Composing Error Concealment Pipelines for Dynamic 3D Point Cloud Streaming

I-CHUN HUANG, National Tsing Hua University, Taiwan
YUANG SHI, National University of Singapore, Singapore
WEI TSANG OOI, National University of Singapore, Singapore
CHUN-YING HUANG, National Yang Ming Chiao Tung University, Taiwan
CHENG-HSIN HSU, National Tsing Hua University, Taiwan

Dynamic 3D point clouds enable the immersive user experience and thus have become increasingly more popular in volumetric video streaming applications. When being streamed over best-effort networks, point cloud frames may suffer from lost or late packets, leading to non-trivial quality degradation. To solve this problem, we proposed the very first error concealment pipeline framework, which comprises five stages: pre-processing, matching, motion estimation, prediction, and post-processing. Alternative algorithms can be developed for each stage, while algorithms of different stages could be mixed and matched into pipelines for end-to-end performance evaluations. We discussed the design goal and proposed multiple algorithms for each stage. These algorithms were then quantitatively compared using dynamic 3D point cloud sequences with diverse characteristics. Based on the comparison results, we proposed four representative pipelines for (i) diverse degrees of motion variance, i.e., minor versus significant, and (ii) different application requirements, i.e., high quality versus low overhead. Extensive end-to-end evaluations of our proposed pipelines demonstrated their superior concealed quality over the 3D frame-copy method in both: (i) 3D metrics, by up to 5.32 dB in GPSNR and 1.7 dB in CPSNR; as well as (ii) 2D metrics, by up to 2.22 dB in PSNR, 0.06 in SSIM, and 11.67 in VMAF. Adding to that, a user study with 15 subjects indicated that our best-performing pipeline achieved 100% preference winning rate over the state-of-the-art learning-based interpolation algorithms while consuming merely up to 8.55% of running time.

ACM Reference Format:
I-Chun Huang, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. 2024. Composing Error Concealment Pipelines for Dynamic 3D Point Cloud Streaming. 1, 1 (February 2024), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Immersive technologies have brought the user experience of multimedia applications to the next level in several business sectors, including entertainment, education, healthcare, and retail. A recent market report forecasts that the global immersive technology market will grow rapidly from 22.6 billion in 2021 to 138.5 billion USD in 2030 [36]. Immersive multimedia applications employ 3D content, which can be represented in various formats, including RGB-D videos, dynamic meshes, voxel sequences, and dynamic 3D point clouds. Among them, 3D point clouds can be natively

Authors' addresses: I-Chun Huang, National Tsing Hua University, Hsin-Chu, Taiwan; Yuang Shi, National University of Singapore, Singapore, Singapore; Wei Tsang Ooi, National University of Singapore, Singapore, Singapore; Chun-Ying Huang, National Yang Ming Chiao Tung University, Hsin-Chu, Taiwan; Cheng-Hsin Hsu, National Tsing Hua University, Hsin-Chu, Taiwan.

XXXX-XXXX/2024/2-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

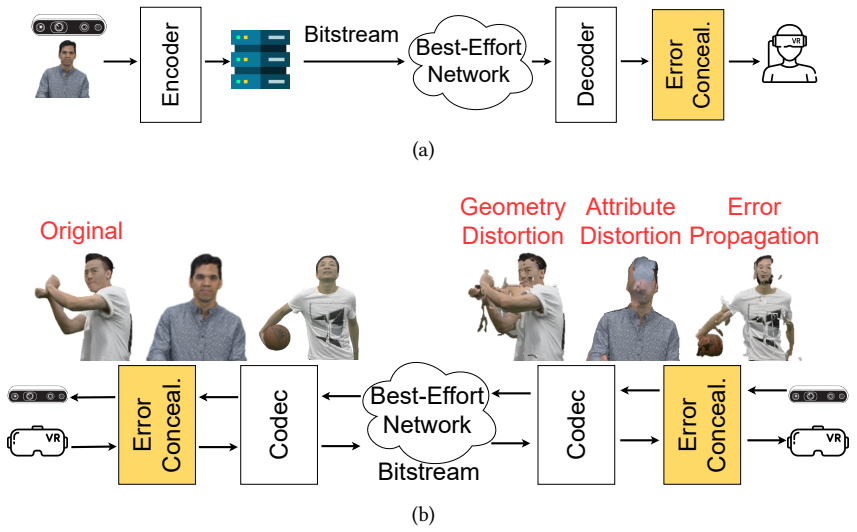


Fig. 1. Dynamic 3D point cloud streaming scenarios: (a) on-demand streaming, which is quality sensitive, and (b) teleconferencing, which is delay sensitive. Different types of distortion are also illustrated in (b).

captured by commodity sensors while preserving non-quantized *geometry* and *attribute* data. The geometry data are essentially coordinates of a set of unordered 3D points, while the attribute data include color, normal, reflectivity, etc. Hence, 3D point clouds are popular with multimedia applications [11] for (i) machines, such as autonomous driving and robotics, where accurate scene understanding is crucial, and (ii) humans, such as telecommunications, culture heritage, metaverses, tele-surgery, and remote sensing, where six degree-of-freedom interactions are critical.

In this article, we consider dense point clouds meant for human consumption. Particularly, Fig. 1 illustrates two typical dynamic 3D point cloud streaming scenarios: (i) *on-demand* with one-way streaming of pre-recorded content, where visual quality is important, and (ii) *teleconferencing* with two-way streaming of live-captured content, where low end-to-end delay is crucial. In both scenarios, streaming raw dynamic 3D point clouds leads to extremely high bandwidth requirements and thus is infeasible.

Point cloud codecs, such as the open-source Draco and standardized MPEG Geometry-based Point Cloud Compression (G-PCC) and Video-based Point Cloud Compression (V-PCC) [42], could help mitigate the pressure caused by high bandwidth requirements. Among them, V-PCC [20] projects 3D point cloud frames into multiple 2D images, which are, in turn, encoded with highly optimized, often commodity 2D video codecs. The coded bitstream is sent over networks before being decoded and projected into 3D point cloud frames. V-PCC has two strengths compared to other codecs. First, V-PCC has been shown to achieve high coding efficiency [9, 20]. Second, 2D video codecs have been massively produced, often in hardware, leading to cost- and energy-effective deployment of immersive multimedia applications. However, like other multimedia codecs, lost or late packets could result in distorted 3D point clouds at the receiver, which leads to a poor user experience. For example, a recent work [7] conducted subjective experiments under different packet loss rates when streaming using the V-PCC codec. Their results showed that the subjective quality could drop by more than 1 to 1.5 in a 5-point Mean Opinion Score (MOS) under merely a 0.5% packet loss rate, indicating an urgent need for error concealment methods in dynamic 3D

point cloud streaming. We carried out similar experiments and gave sample distorted 3D point clouds from the V-PCC reference software in Fig. 1(b), due to: (i) geometry, i.e., coordinate loss; (ii) attribute, i.e., color loss; and (iii) error propagation from already distorted frames. We observed that V-PCC is vulnerable to lost or late packets because it relies on the *2D Frame-Copy* (2DFC) method for error concealment. Because the geometry and attribute data are not aligned across 3D point cloud frames, V-PCC could produce catastrophically distorted 3D point clouds at the receiver. Similar observations were also made in the works of Wu et al. [43] and Hung et al. [17].

This article tackles the problem of distorted 3D point cloud frames caused by unsuccessful transmission over best-effort networks. Corruption in attribute data can be effectively concealed by copying the attributes of the nearest points in the preceding frame to the corrupted frame [17]. In contrast, geometry distortion is typically catastrophic and needs to be concealed by rebuilding the missing frame from scratch [43]. This task, unfortunately, can not be properly accomplished by existing hole-filling [8, 14, 27, 28, 38] or point-cloud completion methods [4, 41, 44], which rely on intra-frame operations and do not work when large portions of decoded frames are gone. Concealing geometry distortion by interpolating between two received point cloud frames was first proposed by Wu et al. [43]. Multiple state-of-the-art neural-based interpolation algorithms [2, 37, 49] for point clouds have also been proposed, but they are computationally expensive and often support a fixed, small, number of points. Therefore, it is unclear how they perform in different dynamic 3D point cloud streaming scenarios (see Fig. 1) in typical best-effort networks.

Concealing catastrophically distorted point cloud frames is no easy task for several reasons. One key challenge is matching unsorted points across frames to estimate point motions, which at first glance may look similar to 3D pose estimation [39], stereo matching [12, 23], or optical/scene flow estimation [18] problems. Whether algorithms designed for those problems can be augmented for matching points of 3D point cloud frames remains largely unknown because: (i) dynamic point clouds often involve non-rigid transforms, and (ii) point cloud frames consist of diverse numbers of points. For example, Ingale and Udayan [18] reported that existing optical/scene flow estimation algorithms are mostly tailored for non-rigid motions, which suffer from large motions that could be caused by consecutive frame drops.

To the best of our knowledge, this article is the first end-to-end investigation on error concealment of dynamic 3D point cloud streaming, while our initial results were presented in a conference version [17]. This article makes the following contributions:

- We propose a general *multi-stage* pipeline framework for error concealment of 3D point cloud streaming in Sec. 2. The framework can incorporate alternative algorithms in each stage.
- We develop and quantitatively compare a suite of algorithms for individual stages in Secs. 3–6. The algorithms in different stages can be mixed and matched into diverse error concealment pipelines for diverse usage scenarios.
- We construct multiple representative pipelines following the insights gained in per-stage comparison and extensively evaluate these pipelines through end-to-end experiments in Sec. 7.

The end-to-end evaluation results depict the advantages of our proposed pipelines tailored for dynamic 3D point cloud streaming: (i) with diverse degrees of motion variance, i.e., from minor to significant, and (ii) under different application requirements, i.e., high-quality or low overhead. More concretely, in objective experiments, our best-performing pipeline outperforms the *3D Frame-Copy* (3DFC) method by up to (on average): (i) 5.32 dB (3.01 dB) in Geometry Peak Signal-to-Noise Ratio (GPSNR), (ii) 2.22 dB (1.28 dB) in Peak Signal-to-Noise Ratio (PSNR), and (iii) 11.67 (5.59) in Video Multi-Method Assessment Fusion (VMAF) [1] among seven dynamic 3D point cloud sequences from the MPEG dataset [6]. In subjective tests, our best-performing pipeline achieves a 100% winning

rate on the overall video preference over the state-of-the-art learning-based algorithms [2, 49]. Such improved concealment quality could attract and retain users of immersive multimedia applications. More importantly, as we made our implementation publicly available [15], our proposed pipeline framework can facilitate and stimulate future innovation on the algorithms in one or multiple stages to further optimize error concealment of dynamic 3D point cloud streaming in different scenarios.

2 ERROR CONCEALMENT PIPELINE

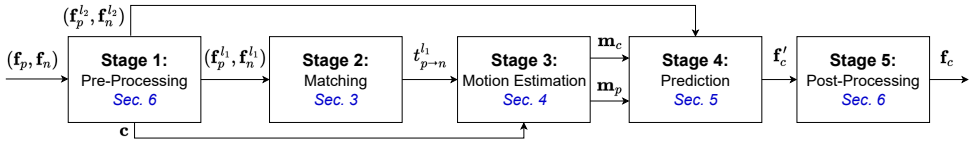


Fig. 2. The proposed error concealment pipeline framework for dynamic 3D point cloud streaming.

2.1 Overview

Fig. 2 illustrates the framework of our error concealment pipelines. Each pipeline generates an error-concealed point cloud frame $\mathbf{f}_c$ using the preceding frame $\mathbf{f}_p$ and the next frame $\mathbf{f}_n$, where each frame is composed of a set of points. We let $\mathbf{f}.i$ be the frame index of $\mathbf{f}$, and $\mathbf{f}_c.r = (\mathbf{f}_c.i - \mathbf{f}_p.i)/(\mathbf{f}_n.i - \mathbf{f}_p.i)$ be the relative *position* of $\mathbf{f}_c$ within the consecutive lost or late frames. To control distortion, we started concealing $\mathbf{f}_c$ from the closer reference frame (either $\mathbf{f}_p$ or $\mathbf{f}_n$), which we refer to as the *anchor* frame, while the other frame is called the *current* frame. Specifically, $\mathbf{f}_p$ is the anchor frame iff $\mathbf{f}_c.r \leq 0.5$. Without loss of generality, we assume $\mathbf{f}_p$ is the anchor frame in our following discussion.

Our pipeline framework consists of five stages: (1) pre-processing, (2) matching, (3) motion estimation, (4) prediction, and (5) post-processing. The *pre-processing* stage downsamples point clouds to reduce the computational overhead of the following stages. Specifically, it voxelizes $\mathbf{f}_p$ and $\mathbf{f}_n$ with voxel lengths l_1 and l_2 (where $l_1 \geq l_2$), resulting in two pairs of downsampled point clouds: $(\mathbf{f}_p^{l_1}, \mathbf{f}_n^{l_1})$ and $(\mathbf{f}_p^{l_2}, \mathbf{f}_n^{l_2})$. The coarser-grained $(\mathbf{f}_p^{l_1}, \mathbf{f}_n^{l_1})$ is utilized in the *matching* and *motion estimation* stages, which are computationally demanding. In contrast, the finer-grained $(\mathbf{f}_p^{l_2}, \mathbf{f}_n^{l_2})$ is adopted in the *prediction* stage to produce a better quality $\mathbf{f}_c'$. The *pre-processing* stage also computes a minimal 3D cube of a side length of h to bound all points of $\mathbf{f}_p$ and $\mathbf{f}_n$. It assigns all points in $\mathbf{f}_p$ to a set of *cubes* $\mathbf{c}$ with a side length of l_c. The purpose of $\mathbf{c}$ is to (i) reduce the workload of the computationally-intensive *motion estimation* stage, and (ii) avoid inconsistent motion vectors among nearby points. The *matching* stage creates a matching table $t_{p\to n}^{l_1}$ from $\mathbf{f}_p^{l_1}$ to $\mathbf{f}_n^{l_1}$, which records the point-to-point correspondence. The *motion estimation* stage generates motion vectors for either individual points or cubes using $t_{p\to n}^{l_1}$. The point-based motion vectors are denoted as $\mathbf{m}_p$, while the cube-based motion vectors are denoted as $\mathbf{m}_c$. The *prediction* stage produces an interpolated point cloud $\mathbf{f}_c'$ using $\mathbf{m}_p$ or $\mathbf{m}_c$ on $(\mathbf{f}_p^{l_2}, \mathbf{f}_n^{l_2})$. The *post-processing* stage applies refinement algorithms on $\mathbf{f}_c'$ to get $\mathbf{f}_c$. Our pipeline framework is general since (i) each stage can be instantiated by alternative algorithms, and (ii) individual stages are optional and can be bypassed.

2.2 Design Objectives

The eventual goal of error concealment algorithms is generating $\mathbf{f}_c$ with high visual quality, which can be quantified in 3D and 2D quality metrics [42]. Our considered 3D metrics include (i) *GPSNR*,

which is the PSNR of the Chamfer distance, i.e., the average distance of pair-wise closest points of two frames; (ii) *Hausdorff distance*, which is the maximal shortest distance between the closest points of two frames; and (iii) *CPSNR (color PSNR)*, which is the PSNR of the luminance distortion between the closest points of two frames. Our considered 2D metrics include (i) the average *PSNR* of the foreground object in the point cloud sequence; (ii) the average Structural SIMilarity [40] (*SSIM*) of the foreground object in the point cloud sequence; and (iii) *VMAF*, a data-driven quality metric from Netflix [1] of the rendered point cloud sequence. Zerman et al. [50] reported that most VR users view 3D avatars on their horizontal planes. Hence, we render eight inward 2D images at 45° intervals with Open3D [51] and report the average 2D metrics across all eight images. In this article, we consider two dynamic 3D point cloud datasets: public data [34] and an MPEG dataset [6]. For consistency, we normalize the range of the point coordinates of the sequences in Sun et al. [34] to that of the MPEG sequences [6]. We employ a rendering point size of 8 and 3 in these two datasets, respectively, as they have diverse point densities.

As point matching is an essential and challenging stage[1] in the pipeline, it is crucial to evaluate the accuracy of a given point-matching algorithm. To the best of our knowledge, the quality of point matching across frames has never been investigated in the literature. Hence, we propose two metrics for measuring the *temporal* and *spatial smoothness* of a matching table $t_{p\to n}$. Let $t^*_{p\to n}$ denote the ground-truth matching table. For any point $p_i \in \mathbf{f}_p$, we write the point-wise spatial smoothness based on coordinate distance as:

$$d_g(t_{p\to n}, t^*_{p\to n}, p_i) = \Delta_{cor}\big(t_{p\to n}(p_i), t^*_{p\to n}(p_i)\big), \tag{1}$$

where $\Delta_{cor}(p, q)$ represents the 3D Euclidean distance of p and qs' coordinates. We also write the point-wise temporal smoothness on the angle between two motion vectors as:

$$d_r(t_{p\to n}, t^*_{p\to n}, p_i) = \cos^{-1}\frac{\overrightarrow{p_i t_{p\to n}(p_i)} \cdot \overrightarrow{p_i t^*_{p\to n}(p_i)}}{\left|\overrightarrow{p_i t_{p\to n}(p_i)}\right|\left|\overrightarrow{p_i t^*_{p\to n}(p_i)}\right|}, \tag{2}$$

where $\overrightarrow{pq}$ represents the motion vector from p to qs' 3D coordinates. Next, we generalize these two smoothness metrics to the frame level as:

$$d_g(t_{p\to n}) = \Sigma_{p_i\in\mathbf{f}_p} d_g(t_{p\to n}, t^*_{p\to n}, p_i)/|\mathbf{f}_p|; \tag{3}$$

$$d_r(t_{p\to n}) = \Sigma_{p_i\in\mathbf{f}_p} d_r(t_{p\to n}, t^*_{p\to n}, p_i)/|\mathbf{f}_p|. \tag{4}$$

Smaller $d_g(\cdot)$ and $d_r(\cdot)$ values lead to higher spatial and temporal smoothness, respectively.

3 MATCHING STAGE

3.1 Motivation

The goal of the matching stage is to compute the matching table $t^{l_1}_{p\to n}$ from the anchor ($\mathbf{f}^{l_1}_p$) to current ($\mathbf{f}^{l_1}_n$) frames. Although the quality of the matching table $t^{l_1}_{p\to n}$ strongly affects the following stages, finding high-quality $t^{l_1}_{p\to n}$ remains very challenging for real-world dynamic point cloud sequences because frames are composed of different numbers of unordered points, and transforms are often non-rigid. As an illustrative example, we implement the **Nearest-Neighbor (NN)** algorithm to find the matching table: $t^{l_1}_{p\to n} = \left\{ \text{argmin}_{q\in\mathbf{f}^{l_1}_n} \Delta_{cor}(p, q) \middle| p \in \mathbf{f}^{l_1}_p \right\}$. We use the sequences with ground truth matched points in Sun et al. [34] in our experiments. Fig. 3 visualizes sample results from two frames of a dancing woman, in which matched points are connected by lines. Fig. 3(b) reveals that NN matches some points of the avatar's fingertips to those of her palms and some points of her

[1]This is largely due to the non-rigid transforms [18].

(a) (b)

Fig. 3. Point matching results between overlaid $(\mathbf{f}_p, \mathbf{f}_n)$: (a) ground truth and (b) NN matching. Only 2% of matched points are shown as samples for clarity.

left leg/foot to those of her left thigh. These mismatched points would lead to significantly wrong motion vectors (both in direction and magnitude) in later stages and result in catastrophically corrupted $\mathbf{f}_c$. Hence, better matching algorithms are needed.

3.2 Proposed Algorithms

We propose two efficient point-matching algorithms below.

Query-Radius (QR). The algorithm searches for the best matching point within a given *search radius* τ to control the computational complexity. Particularly, for each point $p \in \mathbf{f}_p^{I_1}$, we define $\mathbf{s}(p,\tau) = \left\{q \in \mathbf{f}_n^{I_1} \middle| \Delta_{cor}(p,q) < \tau\right\}$ as the candidate point set. Next, we find the most similar point q^* from $\mathbf{s}(p,\tau)$ by: $q^* = \text{argmin}_{q \in \mathbf{s}(p,\tau)} \Delta(p,q)$, with a utility function:

$$\Delta(p,q) = \alpha\Delta_{cor}(p,q) + (1-\alpha)\Delta_{rgb}(p,q) + \beta t(q). \tag{5}$$

Here, $\Delta_{rgb}(p,q)$ represents the Euclidean distance of the RGB vectors of points p and q. We also write per-channel color distance as $\Delta_r(p,q)$, $\Delta_g(p,q)$, and $\Delta_b(p,q)$. In addition, $t(q)$ represents q's previous matching times, which can be interpreted as a penalty term to encourage 1-1 matching. α, β are parameters to tune the relative importance among the three terms of $\Delta(p,q)$. Last, for any $p \in \mathbf{f}_p^{I_1}$ with $\mathbf{s}(p,\tau) = \emptyset$, we match p to $\emptyset$ in the resulting $t_{p\to n}^{I_1}$.

Adaptive QR (AQR). This algorithm extends QR by adapting the τ value for each point. Larger τ leads to longer running time and higher chances of incorrect matches (like from left to right shoe), while smaller τ leads to a higher chance of missing the most suitable matches. The AQR algorithm is developed with two design rationales derived from some pilot experiments. First, color similarity carries a non-trivial weight on point matching between the anchor and current frames (in the temporal domain). Hence, we cluster the points in the current frame based on their colors. Second, (spatially) close-by points should be treated similarly when searching for matching points (and motion vectors). Hence, we cluster the points in the anchor frame based on their coordinates. The detail of the AQR algorithm is given below.

The AQR algorithm first clusters all points of $\mathbf{f}_n^{I_1}$ by their colors in two passes. The first pass scans through $\mathbf{f}_n^{I_1}$ and incrementally builds a set of point clusters $\mathbf{u}_n$. More specifically, for each $p \in \mathbf{f}_n^{I_1}$, we add $\{p\}$ to $\mathbf{u}_n$ iff $\max_{u_j \in \mathbf{u}_n}\left\{\min_{q_j \in u_j} \Delta_r(p,q_j), \min_{q_j \in u_j} \Delta_g(p,q_j), \min_{q_j \in u_j} \Delta_b(p,q_j)\right\} \geq 32$, where 32 was empirically selected. After the first pass, we get a set of clusters, where each cluster

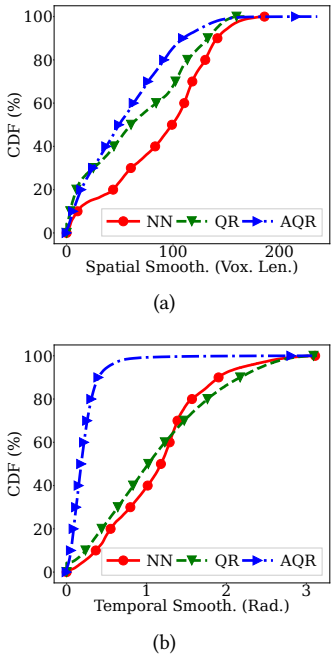


Fig. 4. Comparison of matching algorithms of a sample frame with 4 consecutive frame drops: (a) spatial and (b) temporal smoothness.

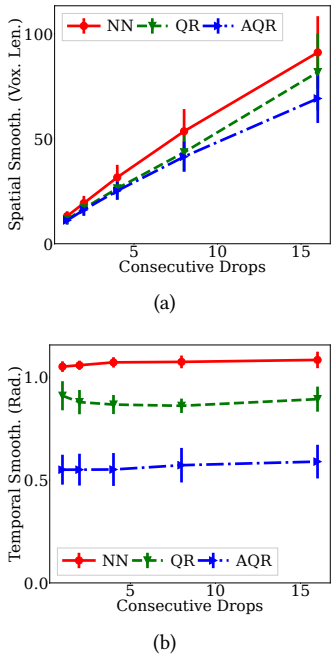


Fig. 5. Comparison of matching algorithms with diverse numbers of frame drops: (a) spatial and (b) temporal smoothness.

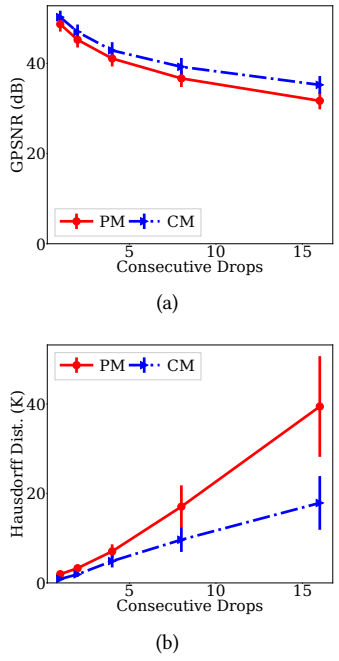


Fig. 6. 3D quality comparison of motion estimation algorithms: (a) GPSNR and (b) Hausdorff distance.

contains a single point. The second pass scans through $f_n^{l_1}$ again and assigns each point to the cluster with the highest RGB color similarity with individual clusters' points. The assignment leads to the final *color-based clusters* $\mathbf{u}_n$. The AQR algorithm also clusters all points of $\mathbf{f}_p^{l_1}$ by their coordinates. More specifically, it voxel-downsamples $\mathbf{f}_p^{l_1}$ into the *coordinate-based clusters* $\mathbf{u}_p$ using a voxel length of $h/10$. Points in $\mathbf{f}_p^{l_1}$ are associated with the closest clusters in $\mathbf{u}_p$.

With $\mathbf{u}_n$ and $\mathbf{u}_p$ in hand, we next find the best τ value for each point $p_i \in f_p^{l_1}$. We locate the color-based cluster $u_j \in \mathbf{u}_n$ with the smallest $\Delta_{rgb}(p_i, u_j)$. Here, we consider the mean RGB color of all points of u_j. Then, we perform an NN search within u_j to get $q_i = \text{argmin}_{q \in u_j} \Delta_{cor}(p_i, q)$. Upon iterating through all $p_i \in f_p^{l_1}$, we compute the τ_i^* for all points in the coordinate-based cluster $u_k \in \mathbf{n}_p$ by:

$$\tau_k = \gamma \max_{p_i \in u_k} \Delta_{cor}(p_i, q_i), \tag{6}$$

where the scaling factor $\gamma \in \mathbb{R}^+$ is a system parameter. Notice that we go with the largest search radius to accommodate the point with the largest motion vector.

After getting the best search radius τ_i for each $p_i \in \mathbf{f}_p^{l_1}$, we invoke the QR algorithm with τ_i to get the matching table. One last detail is, before doing that, we sort points $p_i \in \mathbf{f}_p^{l_1}$ by their $\Delta_{cor}(p_i, q_i)$. By doing so, the AQR algorithm matches the points with smaller motion vectors (such as the torso of an avatar) earlier. Because these points get larger penalty $t(q_i)$ sooner, they are less likely to be matched with points having larger motion vectors (such as the limbs of an avatar).

AQR can be performed with either forward ($t^{l_1}_{p\to n}$) or backward ($t^{l_1}_{n\to p}$) point matching. We found that by performing the AQR algorithm in the direction with a larger average magnitude of matching tables as computed by the NN algorithm, we can significantly improve the smoothness. For example, with a dancing man sequence [34], up to 5.90% and 34.02% reductions on $d_g(\cdot)$ and $d_r(\cdot)$ were observed. Hence, we swapped $\mathbf{f}_p$ and $\mathbf{f}_n$ before running the AQR algorithm whenever backward point matching was more promising.

3.3 Comparison

We compared NN, QR, and AQR using a dancing man sequence [34] under different numbers of consecutive frame drops in {1, 2, 4, 8, 16}. We ran QR on all sample frames and chose the smallest τ to ensure that 95+% of points p in every frame had nonempty $\mathbf{s}(p, \tau)$. We sampled an $\mathbf{f}_p$ frame every 10 frames, leading to 50 frames in total. The voxel length l_1 was set to one[2]. The following system parameters were empirically chosen: $\alpha = 0.9$, $\beta = 0.1$, and $\gamma = 1.5$.

Fig. 4 reports the Cumulative Distribution Function (CDF) of spatial smoothness and temporal smoothness from a sample frame with four consecutive drops. We observe that AQR can pick a matching point closer to the ground truth than QR and NN. In this sample frame, AQR achieves 80% point matching accuracy with errors of less than 90 voxel length in d_g and 0.3 radians in d_r. On the other hand, QR and NN have lower accuracy with errors of 111 and 128 voxel length in d_g and 1.76 and 1.56 radians in d_r, respectively.

Fig. 5 gives the average smoothness results with 95% confidence intervals[3]. We observe that as the number of consecutive frame drops increases, the spatial smoothness worsens, and the gap between AQR and NN enlarges. In contrast, the temporal smoothness remains stable under different numbers of consecutive frame drops. Overall, AQR outperforms NN by at least 16% and 47% in spatial and temporal smoothness, respectively. QR's performance lies between NN and AQR, and highly depends on the selection of τ.

NN, QR, and AQR build k-d trees for searching in space. The dominating time complexity of k-d tree searches on $\mathbf{f}$ is $O(|\mathbf{f}| \log |\mathbf{f}|)$ on average. As τ increases, the time complexity approaches $O(|\mathbf{f}|^2)$ in the worst case. In our experiments, although NN runs fast, it produces matching tables with inferior smoothness levels. QR improves the smoothness over NN. However, proper τ values must be manually found in a trial-and-error fashion. AQR achieves the best smoothness levels but takes longer to compute τ values than QR.

4 MOTION ESTIMATION STAGE

4.1 Motivation and Proposed Algorithms

The motion estimation stage calculates the resulting motion vectors from the anchor ($\mathbf{f}^{l_1}_p$) to current ($\mathbf{f}^{l_1}_n$) frames. The motion vectors are either point-based $\mathbf{m}_p$ or cube-based $\mathbf{m}_c$. We propose two motion estimation algorithms in the following.

Point Motion (PM) generates point-based motion vectors by:

$$\mathbf{m}_p = \left\{ \overrightarrow{p_i t_{p\to n}(p_i)} \middle| p_i \in \mathbf{f}^{l_1}_p \right\}. \tag{7}$$

Cube Motion (CM) generates cube-based motion vectors by:

$$\mathbf{m}_c = \left\{ \Sigma_{p_i \in c_j} \overrightarrow{p_i t_{p\to n}(p_i)} / |c_j| \middle| c_j \in \mathbf{c} \right\}. \tag{8}$$

[2]In this article, we assume coordinates are integers. Therefore, downsampling with a voxel length of 1 means no downsampling.

[3]Throughout this article, we plot 95% confidence intervals as error bars, if not otherwise specified.

We note that the CM algorithm aims to mitigate the point-matching *outliers* in matching tables, whose motion vectors are quite different from those of the nearby points. With that said, the PM algorithm still has a chance to be more accurate if their smoothness levels are small.

4.2 Comparison

We take the matching table produced by NN in Sec. 3 as input to validate whether CM mitigates incorrect point matching. More precisely, we use the anchor frame ($\mathbf{f}_p$) and motion vectors ($\mathbf{m}_p$ or $\mathbf{m}_c$) to estimate the current frame $\mathbf{f}'_n$. We then compare it with the ground-truth current frame $\mathbf{f}_n$ to quantify the 3D point cloud quality in the GPSNR and Hausdorff distance. The system parameter $l_c = 128$ is empirically chosen.

Fig. 6 gives the average 3D point cloud quality under different numbers of consecutive frame drops with a dancing man sequence [34]. This figure reveals that CM improves the 3D quality by 3.54 dB in GPSNR at most and 54.67% in Hausdorff distance on average. This demonstrates that CM could mitigate the errors in point matching. Both PM and CM need to iterate through all the points in $\mathbf{f}$ and query the corresponding matching in $t_{p\to n}$, which leads to a time complexity of $O(|\mathbf{f}|)$.

5 PREDICTION STAGE

5.1 Motivation

The prediction stage aims to generate $\mathbf{f}'_c$ in good visual quality using the anchor frame and motion vectors. Doing so with prediction algorithms is no easy task because both spatial and temporal smoothness must be maintained. It has been reported that ill-designed prediction algorithms often lead to severe *cracks* [17], which dramatically degrade the visual quality and reduce the quality of experience. Prediction algorithms are critical to mitigate these cracks.

5.2 Proposed Algorithms

For the sake of presentation, we first generalize $\mathbf{m}_c$ into functions returning the corresponding motion vectors, i.e.: (i)$\mathbf{m}_c(c_j)$ for any cube $c_j \in \mathbf{c}$ and (ii) $\mathbf{m}_c(p_i) = \mathbf{m}_c(c_j)$ for any point $p_i \in \mathbf{f}_p^{l_2}$, where p_i falls in $c_j \in \mathbf{c}$. Similarly, we generalize $\mathbf{m}_p$ into a function $\mathbf{m}_p(p_i)$ for any $p_i \in \mathbf{f}_p^{l_2}$, which returns the motion vector of the closest voxel in the coarser-grained $\mathbf{f}_p^{l_1}$. We propose three prediction algorithms below.

Point-based Prediction (PP). It generates $\mathbf{f}'_c$ from $\mathbf{m}_p$ by:

$$\mathbf{f}'_c = \{p_i + \mathbf{f}_c.r \times \mathbf{m}_p(p_i) | p_i \in \mathbf{f}_p^{l_2}\}. \tag{9}$$

Cube-based Prediction (CP). The set of cubes $\mathbf{c}$ (created in the pre-processing stage) consists of mutually disjoint cubes with a side length l_c. In our pilot tests, we noticed that after applying $\mathbf{m}_c$ to $\mathbf{c}$ for $\mathbf{f}'_c$, the *predicted cubes*:

$$\mathbf{c}_c = \{c_j + \mathbf{f}_c.r \times \mathbf{m}_c(c_j) | c_j \in \mathbf{c}\} \tag{10}$$

are no longer mutually disjoint. Because of that, adjacent cubes with larger *gaps* often result in more severe cracks. Hence, we developed an adaptation approach to *enlarge* the side lengths of selected cubes to eliminate these gaps as follows.

We let $\mathbf{c}'_c = \mathbf{c}_c$ be all cubes that need to be inspected. We sort all $c_j \in \mathbf{c}_c$ on their magnitude $|\mathbf{m}_c(c_j)|$ in descending order to fill up larger gaps earlier. Next, we iterate through all $c_j \in \mathbf{c}_c$, and compute the cube center distance between c_j and all adjacent cubes $c_k \in \mathbf{c}'_c$. Here, *adjacent* cubes c_k are the six cubes that share a face with c_j. To find the six adjacent cubes of c_j, we utilize l_c and the coordinates in $\mathbf{c}$ (before applying $\mathbf{m}_c$), which can be done in constant time. Among the six cube center distances after applying $\mathbf{m}_c$, we find the largest one, say l_m. If $l_m > l_c$, we enlarge c_j's side

length to be l_m. Next, we remove c_j from $\mathbf{c}'_c$ and move to the next $c_j \in \mathbf{c}_c$. After $|\mathbf{c}_c|$ iterations, we have $|\mathbf{c}'_c| = 0$ and a $\mathbf{c}_c$ with *customized* cube side lengths leading to minimal gaps. Using the updated $\mathbf{c}_c$, we recompute the cube motion $\mathbf{m}_c$ using Eq. (8). Last, with the updated $\mathbf{m}_c$, we generate $\mathbf{f}'_c$ by:

$$\mathbf{f}'_c = \{p_i + \mathbf{f}_c.r \times \mathbf{m}_c(c_j) \big| p_i \in \mathbf{f}_p^{l_2}\}. \tag{11}$$

Neighboring-cube-based Prediction (NP). Although CP minimizes the gaps between adjacent cubes, our pilot tests revealed that CP may produce irregular surfaces due to sudden jumps in the side lengths among adjacent cubes. Hence, we propose a prediction algorithm that takes the motion vectors of all *neighboring* cubes into consideration below. First, for any cube $c_j \in \mathbf{c}$, we define neighboring cubes as those non-c_j cubes $c_k \in \mathbf{c}$ that share at least one vertex with c_j. We collectively refer to all neighboring cubes of c_j as $\mathbf{n}_j$, where $|\mathbf{n}_j| = 26$. When predicting a point $p_i \in \mathbf{f}_p^{l_2}$ that falls in $c_j \in \mathbf{c}$, we compute a weighted sum of all motion vectors of $\mathbf{n}_j \cup \{c_j\}$. We define a weight for p_i and $c_k \in \mathbf{n}_j \cup \{c_j\}$ as:

$$e_{i,k} = 1/\Delta_v(p_i, c_k), \tag{12}$$

where $\Delta_v(p_i, c_k)$ denotes the volume of a cuboid with p_i and the center of c_k as two opposite vertices. It is not hard to see that closer cubes have higher weights. With the above notations, NP generates $\mathbf{f}'_c$ by:

$$\mathbf{f}'_c = \left\{ p_i + \mathbf{f}_c.r \times \frac{\sum_{c_k \in \mathbf{n}_j \cup \{c_j\}} e_{i,k} \mathbf{m}_c(c_k)}{\sum_{c_k \in \mathbf{n}_j \cup \{c_j\}} e_{i,k}} \Big| p_i \in \mathbf{f}_p^{l_2} \right\}. \tag{13}$$

5.3 Comparison

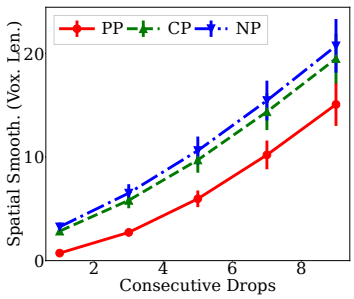
To compare PP, CP, and NP, we let $l_1, l_2 = 1$. We conceal the most challenging middle frame in different numbers of consecutive frame drops in {1, 3, 5, 7, 9}. We predict $\mathbf{f}'_c$ by running PP, CP, and NP on ground-truth motion vectors. Other settings are consistent with those used in Secs. 3 and 4.

(a) (b) (c)

Fig. 7. Sample 2D-rendered views from different prediction algorithms: (a) PP, (b) CP, and (c) NP.

Fig. 7 presents sample 2D-rendered views of a dancing man sequence [34]. Among the three algorithms, we observe that CP (Fig. 7(b)) moves points in each cube toward a single direction, leading to irregular surfaces on the avatar's body and arms, while NP (Fig. 7(c)) mitigates this issue. Fig. 8 gives overall spatial smoothness $d_g(\cdot)$ under different numbers of consecutive frame drops. We observe that the difference among different prediction algorithms is at most 6 voxel length, which is negligible (~0.6%) compared to the height of the avatar. Similarly, the temporal smoothness difference is also small (figures not shown for brevity). Fig. 9 reports the 2D quality of rendered views, which demonstrates that (i) PP outperforms CP by 1.25 dB in PSNR and 0.06 in

SSIM, and (ii) CP outperforms NP by 0.26 dB in PSNR and 0.01 in SSIM. Note that the observed performance difference is with ground-truth (perfect) motion vectors, which could be different in end-to-end pipelines.



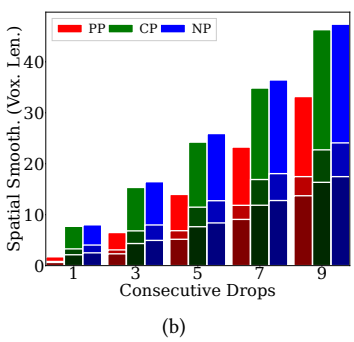
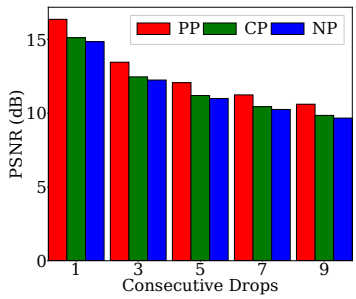


Fig. 8. Smoothness comparison of prediction algorithms: (a) average and (b) 95, 75, 50 percentiles.



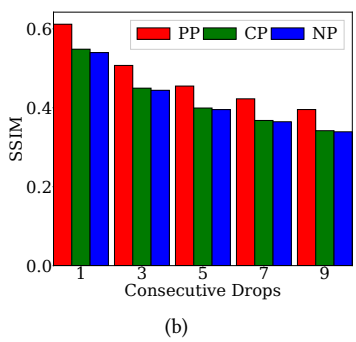


Fig. 9. 2D quality comparison of prediction algorithms: (a) PSNR and (b) SSIM.

Fig. 10. Visual results of sample point clouds: (a) without and (b) with PSR.

PP iterates through all points in Eq. (9) and has a time complexity of $O(|\mathbf{f}|)$. As for CP, the motion vectors $\mathbf{m}_c$ are firstly applied to the cubes $\mathbf{c}$, whose time complexity is $O(|\mathbf{c}|)$. Subsequently, the cubes in $\mathbf{c}$ are sorted according to their motion quantified by $|\mathbf{m}_c|$, which takes $O(|\mathbf{c}| \log |\mathbf{c}|)$. For each cube $c_j \in \mathbf{c}$, we compute l_m from its six adjacent cubes in $O(1)$ using their coordinates and l_c, so the complexity for all cubes is $O(|\mathbf{c}|)$. The final step recalculates the motions of the updated cubes and applies them to points in $\mathbf{f}$, whose time complexity is $O(|\mathbf{f}|)$. Thus, the overall time complexity of CP is $O(|\mathbf{c}| \log |\mathbf{c}| + |\mathbf{f}|)$. For NP, initially, 26 neighbors are identified for each cube, yielding a time complexity of $O(|\mathbf{c}|)$. Then, for every point in $\mathbf{f}_p^{l_2}$, the calculation of Eq. (13) takes a time complexity of $O(|\mathbf{f}|)$. Hence, the overall time complexity of NP is $O(|\mathbf{c}| + |\mathbf{f}|)$. The complexity of our three prediction algorithms can be sorted in descending order as CP, NP, and PP. In actual experiments, NP is, however, slower than CP because $|\mathbf{f}|$ is typically much larger than $|\mathbf{c}|$.

6 PRE- AND POST-PROCESSING STAGES

6.1 Motivation and Algorithms

To mitigate the high complexity caused by too many points, the pre-processing stage downsamples point clouds using l_1 and l_2. If $l_1, l_2 \neq 1$, $|\mathbf{f}'_c| < |\mathbf{f}_p|, |\mathbf{f}_n|$. We use voxel downsampling as it takes

advantage of the discrete structure and the regular division of 3D space, endowing the unorganized point cloud with richer spatial knowledge and correlation among filled voxels [46].

The post-processing stage upsamples $\mathbf{f}'_c$ to acquire a similar point number (resolution) $\mathbf{f}_c$ and applies *surface refinement* algorithms to improve spatial smoothness by repairing the cracks and artifacts. Our earlier work [33] revealed that a simple yet effective surface reconstruction method, i.e., Screened Poisson Surface Reconstruction (PSR) [22], can be adopted for this task. PSR is based on the observation that the oriented point samples can be regarded as samples of the gradient of the 3D model's indicator function, which takes normal vectors of points in $\mathbf{f}'_p$ as input. Other simpler surface refinement algorithms (e.g., bilinear interpolation and moving least squares [24]) can be applied as well in the proposed framework. These algorithms, however, cannot handle point clouds that are distributed in a non-uniform manner or with noise/outliers, which are inevitable in a practical setup. On the contrary, PSR considers all the points at once, without considering heuristic spatial partitioning or blending, making it highly resilient to noise [13].

6.2 Comparison

To understand the PSR's potential, we compared the concealed frame quality without and with it on the first 200 frames of MPEG Red&Blk sequence [6] with 3 consecutive frame drops. We employed AQR, CM, and NB in the middle stages for experiments with normal estimates using Open3D [51] on $\mathbf{f}'_c$. Our experiments depicted that PSR improves PSNR, SSIM, and VMAF by 0.59 dB, 1.42%, and 0.96 on average. To visualize the difference, Fig. 10 provides an illustration of the rendered point clouds without and with PSR, demonstrating its effectiveness on crack filling. PSR consumes considerable computational resources, even in situations where the normals are pre-computed and given. Alleviating this overhead is our future work.

To evaluate the tradeoff between 2D quality and running time, we tested different downsample ratios $s \in \{0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ by adjusting l_2 so that $s \times |\mathbf{f}_p| = |\mathbf{f}_p^{l_2}|$. We used PSR for surface refinement. We picked 20 random frames from Red&Blk [6] and evaluated them for 10 rounds to acquire the average quality and running time. Fig. 11 shows the running time and quality under different s. We also added a baseline without downsampling and PSR for comparison. First, as we found, PSR with $s > 0.3$ achieves better quality than the baseline. This observation demonstrates the feasibility of leveraging the spatial redundancy to reduce the overhead of spatial refinement by downsampling. Moreover, we observed that the VMAF of the point cloud first increases as the running time increases and reaches its peak at $s = 0.7$. This interesting trend suggests that downsampling with a proper ratio can remove outliers of points, thus improving the quality of concealed point clouds.

7 EXPERIMENTS

7.1 Our Proposed Representative Pipelines

Table 1. Alternative Algorithms in Individual Stages

Algorithm \ Stage	Pre-Proc.	Matching	M. Est.	Pred.	Post-Proc.
Adopted	Downs. [46]	NN (baseline)	-	-	Ups. + PSR [22]
Proposed	-	QP, AQR	PM, CM	PP, CP, NP	-

We have implemented error concealment framework and algorithms in C++ and Python using libraries including PCL [32] and Eigen [10]. Our implementation [15] includes the alternative algorithms proposed in Secs. 3–6, as summarized in Table 1. In addition, we concatenate the

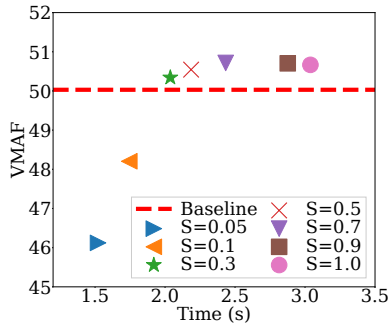


Fig. 11. Tradeoff between quality and running time of the post-processing stage with PSR as the surface refinement algorithm.

proposed algorithms in different stages to form representative error concealment pipelines for diverse usage scenarios, as listed in Table 2. More details are given below.

- **Fast (F)**: We combined QR with PP for minimum time complexity. We opted not to use an even simpler NN because of its inferior point matching quality. On the other hand, AQR is too heavy but brings marginal quality improvement compared to QR with a well-selected τ value.
- **Balance (B)**: We combined NN with CM for a good tradeoff between quality and time complexity. While NN is the fastest matching algorithm, it may produce noisy matching tables. Therefore, we employed CM to mitigate the noise.
- **Quality (Q)**: We combined AQR with NP for the highest quality. We employed downsampling voxel lengths $(l_1, 1)$ to speed up AQR by reducing its problem size. However, AQR occasionally matches points that are too distant, resulting in incorrect motion vectors. NP was therefore selected to cope with mismatched points.
- **Quality+ (Q+)**: Q+ is built upon Q, but uses the more expensive PSR for surface refinement. Hence, we adopted downsampling voxel lengths (l_1, l_2) to speed up both AQR and NP.

Table 2. Our Representative Error Concealment Pipelines

Pipeline \ Stage	Pre-Proc.	Matching	M. Est.	Pred.	Post-Proc.
Fast (F)	-	QR	PM	PP	-
Balance (B)	-	NN	CM	CP	-
Quality (Q)	Downs.(l_1, 1)	AQR	CM	NP	-
Quality+ (Q+)	Downs.(l_1, l_2)	AQR	CM	NP	Ups. + PSR

Table 3. Dynamic 3D Point Cloud Sequences

Property \ Sequence	Queen	Loot	Red&Blk	Soldier	LongDress	Basketball	Dancer
Cplx.	Low	Low	Low	Low	Medium	High	High
Pt. # (M)	1.00	0.78	0.70	1.50	0.80	2.90	2.60
Coor.	[0, 1023]	[0, 1023]	[0, 1023]	[0, 1023]	[0, 1023]	[0, 2047]	[0, 2047]

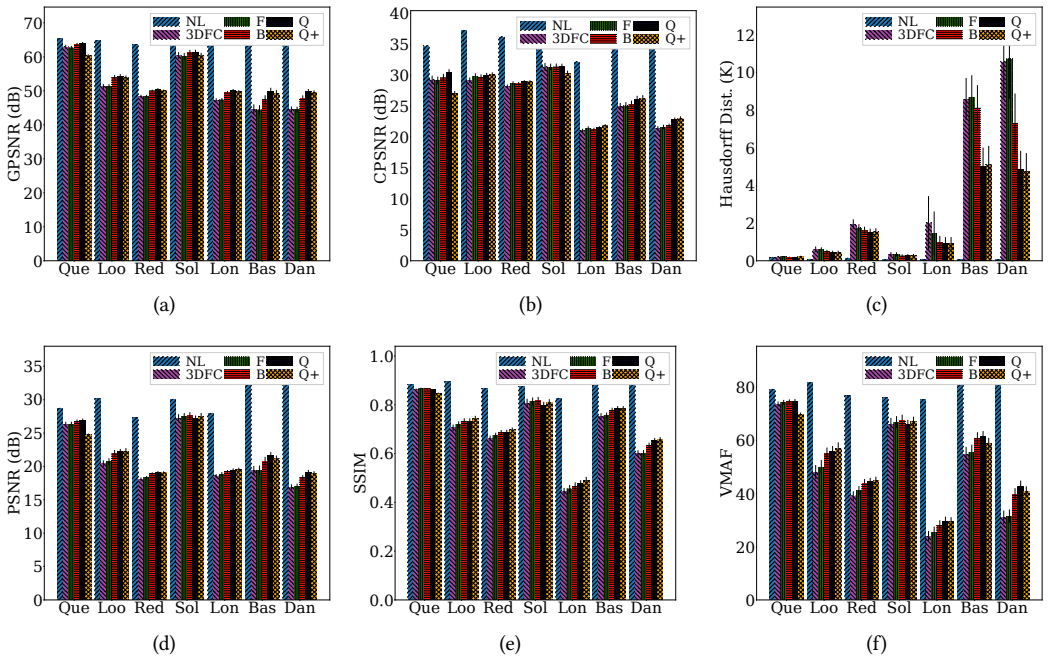


Fig. 12. Overall quality of concealed frames from individual sequences with a single frame drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

7.2 Setup

We have found that the simplistic 2DFC suffers from significant quality degradation as briefly illustrated in Fig. 1(b) and detailed in the preliminary conference version [17] of this article. For brevity, in this article, we employed 3DFC as the baseline. 3DFC duplicates the anchor frame as the concealed frame. While 3DFC is lightweight, it could lead to jerky playback as the same frame may be duplicated multiple (specifically, when $\mathbf{f}_n.i - \mathbf{f}_p.i - 1 > 1$) times. We also report No Loss (NL) as an unrealistic benchmark in some tests. Table 3 lists seven MPEG sequences [6] in the ascending complexity levels. Each sequence contains a human avatar, which is a representative 3D object for point cloud streaming. We consider the first 200 point cloud frames in each sequence in our experiments. We evaluated the performance of the four representative pipelines using V-PCC reference software [30] as the codec with the default settings defined in Common Test Conditions (CTC) [29]. We adopted the more error-resilient All-Intra mode of V-PCC at 30 frames-per-second (fps).

To evaluate the representative pipelines under different network conditions, we vary the number of frame drops from 1 to 5. The following parameters were heuristically decided through a grid search: (i) $l_1 = 15, l_2 = 1.9$, and $l_c = 128$ in the pre-processing stage and (ii) $\tau = 2$, $\alpha = 0.9, \beta = 0.1$, and $\gamma = 1.5$ in the matching stage. We ran our algorithms on an AMD Ryzen 7 5800X CPU at 3.8 GHz. We report the average results across all concealed frames in the rest of this section.

7.3 Results

Quality comparison over different sequences. We first discuss the results from our pipelines F, B, and Q, which do not employ surface refinement algorithms. Fig. 12 plots the overall quality of

individual sequences under a single frame drop. We observe that B and Q consistently outperform 3DFC in terms of all the metrics. However, F trails 3DFC with some sequences. Moreover, Q constantly outperforms B, while B constantly outperforms F across all sequences. This confirms that more complicated matching and prediction algorithms lead to better 2D and 3D quality.

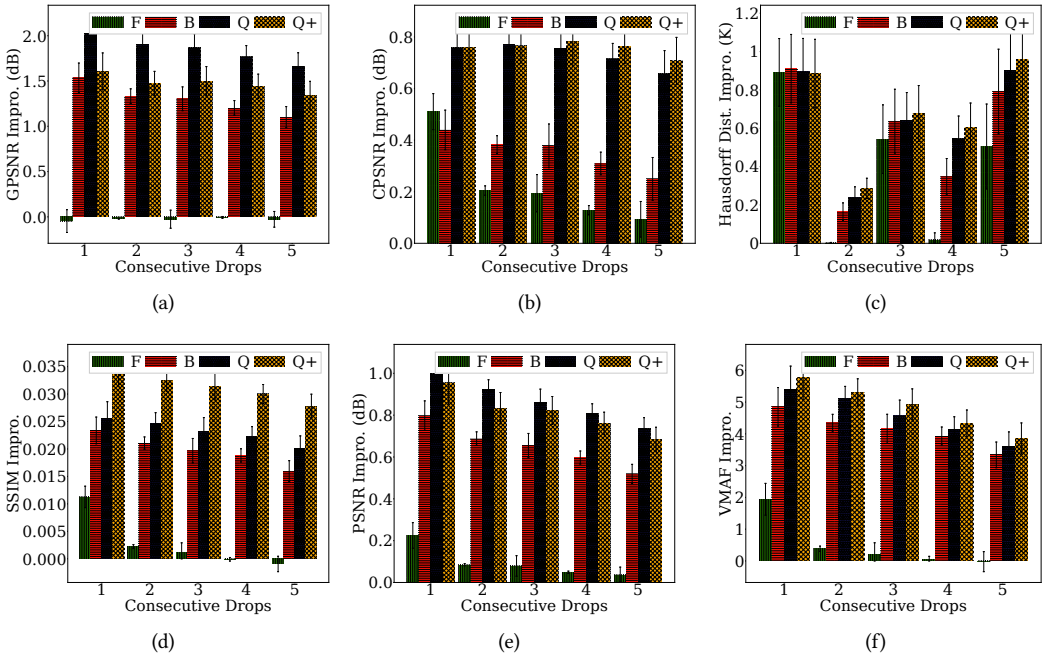


Fig. 13. Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Red&Blk are shown.

Quality comparison under different numbers of consecutive frame drops. We next zoom into two sample sequences to study the implications of different numbers of consecutive drops. Figs. 13 and 14 give the overall quality improvement of our reference pipelines over 3DFC with (low-complexity) Red&Blk and (high-complexity) Dancer, respectively. With Red&Blk, Q outperforms 3DFC by at most 2.03 dB in GPSNR, 0.76 dB in CPSNR, 0.025 in SSIM, and 5.40 in VMAF across all numbers of consecutive frame drops. With Dancer, Q outperforms 3DFC by at most 5.32 dB in GPSNR, 6.15 K in Hausdorff distance, 2.20 dB in PSNR, and 11.68 in VMAF across all numbers of consecutive frame drops. We also observe that the gaps decrease when more consecutive frames are dropped. This can be attributed to the larger errors in the matching and prediction stages. Take the fast-moving Dancer as an example; compared to single frame drops, dropping 5 frames degrades Q's quality improvement by 3.56 dB in GPSNR, 3.16 K in Hausdorff distance, 1.56 dB in PSNR, and 9.86 in VMAF. Designing error concealment pipelines that are more resilient to longer consecutive frame drops is among our future tasks.

Implications of the surface refinement algorithm. We next discuss the results from the PSR-enhanced pipeline Q+, which are also reported in Figs. 12–14. Fig. 12 shows that excluding Queen, Q+ delivers similar quality compared to Q: the maximal gaps across the other six sequences are 0.55 dB in GPSNR, 0.10 K in Hausdorff distance, 0.44 dB in PSNR, and 2.52 in VMAF. A closer

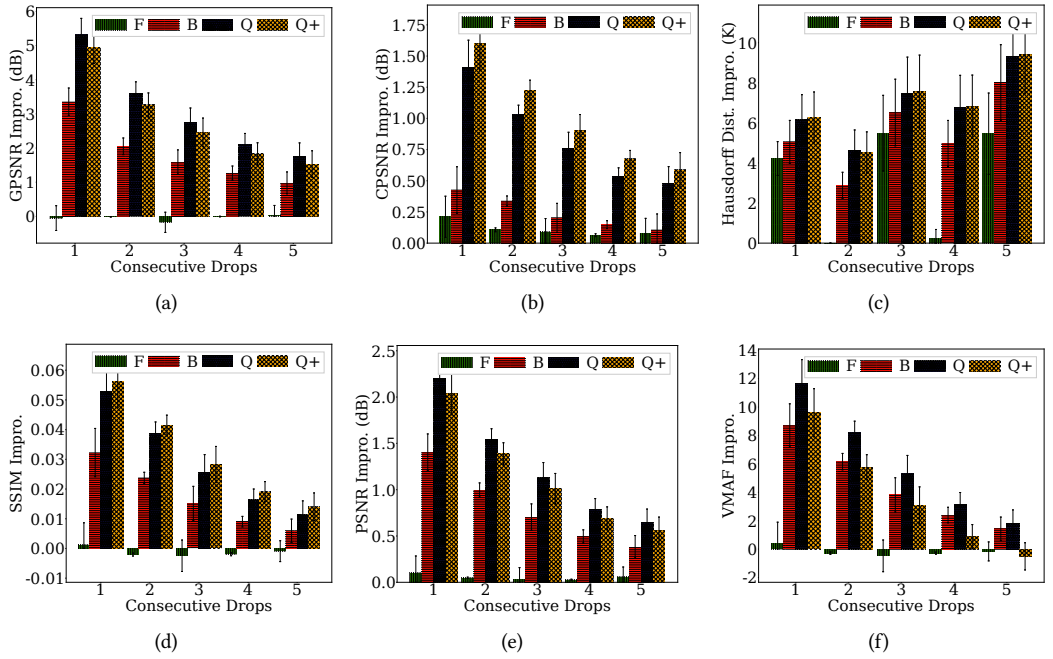


Fig. 14. Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Dancer are shown.

look at why Q+ performs badly with Queen reveals that it is largely caused by imperfect normal estimation done by Open3D [51]. For example, the normals of the tablet held by the avatar are clearly ill-estimated (by Open3D), which significantly affects the performance of PSR. Furthermore, Figs. 13 and 14 reveal that pipelines Q+ and Q achieve similar quality improvement over 3DFC for different numbers of consecutive frame drops. Take Red&Blk as an example; the gaps between Q+ and Q are: (i) [-0.42, -0.32] dB in GPSNR, (ii) [0.00, 0.05] dB in CPSNR, (iii) 0.01 in SSIM, and (iv) [0.20, 0.36] in VMAF. Take Dancer as another example; the gaps between Q+ and Q are: (i) [-0.37, -0.23] dB in GPSNR, (ii) [-0.08, 0.13] K in Hausdorff distance, (iii) [-0.16, -0.09] dB in PSNR, and (iv) [-2.43, -2.08] in VMAF. We conclude that pipeline Q+ achieves comparable concealed quality to Q.

Per-frame running time of different pipelines. We selected 24 random frames from Loot, LongDress, and Dancer to measure the average per-stage running time under different sequence complexity levels. All the evaluations use a single CPU core without parallelization. Moreover, for pipeline Q+, we assume that normals are precomputed. Table 4 reports the per-frame average running time. We observe that F is consistently faster than B, while B is consistently faster than Q. On the other hand, Q+ is faster than Q, which can be attributed to the downsampling settings. Our proposed pipeline F, B, and Q+ could work for on-demand streaming, in which larger buffers (say a few seconds) are possible, while teleconferencing dictates further optimizing the running time in various ways. For example, parallelization techniques, including but not limited to Single Instruction Multiple Data (SIMD) and General-Purpose Graphics Processing Unit (GPGPU), can be adopted. Moreover, it was reported that GPGPU-based surface reconstruction, such as PSR, could improve the running time by at least two orders of magnitude [51]. Table 4 also gives the fraction of running times consumed by individual stages, shedding some light on the strategies for optimizing

Table 4. Per-frame Running Time Break-down from Loot/LongDress/Dancer: Second (Fraction)

Pipeline \ Stage	Pre-Proc.	Matching	M. Est.	Pred.	Post-Proc.	Total
		QR	**PM**	**PP**		
Fast (F)	-	1.05/0.96/2.66 (93%/93%/93%)	0.02/0.02/0.03 (2%/2%/1%)	0.06/0.06/0.19 (5%/5%/6%)	-	1.13/1.03/2.88
		NN	**CM**	**CP**		
Balance (B)	-	0.99/1.34/14.64 (25%/30%/54%)	1.13/1.11/4.52 (27%/25%/17%)	1.98/1.98/7.93 (48%/45%/29%)	-	4.11/4.44/27.08
	Downs.(l_1, 1)	**AQR**	**CM**	**NP**		
Quality (Q)	0.04/0.01/0.15 (<1%/<1%/<1%)	1.94/2.17/18.81 (18%/18%/33%)	0.01/0.01/0.03 (<1%/<1%/<1%)	8.69/9.49/36.10 (81%/81%/66%)	-	10.67/11.71/55.09
	Downs.(l_2, 1)	**AQR**	**CM**	**NP**	**Ups. + PSR**	
Quality+ (Q+)	0.11/0.11/0.36 (2%/2%/2%)	1.77/1.89/9.53 (36%/36%/43%)	0.01/0.01/0.03 (<1%/<1%/<1%)	2.80/3.07/11.48 (58%/58%/52%)	0.17/0.18/0.52 (3%/3%/2%)	4.86/5.27/21.92

individual pipelines. For instance, in our proposed pipeline Q, NP accounts for up to 81% of the running time. Fortunately, as points are calculated independently in NP, it is not hard to parallelize NP for a shorter running time.

8 USER STUDY

8.1 Design

We carried out a user study with 15 subjects (4 female) with a mean age of 23.83 years old. We tested each subject for visual acuity and color blindness. None of them were removed as a result. We rendered two 2D videos of 200 dynamic point cloud frames, in which corrupted frames were concealed by a pair of: (i) one of our proposed representative pipelines and (ii) one of the baseline algorithms. The rendered 2D videos were displayed side-by-side in a random order on a 23" monitor at 1920X1080 and 60 fps following the BT.500-13 standard [19].

We considered two tests with different baseline algorithms: (i) 3DFC and (ii) state-of-the-art learning-based interpolation algorithms [2, 49][4][5]. More precisely, for the 3DFC test, we consider a more challenging setup with five consecutive frame drops to evaluate our proposed representative pipelines: {F, B, Q, Q+}. For the learning-based test, because the baseline algorithms [2, 49] are rather slow and only work for single frame drops, we only consider the promising pipeline Q with single frame drops.

For each video pair, we asked subjects to select the better video in terms of three aspects:

- *Spatial smoothness*, which accounts for artifacts like cracks, irregular surfaces, and blurred edges.
- *Temporal smoothness*, which covers varying fps and stalls.
- *Preference*, which represents the overall quality.

We report *winning rates* of individual questions, which are the fractions of votes on our proposed representative pipelines.

For each subject, we first explained the definitions of the three subjective questions. We then held a dry-run session with a training-only sequence to familiarize them with the test procedure and user interface. The actual testing video pairs were then played to the subject. We looped each

[4]We only consider the state-of-the-art algorithms with official implementations in the public domain to avoid glitches in third-party realizations.

[5]Because Zeng et al. [49] only take point clouds with 1024 points as input, we divide each point cloud frame into multiple sub point clouds with $\leq$ 1024 points before sending them into the neural network.

pair of videos, and subjects took ~20 seconds before providing their answers. In total, it took each subject ~18 minutes to complete his/her user study.

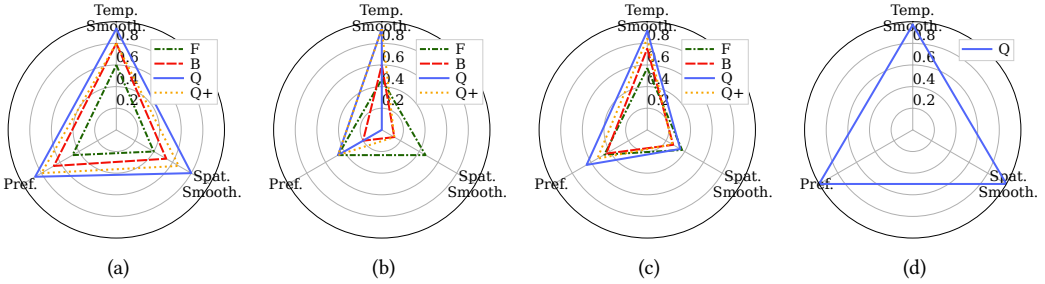


Fig. 15. Winning rates of three subjective questions: (a)–(c) from the 3DFC test and (d) from the learning-based test. Sample results from (a) best-performing Red&Blk and (b) worst-performing Basketball. Average results across all sequences, compared to (c) 3DFC and (d) state-of-the-art learning-based interpolation algorithms.

8.2 Results

Fig. 15 presents the sample winning rates: 3DFC test results from the best- and worst-performing sequences in Figs. 15(a) and 15(b), respectively; average 3DFC results in Fig. 15(c); and overall results against the state-of-the-art learning-based interpolation algorithms in Fig. 15(d). Note that in the second test, we compare our proposed pipeline Q against both state-of-the-art learning-based interpolation algorithms, but we did not analyze the results from these two interpolation algorithms separately. We make the following observations. First, *our representative pipelines result in better temporal smoothness.* Fig. 15(c) shows that compared to 3DFC, our Q, B, and F pipelines achieve average winning rates of 91.43%, 75.24%, and 57.14%, respectively. Fig. 15(b) reveals that even with the worst-performing sequence, Q and B still result in 50+% winning rates. Fig. 15(d) demonstrates that our pipeline Q achieves a 97.14% average winning rate over the two state-of-the-art learning-based interpolation algorithms [2, 49].

Second, *our representative pipelines lead to better spatial smoothness compared to the state-of-the-art learning-based interpolation algorithms.* Fig. 15(d) depicts that our pipeline Q delivers a 100% average winning rate over the state-of-the-art learning-based interpolation algorithms. Compared to 3DFC, Fig. 15(a) reveals that the B, Q, and Q+ pipelines achieve as high as 80% winning rate. However, Fig. 15(c) shows that, when considering all seven sequences, none of our pipelines achieve 50+% average winning rates. This is understandable as 3DFC never incurs spatial artifacts, even though it could suffer from jerky playouts.

Third, *our representative pipeline Q achieves better preference.* Fig. 15(c) demonstrates that only pipeline Q leads to a 64.76% average winning rate over 3DFC. We interviewed the subjects and found that inferior spatial smoothness is more noticeable than inferior temporal smoothness. Even if spatial artifacts, such as cracks, only appear in a single point cloud frame, they are easily perceived and remembered by subjects. This gives 3DFC an edge over our reference pipelines. For visual inspection, Fig. 16(a) gives a sample concealed frames of the worst-performing Basketball sequence from our pipeline Q under five consecutive frame drops. This figure shows noticeable artifacts such as the squeezed basketball and displaced arms, explaining why subjects may prefer 3DFC over our pipeline Q.

(a) (b)

Fig. 16. Sample error concealed 3D point cloud frames using: (a) our proposed representative pipeline with five consecutive frame drops and (b) Akhtar et al. [2] with a single frame drop.

Last, our representative pipeline Q subjectively outperforms the state-of-the-art learning-based interpolation algorithms. Fig. 16(b) shows a sample frame concealed from a state-of-the-art learning-based interpolation algorithm under a single frame drop. Compared to Fig. 16(a), it is not hard to see why subjects prefer our pipeline Q. In fact, Fig. 15(d) confirms that our pipeline Q leads to a 100% winning rate over the state-of-the-art learning-based interpolation algorithms. We take a closer look and quantify the severity of cracks by counting the number of *see-through pixels* in 2D-rendered images. A pixel of a rendered image is defined as see-through iff either it or its corresponding pixel in the ground-truth image is in the background color (but not both). In addition, pixels within 10 pixels to the avatar boundary are not considered as see-through pixels. Table 5 compares the mean (maximum) see-through pixels. Across the seven sequences, we found that Akhtar et al. [2] and Zeng et al. [49] suffer from averagely 1.83–12.00 and 1.68–18.00 times of see-through pixels than our Q pipeline, respectively. Such huge gaps explain why the state-of-the-art learning-based interpolation algorithms [2, 49] were not preferred by subjects. In addition, Akhtar et al. [2] took 11.70–15.66 times running time compared to Q, while Zeng et al. [49] took 102.22–182.20 times. *Hence, the state-of-the-art learning-based interpolation algorithm [2, 49] cannot effectively solve the error concealment problem.*

Table 5. Number of Continuous See-through Pixels: Mean (Max)

Algorithm \ Sequence	Queen	Loot	Red&Blk	Soldier	LongDress	Basketball	Dancer
Proposed Pipeline Q	1 (167)	15 (576)	53 (793)	4 (335)	38 (937)	161 (2123)	85 (1495)
Akhtar et al. [2]	12 (1204)	119 (3708)	97 (1301)	43 (2317)	190 (2703)	548 (3870)	326 (3165)
Zeng et al. [49]	18 (764)	55 (1118)	89 (1417)	56 (1010)	79 (1086)	1325 (29097)	286 (5321)

In summary, our best-quality pipeline Q outperforms 3DFC in temporal smoothness (91.43% average winning rate) and preference (64.76%); it also outperforms the state-of-the-art learning-based interpolation algorithms [2, 49] in temporal smoothness (97.14%), spatial smoothness (100%), and preference (100%), as well as running time ($< 1/11.70 = 8.55\%$).

9 CONCLUDING REMARKS

In this article, we proposed the very first pipeline framework for concealing lost or late frames in dynamic 3D point cloud streaming. Our multi-stage framework is general, as a stage can be skipped, and every stage can be realized by alternative algorithms. We proposed, developed, implemented, and evaluated multiple algorithms for individual stages. Based on the per-stage evaluation results, we proposed four representative pipelines for diverse streaming scenarios. Extensive objective experiments and subjective tests revealed the merits of our proposed representative pipelines. In particular, we: (i) improved at most 5.32 dB in GPSNR, 2.22 dB PSNR, and 11.67 in VMAF compared to 3DFC, (ii) achieved better temporal smoothness than 3DFC, and (iii) achieved a 100% winning rate on preference over the state-of-the-art learning-based interpolation algorithms.

Table 6. Recommended Error Concealment Pipelines

Requirement \ Motion Variance	Minor	Medium	Significant
High Quality	Q	Q	3DFC
Low Overhead	F	Q+	3DFC

Our proposed reference pipelines have different pros and cons. Table 6 gives our recommendations. We classify the usage scenarios into two dimensions: requirement and motion variance between the anchor and current frames. When high quality is the main requirement, we recommend our pipeline Q for the best possible quality as long as motion variance is minor or medium. When low overhead is the main requirement, we recommend our efficient pipeline F under minor motion variance, and more comprehensive Q+ under medium motion variance. Last, with significant motion variance, our representative pipelines may cause occasional spatial artifacts, which could degrade the objective and subjective visual quality. Hence, we suggest using 3DFC for such challenging scenarios. We believe there is room for innovation for high motion scenarios, and we welcome researchers to experiment with existing or to-be-developed algorithms using our framework.

REFERENCES

[1] Anne Aaron, Zhi Li, Megha Manohara, Joe Yuchieh Lin, Eddy Chi-Hao Wu, and C.-C Jay Kuo. 2015. Challenges In Cloud Based Ingest And Encoding For High Quality Streaming Media. In *Proc. of the IEEE International Conference on Image Processing (ICIP'15)*. Quebec City, Canada, 1732–1736.

[2] Anique Akhtar, Zhu Li, Geert Van der Auwera, and Jianle Chen. 2022. Dynamic Point Cloud Interpolation. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'22)*. Singapore, 2574–2578.

[3] Toby Breckon and Robert Fisher. 2008. Three-Dimensional Surface Relief Completion Via Nonparametric Techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 12 (2008), 2249–2255.

[4] Jingdao Chen, John Yi, Mark Kahoush, Erin Cho, and Yong Cho. 2020. Point Cloud Scene Completion Of Obstructed Building Facades With Generative Adversarial Inpainting. *Sensors* 20, 18 (2020), 5029:1–5029:27.

[5] J. Davis, S.R. Marschner, M. Garr, and M. Levoy. 2002. Filling Holes In Complex Surfaces Using Volumetric Diffusion. In *Proc. of the First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*. Padova, Italy, 428–441.

[6] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip Chou. 2017. 8i Voxelized Full Bodies–A Voxelized Point Cloud Dataset. http://plenodb.jpeg.org/pc/8ilabs/.

[7] Emil Dumic and Luis da Silva Cruz. 2023. Subjective Quality Assessment Of V-PCC-Compressed Dynamic Point Clouds Degraded By Packet Losses. *Sensors* 23, 12 (2023), 5623:1–5623:28.

[8] Zeqing Fu and Wei Hu. 2021. Dynamic Point Cloud Inpainting via Spatial-Temporal Graph Learning. *IEEE Transaction on Multimedia* 23 (2021), 3022–3034.

[9] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai. 2020. An Overview Of Ongoing Point Cloud Compression Standardization Activities: Video-based (V-PCC) and Geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020), e13:1–e13:17.

[10] Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

[11] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2021. Deep Learning For 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 12 (2021), 4338–4364.
[12] Mohd Saad Hamid, NurulFajar Abd Manap, Rostam Affendi Hamzah, and Ahmad Fauzan Kadmin. 2022. Stereo Matching Algorithm Based On Deep Learning: A Survey. *Journal of King Saud University-Computer and Information Sciences* 34, 5 (2022), 1663–1673.
[13] Hugues Hoppe. 2008. Poisson Surface Reconstruction And Its Applications. In *Proc. of the ACM Symposium on Solid and Physical Modeling (SPM'08)*. New York, NY, 10.
[14] Wei Hu, Zeqing Fu, and Zongming Guo. 2019. Local Frequency Interpretation And Non-local Self-similarity On Graph For Point Cloud Inpainting. *IEEE Transactions on Image Processing* 28, 8 (2019), 4087–4100.
[15] I-Chun Huang. 2023. Composing Error Concealment Pipelines for Dynamic 3D Point Cloud Streaming. https://github.com/Huang-I-Chun/error_concealment_pipeline.
[16] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. 2014. Image Completion Using Planar Structure Guidance. *ACM Transactions on graphics* 33, 4 (2014), 1–10.
[17] Tzu-Kuan Hung, I-Chun Huang, Samuel Rhys Cox, Wei Tsang Ooi, and Cheng-Hsin Hsu. 2022. Error Concealment Of Dynamic 3D Point Cloud Streaming. In *Proc. of the ACM Multimedia (MM'22)*. Lisboa, Portugal, 3134–3142.
[18] Anupama K. Ingale and Divya Udayan J. 2021. Real-time 3D reconstruction techniques applied in dynamic scenes: A systematic literature review. *Computer Science Review* 39 (2021), 100338:1–100338:13.
[19] ITU-R BT.500-13 2012. *Methodology For The Subjective Assessment Of The Quality Of Television Pictures ITU-R Recommendation BT.500-13*. Document. International Telecommunication Union.
[20] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi. 2019. Video-Based Point-Cloud-Compression Standard In MPEG: From Evidence Collection To Committee Draft. *IEEE Signal Process Mag* 36, 3 (2019), 118–123.
[21] Mohammad Kazemi, Mohammad Ghanbari, and Shervin Shirmohammadi. 2021. A Review Of Temporal Video Error Concealment Techniques And Their Suitability For HEVC And VVC. *Multimedia Tools and Applications* 80 (2021), 12685–12730.
[22] Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics* 32, 3 (2013), 1–13.
[23] Deepika Kumari and Kamaljit Kaur. 2016. A Survey On Stereo Matching Techniques For 3D Vision In Image Processing. *Int. J. Eng. Manuf* 4 (2016), 40–49.
[24] Peter Lancaster and Kes Salkauskas. 1981. Surfaces Generated By Moving Least Squares Methods. *Mathematics of computation* 37, 155 (1981), 141–158.
[25] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. 2019. Copy-and-paste Networks For Deep Video Inpainting. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. Seoul, Korea, 4413–4421.
[26] Peter Liepa. 2003. Filling Holes In Meshes. In *Proc. of the Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP'03)*. Goslar, Germany, 200–205.
[27] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. 2014. Partial Difference Operators On Weighted Graphs For Image Processing On Surfaces And Point Clouds. *IEEE Transactions on Image Processing* 23, 9 (2014), 3896–3909.
[28] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. 2015. PDE-Based Graph Signal Processing For 3-D Color Point Clouds: Opportunities For Cultural Heritage. *IEEE Signal Process Mag* 32, 4 (2015).
[29] MPEG/N0038 2020. *Common Test Conditions For V3C And V-PCC*. Document. ISO/IEC JTC1/SC29/WG7 MPEG 3D Graphics Coding.
[30] MPEG/N19519 2020. *V-PCC Test Model v11*. Document. ISO/IEC JTC1/SC29/WG11 MPEG 3DG.
[31] Mark Pauly, Niloy Mitra, Joachim Giesen, Markus Gross, and Leonidas Guibas. 2005. Example-based 3D Scan Completion. In *Proc. of the Symposium on Geometry Processing (SGP'05)*. Vienna, Austria, 23–32.
[32] Radu Rusu and Steve Cousins. 2011. 3D Is Here: Point Cloud Library (PCL). In *Proc. of the ICRA'11*.
[33] Yuang Shi, Pranav Venkatram, Yifan Ding, and Wei Tsang Ooi. 2023. Enabling Low Bit-Rate MPEG V-PCC-encoded Volumetric Video Streaming With 3D Sub-sampling. In *Proc. of the 14th Conference on ACM Multimedia Systems (MMsys'23)*. Vancouver, Canada, 108–118.
[34] Yuan-Chun Sun, I-Chun Huang, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. 2023. A Dynamic 3D Point Cloud Dataset For Immersive Applications. In *Proc. of the ACM Multimedia Systems Conference (MMsys'23)*. Vancouver, Canada, 376–383.
[35] Joan Verdera, Vicent Caselles, Marcelo Bertalmio, and Guillermo Sapiro. 2003. Inpainting Surface Holes. In *Proc. of the International Conference on Image Processing (ICIP'03)*. Barcelona, Spain, II–903.
[36] Verified Market Research. 2023. *Immersive Technology Market Size and Forecast*. https://www.verifiedmarketresearch.com/product/immersive-technology-market/.
[37] Irene Viola, Jelmer Mulder, Francesca De Simone, and Pablo Cesar. 2019. Temporal Interpolation Of Dynamic Digital Humans Using Convolutional Neural Networks. In *Proc. of the IEEE International Conference on Artificial Intelligence*

and Virtual Reality (AIVR'19). San Diego, CA, 90–97.

[38] Jianning Wang and Manuel Oliveira. 2007. Filling Holes On Locally Smooth Surfaces Reconstructed From Point Clouds. *Image Vis Comput* 25, 1 (2007), 103–113.

[39] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. 2021. Deep 3D Human Pose Estimation: A Review. *Computer Vision and Image Understanding* 210 (2021), 103225:1–103225:21.

[40] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility To Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[41] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. 2020. Point Cloud Completion By Skip-attention Network With Hierarchical Folding. In *Proc. of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR'20)*. Seattle, WA, 1939–1948.

[42] Cheng-Hao Wu, Chih-Fan Hsu, Tzu-Kuan Hung, Carsten Griwodz, Wei Tsang Ooi, and Cheng-Hsin Hsu. 2023. Quantitative Comparison Of Point Cloud Compression Algorithms With PCC Arena. *IEEE Transaction on Multimedia* 25 (2023), 3073–3088.

[43] Cheng-Hao Wu, Xiner Li, Rahul Rajesh, Wei Tsang Ooi, and Cheng-Hsin Hsu. 2021. Dynamic 3D Point Cloud Streaming: Distortion And Concealment. In *Proc. of the ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV'21)*. Istanbul, Turkey, 9–14.

[44] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. 2020. GRnet: Gridding Residual Network For Dense Point Cloud Completion. In *Proc. of the European Conference on Computer Vision (ECCV'20)*. Virtual, 365–381.

[45] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. 2019. Deep Flow-guided Video Inpainting. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. Long Beach, CA, 3723–3732.

[46] Yusheng Xu, Xiaohua Tong, and Uwe Stilla. 2021. Voxel-based Representation Of 3D Point Clouds: Methods, Applications, And Its Potential Use In The Construction Industry. *Automation in Construction* 126 (2021), 103675:1–103675:26.

[47] Mengyu Yang, Di Wu, Zelong Wang, Miao Hu, and Yipeng Zhou. 2023. Understanding And Improving Perceptual Quality Of Volumetric Video Streaming. In *Proc. of the IEEE International Conference on Multimedia and Expo (ICME'23)*. Brisbane, Australia, 1979–1984.

[48] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative Image Inpainting With Contextual Attention. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. Salt Lake City, UT, 5505–5514.

[49] Yiming Zeng, Yue Qian, Qijian Zhang, Junhui Hou, Yixuan Yuan, and Ying He. 2022. IDEA-Net: Dynamic 3D Point Cloud Interpolation Via Deep Embedding Alignment. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. New Orleans, LA, 6338–6347.

[50] Emin Zerman, Radhika Kulkarni, and Aljosa Smolic. 2021. User Behavior Analysis Of Volumetric Video In Augmented Reality. In *Proc. of the International Conference on Quality of Multimedia Experience (QoMEX'21)*. Virtual, 129–132.

[51] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2023. Open3D: A Modern Library for 3D Data Processing. http://www.open3d.org/.

A RELATED WORK

In this section, we survey the literature on error concealment methods of different types of 2D/3D content, along with methods for relevant yet different problems on 3D point clouds.

A.1 Error Concealment of 2D Video Frames

Error concealment of 2D video frames due to lost or late packets has been thoroughly studied in the literature [21]. A straightforward concealment method replaces a corrupted video frame with the most recently decoded one. A more advanced method decodes as many pixels as possible and spatially or temporally conceals the corrupted regions. Spatial concealment uses the surrounding pixels of the same video frame for concealment. Two classical methods are: (i) *copy-over*, where a nearby region is replicated to replace a corrupted one, and (ii) *averaging*, where the average color of multiple nearby regions is used to fill in a corrupted one. Temporal concealment employs motion vectors and their referred pixels to conceal corrupted regions. Each corrupted or missing motion vector is approximated using the adjacent motion vectors successfully received. While 2D video error concealment methods are not directly applicable to dynamic 3D point cloud streaming, they can be integrated with video-based codecs, such as MPEG V-PCC [20]. *That said, it has been shown that concealing missing 3D point clouds in the 2D domain leads to unacceptable concealed quality [17, 43].*

A.2 Inpainting of 2D/3D Content

Inpainting refers to filling up small missing regions of content. Inpainting has been applied to various 2D/3D content formats, including images [48], videos [25, 45], meshes [5, 26, 35], and point clouds [8, 14, 27, 28, 38]. For 2D content, Yu et al. [48] developed a generative neural network to inpaint missing regions using both global image structures and local image features. Their trained neural network successfully inpainted images with holes at different locations and in various sizes. Inpainting videos are more challenging because of the temporal coherency. To tackle this challenge, Lee et al. [25] trained a deep neural network to duplicate regions in successfully decoded frames to fill the missing regions. Their neural network delivered visually appealing and temporally coherent inpainted videos. Xu et al. [45] also employed a deep neural network to generate motion vectors across video frames. They then used the motion vectors to propagate the decoded pixels to fill up the missing regions. For 3D content, inpainting 3D meshes is also known in the literature as surface hole filling. For instance, Verdera et al. [35] generalized image inpainting algorithms into a geometric partial differential equation system. They then solved the equation system to inpaint missing regions of 3D meshes. Similarly, Liepa [26] leveraged the meshes in surrounding regions to inpaint the triangular meshes in the missing regions. Davis et al. [5] constructed a surface function to follow the meshes close to each missing region and then diffused the function to cover the missing region. Inpainting 3D point clouds has been recently considered. For example, He et al. [14] first detected holes and formulated an optimization problem to find the most similar intra-frame regions. Fu et al. [8] utilized intra-frame self-similarity and inter-frame consistency to inpaint the missing regions of point clouds. *The existing inpainting methods are suitable for fixing distortion in small regions caused by imperfect capturing devices, settings, and processes. Because lost or late packets often result in catastrophic distortion in larger regions, inpainting methods are inapplicable to the error concealment problem.*

A.3 Completion of 2D/3D Content

Completion refers to generating large missing regions of content. *Completion* has also been applied to different 2D/3D content formats, such as images [16], meshes [3, 31], and point clouds [4, 41, 44].

For 2D content, Huang et al. [16] built multiple planes from existing regions. They then employed the resulting planes to derive the offsets and transformations of image patches from completing the missing regions. For completing 3D meshes, Breckon and Fisher [3] took a two-stage approach with the global fitting of geometric surface and the local propagation of texture detail. Pauly et al. [31] took a different approach using an object database of 3D meshes. In particular, they queried the database for the 3D object closest to each incomplete one. The retrieved 3D object is then warped and blended with the incomplete one. Completing 3D point clouds has also been considered. For instance, Chen et al. [4] considered the problem of completing the building facades in three steps: (i) projecting 3D point clouds onto building facades, (ii) applying generative adversarial 2D inpainting algorithms, and (iii) converting the building facades back to 3D point clouds. Wen et al. [41] proposed to complete 3D point clouds by analyzing local structure detail and leveraging a structure-preserving deep neural network. The existing completion methods are mostly designed for: (i) 3D objects other than point clouds and (ii) recreating missing regions using spatially nearby regions. *In contrast, the distortion caused by lost or late packets could totally destroy the structure of point cloud frames, rendering these completion methods inapplicable.*

A.4 Error Concealment of 3D Point Clouds

The error concealment problem addressed in this article aims to improve the rendered quality of the received 3D point clouds. Similar targets have been considered in the literature, e.g., Yang et al. [47] chose to incorporate a super-resolution network to enhance the perceptual quality of the rendered images from a 3D point cloud in their streaming system. Compared to different user behaviors, the quality degradation caused by lost or late packets is much more severe. Additionally, the super-resolution network's input and output are 2D images rather than 3D point clouds. *Hence, their neural networks can not solve the error concealment problem studied in this article.* Another cluster of prior works performed temporal interpolation [2, 37, 49] of dynamic 3D point clouds using deep neural networks mainly for increasing the frame rate. Particularly, Viola et al. [37] used a neural network to estimate motion vectors of downsampled point clouds and used the motion vectors to generate intermediate point cloud frames. The resulting frame is upsampled to get the final output. Zeng et al. [49] aligned the preceding and following point cloud frames point-wise and performed linear interpolation using motion vectors between them to get the output frame. To relax the limitation of the pre-determined number of points per frame, Akhtar et al. [2] proposed a neural network to extract multi-scale features. They hierarchically fused interpolated frames at different scales for the output point cloud frame. *These interpolation neural networks may not be general across datasets or only work for point cloud frames with a pre-determined number of points as input.* Hence, they are less applicable to our error concealment problem in dynamic 3D point cloud streaming. *With that said, we will subjectively compare our proposed representative pipelines against two state-of-the-art interpolation algorithms [2, 49] in Sec. 8. Our user study reveals that these state-of-the-art algorithms suffer from inferior interpolated frame quality while consuming a staggering amount of computational power.*

To the best of our knowledge, Wu et al. [43] is the first work demonstrating the need to conceal 3D point cloud frames due to lost or late packets in the 3D space. The current article takes a step further and proposes: (i) multi-staged error concealment pipelines with a rich set of alternative algorithms for each stage, (ii) through end-to-end and stage-wise performance evaluations, and (iii) a user study to quantify the user experience achieved by different pipelines. Preliminary results of the current article were given in Hung et al. [17].