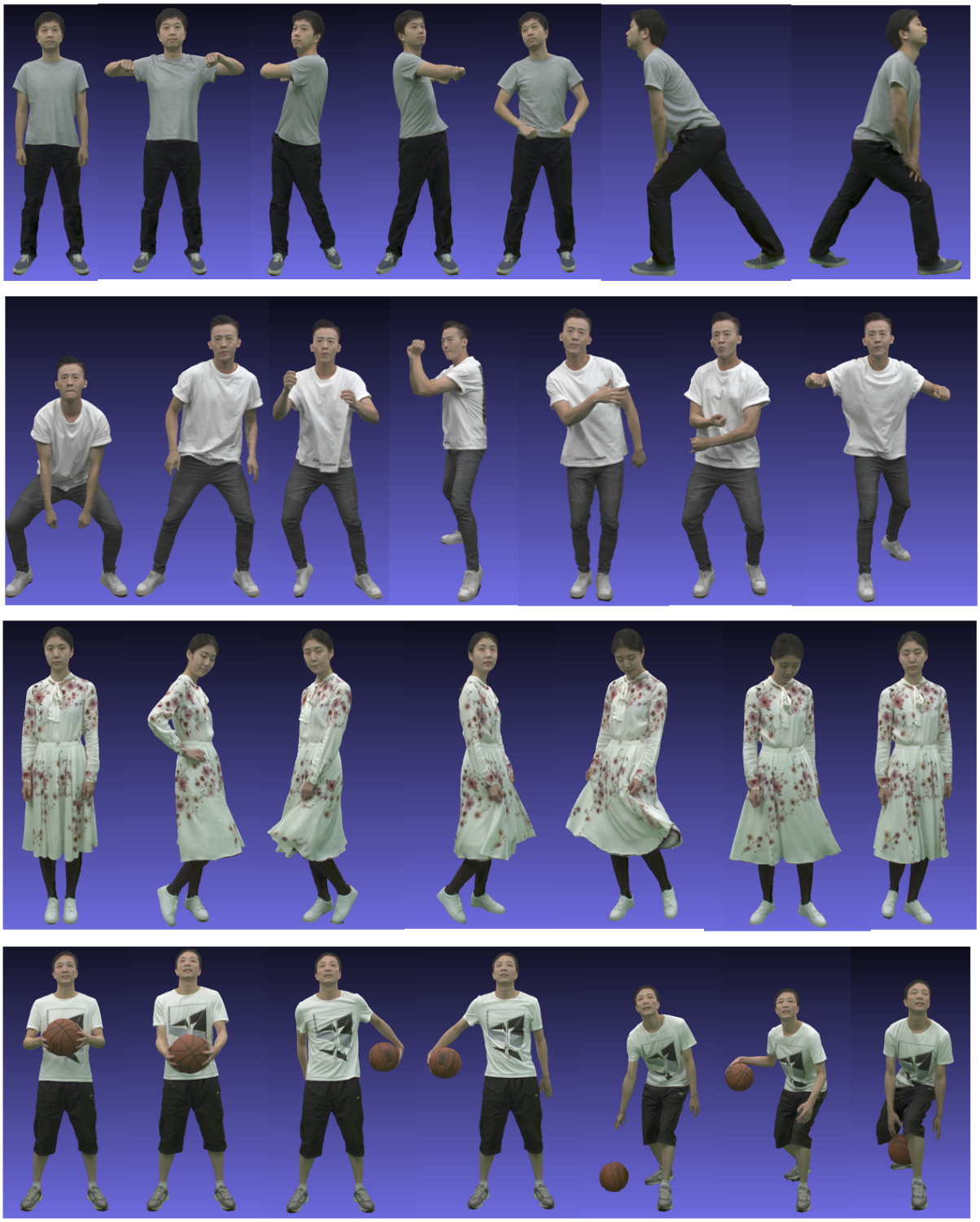

Figure 2.6: Example frames from sequences. From top to bottom, the sequences are *Exercise*, *Dancer*, *Model* and *Basketball*.

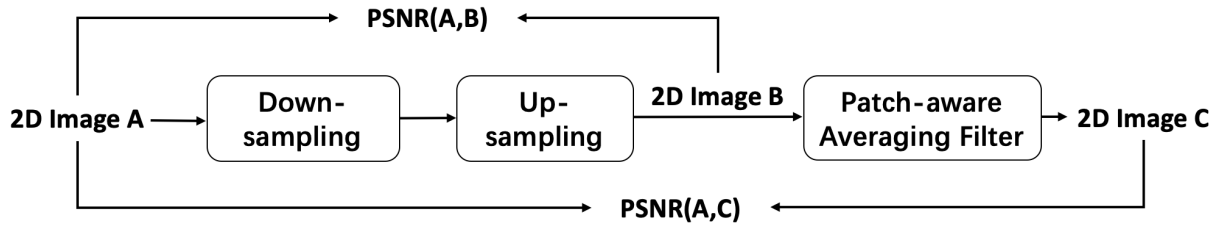

Figure 2.7: PSNR for 2D evaluation

MSE is $\mathrm{MSE}(A, B) = \max(\mathrm{MSE}_{AB}, \mathrm{MSE}_{BA})$. In Fig. 2.8, $\mathrm{MSE}(A, C)$ includes the filter and $\mathrm{MSE}(A, B)$ does not.

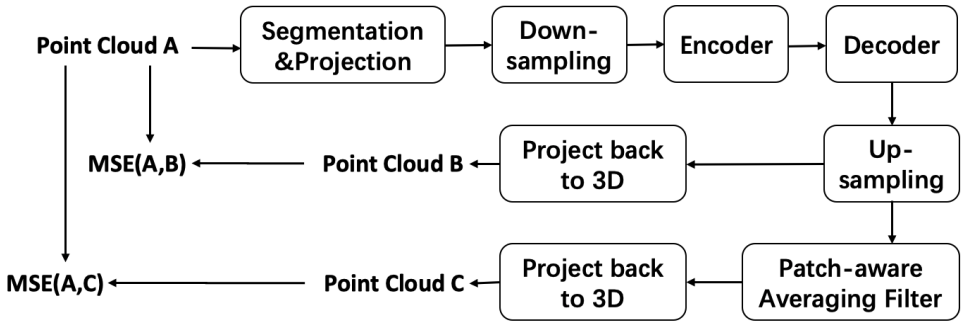

Figure 2.8: MSE for 3D evaluation

2.2.3 Evaluation

For each test sequence, we randomly pick three frames to evaluate with both the 2D metric and 3D metric. Six scale factors, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, are chosen with four different choices for interpolation:

NN2NN: Nearest neighbor scaling at the encoder and decoder.

NN2B: Nearest neighbor scaling at the encoder and bicubic scaling at the decoder.

B2NN: Bicubic scaling at the encoder and nearest neighbor scaling at the decoder.

B2B: Bicubic scaling at the encoder and decoder.

First, we want to show that the scaling module has advantages at low bit rates. Fig. 2.9 plots $20log_{10}(\mathrm{MSE})$ vs. the number of bits after compression using the *Dancer* sequence and

NN2NN. Curves of different colors denote different scale factors. The blue curve corresponds to no scaling. When the bit rate is low, scaling reduces MSE and allows achieving lower bit rates.

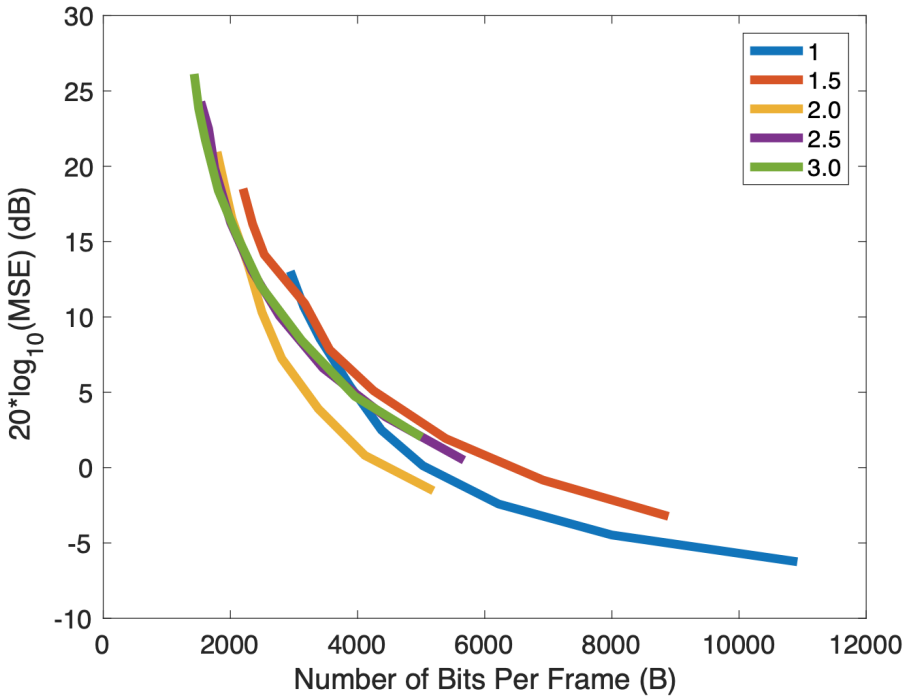

Figure 2.9: MSE vs. Number of bits for different scale factors

Then, the patch-aware averaging filter is evaluated. For 2D PSNR, we have $\mathrm{PSNR}_1 = \mathrm{PSNR}(I, I_{up})$ and $\mathrm{PSNR}_2 = \mathrm{PSNR}(I, I'_{up})$, where I is the original image, I_{up} is the image after both scaling down and scaling up, and I'_{up} is I_{up} processed with the patch-aware averaging filter. Table 2.1 shows $\Delta_{PSNR} = \mathrm{PSNR}_2 - \mathrm{PSNR}_1$ with different interpolation choices. We see that for almost every scale factor and every sequence, the patch-aware averaging filter produces a quality improvement.

For 3D MSE, we evaluate the method in a similar way. MSE_1 is computed between the input PC and output PC from compression and decompression without the patch-aware averaging filter. MSE_2 uses the output PC after the patch-aware averaging filter. Five quantization parameters (QPs) are chosen: 0, 10, 20, 30 and 40. The scale factors are the same as for the PSNR evaluation. Then $\Delta_{MSE} = \mathrm{MSE}_2 - \mathrm{MSE}_1$. In Fig. 2.10, Δ_{MSE} vs. QP curves are plotted for

different interpolation choices and different r. Only plots for *Dancer* are shown here. Almost all curves are below the x axis which means the patch-aware averaging filter produces a lower MSE.

Table 2.1: Δ_{PSNR}

Factors	1.25	1.5	1.75	2.0	2.25	2.5
NN2NN:						
dancer	5.463	5.235	7.230	-0.008	7.334	7.183
basketball	4.659	6.800	7.802	0.012	7.415	8.166
model	5.299	5.600	7.017	-0.131	6.455	7.610
exercise	6.419	6.389	7.838	-0.198	8.119	8.149
NN2B:						
dancer	7.991	8.200	7.587	4.194	5.474	4.576
basketball	8.417	9.006	8.281	4.928	5.819	4.663
model	8.225	8.238	7.293	4.497	5.109	4.663
exercise	9.684	9.256	8.287	5.304	5.721	5.000
B2NN:						
dancer	6.817	6.161	6.690	0.042	5.253	4.693
basketball	7.482	7.534	7.625	0.070	5.137	4.732
model	6.485	6.403	6.893	-0.094	4.388	4.932
exercise	7.343	7.191	8.021	-0.065	5.600	5.306
B2B:						
dancer	6.924	7.647	7.506	7.366	5.720	4.704
basketball	7.572	8.403	8.080	8.363	6.135	4.912
model	6.687	7.449	7.161	7.301	5.402	4.755
exercise	7.581	8.455	8.194	8.485	5.801	5.245

Three filter types, patch-aware averaging filter (A), patch-aware median filter (M) and patch-aware weighted filter (W) of 3×3 Gaussian kernel, with two interpolation choices *NN2NN* and *NN2B* are compared. $A - M$ is the difference of MSE between A and M and other notations use similar definitions. The curves for different QPs and r are shown in Fig. 2.11 for *Dancer*. In the plots, all three filter types have similar performance because the Δ_{MSE} are all within 0.01 and the averaging filter is slightly better than the other two especially for QP ≥ 20. When QP $\leq$ 20 and $r \leq 2.0$ for *NN2B*, the yellow curves are above x axis which means median filtering is a little better than averaging filtering.

To choose which interpolation can generate the best quality, the following evaluation method is applied. For each sequence and interpolation method, $mse_{QP,1}$ is first subtracted from the MSE value $mse_{QP,r}$ corresponding to different scale factors for the same QP. The resulting

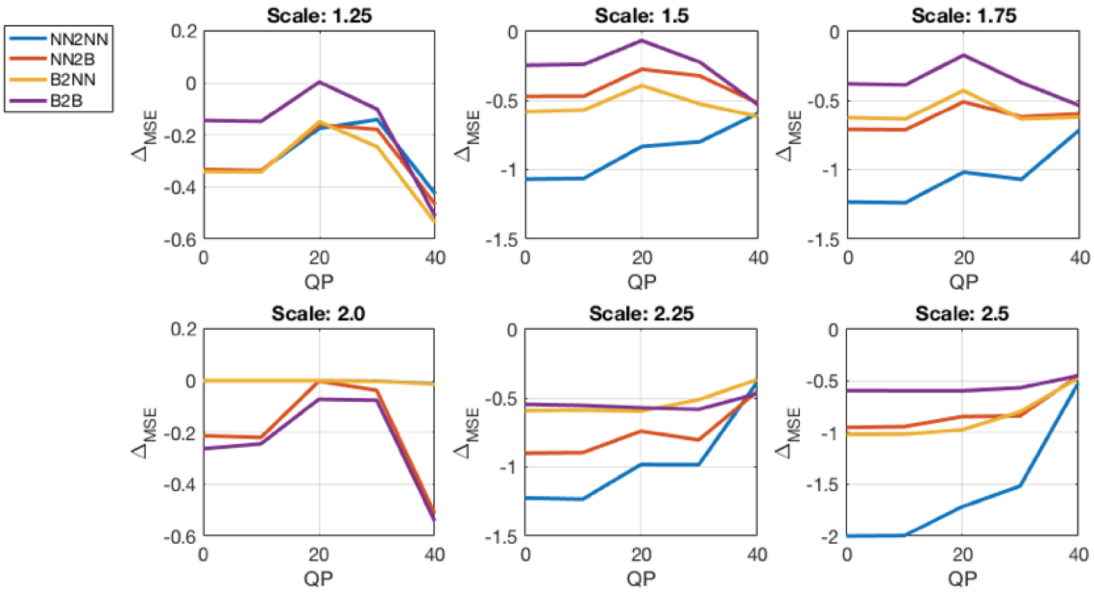

Figure 2.10: Δ_{MSE} vs. QP for different scale factors of sequence *Dancer*

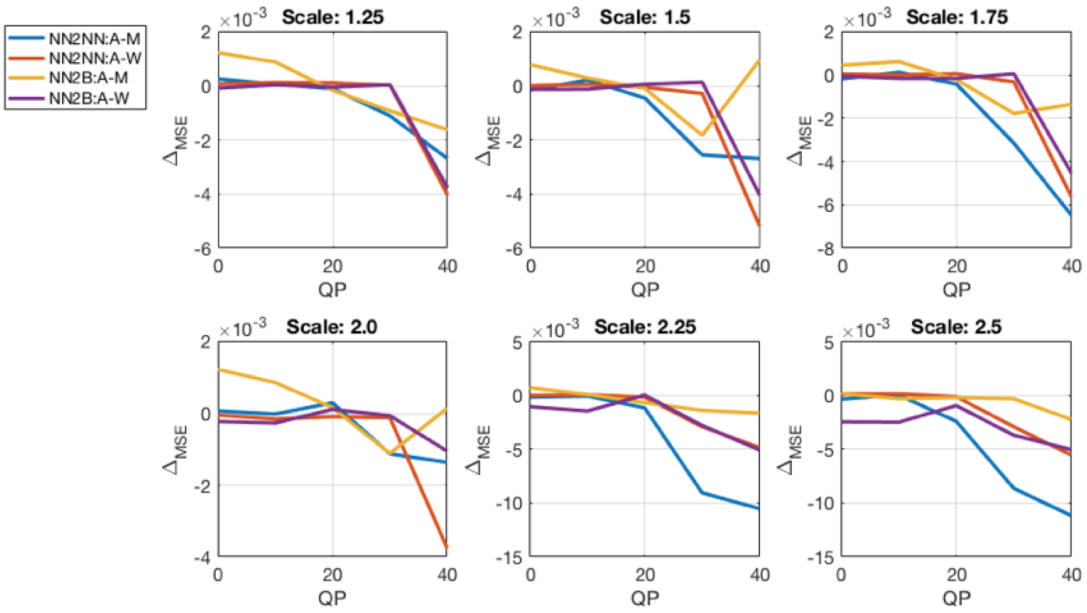

Figure 2.11: Δ_{MSE} vs. QP for different scale factors of sequence *Dancer* for filter comparisons

values are normalized to $[0, 1]$ across all interpolation methods. Lining up normalized values for all QPs and all factors, we calculate the mean value for four interpolation choices shown in Table 2.2. A lower mean value denotes better MSE. We notice that the mean value for *NN2NN* is the lowest among the four choices for every sequence, so *NN2NN* is the best combination among the four interpolation choices.

Table 2.2: MEAN_{MSE}

Interpolation	NN2NN	NN2B	B2NN	B2B
dancer	**0.170**	0.343	0.736	0.593
basketball	**0.260**	0.390	0.747	0.516
model	**0.266**	0.321	0.751	0.484
exercise	**0.227**	0.474	0.742	0.564

To explore why NN2NN has the best performance among the four interpolation configurations, we create a synthesized PC which contains a cube rotating along one axis as shown in Fig. 2.12. The PSNR after patch-aware averaging filtering vs. rotation angle for scale factor 1.25 is plotted in Fig. 2.13. With the increase of angle, the PSNR decreases for all interpolation methods and then increases again. Then we analyze the error pixel distribution. An edge1 error pixel is defined as an edge1 pixel which has nonzero error in $I_{err} = abs(I_g - I_{out})$. Definitions of edge2 error pixels and face error pixels are similar. For edge2 pixels, NN interpolation only considers the nearest pixel, so NN2NN has the smallest number of edge2 error pixels. For edge1 pixels, although NN interpolation may generate large edge1 errors, those errors could be fixed by filtering. For face errors, when the rotation angle is 40° and 50°, NN interpolation produces more face error pixels compared with Bicubic (B), because those rotation angles have large intersection angles with the main projected direction and B will generate a more smooth surface under those cases. However, there are few patches with those intersection angles. Hence, NN performs better in general.

To review our proposed filter from another aspect, for each category, the number of error pixels, their total and average absolute error are shown in the first row of Fig. 2.14. Although edge1 total error is lower than face total error, for average error per pixel, edge1 error pixels rank

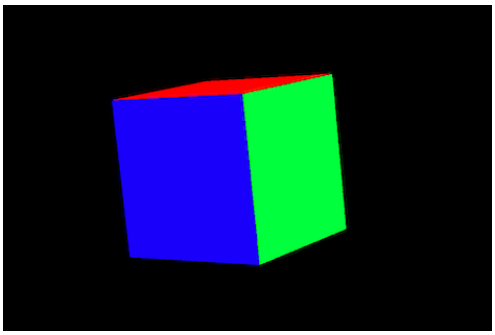

Figure 2.12: Example from cube sequence

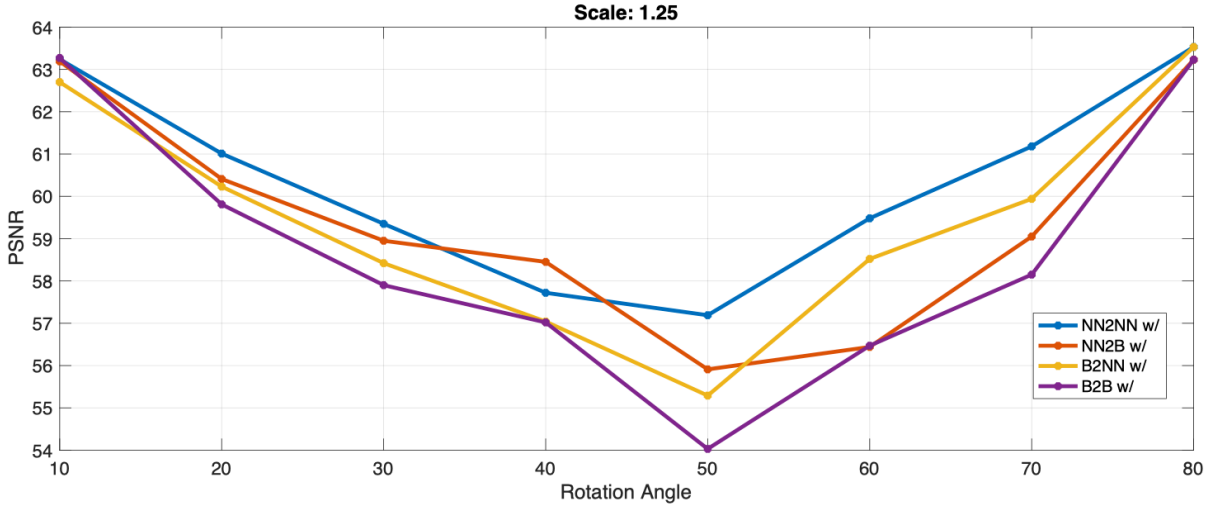

Figure 2.13: PSNR for cube rotation sequence

the highest among the three categories. After filtering, the corresponding plots are shown in the second row of Fig. 2.14. The average error for edge1 error pixels is greatly reduced.

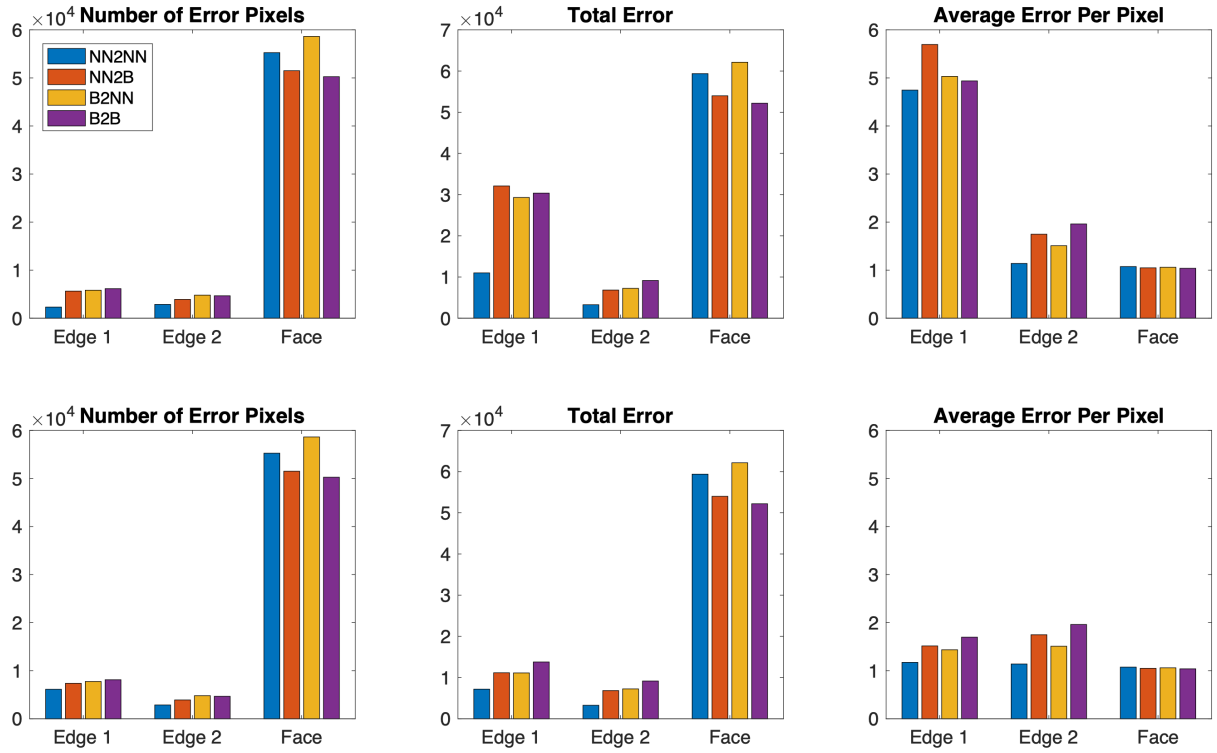

Figure 2.14: Statistics for different categories for sequence *Dancer*. The first row does not use the proposed filter and the second row uses it. The scale factor is 1.25.

For visual quality, Fig. 2.15 shows examples without and with the patch-aware averaging filter. Our filter eliminates the outliers and greatly improves visual quality.

2.3 Conclusion

In this chapter, we add scaling modules to the original V-PCC framework to improve compression flexibility. A patch-aware averaging filter is proposed to remove most outlier points caused by scaling. The averaging filter generates slightly lower MSE than the median filter and a weighted filter. The experimental results on 2D PSNR and 3D MSE show that our method performs well both on objective evaluation and on subjective visual quality. Among the four

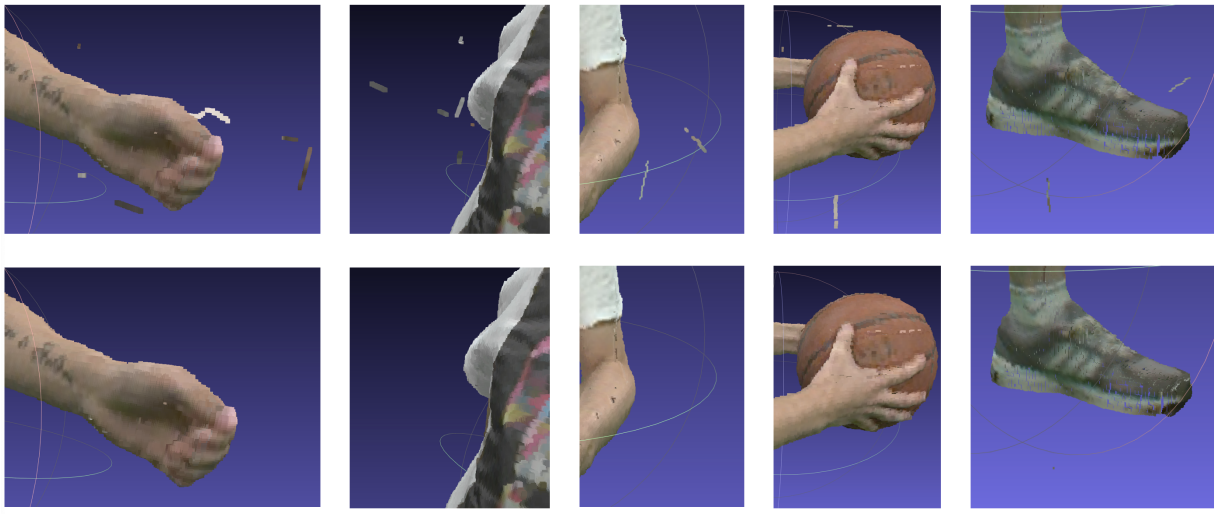

Figure 2.15: The first row shows images without filtering and the second row is with filtering.

different interpolation choices, *NN2NN* achieves the best performance in terms of MSE.

Chapter 2, in part, is a reprint of the material as it appears in the paper: K. Cao, Y. Xu and P. Cosman, "Patch-aware Averaging Filter For Scaling in Point Cloud Compression," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Visual Quality of Compressed Mesh and Point Cloud Sequences

Subjective and objective measures of quality around the two main representation formats for 3D content, point cloud and mesh, has not been researched deeply, especially for quality after compression. Difficulties arise from the lack of promising objective metrics. There are few objective metrics that work on both mesh and PC representations, and most current objective metrics are poorly correlated with human perception [21–26]. Most current work on quality evaluation focuses on geometry distortion and ignores texture distortion, however, the overall visual quality is affected by both. Subjective experiments under varying conditions, as well as objective metrics correlated with perception, are needed.

In this chapter, the contributions are:

- We compare PC sequence compression and mesh sequence compression which allows us to determine which representation is preferred depending on factors of sequence content, bit rate and observation distance. We use the evaluation method in [17] from MPEG, in which a virtual viewing trajectory is chosen and the 3D sequence is rendered into 2D for subjective viewing.

- A model of subjective rating estimation is proposed which consists of two parts, bit rate quality factor (BQF) assessing the quality of compression based on bit rate, and observation distance correction factor (ODCF) making a correction over BQF with observation distance. Our model fits well with subjective ratings.

This chapter is structured as follows. Sec. 3.1 introduces prior work on compression and quality evaluation of meshes and point clouds. Sec. 3.2 describes the subjective experiment. Experimental results and the estimation model are in Sec. 3.3. Sec. 3.4 provides compression suggestions based on the subjective test results, and conclusions are in Sec. 3.5.

3.1 Related Work

3.1.1 3D Mesh Compression

PC compression has been described in detail in Chapter 1. For mesh compression, triangle fan-based compression (TFAN) [27] enumerates triangle connection cases to encode the connection information of a mesh so as to improve compression efficiency. We choose TFAN as our mesh compression method because MPEG adopted TFAN in their mesh coder. Google's open-source compression tool, Draco, is based on the corner-table method [28] and achieves real-time compression and decompression. Similar connectivity pattern information is used to improve encoding efficiency for mesh compression in [29]. The whole mesh is divided into a few sub-meshes and compressed with a graph Fourier transform in [30] which approximates the connectivity of a sub-mesh with a sparse matrix.

3.1.2 3D Content Quality Assessment

Subjective quality evaluation and computable quality evaluation of 3D representations are both active research topics. Subjective quality for 3D content is much less well studied than

for 2D video. Prior work on subjective evaluation and objective metrics for PCs are summarized in [31]. Most recent work focused on evaluating subjective quality or Quality of Experience (QoE) for static 3D objects, including comparison with computable metrics. In [17], MPEG adopted a subjective experiment approach to evaluate the quality of compressed PCs, in addition to the conventional point-to-point and point-to-plane metrics. The impact of different noise levels on QoE of PCs was studied in [32]. Augmented reality head-mounted displays were used in subjective evaluation of PCs in [33]. In [34], different subjective methodologies were studied, such as Absolute Category Rating (ACR) and Double Stimulus Impairment Scale (DSIS). The authors concluded that they performed similarly for PC subjective experiments involving the quality of compression-like distortions. In [23], 20 subjects subjectively scored PC compressed videos from 1 (Bad) to 5 (Excellent), however, this research only considered different encoding configurations without comparing compression bit rates. Also, subjects were allowed to observe the sequence with free viewpoint which might lead to a variety of results since people may focus on different local details. The impact of different reduction methods for PCs on the QoE of rendered images was investigated in [35].

There is less research on subjective quality of 3D sequences. A subjective experiment on V-PCC compression for two PC sequences was carried out in [36]; the authors found that perceptual quality was more affected by texture distortion than geometry distortion. The impact of different rendering configurations on QoE in VR-based training was studied in [37]. Better immersion and faster interaction with 3D content was found to affect subjective quality in [38]. QoE of adaptive PC streaming was investigated in [39] with different network configurations.

One prior work compared the visual quality of mesh and PC representations. In [40], a comparison between colored mesh and PC compression concluded that mesh compression generates better visual quality at high bit rates while PC compression is better at low bit rates. Our work differs from [40] in that we consider multiple observation distances, and our compression pipelines allow PC scaling and mesh simplification, which can change the visual tradeoffs. In

addition, we include a model for estimating quality ratings based on bit rate and observation distance.

Computable quality metrics for 2D and 3D sequences can be categorized as Full-Reference (FR) metrics, which have access to the original sequence as well as the distorted one, Reduced-Reference (RR) metrics, which have access to the distorted sequence and to some key parameters extracted from the original sequence, and No-Reference (NR) metrics [41], which do not have access to the original reference sequence. NR metrics have been further subdivided into pixel-based (NR-P) metrics which use the distorted sequence to evaluate quality, and bit-stream-based (NR-B) quality predictors which do not make use of the distorted sequence directly, but rather use basic bit stream parameters such as the bit rate and packet loss rate to predict quality. Because of different properties of mesh and PC representations, different metrics are applied. A survey of perceptually-based computable metrics for visual impairment of 3D objects appears in [42].

Among FR metrics of mesh quality, root mean square error (RMS) and Hausdorff distance (HD) were adopted as straightforward metrics, but they were found to be poorly correlated with human perception [21, 22]. Curvature was computed on different scales of a distorted mesh and its reference in [21], based on which a correspondence was found to generate a mapping between meshes. Then, inspired by the work of [43], a structural similarity index on 3D, named Mesh Structural Distortion Measure (MSDM) was implemented on all scales. A final score was computed as a weighted combination considering all scales. Similarly, based on curvature, [44] took visual masking and saturation effects into consideration to correct scores directly from curvature. Quality for watermarked meshes was explored in [45] by measuring the difference of surface roughness between watermarked and original meshes. Measuring distance between curvature tensors of two triangle meshes under comparison was proposed in [46]. A RR method was developed in [47] that considered distributions of extracted parameters from dihedral angles of distorted and reference meshes. The Kullback-Leibler divergence between the distributions was calculated as a perceptual distance. In [48], a local roughness measure was derived from

Gaussian curvature. A NR method proposed in [49] fed distributions of dihedral angles into a trained support vector regression to predict quality scores. With the development of neural networks, [50] took mean curvature as input to a regression neural network to predict a quality score without a reference mesh. All of this prior work focused on meshes without color.

For computable metrics of compressed PCs, similar to [21, 22], simple point-to-point and point-to-plane FR objective metrics, such as RMS and HD, showed no correlation with perceptual quality [23–26]. The conclusion that point-to-point and point-to-plane metrics are limited in predicting subjective quality ratings especially for V-PCC was also verified in [51, 52]. In [26], point-to-point and point-to-plane metrics were studied for a PC de-noising algorithm; they concluded that the point-to-plane metric is more correlated with perceptual quality than is a point-to-point metric for a PC with no noise. In [53], the normal vectors used for the point-to-plane metric were averaged within a small local region to avoid the error of estimated normal from geometric distortions. In [54], a plane-to-plane FR metric measured angular similarity through the intersection angle of normal vectors between two corresponding points. While the aforementioned work also focused only on geometric distortion, two studies considered color attributes as well [55, 56]. In [55], the authors rendered a 3D point cloud onto a 2D plane, then applied traditional FR image quality metrics to measure the 2D image quality as input to their prediction model for perceptual quality of the 3D PC. Different objective metrics were proposed in [56] based on geometry, normal vectors, curvature and color separately, and the color-based metric was found to best match perceptual quality.

The current work differs from this prior work in several ways. We provide a comparison of PC and mesh compression across many different bit rates and across three different observation distances, in order to explore the effects that both rate and observation distance have on perceived quality. Unlike most prior work, we consider 3D content with color. We include the possibility of reducing the number of triangles or number of points. Lastly, we develop a simple NR quality predictor which can predict subjective quality scores for both mesh and PC compression as

functions of bit rate and observation distance. A list of terms and symbols used in this chapter is provided in Table 3.1.

Table 3.1: Definitions of Terms and Symbols

Terms & Symbols	Definitions
Mesh Compression:	
N_t	Number of triangles per frame
L	Atlas image size after scaling
q	Quantization parameters for atlas image sequences
β	Quantization step size for vertex coordinates
γ	Quantization step size for vertex-atlas mapping
PC Compression:	
s	Down-scaling factor for PC sequence
l_x, l_y	Size of projected image
$QP_{geometry}$	Quantization parameter for geometry
$QP_{texture}$	Quantization parameter for texture
MOS Estimation Model:	
r	Bit rate
r_{max}	Maximum bit rate (25Mbps in our case)
d	Observation distance
BQF	Bitrate Quality Factor, defines general trend of quality when bitrate changes
ODCF	Observation Distance Correction Factor, equals ratio of quality score at distance d to quality score at maximum observation distance, for a given bitrate
w	Parameter that controls BQF
$slope()$	Slope function of ODCF
$b()$	Intercept function of ODCF
m, p, t	Parameters of slope function
ρ	Pearson Correlation
MSE	Mean Squared Error for fitting evaluation
Choice of Number of Triangles and Scale Factor:	
k, a, c	Parameters involved in number of triangles and scale factor recommendation

3.2 Subjective Experiment

Here we design a subjective test to compare the quality under compression of PC and mesh representations of 3D content.

3.2.1 Test Sequences

We use the same four 3D dynamic sequences as in Chapter 2: *Basketball*, *Dancer*, *Model* and *Exercise*; their content and properties are described in Chapter 2.

Fig. 3.1(a) shows the camera setup for data acquisition, consisting of 75 stationary cameras, of which 25 are color and 50 infrared. Fig. 3.1(b) shows the data acquisition pipeline. The whole pipeline is similar to [57]. The 50 infrared cameras lead to 25 depth images which are aligned with the 25 color images for each frame in the 'Generating Correspondence' module. Those 25 image pairs for each frame constitute the raw data of each sequence.

For the mesh version of each sequence, starting from the raw data, all image pairs are fed into a foreground segmentation algorithm. The foreground segmentation algorithm is adopted from [57]; it considers a confidence map for an RGB image, IR image and Shape from Silhouette jointly in order to generate a good segmentation. After foreground segmentation, a triangle mesh is reconstructed with a given rough number of triangles. Because the captured content is dynamic, a mesh tracking method, non-rigid Iterative Closest Point [58], is applied to ensure topology consistency in each group of frames. The reconstructed mesh has vertices with coordinates ranging from 0 to 2048. At the last step, a texture atlas image, containing the color information of size 2048×2048, is generated.

For the PC version of each sequence, we first generate a mesh with 100k triangles per frame as the source mesh. The number of triangles is high enough to ensure good quality. Then, we sample across the mesh surface with an interval of 0.5 which leads to a source PC with approximately 2.5 million points per frame.

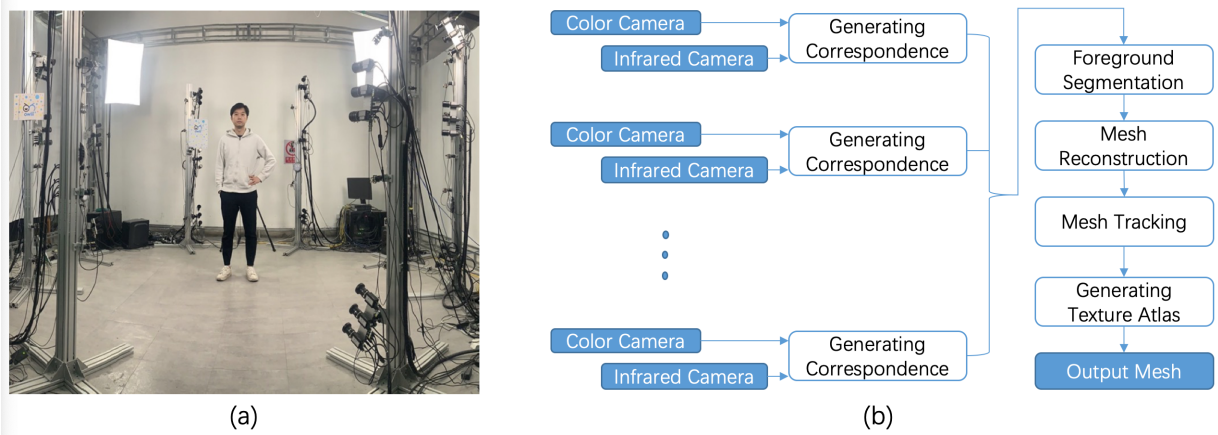

Figure 3.1: (a) Camera setup to capture 3D dynamic sequences. (b) Pipeline of data acquisition process

3.2.2 Sequence Preparation

Each test sequence is encoded at five target bit rates: 3Mbps, 6Mbps, 9Mbps, 15Mbps, and 25Mbps. Those target bit rates are chosen to cover a large range which will satisfy many potential applications. We render the 3D content as depicted in Fig. 3.2. Using OpenGL in a Linux OS, we obtain rendered 2D images of the 3D sequences out of which we generate video sequences that are shown to subjects. The observation distance, which is the distance between the sequence centroid and the virtual rendering camera, takes the values 1.5m (close), 3.0m (middle) and 5.0m (far) where m represents meters. The observation distances chosen represent a significant range of possible observation distances. Observing at 1.5m or closer generally does not allow the whole body to be seen, while observing at 5m or farther means the body appears rather small. By projecting each point or face to the image plane, OpenGL simulates what the 3D content will look like from a camera positioned at the specified observation distance from the center of the object. The virtual viewing trajectory makes a full 360 degree turn around the main figure, while making a small variation in height (somewhat higher and lower than the midpoint). The average distance between the virtual rendering camera and the figure is roughly the observation distance (actual distance could be a little closer or further, less than 5% difference). These virtual viewing

trajectories are chosen to present a large set of angles and details to be examined by the subjects. During the rendering process, the size of rendered videos is fixed to 2048×2048.

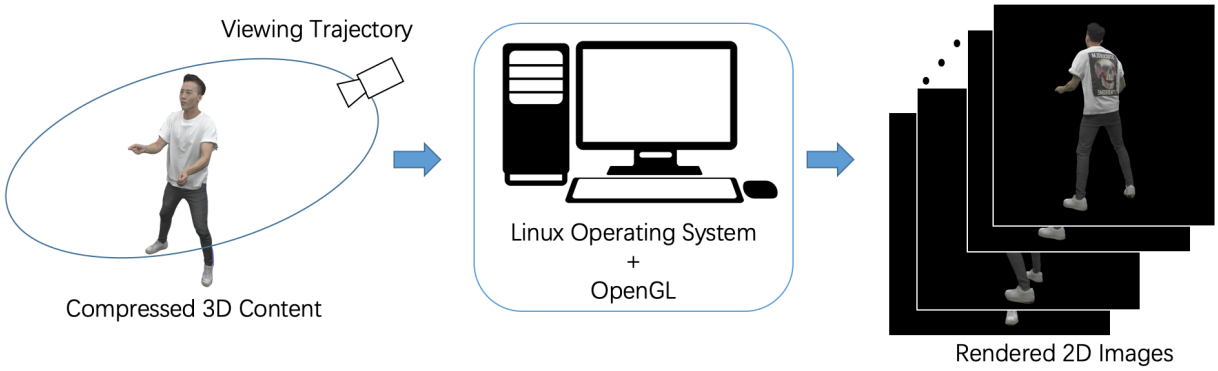

Figure 3.2: Illustration of virtual viewing trajectory and rendering process

The observation distance is defined as the distance between the sequence centroid and the virtual camera. In a real application, observation distance would typically be an input from the user, who chooses the distance from which to view the content. Different observation distances could correspond to different applications. For example, teleconferencing might involve a close distance, whereas a user watching a performance might choose a far distance. Once an observation distance (and angle) are chosen by the user, that determines the rendering parameters which are needed to achieve that view, but it does not determine the encoding parameters.

Then, for each target bitrate and observation distance, we aim to pick the compression parameters with the best visual quality for evaluation. Compression parameter sets are different for mesh and point cloud representations, and they are chosen as follows:

For mesh representation: This consists of two parts, the atlas image and vertex information. Atlas images are formed into a sequence and compressed with any video codec. We used FFmpeg (version 3.4.1) to compress it, involving two compression factors: atlas image size L and compression quantization parameter (QP) q. The vertex information contains vertex coordinates, vertex connection information, and vertex-atlas mapping information. Vertex coordinates are quantized with step size β. Vertex connection information is compressed losslessly

with TFAN [27]. The vertex-atlas mapping determines the correspondence between a vertex coordinate and a pixel position in the atlas image which is a two dimensional vector for each vertex. Vertex-atlas mapping information is quantized with step size γ. All the vertex information after quantization and compression is further compressed as a subtitle bitstream in the FFmpeg framework. The number of triangles per frame N_t can also be controlled as a pre-processing step. The overall parameter set for mesh compression is formulated as the vector $(N_t, L, q, \beta, \gamma)$.

Starting with the raw data for each sequence, 6 versions of the mesh sequence are generated with different numbers of triangles per frame: 3k, 5k, 8k, 10k, 12k and 15k. For each version and each target bit rate, the compression parameter set that generates the best visual quality is manually chosen. That leads to 6 different compression parameter sets in total for each target bit rate. Then, 2D video is rendered according to a certain observation distance. We manually pick one parameter set from those 6 parameter sets as the one with the best 2D visual quality under this bit rate and this distance. For each observation distance and target bit rate, we repeat this procedure. In the end, given 5 bit rates and 3 distances for each sequence, we have 15 different compression parameter sets for each of the four test sequences.

For PC representation: We use V-PCC for PC compression. An important V-PCC parameter is the projected image size, l_x and l_y; the core idea of V-PCC is to project a PC onto a 2D image for both geometry and texture. Then, a conventional video codec is applied with QPs for geometry and texture ($QP_{geometry}$ and $QP_{texture}$). We used FFmpeg (version 3.4.1) as the video codec. Note that the size parameters l_x and l_y for PC compression, and L for mesh compression, are used only in the compression step, where they may differ for different bit rates or observation distances. However, following decompression, for purposes of rendering and display, all cases use the same interface and display size of 2048 $\times$ 2048.

Another important parameter, similar to the number of triangles per frame in mesh compression, is the down-scaling factor s which controls the number of cloud points. Down-scaling is performed over the source PC. Given a point (x, y, z) in the original PC, the scaled point

is $(x/s, y/s, z/s)$. Since the PC compression algorithm only takes integer input, the coordinates $(x/s, y/s, z/s)$ are rounded to integer. After shrinking all point coordinates and rounding to integers, some points that were distinct in the original PC will map to the same position in the down-scaled PC. These duplicate points are removed. In this way, a down-scaling of coordinates directly leads to a down-sampling of points. So the down-scaling reduces the number of points. Scaling-up is implemented when we render the scaled PC.

We have 6 scale factors, 1.0, 1.25, 1.5, 1.75, 2.0 and 2.5. The compression parameter set for PC compression is $(s, l_x, l_y, QP_{geometry}, QP_{texture})$. For each target bit rate and each observation distance, we manually pick the parameter set that provides the best visual quality. In the end we have 15 parameter sets for 5 bit rates and 3 distances for each sequence.

In Table 3.2, we show compression parameter sets that are picked for our experiment for *Dancer*. Those parameter sets achieve the best visual quality for the given target bit rate and observation distance. To generate the candidates, we traversed all possible combinations of parameters that fit the bit rate constraint. After rendering all candidates to 2D videos, the investigators manually picked the parameter set with the best visual quality among the candidates for each observation distance. It is not always the case that the lowest QP was selected as the best candidate, because, for example, to meet the bit rate constraint, the lowest QP might occur with heavy down-scaling of the point cloud, a combination that might lead to poor quality. When we manually pick the parameters, the parameters are hidden from us to avoid bias, so we choose the parameters based on visual quality of the rendered videos. Table 3.2 shows a clear trend in different parameter sets for different bit rates, for example, for lower bit rates, a smaller number of triangles is a good choice for meshes, and heavier down-scaling is a good choice for point clouds. But Table 3.2 does not show a clear trend for observation distances, suggesting that some different combinations of parameters which achieve the same bit rate look visually similar.

There are 120 rendered videos in total, corresponding to 2 representations (mesh and point cloud), 4 sequences, and 15 sets of compression parameters (3 distances $\times$ 5 bit rates).

Each video lasts 10 seconds. Fig. 3.3 and Fig. 3.4 present rendered images of the compressed PC and mesh from the 250th frame of *Model*. From the example frames, we notice that observation distance plays an important role in visual quality. At low bit rate, PC compression causes some cracks and outliers. While mesh compression appears to better preserve the geometry compared to PC compression, the texture information appears to suffer more distortion.

Table 3.2: Parameter sets for different target bitrates and observation distances for sequence *Dancer*

Distance	Close (1.5m)	Middle (3.0m)	Far (5.0m)
Mesh			
3 Mbps	(3k, 512, 31, 0.5, 1/512)	(3k, 512, 31, 0.5, 1/512)	(3k, 512, 31, 0.5, 1/512)
6 Mbps	(3k, 1024, 27, 0.3, 1/1024)	(3k, 1024, 27, 0.3, 1/1024)	(3k, 1024, 27, 0.3, 1/1024)
9 Mbps	(5k, 1024, 23, 0.1, 1/1024)	(8k, 1024, 26, 0.2, 1/1024)	(5k, 1024, 23, 0.1, 1/1024)
15 Mbps	(10k, 2048, 28, 0.2, 1/2048)	(12k, 2048, 31, 0.1, 1/2048)	(10k, 2048, 28, 0.2, 1/2048)
25 Mbps	(15k, 2048, 23, 0.1, 1/2048)	(15k, 2048, 23, 0.1, 1/2048)	(15k, 2048, 23, 0.1, 1/2048)
Point cloud			
3 Mbps	(2, 1280, 1408, 32, 37)	(2, 1280, 1408, 32, 37)	(2, 1280, 1408, 32, 37)
6 Mbps	(1.75, 1472,1616, 29, 31)	(1.75, 1472,1616, 29, 31)	(1.5, 1712, 1872, 28, 35)
9 Mbps	(1, 2560, 2656, 28, 39)	(1.25, 2048, 2240, 27, 33)	(1.25, 2048, 2240, 27, 33)
15 Mbps	(1, 2560, 2656, 21, 35)	(1.25, 2048, 2240, 22, 28)	(1.25, 2048, 2240, 22, 28)
25 Mbps	(1, 2560, 2656, 20, 26)	(1.25, 2048, 2240, 18, 24)	(1, 2560, 2656, 20, 26)

3.2.3 Experiment Design

Thirty subjects (22 males, 8 females, age range 19-37) participated in our two-part experiment. For display, we used an Acer 24-Inch LED backlight monitor. In the first part, the subject scores each video from 0 (worst quality) to 100 (best quality). Before starting, subjects are shown a few example videos to calibrate their scoring standard. The 120 videos are shown in a random order. The user interface for the first part of the experiment is shown in Fig. 3.5. The size of the user interface (and of its displayed video) remains the same for all cases. Subjects input their score in the box according to the scale. After clicking 'Submit Score', the score is recorded and the next random video appears. Including training, this part takes approximately 30 minutes and includes 120 videos.

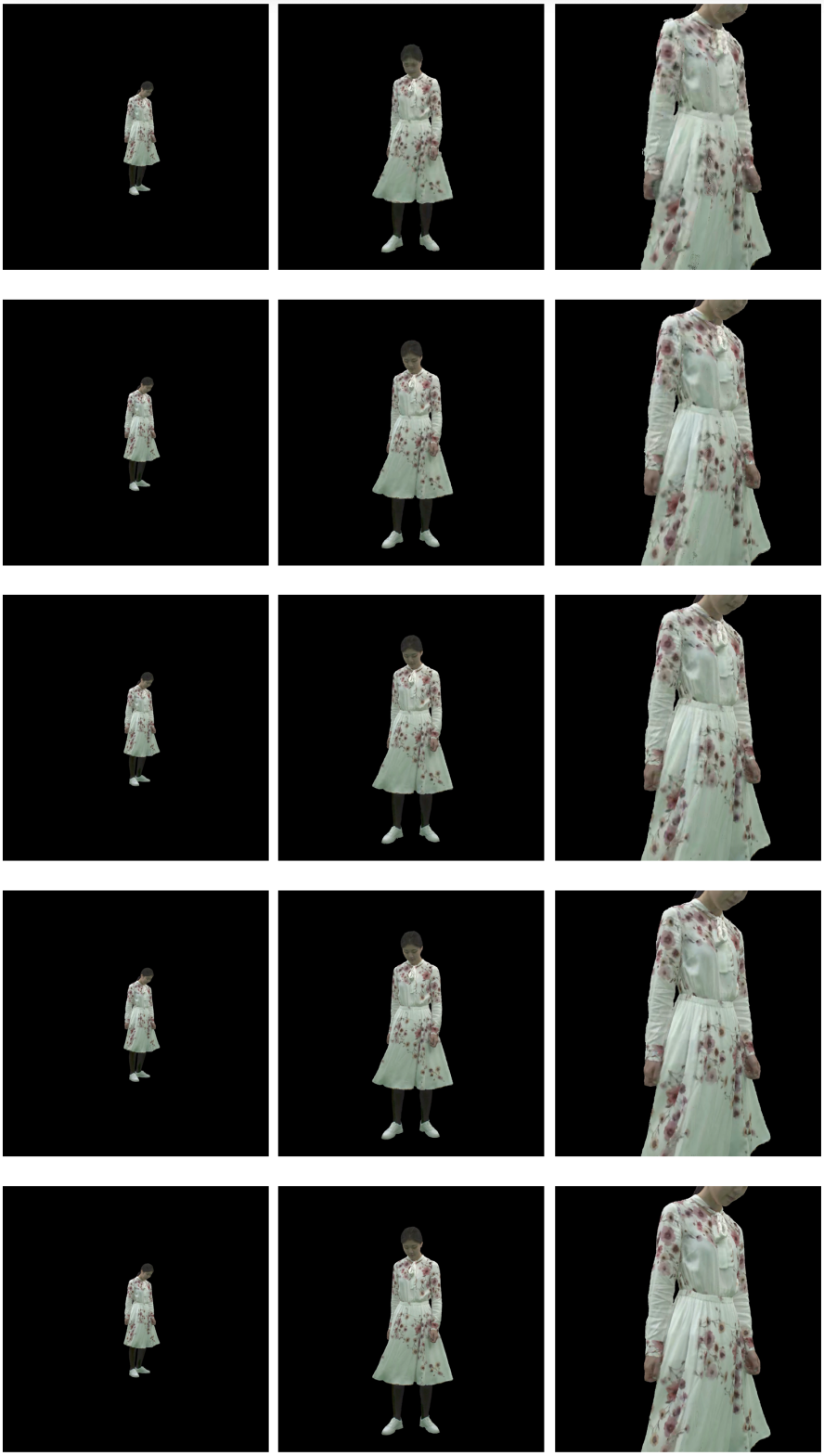

Figure 3.3: Rendered versions of the compressed PC from the 250th frame of *Model*. The bit rates are 3Mbps, 6Mbps, 9Mbps, 15Mbps and 25Mbps from the top to bottom row. From left to right, the observation distance decreases from 5m to 1.5m.

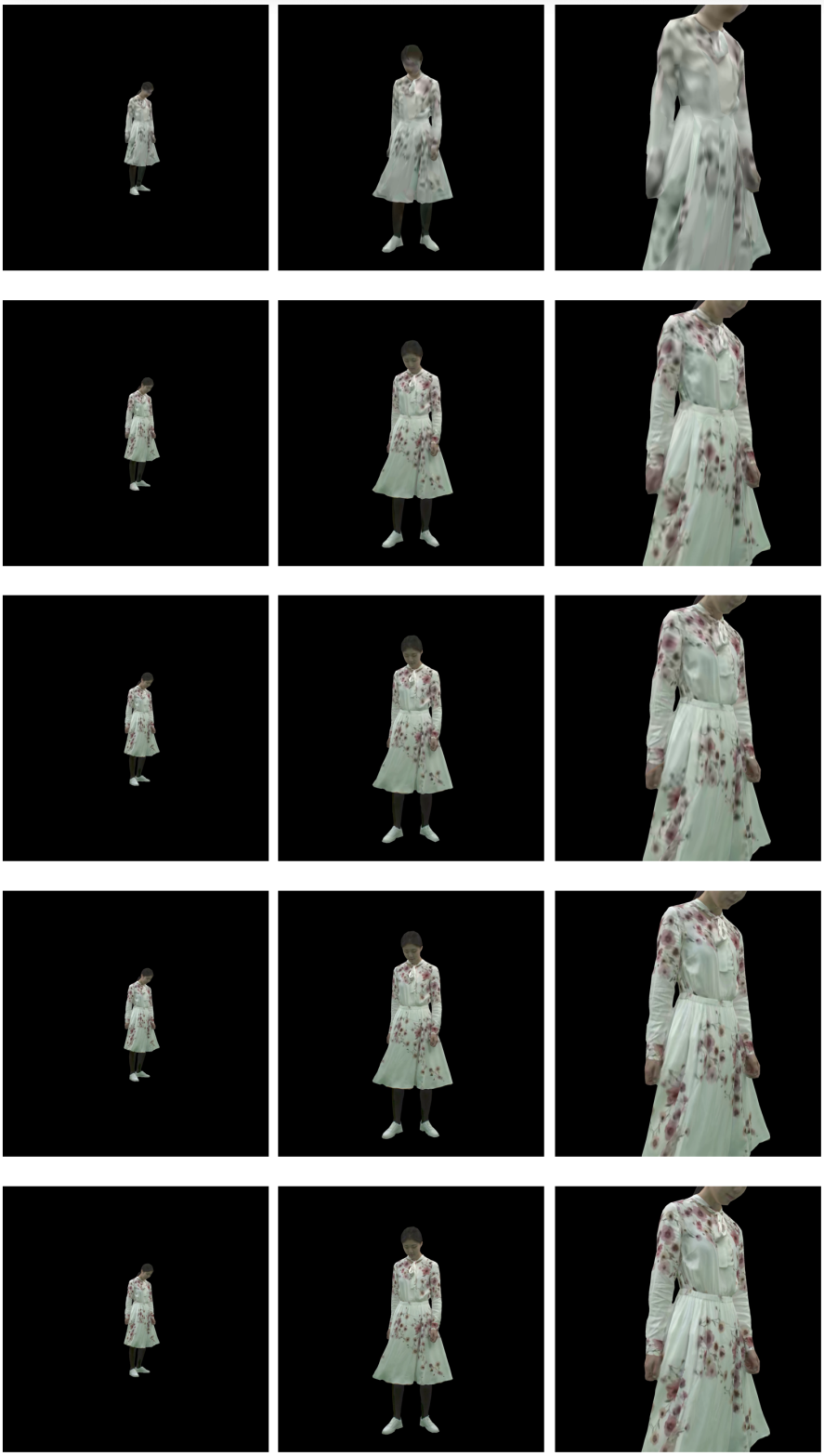

Figure 3.4: Rendered versions of the compressed mesh from the 250th frame of *Model*. The bit rates are 3Mbps, 6Mbps, 9Mbps, 15Mbps and 25Mbps from the top to bottom row. From left to right, the observation distance decreases from 5m to 1.5m.

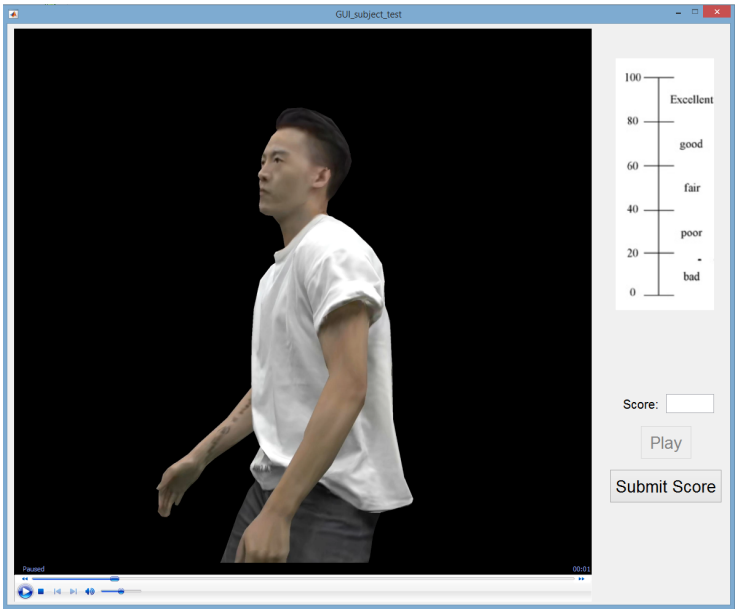

Figure 3.5: User interface for the first part of experiment

The second part aims to compare two videos (A and B) side by side, asking the subject which one is better based on visual quality. The videos are for the same sequence, bit rate and distance; one is from mesh representation and the other is from PC representation. The user interface for this part is shown in Fig. 3.6. The order of videos is random as is their placement as A or B. There are five choices: A is far better, A is a little better, they are similar, B is a little better and B is far better. The subject makes a choice and clicks 'Submit' to proceed to the next pair. This part of the experiment takes about 15 minutes and includes 60 video pairs.

3.3 Experiment Results

3.3.1 Ratings of Mesh and Point Cloud Sequences

For the first part of the experiment, viewers rated sequences separately, and for this data we need to normalize scores and handle outliers. Given the rating range from 0 to 100, scores from different viewers tend to fall in quite different subranges, so we first normalize raw scores.

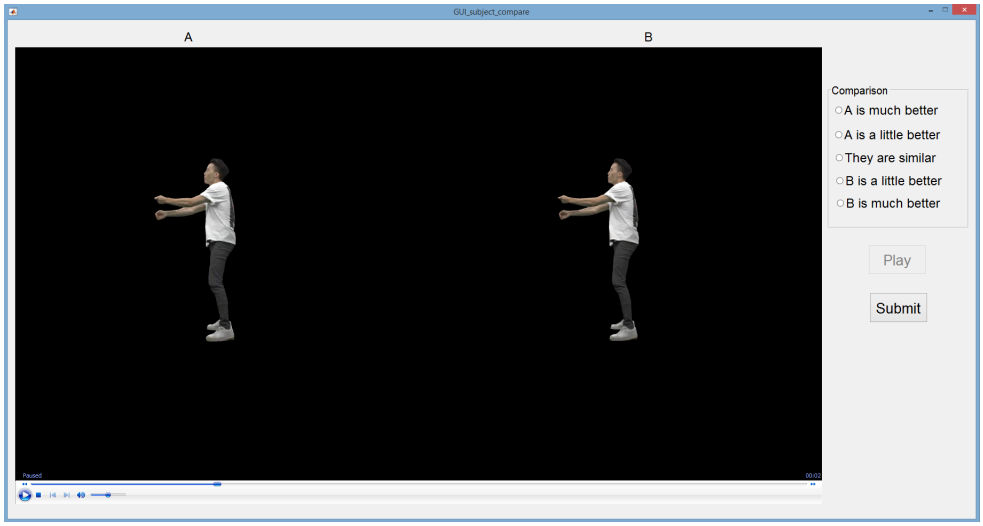

Figure 3.6: User interface for the second part of experiment

We find the minimum and maximum scores given by each viewer for a specific sequence. For each sequence, the median of the minimum scores across viewers is denoted S_{min} (likewise S_{max}), and all viewers' scores for the sequence are normalized to the range from S_{min} to S_{max}.

We adopt a screening method from [59] to eliminate scores that are outliers or inconsistent. This method, also used in [60], makes use of the fact that our test contains videos at different bit rates.

1) First, we aim to remove outlier scores. For each video ζ, we determine the mean, standard deviation and kurtosis, denoted $\bar{u}_\zeta$, σ_ζ and $\beta_{2\zeta}$. When $2 < \beta_{2\zeta} < 4$, the distribution of scores for that video is close to normal, and a score outside the range $[\bar{u}_\zeta - 2\sigma_\zeta, \bar{u}_\zeta + 2\sigma_\zeta]$ could be regarded as an outlier. If $\beta_{2\zeta}$ is not between 2 and 4, the range for inlier is enlarged to $[\bar{u}_\zeta - \sqrt{20}\sigma_\zeta, \bar{u}_\zeta + \sqrt{20}\sigma_\zeta]$. For each viewer, we will reject the 30 scores of this viewer for a given sequence if there are two or more scores above the upper end of the inlier range, or if there are two or more scores below the lower end of the inlier range.

2) If scores from a viewer are consistent, the rating for a lower bit rate should not be larger than that for a higher bit rate. All 30 scores of a viewer for a certain sequence are rejected if there are more than two times that the user gives a score at any lower bit rate more than K

times larger than the score given by the same user at any higher bit rate for the same sequence and observation distance. Here $K = 1.3$ is chosen empirically; the consistency is good enough to show the scoring trend without rejecting too many scores from viewers.

After screening, there are on average 22 user ratings for each sequence. We average viewer scores for the same video to determine its mean opinion score (MOS). We divide all MOS values with the highest score for each sequence and each representation to normalize the highest score to 1.

We plot curves of normalized MOS vs. bit rate in Fig. 3.7. The curves show that increasing bit rate produces better visual quality, and for a given bit rate, closer observation distances generally receive lower quality scores than farther observation distances. In Fig. 3.8, we plot curves of MOS vs. distance. Given a fixed bit rate, the relationship appears close to linear, and the slope and intercept for each line depend on the bit rate. In Section 3.3.3 we will use the data in Figs. 3.7 and 3.8 to create a model that predicts the MOS as a function of bit rate and observation distance.

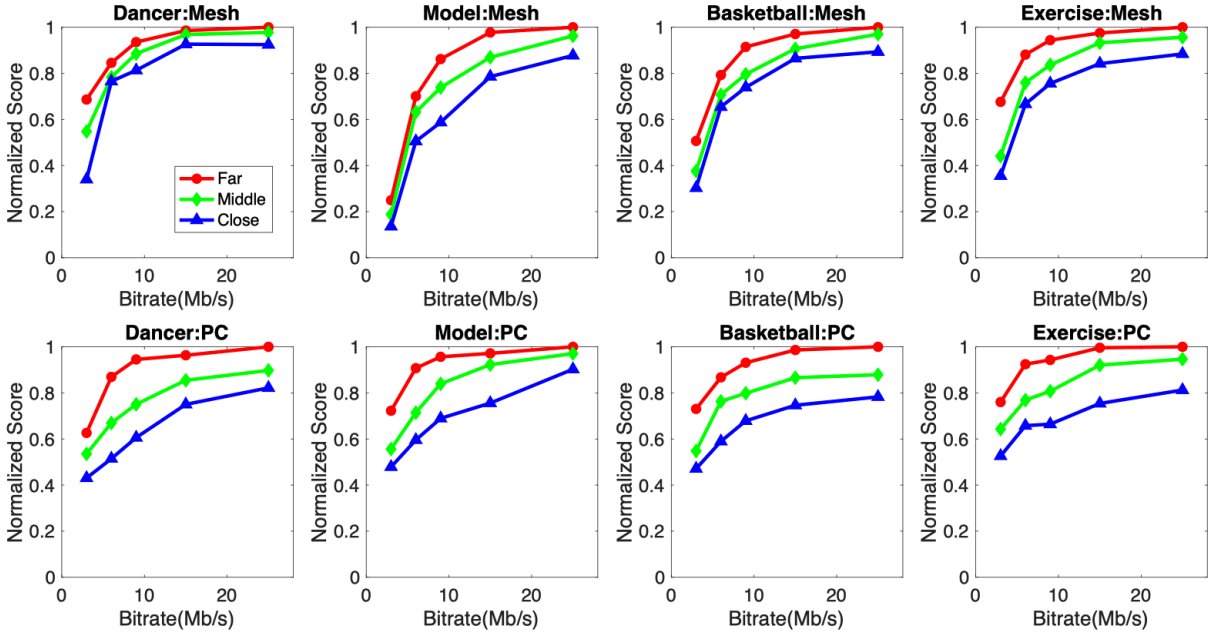

Figure 3.7: Curves of MOS vs. Bit rate for both mesh and PC compression at different observation distances

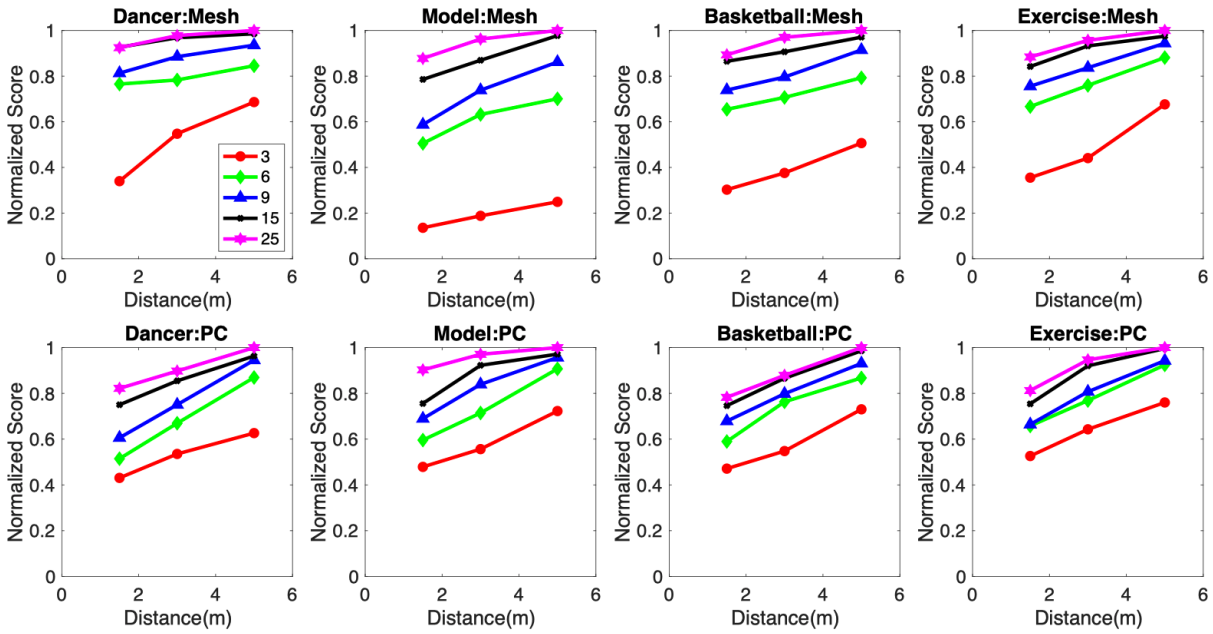

Figure 3.8: Curves of MOS vs. Distance for both mesh and PC compression for different bit rates (Mbps)

3.3.2 Preference for Mesh and Point Cloud Compression

In the second part of the experiment, viewers compare sequences side-by-side, and here we use all data points and for this preference there is no normalization done. Bar charts of the data are shown in Fig. 3.9. The sequence *Dancer* contains fast motion and the sequence *Model* contains detailed texture. Sequence-specific conclusions that we draw from these plots are:

1. For *Basketball*, *Exercise* and *Model*, people prefer PC over mesh compression at low bit rates (e.g. 3Mbps), especially for *Model*, whose texture is richest. Mesh texture appears more blurry than PC texture at low bit rates.

2. For *Dancer* at relatively low bit rates, mesh compression is better than PC. A PC is composed of discrete points which might cause bad artifacts (such as holes) when the motion is fast, as in *Dancer*.

Combining all sequences, the left bar chart in Fig. 3.10 shows the total count of people who preferred mesh compression in the comparison. In the middle is the count of those choosing

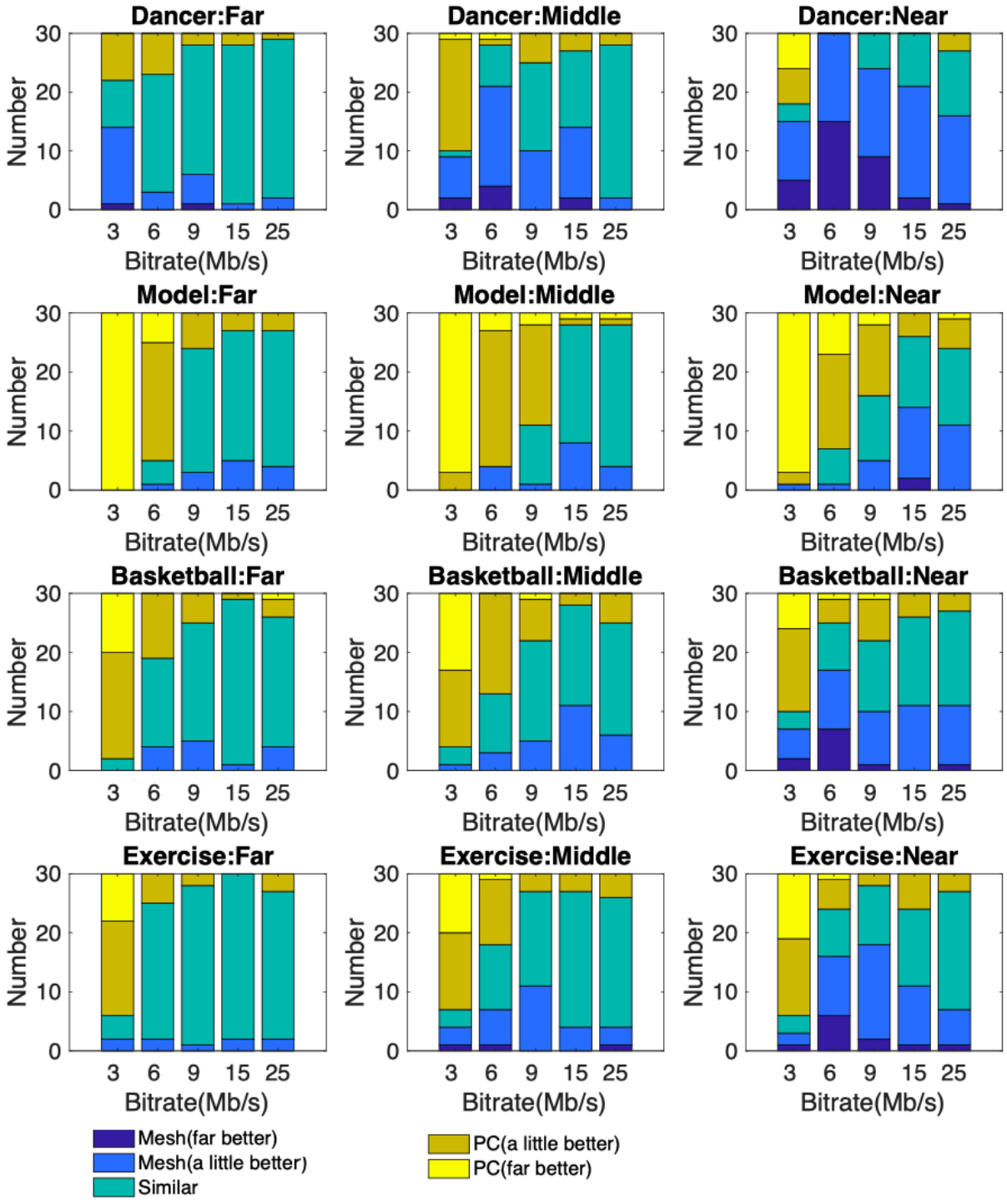

Figure 3.9: Preferences for mesh or PC compression for four different sequences at three different observation distances, across five bit rates

”similar”, and PC preference is at the right. The differences between Fig. 3.10 and Fig. 3.9 are that Fig. 3.9 shows preferences for each sequence while Fig. 3.10 combines all sequences, and Fig. 3.10 combines ’a little better’ and ’far better’ together. Conclusions that we draw from these plots are:

1. There is a general trend that PC compression is preferred at low rates, and the two different representation types become more similar as the bit rate increases.

2. When we observe the sequence from afar, the two representations are similar except at low bit rates.

3. With decreasing observation distance, the preference for mesh compression increases.

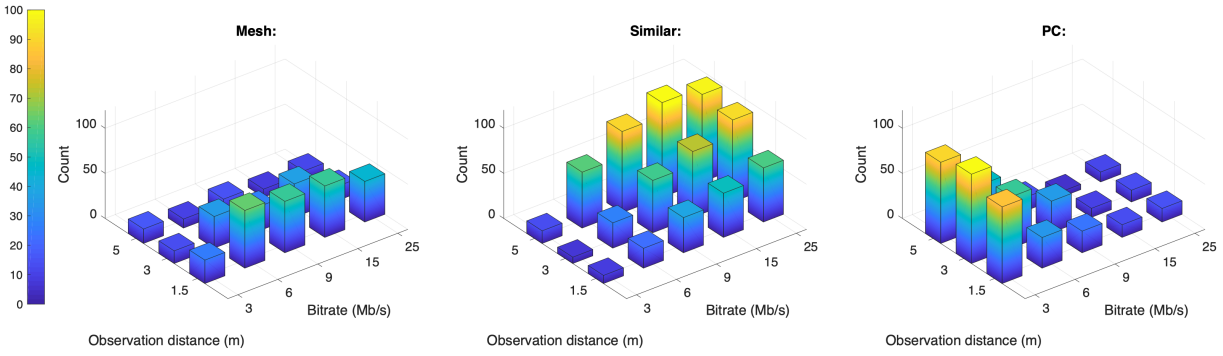

Figure 3.10: Bar chart for preferences: Mesh preferred, similar, and PC preferred

3.3.3 Opinion Score Model

This section proposes an opinion score model that takes bit rate and observation distance as inputs. The model predicts the MOS score using:

$$MOS(r, d) = BQF(r) * ODCF(r, d) \tag{3.1}$$

Here r is bit rate and d is observation distance. Because the video version with the highest bit rate

and farthest distance always gets the highest score, we have $MOS(25Mbps, 5) = 1$. $BQF(r)$ is Bitrate Quality Factor with the formulation adopted from [60]:

$$BQF(r) = \frac{1 - e^{-w\frac{r}{r_{max}}}}{1 - e^{-w}} \tag{3.2}$$

where r_{max} is the maximum bit rate which we take as 25Mbps and w is a parameter for $BQF(r)$. $ODCF(r, d)$ is Observation Distance Correction Factor which we define as the ratio between the score of distance d and the score of the far distance (5) for a certain bit rate r, so $ODCF(r, 5) = 1$. That leads to $MOS(r, 5) = BQF(r)$ and we could use the MOS for far distance to estimate the parameter for $BQF(r)$.

Then, we would like to determine the function $ODCF(r, d)$. We compute the $ODCF(r, d)$ value using Eq. 3.1. From the roughly linear curves of MOS vs. distance shown in Fig. 3.8, and taking the slope and intercept for each line to be dependent on the bit rate, we obtain:

$$ODCF(r, d) = slope(r) * d + b(r) \tag{3.3}$$

where $slope(r)$ is the slope and is a function of bit rate, and $b(r)$ is also a function of r. While the linear fit involves only three points for $ODCF(r, d)$, the three distances represent a wide range of likely observation distances, and the linear fit is simple and performs well in quality estimation, as will be shown in the following evaluation. Since $ODCF(r, 5) = 1$, Eq. 3.3 yields $b(r) = 1 - slope(r) * 5$. We want to determine a relation between $slope(r)$ and r. First, for all 5 bit rates, we fit a linear model to $ODCF(r, d)$ and generate 5 different $slope(r)$. Curves of $slope(r)$ are shown in Fig. 3.11. The relation for mesh compression looks like an inverse proportional function $slope(r) = m/r$ and that for PC compression is a linear function $slope(r) = p * r + t$. Here m, p, t are all function parameters.

Then the final MOS could be computed as the multiplication of $BQF(r)$ and $ODCF(r, d)$:

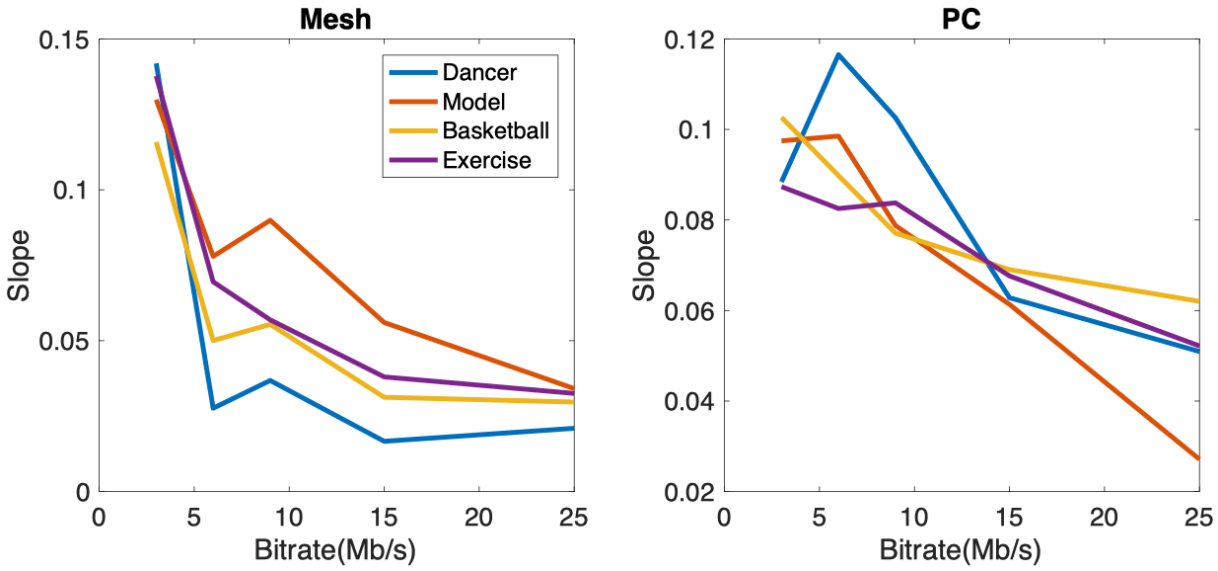

Figure 3.11: Slope vs. Bit rate from which we estimate $slope(r)$

$$MOS_{mesh}(r,d) = BQF(r) * (\frac{m}{r} * d + 1 - 5 * \frac{m}{r}) \tag{3.4}$$

$$MOS_{PC}(r,d) = BQF(r) * ((p * r + t) * d + 1 - 5 * (p * r + t)) \tag{3.5}$$

The fitting is performed using Matlab's built-in functions *fittype()* and *fit()*. Algorithm 1 shows the fitting process as pseudo-code. When using all four sequences for parameter estimation, the fitting result is shown in Fig. 3.12. Pearson Correlation (ρ), Mean Squared Error (MSE), Spearman's Rank Correlation Coefficient (SRCC) [61], Kendall Rank Correlation Coefficient (KRCC) [61] and perceptually weighted rank correlation (PWRC) [62] are used for evaluation. The evaluation results are in Table 3.3.

For comparison, we use the MOS prediction model from [55] which takes $MOS_{predicted} = Ax^3 + Bx^2 + Cx + D$ where A, B, C and D are parameters to be estimated and x is the score from the FR 3D objective quality metric VIFp (Visual Information Fidelity, pixel domain version) which was shown to achieve the best correlation with human perceptual quality in MOS prediction

in [55]. VIFp is carried out on the projected 2D images of 3D content, and it considers color information and geometry information jointly, giving an overall score for input 3D content. VIFp can handle both mesh and PC representations. As in [55], we render each frame of each sequence into 2D along six axis directions, then compute the averaged VIFp of all six directions as the final VIFp value of this frame with the provided tool, Video Quality Measurement Tool (VQMT) [63]. VIFp of the whole sequence is computed as the averaged VIFp across frames. Then we apply the VIFp as input to estimate the parameters of their model based on our measured MOS data. The comparison results of predicted MOS are in Table 3.3. Compared to the model from [55] which is also fit to our collected MOS scores, our model better predicts MOS.

Algorithm 1 Model Fitting for $MOS(r, d)$

Require: Scores for each videos
Consider when $d = 5$, $ODCF(r, 5) = 1$
$MOS(r, 5) = BQF(r)$
Fit model $BQF(r)$ with scores at far distance to get w

Compute $ODCF(r, d) = MOS(r, d)/BQF(r)$
Fit linear model $ODCF(r, d) = slope(r) * d + b(r)$
Based on definition, $ODCF(r, 5) = 1$
$b(r) = 1 - slope(r) * 5$
$ODCF(r, d) = slope(r) * d + b(r) = slope(r) * (d - 5) + 1$

if Sequence is Mesh **then**
$slope(r) = m/r$
Fit $ODCF(r, d)$ to get m
else {Sequence is PC}
$slope(r) = p * r + t$
Fit $ODCF(r, d)$ to get p, t
end if
$MOS(r, d) = BQF(r) * ODCF(r, d)$

Because the MOS scores for all sequences are used to fit the models, Table 3.3 is overly optimistic, for both our approach and that of [55], about the correlation between actual and predicted MOS scores. Therefore, cross validation of the fitting is carried out by removing the scores from one sequence and estimating the model parameters with the remaining three

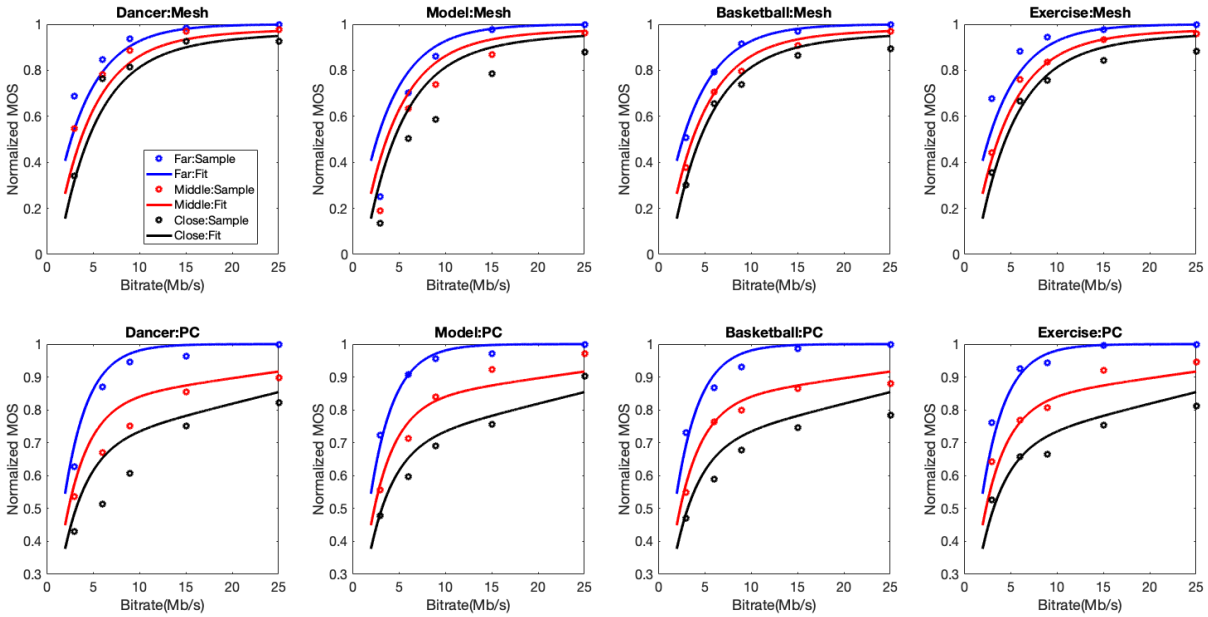

Figure 3.12: Fitting curve of proposed model

Table 3.3: Fitting evaluation of our proposed model and of the VIFp-based model from [55]

Sequence	Mesh					Point Cloud				
	ρ	MSE	SRCC	KRCC	PWRC	ρ	MSE	SRCC	KRCC	PWRC
Our model:										
Dancer	0.977	0.0050	0.979	0.943	0.936	0.983	0.0041	0.993	0.962	0.976
Basketball	0.994	0.0008	0.986	0.943	0.936	0.985	0.0013	0.989	0.943	0.955
Model	0.983	0.0180	0.996	0.981	0.986	0.980	0.0011	0.982	0.924	0.929
Exercise	0.973	0.0029	0.968	0.867	0.878	0.969	0.0014	0.979	0.924	0.933
Model in [55]:										
Dancer	0.870	0.0164	0.964	0.867	0.894	0.946	0.0052	0.946	0.848	0.828
Basketball	0.837	0.0145	0.904	0.810	0.844	0.946	0.0032	0.921	0.790	0.767
Model	0.905	0.0208	0.963	0.880	0.891	0.974	0.0035	0.961	0.867	0.863
Exercise	0.850	0.0104	0.911	0.790	0.818	0.958	0.0033	0.961	0.848	0.854

sequences. Then we compute the correlation measures and MSE between the actual scores and the estimated scores for the sequence left out. Pearson Correlation values, MSE, SRCC, KRCC and PWRC values are shown in Table 3.4 for all four sequences. From Table 3.4, we notice that the correlation coefficients are still promising even though the parameters of the model are estimated from other sequences, and the approach generally outperforms the VIFp model [55] evaluated using the same cross validation strategy.

Table 3.4: Fitting cross validation of our proposed model and of the VIFp-based model from [55]

Sequence	Mesh					Point Cloud				
	ρ	MSE	SRCC	KRCC	PWRC	ρ	MSE	SRCC	KRCC	PWRC
Our model:										
Dancer	0.972	0.0078	0.975	0.924	0.916	0.982	0.0057	0.989	0.943	0.955
Basketball	0.994	0.0008	0.986	0.943	0.936	0.983	0.0015	0.989	0.943	0.955
Model	0.979	0.0292	0.996	0.981	0.986	0.976	0.0013	0.982	0.924	0.929
Exercise	0.964	0.0047	0.939	0.829	0.836	0.966	0.0019	0.979	0.924	0.933
Model in [55]:										
Dancer	0.872	0.0257	0.964	0.867	0.894	0.945	0.0068	0.946	0.848	0.828
Basketball	0.836	0.0156	0.904	0.810	0.844	0.936	0.0042	0.921	0.790	0.767
Model	0.900	0.0289	0.9634	0.880	0.891	0.969	0.0056	0.961	0.867	0.863
Exercise	0.850	0.0111	0.911	0.790	0.818	0.953	0.0057	0.961	0.848	0.854

3.4 Implications for Compression

These subjective test results have implications for the choice of compression method and parameters.

3.4.1 Choice between Point Cloud and Mesh

The second part of the subjective test indicates which representation is better for certain bit rates and observation distances. Fig. 3.9 and Fig. 3.10 provide some guidance on choosing the compression method. If the required bit rate is low, such as 3Mbps, we should choose PC compression regardless of observation distance. Secondly, if the application requires a close

observation distance and the required bit rate is not low, mesh compression should be chosen. For other cases, there is no preference between the representations.

We could also choose the better representation based on the opinion score model. The bit rate and distance are inputs to our proposed models. In the left plot of Fig. 3.13, Δ_{MOS} is defined as the difference between the MOS of mesh and PC compression, $\Delta_{MOS} = MOS_{mesh} - MOS_{point\_cloud}$. The red curve on the surface is where $\Delta_{MOS} = 0$. The surface relates to the bar charts in Fig. 3.10. The right plot of Fig. 3.13 pictures the boundary of $\Delta_{MOS} = 0$ shown as a solid line and the curves where $\Delta_{MOS} = -0.05, -0.02, 0.02, 0.05$ are shown with dashed lines. To the upper left of the solid curve, PC compression is preferred, and to the bottom right, the preference is for mesh. Larger values of Δ_{MOS} indicate stronger preference.

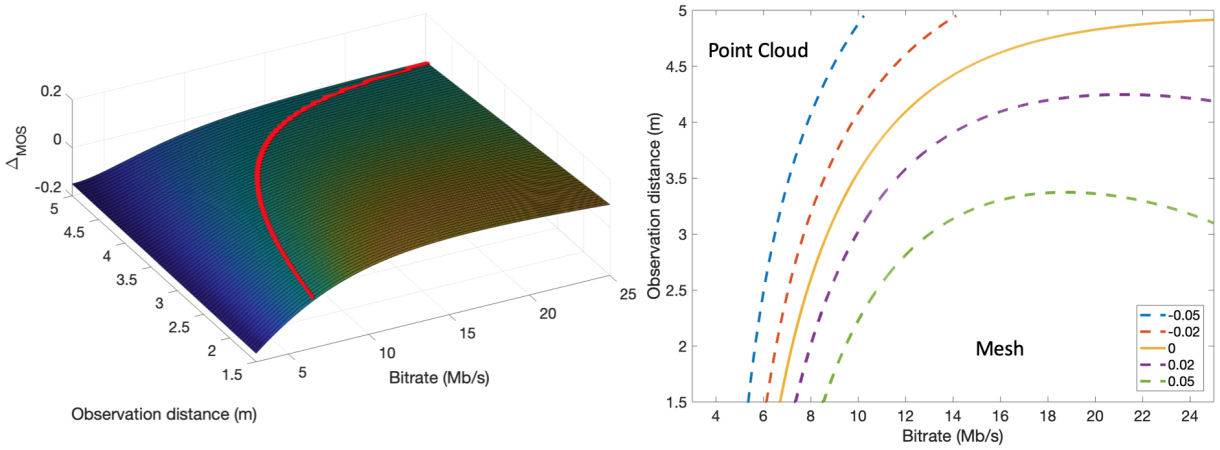

Figure 3.13: (Left) Surface plot of the difference between predicted MOS for mesh and predicted MOS for PC, as a function of bit rate and observation distance, (Right) Isocontours where the difference in predicted MOS takes on the values -0.05, -0.02, 0, 0.02, and 0.05.

3.4.2 Choice of number of triangles and scale factor

There are several compression parameters for PC and mesh compression, including the number of points per frame for a PC which could be controlled by the scale factor, and the number of triangles for a mesh. From the manually chosen compression parameter sets described in

Sec. 3.2 B, we plot the number of triangles N_t and scale factor s vs. bitrate in Fig. 3.14. In this plot, N_t and s are averaged across sequences. From the plot, we notice that the trends are similar across different observation distances, especially for PC compression. The curve for mesh compression at close distance is slightly different because at close distance, people notice more texture details so texture should account for more of the fixed bit rate. That leads to the lower number of triangles for best visual quality.

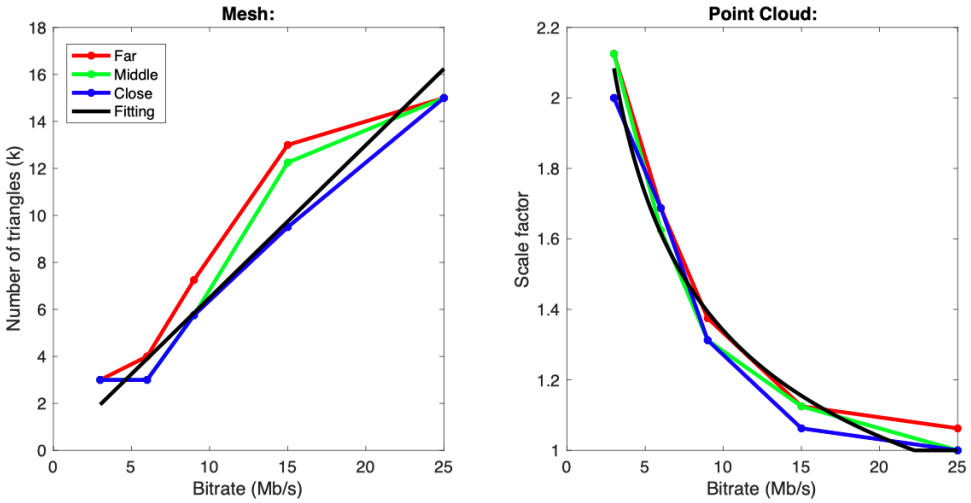

Figure 3.14: Curves of number of triangles vs. bitrate and scale factor vs. bitrate

With increasing bit rate, the number of triangles tends to increase and the scale factor tends to decrease. We do linear and power law fitting as follows:

$$N_t = k * r \tag{3.6}$$

$$s = \begin{cases} a * r^c, & a * r^c > 1 \\ 1, & otherwise \end{cases} \tag{3.7}$$

where $k = 0.6496$, $a = 3.118$ and $c = -0.3667$ are model parameters. The curve fitting result is shown in Fig. 3.14 in black. These models can give a useful rule-of-thumb for estimating the number of triangles or scale factor given the bit rate.

3.5 Conclusion

This study provides several new results regarding subjective quality of compressed 3D content as it relates to choice of representation, observation distance, bit rate, and scaling.

The main conclusions of this chapter are as follows:

- We designed a subjective test to compare the compression quality for PC and mesh representations. PC compression is better for low bit rates, whereas mesh compression is preferred when the observation distance is close and the bit rate is not low. When the bit rate is high, there is little difference between the two representations.

- For the two representations, we propose two models that estimate people's opinion scores and fit the experimental data well under cross-validation. The model can be used to choose a representation based on observation distance and target bit rate.

In addition, when we generated parameter sets for our experiment, we found the general trend that reducing the number of mesh triangles or reducing the number of cloud points improved visual quality at low bit rates. Suggestions for reducing the number of mesh triangles and choosing the point cloud scale factor are provided. Such reductions are not routinely considered part of the compression pipeline for 3D content, but our finding fits the well-known result for 2D content that spatial down-sampling is useful at low bit rates (see, e.g., [64, 65]). Such reductions can play an important role in preserving quality at low bit rates for 3D content as well, and would be worthy of a further subjective study.

Chapter 3, in part, is a reprint of the material as it appears in the paper: K. Cao, Y. Xu and P. Cosman, "Visual Quality of Compressed Mesh and Point Cloud Sequences," In *IEEE Access*, 8 (2020): 171203-171217. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Denoising and Inpainting for Point Clouds Compressed by V-PCC

V-PCC achieves a high efficiency for compression, but compressed point clouds suffer from different kinds of artifacts, such as outliers and cracks, especially for low bit rates. There exist various denoising and inpainting methods that can fix many of these outliers and cracks, however these methods are computationally complex. While a defective sensor might lead to an artifact occurring anywhere in a point cloud, the key observation of this chapter is that artifacts which arise as a result of V-PCC standard compression tend to appear in particular regions and can be characterized in ways that are amenable to post-processing algorithms such as denoising and inpainting to remove them. We develop an approach to artifact removal which directly exploits the point cloud representation in its V-PCC compressed framework, both to locate the artifacts and to aid in removing them, and we conduct a subjective experiment to validate our approach to artifact removal. In this chapter, the contributions are:

- We characterize the types and locations of artifacts that arise with V-PCC. In particular, artifacts tend to appear at patch edges and also where the patch projection direction differs significantly from point normal directions.

- Targeting two different kinds of artifacts, we propose simple but effective denoising and inpainting methods.
- We conduct a subjective test of visual quality; the results indicate our proposed method significantly improves the visual quality of compressed PC sequences and static objects.

This chapter is structured as follows. Sec. 4.1 introduces prior work on PC denoising and inpainting. Sec. 4.2 categorizes different types of artifacts from V-PCC at low bit rates and presents our proposed algorithm. Sec. 4.3 describes the subjective test to evaluate our proposed algorithm and analyzes its results. Conclusions are in Sec. 4.4.

4.1 Related Work

4.1.1 Point Cloud Denoising and Inpainting

Denoising for 2D images is a broadly studied topic. However, due to the lack of a grid as in 2D images, PC denoising is more challenging. Some conventional methods could be transferred to PCs. The method in [66] adopts a moving least-squares method to recover the underlying smooth surface of unorganized points. Noise is located with spectral analysis for PCs in [67] and then a Laplace-Beltrami operator is adopted to smooth the PC surface. Sparsity is taken into consideration in [68] to reconstruct a smooth point set surface. In [69], a bilateral filter is applied to remove noise from 3D scanners. Methods applying neural networks are also proposed in [70–72] to remove noise from PCs.

Hole filling, or inpainting, is another kind of artifact removal process; it consists of two steps, detecting holes and filling holes. Hole detection is achieved through different criteria to identify whether a point belongs to a boundary or not. In [73], a covariance matrix is generated based on the neighbor points of a given point, and then the given point will be identified as being a boundary point or not based on the distribution of the matrix eigenvalues. The detected boundary

points are connected to construct boundary polygons. Halfdisc, angle, and shape criteria are combined in [74–76] to detect boundary points. The halfdisc criterion is based on the property that the local neighborhood of points located on the surface boundary is homeomorphic to a halfdisc as opposed to the full disc of an interior point. Use of the angle criterion begins by projecting all neighbor points of a center point on the tangent plane and then sorting them according to one direction (clockwise or counter-clockwise). The largest angle between two consecutive neighbor points is defined as the angle for the center point. The shape criterion is based on the covariance matrix of neighbors. Four characteristic situations (boundary, interior, corner/noise and line) are classified based on the eigenvalue distribution. Sharp edges are located with normal vectors in [77].

A survey [78] on algorithms for hole filling in 3D surface reconstruction categorizes the methods into surface-based methods and volume-based methods. Surface-based methods fill holes considering properties of the surface, such as boundary curvature, topology and primitive shape. The method in [73] jointly considers coordinates and normal vectors to solve the surface fitting issue. Polynomial surface fitting is applied in [76]; they claim 4th order is sufficient for the fitting function. The method in [79] assumes the hole will have a cone shape and the fitting algorithm is based on properties of cones. Instead of fitting the surface, missing points are inferred from a spline curve guided tensor voting mechanism in [75]. Primitive shapes are extracted in [80] to help with hole-filling.

In contrast to surface-based methods, volume-based methods for filling holes in PCs usually decompose the point space into grid volumes. The method in [81] builds up a signed function around a surface, then applies a diffusion process with a 7-point 'plus' kernel to try to bridge the gap (or hole). Radius basis functions are applied in [82] to expand the information from points to their neighbors trying to fill the hole. PCs are segmented into thousands of overlapped cube regions in [83, 84] and the similarity between cubes is computed for hole-filling.

As a popular tool, neural networks have also been proposed for PC inpainting [85, 86].

In some case, extra information is used; scanned raw data and depth maps are used to infer 3D information in [87], while [88] takes advantage of temporal correlation to reconstruct PCs based on rank minimization theory.

4.2 Proposed Artifact Removal Method

In this section, we first categorize different types of geometric compression artifacts based on the cause for their appearance. Then, artifact detection and removal algorithms are described for each type. A discussion of hyper-parameters involved in the proposed algorithms is included.

4.2.1 Noise Categorization

From observing compressed and reconstructed PCs, there are mainly two kinds of geometric artifacts, outliers and cracks (Fig. 4.1), that are generated by V-PCC. Outliers are points that stand out from an original smooth surface as shown in Fig. 4.1(a). Outliers tend to appear when the geometry information at the edge of a 2D geometry patch is affected by lossy compression, which can lead to reconstructed points flying out of (or inside) the original surface.

Cracks can be classified into edge cracks (Fig. 4.1(b)) and projection cracks (Fig. 4.1(c)). Edge cracks appear at the edges of patches. Separating the cloud into patches causes some points that are neighbors in the original PC to end up at the edges of different patches, subject to different projections. When putting reconstructed 3D patches back together to generate the whole PC after decompression, the edges might not align exactly, leading to edge cracks.

In the segmentation step, each point is assigned one of 6 main directions as the principal direction. However, when dividing the whole PC into patches, smoothing is applied where points with different principal directions might be classified into the same patch. When the patch projection direction has a large angle with the normal direction of a point, the projection fails to preserve surface information which causes projection cracks on the reconstructed surface.

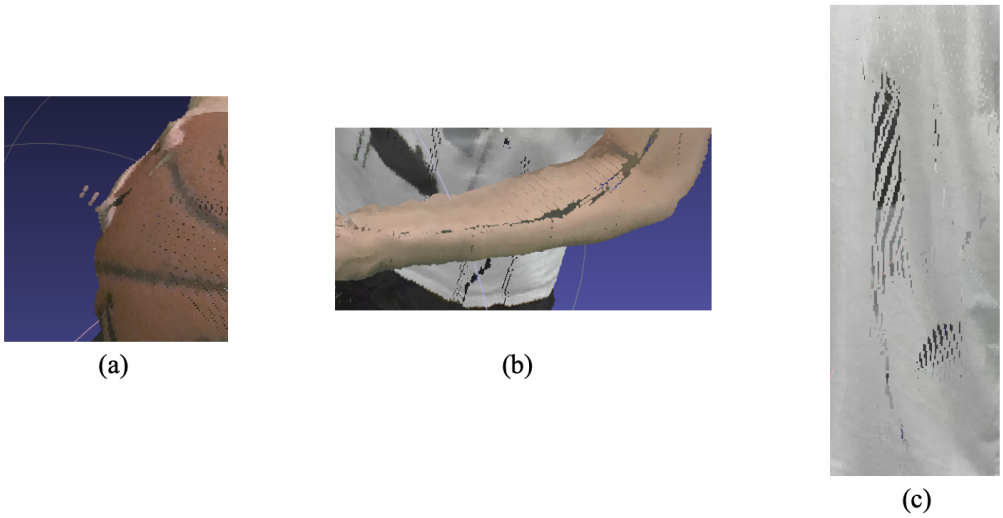

Figure 4.1: Noise types. (a) Outliers. (b) Edge cracks. (c) Projection cracks.

4.2.2 Artifact Region Detection

Since outliers and edge cracks relate to patch edges, we extract information from V-PCC to help locate them. We adopt the same definition as Fig. 2.3 to categorize pixels in a 2D patch into three categories, edge1 pixels, edge2 pixels and interior pixels. In V-PCC, when a 2D patch is converted back to 3D at the decoder, it becomes a cluster of points. 3D points that come from edge1 (or edge2) pixels are called edge1 (or edge2) points. Interior points are converted from interior pixels. After decoding, we treat edge1 points as possible candidates for being outliers, while both edge1 and edge2 points are regarded as where edge cracks might appear.

Given the reason why projection cracks occur, possible projection crack regions could be located wherever the projection direction has a large angle with the surface normal. For a point P with projection direction V_{proj}, we compute the normal vector V_{norm} for P and the angle between them: $\theta = angle(V_{proj}, V_{norm})$. A threshold t_θ is applied; if $\theta > t_\theta$, this point is considered to be a projection crack point.

Fig. 4.2 illustrates detected noise regions. From the visualization, the detected projection and edge crack regions are visually accurate and cover most artifact regions.

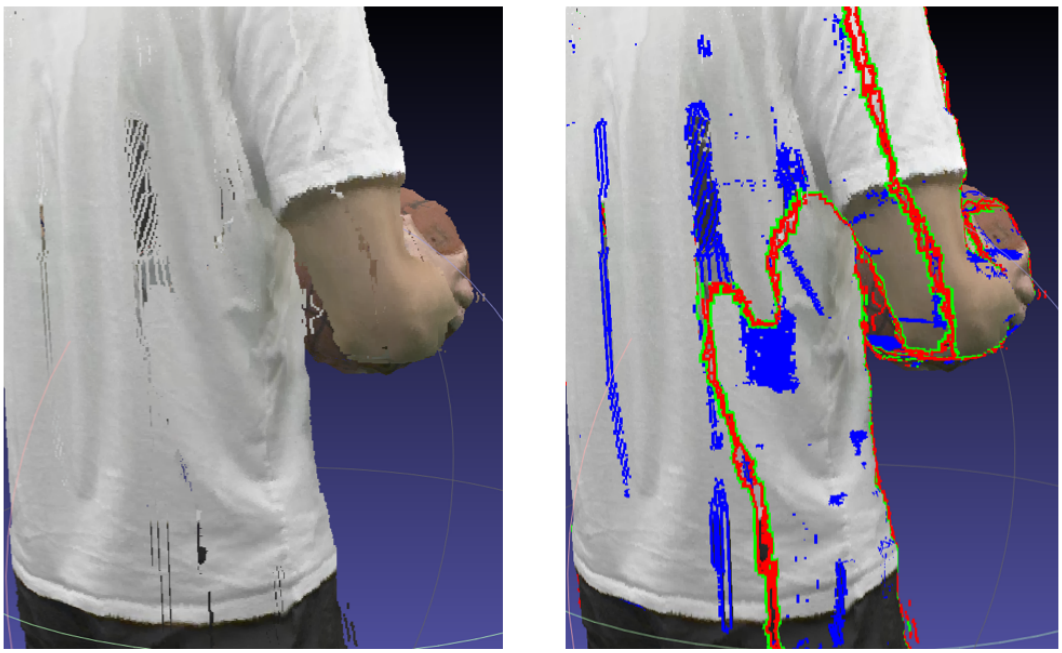

Figure 4.2: Visualization of crack regions. (a) Compressed and reconstructed PC. (b) Red points are candidates for outliers. Red and green points show the edges of V-PCC patches, and are where edge cracks might appear. Blue points show where point normal vectors and projection directions have a large angle, and are where projection cracks might appear.

4.2.3 Outlier Removal and Crack Filling

Outlier removal: Fig. 4.3 shows the flowchart to determine whether an edge1 point is an outlier. For each edge1 point, we find its closest edge2 point in the same patch and compute the distance $d_{outlier}$ between them along the projection direction. If $d_{outlier}$ is greater than a threshold t_o, the edge1 point is classified as an outlier. The orange blocks of Fig. 4.3 provide an example of computing $d_{outlier}$. Fig. 4.4 illustrates the approach. Fig. 4.4(a) shows a 2D patch, where P is an edge1 pixel whose depth is not continuous with its neighbor edge2 pixel. In (b), (c) and (d), we show reconstructed 3D points as seen in three orthogonal planes. We can clearly see that the reconstructed 3D point is an outlier, as shown in the reconstructed PC in (e).

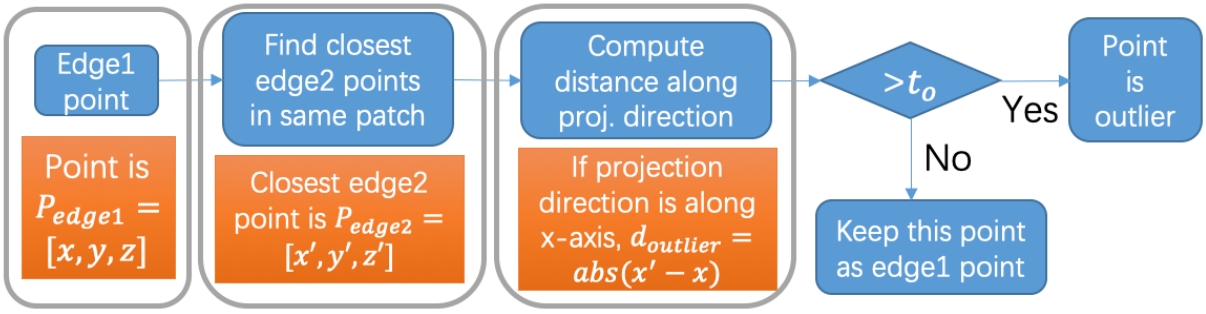

Figure 4.3: Flowchart for identifying outliers

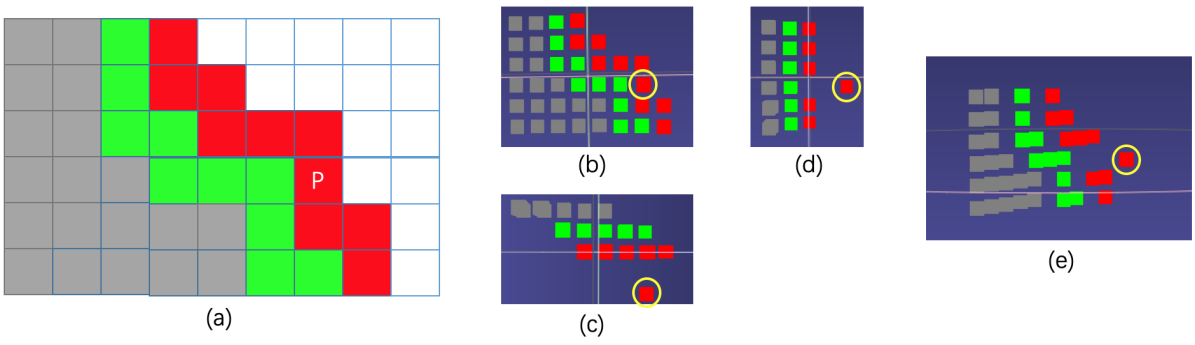

Figure 4.4: Example where $d_{outlier}$ is large. (a) 2D patch, where P is an edge1 pixel; (b), (c) and (d) Reconstructed 3D points as seen in three orthogonal planes; (e) Reconstructed PC, in which P is an outlier.

After identifying an outlier point, we try to correct it by fitting a surface. Fig. 4.5 shows

the pipeline. For each outlier, we first find its nearest k_o neighbors that are non-edge1 points of the same patch, and then fit the points with a polynomial surface of order 4 as suggested in [76]. A new point generated based on the fitted surface is evaluated with the same criterion shown in Fig. 4.3. If the new point satisfies the criterion, the outlier point is deleted and the new point replaces it as a new edge1 point, otherwise, we simply remove the outlier. An example is described in the orange blocks of Fig. 4.5.

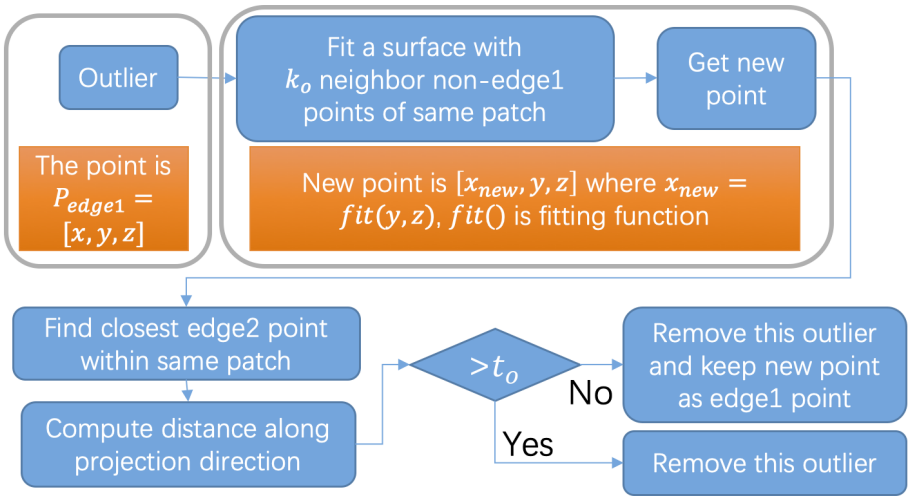

Figure 4.5: Flowchart for removal of identified outliers

A visual example is shown in Fig. 4.6. Fig. 4.6(a) circles a region with outliers; Fig. 4.6(b) zooms in on that region. The marked yellow point is one identified outlier. In Fig. 4.6(c), the red points are the k_o nearest neighbors that are in the same patch as the yellow point. and the fitted surface is shown transparently. The green point in (c) is the outlier after correction, which is on the fitted surface. The green point in Fig. 4.6(d) shows the corrected point. After such processing on all outliers, the section looks as shown in Fig. 4.6(e).

Crack Filling: Fig. 4.7 shows the pipeline for filling cracks. For a crack point P, we find its $k_{surface}$ nearest neighbors and project them along the x, y, and z directions. For each projection direction, (a) a binary image B is generated based on whether a pixel is occupied after projection, (b) binary closing is applied with radius r_{disk} to obtain B', and (c) the difference $B - B'$ is

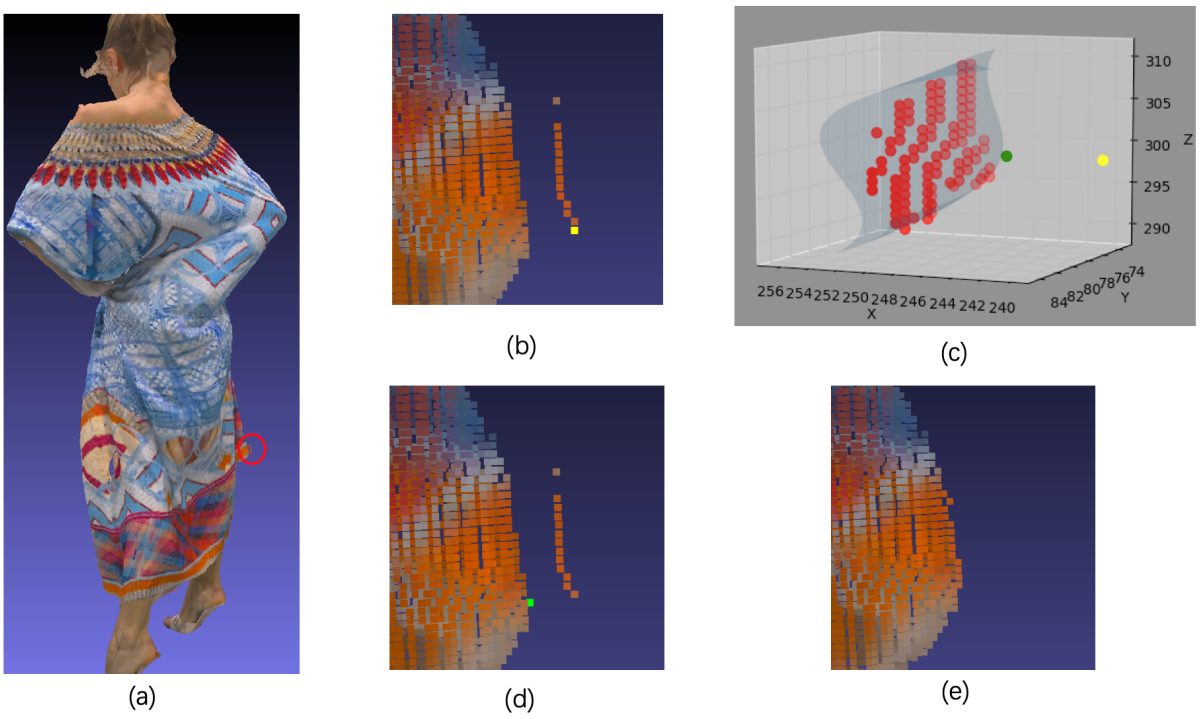

Figure 4.6: Example of removing outliers

computed. The main direction is the direction for which $B - B'$ has the most points, and also for which the number of points in B is not the smallest among the three projection directions. If these two criteria are not satisfied at the same time, we skip this crack point. For the example in Fig. 4.7, the x direction is chosen as the main direction. The second criterion aims to avoid a wrong projection direction for filling the crack; in the case of Fig. 4.7 the y direction runs parallel to the surface and produces the smallest area in the projection, so the y direction should not be the main projection direction for crack filling at that location. After determining the main direction, candidate points will be generated based on the fitted surface and difference between B' and B. For pixels that are in $B' - B$, their pixel positions are fed into the fitted surface function to compute the third component for coordinates. Some of these candidate points might be outliers so they are checked with the outlier detection pipeline (Fig. 4.3) before being added to the PC. The attribute of the added point is assigned to be the same as the attribute of its closest point.

Combining together the modules for outlier detection and removal, and for crack detection

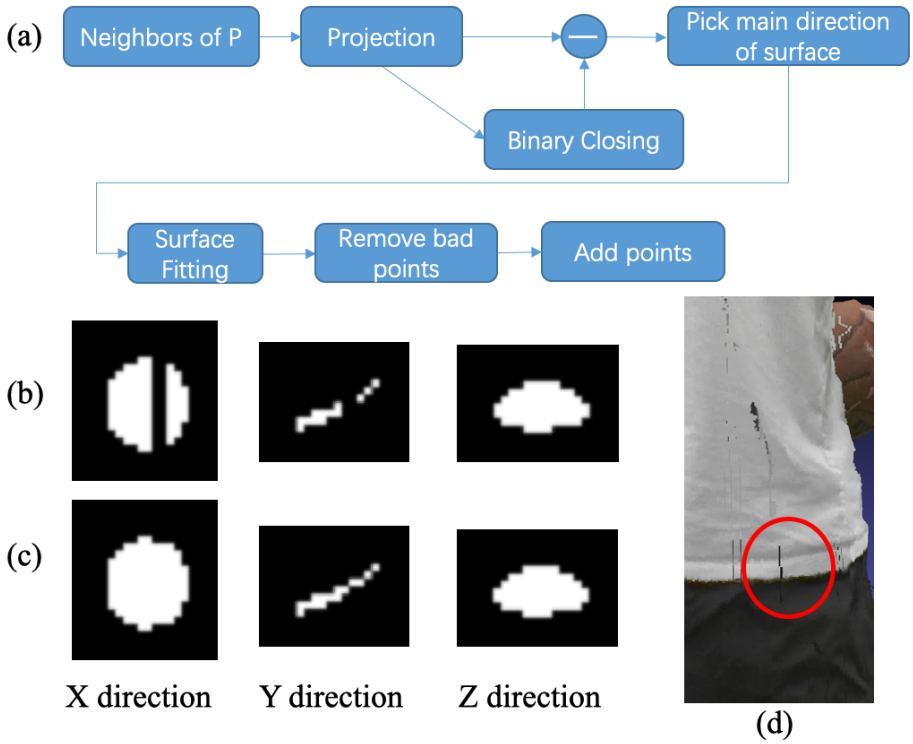

Figure 4.7: (a) Pipeline for filling cracks. (b) Binary images before closing operation. (c) Binary images after closing operation. (d) The red circle denotes the crack region where we implement projection.

and filling, we obtain the whole pipeline for artifact removal (Fig. 4.8). Since the cracks might be large, a loop structure is added to fill large cracks progressively. A parameter $ratio_{added} = \frac{N_n}{N_{n-1}}$ is computed for each iteration of the loop, where N_n is the number of points added in the nth iteration. A threshold th_{stop} is set for $ratio_{added}$ to control stopping.

Discussion of Hyper-parameters: The hyper-parameters are listed in Table 4.1.

Angle threshold t_θ

For a point P, if the angle θ between the normal vector and the projection direction of P is greater than t_θ, we consider P to be a projection crack point. In V-PCC, the best projection direction of each point is picked according to the maximum absolute coordinate value of the normal vector. For example, if a point has normal vector (n_x, n_y, n_z) where max($|n_x|, |n_y|, |n_z|$) =

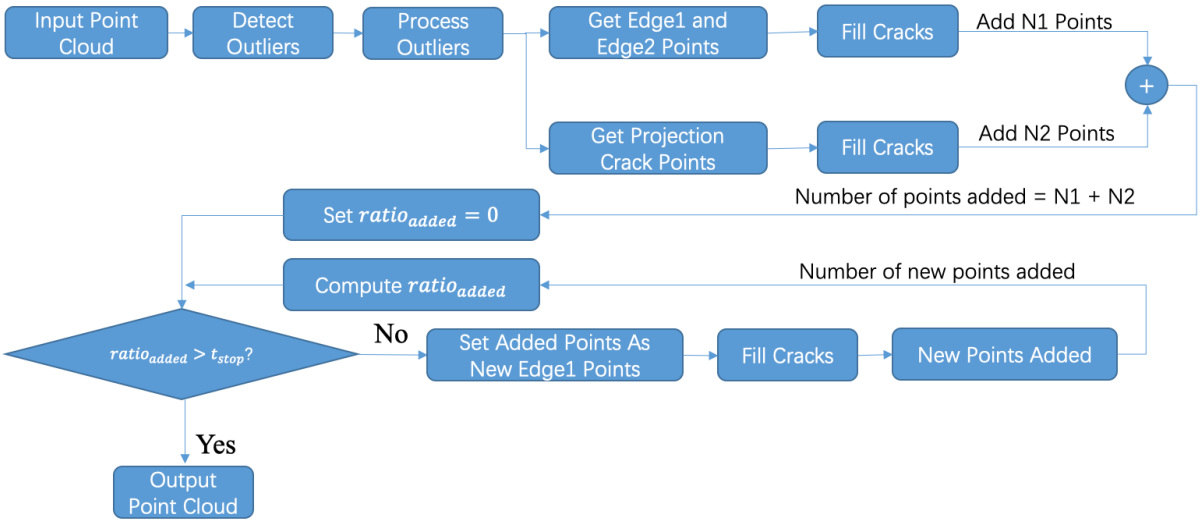

Figure 4.8: Loop pipeline to fill large cracks

Table 4.1: Definitions of Hyper-parameters

Hyper-parameters	Definitions
Locate Projection Crack Regions:	
t_θ	Threshold for angle
Outlier Detection and Removal:	
t_o	Threshold for outlier identification
k_o	Number of nearest neighbors
Cracking Filling:	
r_{disk}	Radius of disk size for binary closing operation
$k_{surface}$	Number of nearest neighbors for surface fitting
t_{stop}	Threshold for stopping loop

n_x, the $+\mathbf{x}$ axis will be chosen as the best projection direction for this point. This best projection direction and the normal direction will have an angle between them which is less than or equal to 45 degrees. However, when dividing the whole PC into patches, to avoid subdividing the PC into too many small patches, smoothing is applied where points with different best projection directions might be classified into the same patch, and a single projection direction is used for the patch. For those points where the best projection direction is not the the projection direction actually used, the angle between the normal and the actual projection direction is greater than 45 degrees. This leads us to choose $t_\theta = 45$.

Threshold t_o for outlier identification

The threshold t_o is used to identify whether a point is an outlier. For the PCs *Basketball*, *Longdress* and *Can*, Table 4.2 provides the average nearest neighbor distance d_n as well as the percentage of points in the original PC that have unit distance to their nearest neighbor. Here, $d_n = \frac{1}{N}\sum_{i=1}^{N} d_i$ where N is the total number of points in the cloud and d_i is the distance from point P_i to its nearest neighbor. The PCs we use as input to V-PCC are dense and have integer coordinates, so nearly all points in the original PC have unit distance to their closest neighbor. Given this high fraction of points with distance 1 to their nearest neighbor, we choose $t_o = 1$.

Table 4.2: Statistics of nearest neighbor distances for original PCs, and estimated crack widths for QP = 20, 30, and 40.

File	Distance	% dist=1	Average Crack Width
Basketball	1.00000	99.99989%	2.298, 2.549, 3.501
Longdress	1.00011	99.97657%	2.565, 2.856, 3.665
Can	1.00093	99.77452%	3.430, 3.676, 4.021

Disk radius r_{disk} for binary closing operation

Table 4.2 has a rough estimate of average crack width for QP=20, 30 and 40. The crack width is estimated as follows. After handling outliers, for each edge1 point, we find its closest

edge1 point from another patch, then compute the distance between those two edge1 points. We take the average of those distances that are greater than $\sqrt{3}$ because two points with distance larger than that cannot be 26-connected in a 3D integer space, so this could be considered a crack. We regard this average as an estimate of average crack width. Crack width generally increases with increasing QP. From the statistics and visual inspection, we choose $r_{disk} = 3$; using smaller values would fill smaller cracks at the cost of increased complexity.

Number $k_{surface}$ of nearest neighbors for surface fitting

Surface fitting performance under different numbers of nearest neighbors was evaluated by randomly choosing 30k starting points for surface fitting from a PC. For each point, the nearest $5 * k_{surface}/4$ points are found, of which a random $k_{surface}$ points are used for surface fitting and the remaining $k_{surface}/4$ points are for testing. Table 4.3 reports the average fitting error (MSE) for *Longdress*; $k_{surface} = 128$ achieves the lowest MSE in this set. The MSE for $k_{surface} = 32$ is large because using only 32 points leads to overfitting. Other PCs showed similar results, with $k_{surface} = 128$ achieving the lowest MSE.

Table 4.3: Average fitting error of different numbers of points

$k_{surface}$	32	64	128	256	512	1024
MSE	63.190	0.766	0.454	0.548	1.191	3.092

Threshold t_{stop} for stopping loop

Fig. 4.9(a) shows the number of points added after each crack filling loop, normalized by dividing by the number of points added in the first loop. The number of points added drops sharply initially, then becomes flatter. Fig. 4.9(b) shows $ratio_{added} = N_n/N_{n-1}$ where N_n is the number of points added in the nth loop. Choosing a stopping criterion of $ratio_{added} > t_{stop} = 0.6$ worked well based on visual inspection; obvious cracks are filled on the surface. Choosing 3 or 4

loops as a stopping point produced similar results to using $ratio_{added} > 0.6$.

With those preset hyper-parameters, the running time for one frame with around 1 million points is approximately 150s compared with decoding time of 6s on an Intel i7 CPU. The overall setup is highly suitable for parallelization, however, since only local points are involved in the outlier removal and crack filling operations; if multi-threads and parallel computing were applied, the time efficiency would improve significantly.

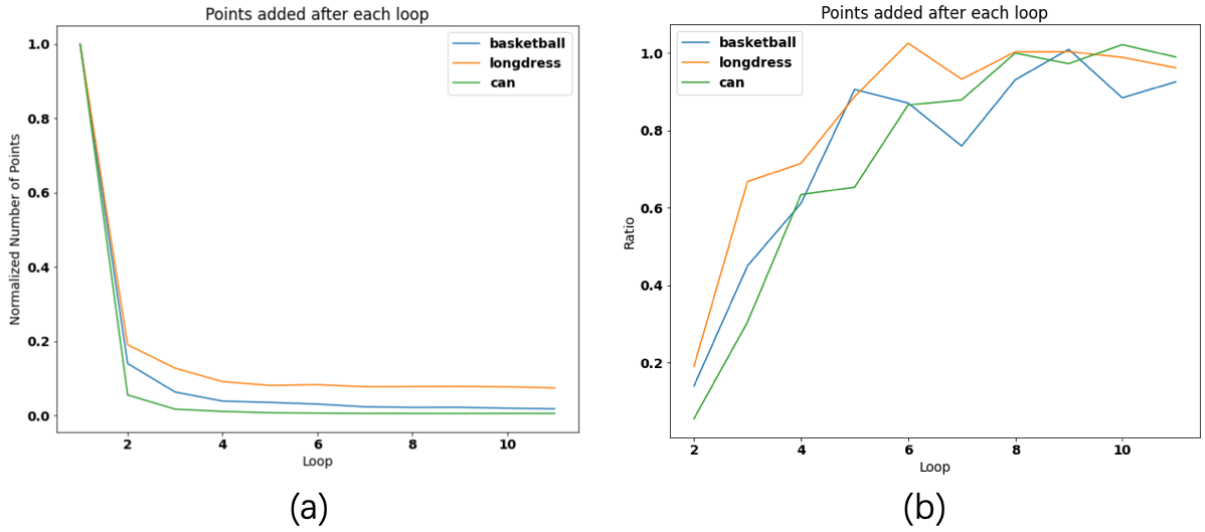

Figure 4.9: (a) Number of points added at each loop. (b) Ratio of numbers of points added at consecutive loops

4.3 Evaluation of Proposed Algorithm

In this section, we describe our subjective experiment to evaluate the proposed algorithm.

4.3.1 Test Materials

We use five 3D dynamic PC sequences (*Exercise*, *Dancer*, *Basketball*, *Redandblack* and *Soldier*), each of which has 300 frames, as well as four static object PCs (*Skull*, *Earth-simple*, *Earth-complex* and *Can*). Each sequence frame and each static PC contains approximately 0.8

million points with color information as attribute. Fig. 4.10 illustrates the content. *Exercise* contains a man wearing solid-color clothing; he exercises slowly, with a large movement range. In *Dancer*, a man in solid-color clothing does an urban dance with fast movement and large movement range. *Basketball* involves two objects, a basketball and a man; he plays with the basketball with a moderate motion speed. Both the ball and the man's shirt have some color variation. In *Redandblack*, a woman moves slowly and swirls her skirt. Lastly, *Soldier* shows a male soldier wearing camo and holding a gun. The motion is slow and motion range is small. For static objects, *Skull* is a head skull with subtle texture and a complicated shape. *Earth-simple* and *Earth-complex* are spheres with different texture complexity levels. *Can* is a regular cylinder shape. We randomly pick one frame from each sequence and use it as a static object so the experiment consists of five sequences and nine static objects in total.

Figure 4.10: Sequence and static object examples. From left to right, the sequences are *Exercise*, *Dancer*, *Basketball*, *Redandblack* and *Soldier*. The static objects are *Skull*, *Earth-simple*, *Earth-complex* and *Can*.

4.3.2 Experiment Design

The test materials are first compressed with V-PCC. We set QP=8 for the texture sequence and use three different QP values for the geometry sequence, 20, 30 and 40, corresponding to

roughly 0.08bpp (bits per point), 0.04bpp and 0.018bpp for the geometry sequence. The artifact removal process is used on the decompressed PCs. For the test, we render the 3D content into 2D videos with a preset camera trajectory as in [89], shown in Fig. 3.2. Using OpenGL in a Linux OS, we obtain rendered 2D images of the 3D content out of which we generate video sequences that are shown to subjects. The virtual viewing trajectory makes a full 360 degree turn around the main figure and the virtual viewing trajectories are chosen to present a large set of angles and details to be examined by the subjects. The motion trajectory of the virtual viewing camera is the same for the PC sequences and for the static PC objects.

The rendering is applied on both the decompressed PC directly from V-PCC and the PC processed with our proposed method. There are 42 pairs of rendered videos in total, corresponding to 3 QP values and 14 contents (5 sequences and 9 static objects). Each pair consists of a video with the artifact removal post-processing and one without. Each video lasts 10 seconds.

Twenty-three subjects (15 males, 8 females, age range 18-33) participated in our subjective experiment. Given the current pandemic circumstances where people are restricted from coming to university labs, we used a web-based user interface so that subjects could participate in the experiment in their own rooms. Before starting, subjects see a few example videos to calibrate their preference standard. The interface has the same layout as Fig. 3.6 in Chapter 3, which shows two videos side by side. Subjects mark their preference based on visual quality. There are five choices: left is much better than right, left is a little better than right, left and right are similar, left is a little worse than right and left is much worse than right. The order of videos is random. In addition to the 42 regular pairs of videos with and without artifact removal, subjects are also presented with 6 duplicated pairs and 2 identical pairs. The duplicated pair is a repeat of a regular pair but with the placement of left and right switched, and an identical pair is one with left and right videos exactly the same. There are 50 pairs shown in total, and the experiment takes approximately 20 minutes.

4.3.3 Experiment Results

Result Screening

First, we used the 6 duplicated pairs and 2 identical pairs to check subject consistency.

For identical pairs, the subject should mark "similar." Fourteen of the subjects marked both of their identical pairs as similar. The other 9 subjects marked one of their identical pairs as "similar" and the other as having one side "a little better" or "a little worse". We considered this acceptable. There were no cases where a subject marked "much better than" or "much worse than" for the identical pairs.

For duplicated pairs, the preference on the second viewing should be consistent with that of the first. We converted the responses to numbers, where 0, 1, 2, 3, 4 denote, respectively, that after processing, the quality is much better, a little better, similar, a little worse, and much worse. If the preference for the first time is m and for the second occurrence of the pair is n, we define for each subject *WeakMismatch* as the number of cases when $abs(m-n)=1$, and *StrongMismatch* as the number of cases when $abs(m-n)>1$. For each subject, we reject all data from that subject if *StrongMismatch* >1 or *WeakMismatch* >4. Table 4.4 show the statistics of the variables for screening. One subject was rejected for having *StrongMismatch* =2; data of the other 22 subjects are considered valid.

Table 4.4: Mismatches among duplicated pairs

Number of duplicated pairs	0	1	2	3	4	5	6
Count of subjects with *StrongMismatch*	17	5	1	0	0	0	0
Count of subjects with *WeakMismatch*	3	10	7	1	2	0	0

Results of Subjective Test

In the following, we use the terms ‘Much Better’, ‘Much Worse’, ‘A Little Better’ etc. to refer to preference choices for the quality after the artifact removal process. Fig. 4.11(a) shows a histogram of preferences combining all QP values and all content. In nearly all cases, the post-processed content was seen as either ‘Much Better’, ‘A Little Better’ or ‘Similar’. Fig. 4.11(b) divides the data by different QP values. We observe that when QP=40, ‘Much Better’ is the most commonly chosen response, whereas for QP=30, ‘A Little Better’ quality is often achieved by the post-processing algorithm. When QP=20, the overall quality is high and there is little noise, so ‘Similar’ is chosen most often. Nonetheless, for nearly one third of the cases, the algorithm shows some improvement.

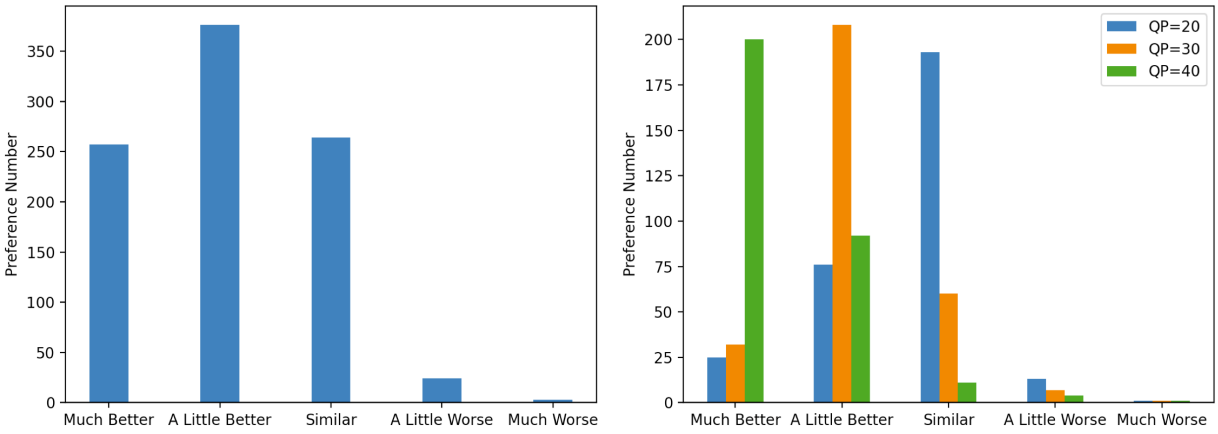

Figure 4.11: (a) Preference bar plot for all QP values and all data. (b) Preference results across different QP values.

In considering different kinds of data, first we look at results for the 5 sequences (Fig. 4.12). For *Soldier*, the percentage of cases where ‘Much Better’ is chosen is lower than for the other sequences, because the texture of *Soldier* is more complex and that tends to mask the distortion. When we look at results for different QP values (Fig. 4.13), conclusions similar to those for Fig. 4.11(b) could be drawn; when QP=20, ‘Similar’ is chosen the most, but with QP=30, the post-processed quality tends to be ‘A Little Better’ and tends to be ‘Much Better’ when QP=40.

One interesting observation is that when QP=40, subjects tended to rate *Basketball*, *Exercise* and *Redandblack*, 'Much Better' more frequently than for the other two sequences. These three sequences are not fast motion, and all have skin showing, where the smooth texture makes PC cracks highly noticeable. In contrast, for *Soldier*, the texture is complicated which makes it harder to notice the noise, and for *Dancer*, the motion is fast which also can mask the noise.

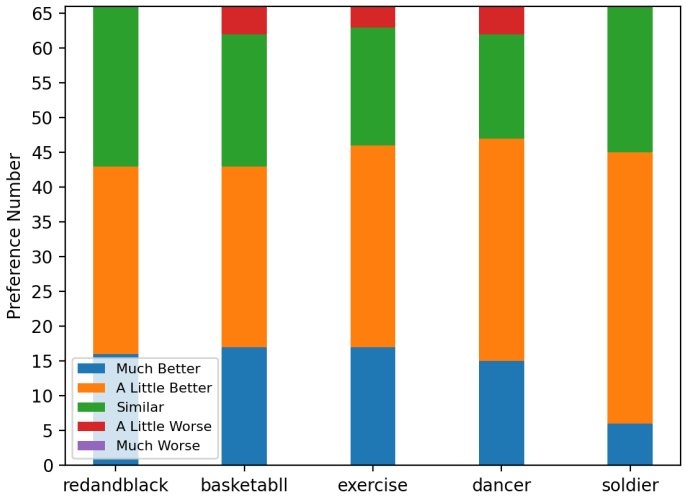

Figure 4.12: Preference for post-processed PC sequences combining all QP values

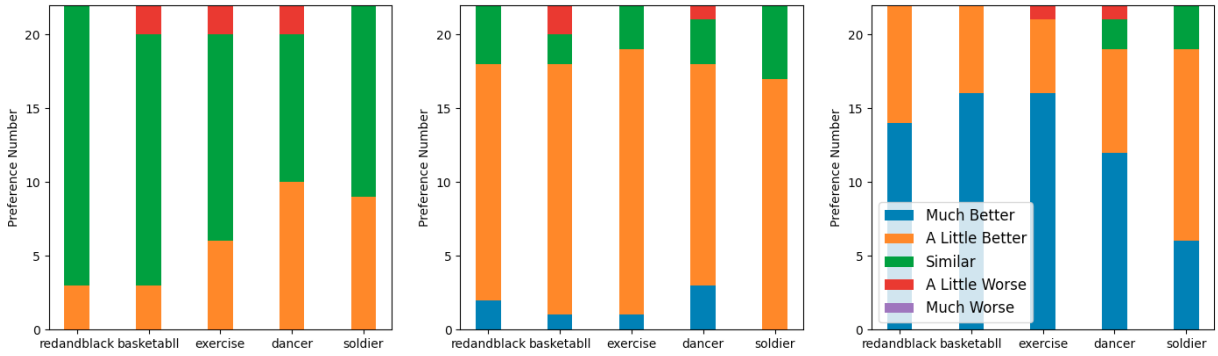

Figure 4.13: Preference for post-processed PC sequences for different QP values. (a) QP = 20, (b) QP = 30, (c) QP = 40.

There are 9 static objects, of which 5 are single frames taken from the 5 sequences. The results for those 5 static frames are shown in Fig. 4.14. Results are somewhat similar to the

dynamic cases. However, for QP=40, *Dancer* has more subjects choosing 'Much Better' quality. Unlike the dynamic case, where there is both fast motion of the dancer in the PC, as well as a slow motion trajectory from the virtual viewing camera circling around the dancer, in the single-frame case, there is only the slow camera trajectory, so the masking effect from fast motion is gone, and subjects more strongly prefer the post-processed version.

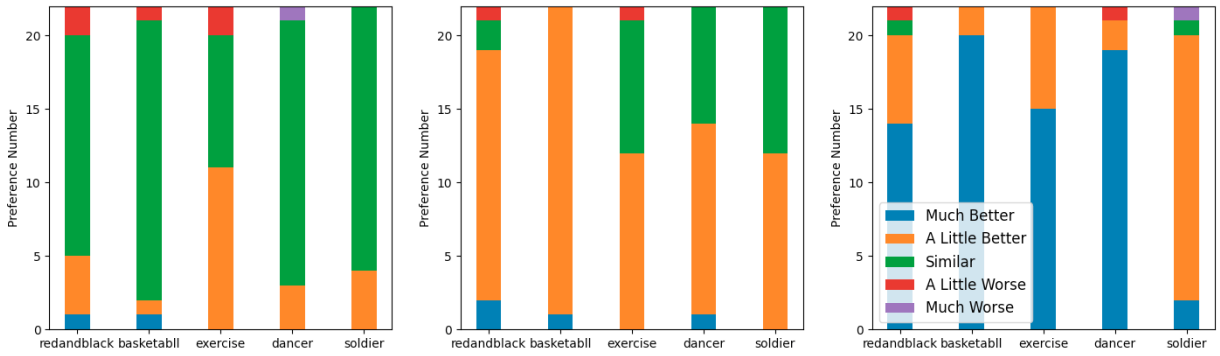

Figure 4.14: Preference for post-processed versions for 5 static frames across different QP values. (a) QP = 20. (b) QP = 30. (c) QP = 40.

Results for the remaining 4 static objects are in Fig. 4.15. *Skull* has subtle fine-grained texture, *Can* is a cylinder, and *Earth-simple* has the same geometry but slightly more simple texture than *Earth-complex*. For *Can*, for all QP values, de-noising achieves 'Much better' quality because the shape is very smooth and simple, and in parts the texture is smooth as well, so the noise is very obvious. When QP=20 and 30, *Earth-simple* has slightly more people preferring the post-processed version than for *Earth-complex*, possibly because there is slightly less texture-masking effect.

Significance Test and Visual Results

To compute a significance test, 'Much Better' and 'A Little Better' are combined together as 'Better'; likewise 'Much Worse' and 'A Little Worse' are combined as 'Worse'. We use the Bradley-Terry model from [90], and handle the responses of 'Similar' as in [91]. We use a one-tailed t-test for which the null hypothesis is that the probabilities for 'Better' and 'Worse'

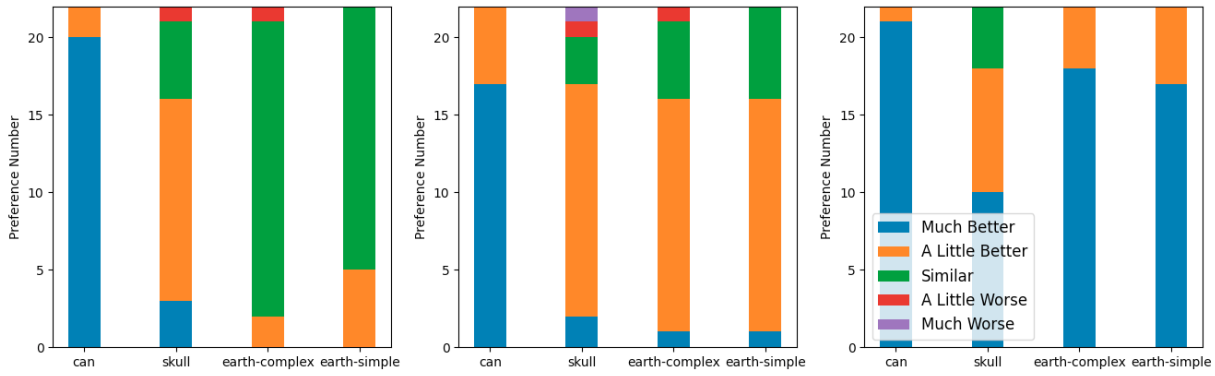

Figure 4.15: Preference for post-processed versions of 4 static objects across different QP values. (a) QP = 20. (b) QP = 30. (c) QP = 40.

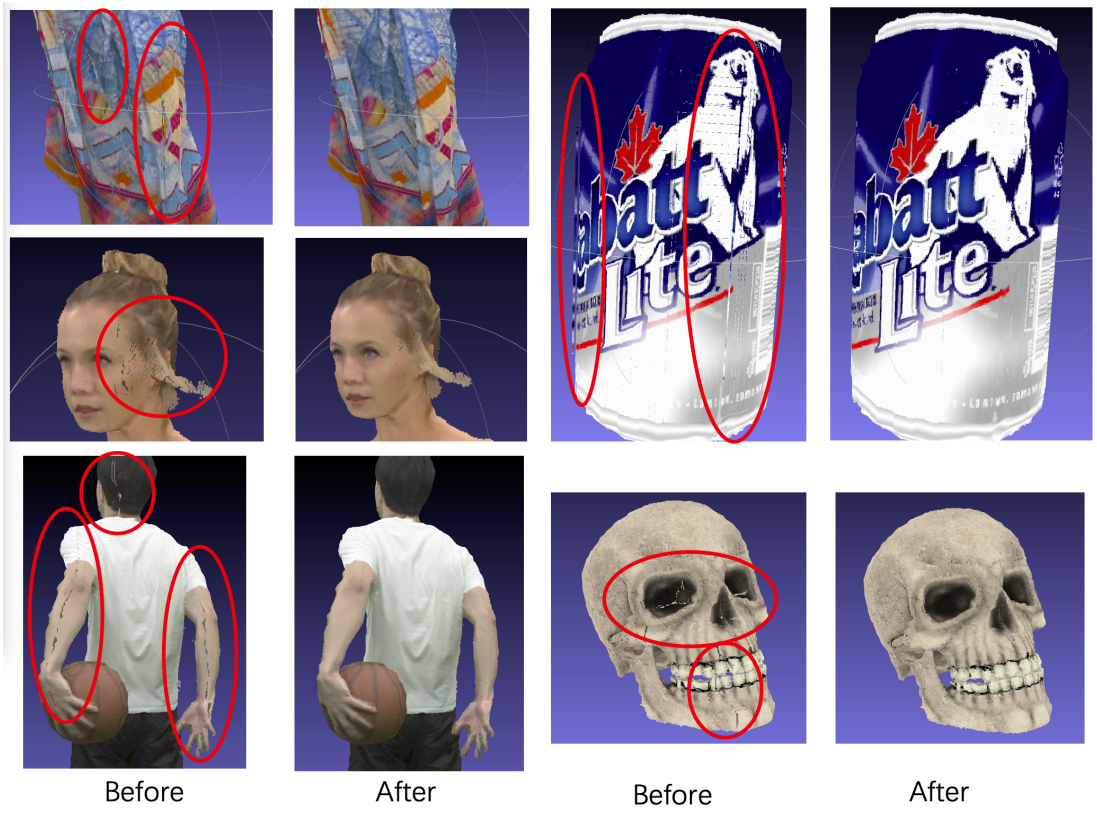

Figure 4.16: Visual examples with noise regions circled.

are the same, against the alternative hypothesis that 'Better' is more likely than 'Worse'. The null hypothesis is rejected for all three QP values; $P = .001$ when QP=20 and $P << .001$ when QP=30 and 40. The algorithm improves visual quality for compressed PCs, and P values decrease with increasing QP.

Fig. 4.16 presents a few visual examples for our whole pipeline to illustrate the improvements.

Comparison with Other Methods

While there is no PC processing method in the literature that aims to both remove outliers and fill holes, we can compare our approach against algorithms that do the tasks separately. Fig. 4.17 presents visual comparisons of our outlier removal approach with the methods in [69] and [72]. A bilateral filtering approach for 2D images is extended to 3D PC space in [69]. It performs well when the noise is Gaussian as shown in [69], but fails on these V-PCC outliers. A neural network *PointCleanNet* is used in [72] to remove outliers; it processes the whole PC with small local patches. The computational complexity is high, with about 20 minutes required to remove outliers on a PC of 0.8 million points with the help of a GPU.

For crack filling, we compare with [83], which claims to outperform its four competitors. The method in [83] divides the whole PC into thousands of overlapped cubes. For any cube that needs inpainting, they find the most similar cube among the remaining cubes and directly rotate and transform it to replace the cube with artifact. Visual results are in Fig. 4.18. While their method might work well for the artifacts in [83], it does not do well for the V-PCC crack artifacts, possibly because many cubes have cracks; the method in [83] might replace a bad cube with another cube that has an artifact, resulting in additional artifacts as shown in Fig. 4.18. As they directly take points from another cube, the points on the boundary of the cube could also add unexpected noise. The computational complexity is high compared with our proposed method because they need to iterate over all cubes to find the most similar one. Using a cube size of

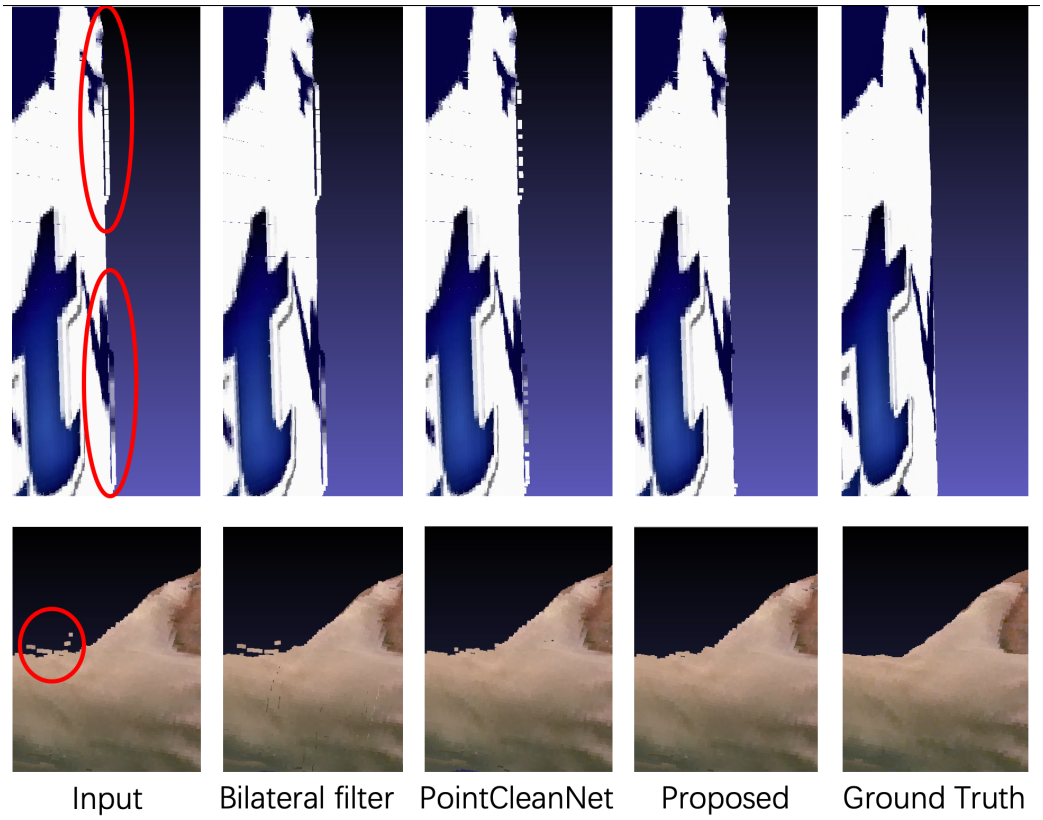

Figure 4.17: Outlier removal: Visual comparison with bilateral filter and PointCleanNet

$30 \times 30 \times 30$, each cube with holes needs 45 seconds to find its most similar cube, determine the rotation and transformation matrix, and replace the points in the original cube. With more than 1000 cubes that are considered to need fixing, the approach is considerably more complex than our proposed method.

Compared to other methods in the literature, the proposed approach performs well on point clouds which have been decompressed and reconstructed using V-PCC, by taking advantage of patch information from the compression to both locate and correct artifacts.

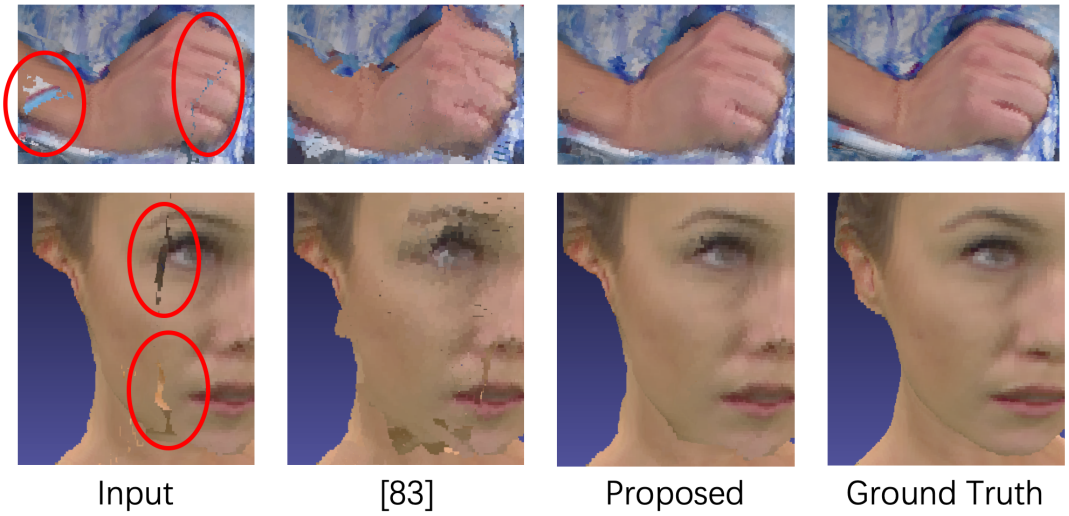

Figure 4.18: Crack filling: Visual comparison with the method in [83].

4.4 Conclusion

In this chapter, we categorize different types of artifacts that occur during the V-PCC pipeline. In particular, outliers and cracks are specific geometric artifacts which tend to occur at V-PCC patch edges and also at places where the patch projection direction is not close to the normal vector for a point. We propose specific methods to locate and remove outliers, and also to locate and fill in cracks. We conducted a subjective test to evaluate our proposed method. From the results of the subjective experiment, we conclude that our artifact removal algorithm

provides significant visual quality improvement for PCs that are reconstructed after V-PCC compression. In general, there is a trend that the artifact removal process tends to achieve more obvious improvement of visual quality with increasing QP. The improvements were more modest but still significant for cases where complex texture or fast object motion could mask the noise, and the improvements were strongest for cases where very smooth texture (such as human skin), slow motion, and simple shapes made the artifacts more obvious.

Chapter 4, in part, is a reprint of the material as it appears in the paper: K. Cao and P. Cosman, "Denoising and Inpainting for Point Clouds Compressed by V-PCC," submitted to *IEEE Transactions on Circuits and Systems for Video Technology*. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Conclusions and Future Work

Adding scaling modules to the original V-PCC framework improves compression flexibility. Similar to the situation for 2D video, downscaling and upscaling as part of the compression pipeline tends to produce better quality when the bit rate is constrained to be low. A patch-aware averaging filter is proposed to remove most outlier points caused by scaling. The averaging filter generates slightly lower MSE than the median filter and a weighted filter. The experimental results on 2D PSNR and 3D MSE show that our method performs well both on objective evaluation and on subjective visual quality. Among the four different interpolation choices, donwscaling the best performance is achieved in terms of MSE when nearest neighbor interpolation is used in the scaling modules at both the encoder and decoder.

Point cloud compression was shown to be better for low bit rates regardless of the observation distance, whereas mesh compression is preferred when the observation distance is close and the bit rate is not low. When the bit rate is high, there is little difference between the two representations.

For the two representations, the proposed two models that estimate people's opinion scores fit the experimental data well under cross-validation and the model can be used to choose a representation based on observation distance and target bit rate.

In addition, a general trend is found that reducing the number of mesh triangles or reducing the number of cloud points improved visual quality at low bit rates when compression. Such reductions are not routinely considered part of the compression pipeline for 3D content, but our finding fits the well-known result for 2D content that spatial down-sampling is useful at low bit rates (see, e.g., [64, 65]). Such reductions can play an important role in preserving quality at low bit rates for 3D content as well, and would be worthy of a further subjective study.

There remains considerable room for further study of subjective quality of PC and mesh compression. We chose the bit rate and the observation distance for a compressed point cloud because they are two very important factors which affect quality, and which also do not require any computation on the actual distorted sequence at the decoder. With increased complexity, there are many other factors which affect quality at a given bit rate and given observation distance, such as the color variation in the texture, the inherent geometric complexity of the sequence, or the rapidity of the motion. Prediction of head and eye movement based on past movement or based on saliency has been carried out for 360 degree video [92, 93], and such work could inform the compression approaches as well as the viewing trajectories for future subjective experiments. For future work, more factors could also also be considered, such as the sequence's complexity in geometry or texture, or the rapidity of motion, to make the prediction of quality more accurate. One limitation of this study is the limited number of sequences used as the test data set. A larger number of test sequences that can cover diverse visual content should also be involved in future work.

Different types of artifacts are categorized that occur during the V-PCC pipeline. In particular, specific geometric artifacts, outliers and cracks, tend to occur at V-PCC patch edges and also at places where the patch projection direction is not close to the normal vector for a point. Those artifacts can be located and removed by our proposed methods, and we showed that this provides significant visual quality improvement for PCs that are reconstructed after V-PCC compression.

There is a trend that the artifact removal process tends to achieve more obvious improvement of visual quality with increasing QP in general. The improvements were more modest but still significant for cases where complex texture or fast object motion could mask the noise, and the improvements were strongest for cases where very smooth texture (such as human skin), slow motion, and simple shapes made the artifacts more obvious.

While the hyper-parameters chosen worked robustly across a range of QP values and different contents, in future work, the algorithm parameters could be tailored to the QP and/or to the level of masking. For example, in cases of fast motion, complex texture, and/or complex geometry, the algorithm could ignore small cracks, or in cases of severe quantization, the algorithm could allow more loops of crack filling.

Bibliography

[1] R. Ma, T. Maugey, and P. Frossard, "Optimized data representation for interactive multiview navigation," *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1595–1609, 2018.

[2] O. Stankiewicz, M. Domański, A. Dziembowski, A. Grzelka, D. Mieloch, and J. Samelak, "A free-viewpoint television system for horizontal virtual navigation," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 2182–2195, 2018.

[3] P. Pourashraf and F. Safaei, "Perceptual pruning: A context-aware transcoder for immersive video conferencing systems," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1327–1338, 2017.

[4] C. Zhang, Q. Cai, P. A. Chou, Z. Zhang, and R. Martin-Brualla, "Viewport: A distributed, immersive teleconferencing system with infrared dot pattern," *IEEE MultiMedia*, vol. 20, no. 1, pp. 17–27, 2013.

[5] H. Sabirin, Q. Yao, K. Nonaka, H. Sankoh, and S. Naito, "Toward real-time delivery of immersive sports content," *IEEE MultiMedia*, vol. 25, no. 2, pp. 61–70, 2018.

[6] G. Makransky and L. Lilleholt, "A structural equation modeling investigation of the emotional value of immersive virtual reality in education," *Educational Technology Research and Development*, pp. 1–24, 2018.

[7] "5 sports that are benefiting from virtual reality," https://www.viar360.com/5-sports-benefiting-virtual-reality/.

[8] "Haptic feedback is making VR surgery feel like the real thing," https://www.theverge.com/2018/8/14/17670304/virtual-reality-surgery-training-haptic-feedback-fundamentalvr.

[9] R. Schnabel and R. Klein, "Octree-based point-cloud compression," *Symposium on Point-Based Graphics*, vol. 6, pp. 111–120, 2006.

[10] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 778–785.

[11] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-based static 3D point clouds geometry coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 284–299, 2019.

[12] Y. Fan, Y. Huang, and J. Peng, "Point cloud compression based on hierarchical point clustering," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*. IEEE, 2013, pp. 1–7.

[13] R. L. de Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, 2016.

[14] P. A. Chou and R. L. De Queiroz, "Motion-compensated compression of dynamic voxelized point clouds," Nov. 30 2017, uS Patent App. 15/168,019.

[15] J.-Y. Chen, C.-H. Lin, P.-C. Hsu, and C.-H. Chen, "Point cloud encoding for 3D building model retrieval," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 337–345, 2014.

[16] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, 2017.

[17] "Call for proposals for point cloud compression v2." ISO/IEC JTC1/SC29/WG11, Hobart, Australia, April, 2017.

[18] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[19] K. Mammou, A. M. Tourapis, D. Singer, and Y. Su, "Video-based and hierarchical approaches point cloud compression." ISO/IEC JTC1/SC29/WG11, Macau, China, October, 2017.

[20] K. Cao, Y. Xu, and P. Cosman, "Patch-aware averaging filter for scaling in point cloud compression," in *IEEE Global Conference on Signal and Information Processing*, 2018.

[21] G. Lavoué, "A multiscale metric for 3D mesh visual quality assessment," in *Computer Graphics Forum*, vol. 30, no. 5. Wiley Online Library, 2011, pp. 1427–1437.

[22] M. Corsini, M.-C. Larabi, G. Lavoué, O. Petřík, L. Váša, and K. Wang, "Perceptual metrics for static and dynamic triangle meshes," in *Computer Graphics Forum*, vol. 32, no. 1. Wiley Online Library, 2013, pp. 101–125.

[23] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2016.

[24] E. Alexiou and T. Ebrahimi, "On subjective and objective quality evaluation of point cloud geometry," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–3.

[25] E. Alexiou, T. Ebrahimi, M. V. Bernardo, M. Pereira, A. Pinheiro, L. A. D. S. Cruz, C. Duarte, L. G. Dmitrovic, E. Dumic, D. Matkovics *et al.*, "Point cloud subjective evaluation methodology based on 2D rendering," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2018, pp. 1–6.

[26] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "Subjective and objective quality evaluation of 3D point cloud denoising algorithms," in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 1–6.

[27] K. Mamou, T. Zaharia, and F. Prêteux, "TFAN: A low complexity 3D mesh compression algorithm," *Computer Animation and Virtual Worlds*, vol. 20, no. 2-3, pp. 343–354, 2009.

[28] J. Rossignac, "3D compression made simple: Edgebreaker with ZipandWrap on a corner-table," in *Shape Modeling and Applications, SMI 2001 International Conference on.* IEEE, 2001, pp. 278–283.

[29] R. Mekuria, M. Sanna, E. Izquierdo, D. C. Bulterman, and P. Cesar, "Enabling geometry-based 3-D tele-immersion with fast mesh compression and linear rateless coding," *IEEE Transactions on Multimedia*, vol. 16, no. 7, pp. 1809–1820, 2014.

[30] A. S. Lalos, I. Nikolas, E. Vlachos, and K. Moustakas, "Compressed sensing for efficient encoding of dense 3D meshes using model-based bayesian learning," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 41–53, 2017.

[31] E. Dumic, C. R. Duarte, and L. A. da Silva Cruz, "Subjective evaluation and objective measures for point clouds - State of the art," in *2018 First International Colloquium on Smart Grid Metrology (SmaGriMet)*. IEEE, 2018, pp. 1–5.

[32] J. Zhang, W. Huang, X. Zhu, and J.-N. Hwang, "A subjective quality evaluation for 3D point cloud models," in *2014 International Conference on Audio, Language and Image Processing*. IEEE, 2014, pp. 827–831.

[33] E. Alexiou, E. Upenik, and T. Ebrahimi, "Towards subjective quality assessment of point cloud imaging in augmented reality," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2017, pp. 1–6.

[34] E. Alexiou and T. Ebrahimi, "On the performance of metrics to predict quality in point cloud representations," in *Applications of Digital Image Processing XL*, vol. 10396. International Society for Optics and Photonics, 2017, p. 103961H.

[35] M. Seufert, J. Kargl, J. Schauer, A. Nüchter, and T. Hoßfeld, "Different points of view: Impact of 3D point cloud reduction on QoE of rendered images," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[36] E. Zerman, P. Gao, C. Ozcinar, and A. Smolic, "Subjective and objective quality assessment for volumetric video compression," *Electronic Imaging*, vol. 2019, no. 10, pp. 323–1, 2019.

[37] R. Schatz, G. Regal, S. Schwarz, S. Suettc, and M. Kempf, "Assessing the QoE impact of 3D rendering style in the context of VR-based training," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2018, pp. 1–6.

[38] K. Desai, S. Raghuraman, R. Jin, and B. Prabhakaran, "QoE studies on interactive 3D tele-immersion," in *2017 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2017, pp. 130–137.

[39] J. van der Hooft, M. T. Vega, C. Timmerer, A. C. Begen, F. De Turck, and R. Schatz, "Objective and subjective QoE evaluation for adaptive point cloud streaming," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[40] E. Zerman, C. Ozcinar, P. Gao, and A. Smolic, "Textured mesh vs coloured point cloud: a subjective study for volumetric video compression," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[41] S. Kanumuri, P. C. Cosman, A. R. Reibman, and V. A. Vaishampayan, "Modeling packet-loss visibility in MPEG-2 video," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 341–355, 2006.

[42] G. Lavoué and M. Corsini, "A comparison of perceptually-based metrics for objective evaluation of geometry processing," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 636–649, 2010.

[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[44] L. Dong, Y. Fang, W. Lin, and H. S. Seah, "Perceptual quality assessment for 3D triangle mesh based on curvature," *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2174–2184, 2015.

[45] M. Corsini, E. D. Gelasca, T. Ebrahimi, and M. Barni, "Watermarked 3-D mesh quality assessment," *IEEE Transactions on Multimedia*, vol. 9, no. 2, pp. 247–256, 2007.

[46] F. Torkhani, K. Wang, and J.-M. Chassery, "A curvature tensor distance for mesh visual quality assessment," in *International Conference on Computer Vision and Graphics*. Springer, 2012, pp. 253–263.

[47] I. Abouelaziz, M. Omari, M. El Hassouni, and H. Cherifi, "Reduced reference 3D mesh quality assessment based on statistical models," in *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2015, pp. 170–177.

[48] K. Wang, F. Torkhani, and A. Montanvert, "A fast roughness-based approach to the assessment of 3D mesh visual quality," *Computers & Graphics*, vol. 36, no. 7, pp. 808–818, 2012.

[49] I. Abouelaziz, M. El Hassouni, and H. Cherifi, "No-reference 3D mesh quality assessment based on dihedral angles model and support vector regression," in *International Conference on Image and Signal Processing*. Springer, 2016, pp. 369–377.

[50] I. Abouelaziz, M. El Hassouni, and H. Cherifi, "A curvature based method for blind mesh visual quality assessment using a general regression neural network," in *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2016, pp. 793–797.

[51] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A comprehensive study of the rate-distortion performance in MPEG point cloud compression," *APSIPA Transactions on Signal and Information Processing*, vol. 8, 2019.

[52] H. Su, Z. Duanmu, W. Liu, Q. Liu, and Z. Wang, "Perceptual quality assessment of 3D point clouds," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3182–3186.

[53] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3460–3464.

[54] E. Alexiou and T. Ebrahimi, "Point cloud quality assessment metric based on angular similarity," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[55] E. M. Torlig, E. Alexiou, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A novel methodology for quality assessment of voxelized point clouds," in *Applications of Digital Image Processing XLI*, vol. 10752. International Society for Optics and Photonics, 2018, p. 107520I.

[56] E. Alexiou and T. Ebrahimi, "Towards a point cloud structural similarity metric," in *2020 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, no. CONF, 2020.

[57] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–13, 2015.

[58] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 28, no. 5, pp. 1–10, 2009.

[59] R. I.-R. BT, "Methodology for the subjective assessment of the quality of television pictures," 2002.

[60] Y.-F. Ou, Z. Ma, T. Liu, and Y. Wang, "Perceptual quality assessment of video considering both frame rate and quantization artifacts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 286–298, 2011.

[61] Z. Wang and A. C. Bovik, "Modern image quality assessment," *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 2, no. 1, pp. 1–156, 2006.

[62] Q. Wu, H. Li, F. Meng, and K. N. Ngan, "A perceptually weighted rank correlation indicator for objective image quality assessment," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2499–2513, 2018.

[63] "VQMT: Video Quality Measurement Tool," https://mmspg.epfl.ch/vqmt.

[64] W. Lin and L. Dong, "Adaptive downsampling to improve image compression at low bit rates," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2513–2521, 2006.

[65] A. M. Bruckstein, M. Elad, and R. Kimmel, "Down-scaling for better transform compression," *IEEE Transactions on Image Processing*, vol. 12, no. 9, pp. 1132–1144, 2003.

[66] I.-K. Lee, "Curve reconstruction from unorganized points," *Computer aided geometric design*, vol. 17, no. 2, pp. 161–177, 2000.

[67] G. Rosman, A. Dubrovina, and R. Kimmel, "Patch-collaborative spectral point-cloud denoising," in *Computer Graphics Forum*, vol. 32, no. 8. Wiley Online Library, 2013, pp. 1–12.

[68] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "L1-sparse reconstruction of sharp point set surfaces," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 5, pp. 1–12, 2010.

[69] J. Digne and C. De Franchis, "The bilateral filter for point clouds," *Image Processing On Line*, vol. 7, pp. 278–287, 2017.

[70] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 386–402.

[71] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 52–60.

[72] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "PointCleanNet: Learning to denoise and remove outliers from dense point clouds," in *Computer Graphics Forum*, vol. 39, no. 1. Wiley Online Library, 2020, pp. 185–203.

[73] P. Chalmovianskỳ and B. Jüttler, "Filling holes in point clouds," in *Mathematics of Surfaces*. Springer, 2003, pp. 196–212.

[74] G. H. Bendels, R. Schnabel, and R. Klein, "Detecting holes in point set surfaces," *Journal of WSCG*, vol. 14, no. 1-3, 2006.

[75] H. Lin, J. Chen, Y. Zhang, W. Wang, and D. Kong, "Feature preserving filling of holes on point sampled surfaces based on tensor voting," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[76] V. S. Teggihalli, R. A. Tabib, A. Jamadandi, and U. Mudenagudi, "A polynomial surface fit algorithm for filling holes in point cloud data," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2019, pp. 515–522.

[77] C. Weber, S. Hahmann, and H. Hagen, "Sharp feature detection in point clouds," in *2010 Shape Modeling International Conference*. IEEE, 2010, pp. 175–186.

[78] X. Guo, J. Xiao, and Y. Wang, "A survey on algorithms of hole filling in 3D surface reconstruction," *The Visual Computer*, vol. 34, no. 1, pp. 93–103, 2018.

[79] M. Ju and M. Wang, "3D point cloud hole repair based on boundary rejection method," in *Proceedings of the Seventh International Symposium of Chinese CHI*, 2019, pp. 105–108.

[80] T.-T. Tran, V.-T. Cao, and D. Laurendeau, "Extraction of reliable primitives from unorganized point clouds," *3D Research*, vol. 6, no. 4, p. 44, 2015.

[81] J. Davis, S. R. Marschner, M. Garr, and M. Levoy, "Filling holes in complex surfaces using volumetric diffusion," in *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. IEEE, 2002, pp. 428–441.

[82] G. Casciola, D. Lazzaro, L. B. Montefusco, and S. Morigi, "Fast surface reconstruction and hole filling using positive definite radial basis functions," *Numerical Algorithms*, vol. 39, no. 1-3, pp. 289–305, 2005.

[83] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 4087–4100, 2019.

[84] Z. Fu, W. Hu, and Z. Guo, "3D dynamic point cloud inpainting via temporal consistency on graphs," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[85] P. Väänänen and V. Lehtola, "Inpainting occlusion holes in 3D built environment point clouds," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 393–398, 2019.

[86] X. Han, Z. Zhang, D. Du, M. Yang, J. Yu, P. Pan, X. Yang, L. Liu, Z. Xiong, and S. Cui, "Deep reinforcement learning of volume-guided progressive view inpainting for 3D point scene completion from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 234–243.

[87] S. Gai, F. Da, L. Zeng, and Y. Huang, "Research on a hole filling algorithm of a point cloud based on structure from motion," *JOSA A*, vol. 36, no. 2, pp. A39–A46, 2019.

[88] E. Vlachos, A. S. Lalos, A. Spathis-Papadiotis, and K. Moustakas, "Distributed consolidation of highly incomplete dynamic point clouds based on rank minimization," *IEEE Transactions on Multimedia*, vol. 20, no. 12, pp. 3276–3288, 2018.

[89] K. Cao, Y. Xu, and P. Cosman, "Visual quality of compressed mesh and point cloud sequences," *IEEE Access*, vol. 8, pp. 171 203–171 217, 2020.

[90] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[91] M. E. Glickman, "Parameter estimation in large dynamic paired comparison experiments," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 48, no. 3, pp. 377–394, 1999.

[92] Y. Zhu, G. Zhai, and X. Min, "The prediction of head and eye movement for 360 degree images," *Signal Processing: Image Communication*, vol. 69, pp. 15–25, 2018.

[93] Y. Zhu, G. Zhai, X. Min, and J. Zhou, "The prediction of saliency map for head and eye movements in 360 Degree Images," *IEEE Transactions on Multimedia*, 2019.