國立清華大學電機資訊學院資訊工程研究所
碩士論文

Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master's Thesis

為動態三維點雲串流設計錯誤隱藏方法
Composing Error Concealment Pipelines for Dynamic 3D Point Cloud Streaming

黃奕淳
I-Chun Huang

學號：111062585
Student ID:111062585

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 113 年 3 月
March, 2024

國立清華大學
資訊工程研究所

碩士論文

為動態三維點雲串流設計錯誤隱藏方法

黃奕淳

112

Abstract

Dynamic 3D point clouds enable an 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 a 100% preference winning rate over the state-of-the-art learning-based interpolation algorithms while consuming merely up to 8.55% of running time.

中文摘要

動態 3D 點雲技術提供了沉浸式使用者體驗，因此在體積視訊串流應用中變得越來越受歡迎。 在透過網路進行串流傳輸時，點雲幀可能會因遺失或延遲的封包而受到大幅度的視覺品質下降。 為了解決這個問題，我們提出了首個錯誤隱蔽的流水線框架，該框架包括五個階段：前處理、匹配、運動估計、預測和後處理。 每個階段都可以開發替代演算法，而不同階段的演算法可以混合搭配成管線，以進行端到端效能評估。 我們討論了設計目標，並為每個階段提出了多種演算法。 然後，我們使用具有不同特徵的動態 3D 點雲序列對這些演算法進行了定量比較。 基於比較結果，我們提出了四個代表性管線，分別適用於：（i）不同程度的運動變化，即輕微與顯著，以及（ii）不同的應用需求，即高品質與低開銷。 對我們提出的管線進行了廣泛的端到端評估，結果表明，它們在以下兩方面的隱蔽品質明顯優於3D 幀複製方法：（i）3D 指標中，GPSNR 提高了多達5.32 dB，CPSNR 提高了1.7 dB；以及（ii）2D 指標中，PSNR 提高了多達2.22 dB，SSIM 提高了0.06，VMAF 提高了11.67。 此外，一項對 15 名受試者的用戶研究表明，我們性能最佳的管線在消耗僅 8.55% 的運行時間的情況下，實現了對最先進的基於學習的插值演算法 100% 的偏好勝率。

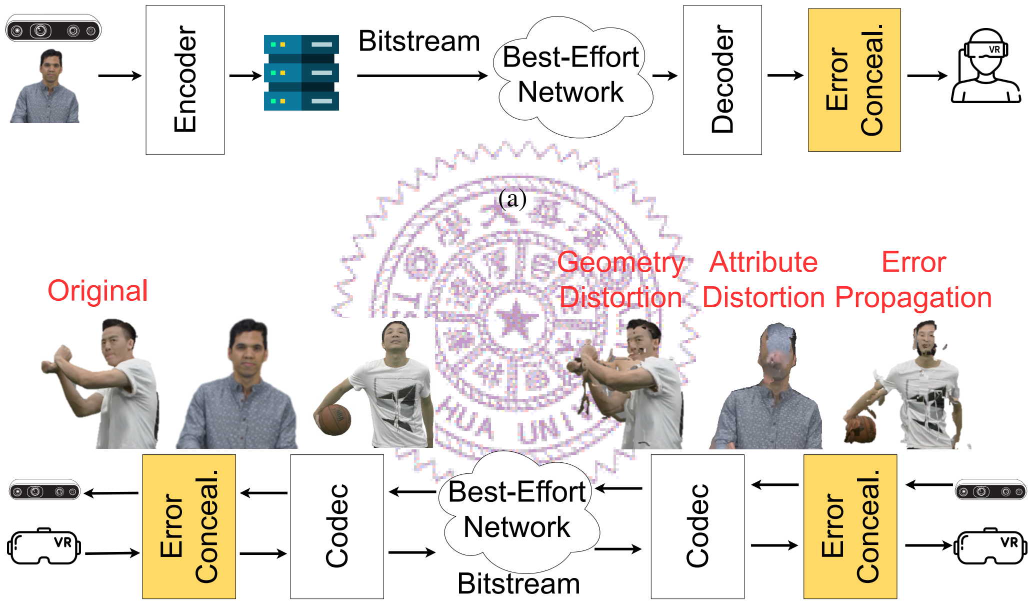
Contents

List of Figures

List of Tables

Chapter 1

Introduction



(b)

Figure 1.1: 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).

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 [42]. 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 captured by commodity sensors while preserving unquantized *geometry* and *attribute* data. The geometry data are essentially coordinates of a set of unordered 3D points, while the attribute data include color, normal, reflectiveness, 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, telesurgery, and remote sensing, where six degree-of-freedom interactions are critical.

In this thesis, we consider dense point clouds meant for human consumption. Particularly, Fig. 1.1 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 G-PCC and V-PCC [48], could help mitigate the pressure caused by high bandwidth requirements. Among them, V-PCC [22] projects 3D point cloud frames onto 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 onto 3D point cloud frames. V-PCC has two strengths compared to other codecs. First, V-PCC has been shown to achieve high coding efficiency [9,22]. 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 can drop by more than 1 to 1.5 in 5-point Mean Opinion Score (MOS) under merely 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 give sample distorted 3D point clouds from the V-PCC reference software in Fig. 1.1(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 Wu et al. [49] and Hung et al. [18, 19].

This thesis 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 [18, 19]. In contrast, geometry distortion is typically catastrophic and needs to be concealed by rebuilding the missing frame from scratch [49]. This task, unfortunately, can not be properly accomplished by existing hole-filling [8,15,30,31,44] or point-cloud completion methods [4,47,50], 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. [49]. Multiple state-of-the-art neural-based interpolation algorithms [2,43,55] for point clouds have also been proposed, but they are computationally expensive and often support a fixed, small, number of points. Therefore, how they perform in different dynamic 3D point cloud streaming scenarios (see Fig. 1.1) in typical best-effort networks is unclear.

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 [45], stereo matching [12,25], or optical/scene flow estimation [20] 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 [20] 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.

1.1 Contributions

This thesis proposed an end-to-end investigation on error concealment of dynamic 3D point cloud streaming, while our initial results were presented in a conference version [18, 19]. This thesis makes the following contributions:

- We propose a general *multi-stage* pipeline framework for error concealment of 3D point cloud streaming in Sec. 4. The framework can incorporate alternative algorithms in each stage.
- We developed and quantitatively compared a suite of algorithms for individual stages in Secs. 4.3–4.6. The algorithms in different stages can be mixed and matched into diverse error concealment pipelines for diverse usage scenarios.
- We constructed multiple representative pipelines following the insights gained in per-stage comparison and extensively evaluated these pipelines through end-to-end experiments in Sec. 5.

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, 55]. Such improved concealment quality could attract and retain users of immersive multimedia applications. More importantly, as we made our implementation publicly available [16], 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.

1.2 Organization

In Ch. 1, we introduce the concept of interpolation and transmission loss in dynamic 3D point cloud streaming. We summarize our contributions along with the challenges and limitations faced during this study. In Ch. 2, we discuss the representation formats of 3D data, highlighting their advantages and disadvantages. We investigate point cloud compression methods used for streaming, analyze the packet loss pattern in the compressed bitstream, and categorize the errors into attribute and geometry error concealment. In Ch. 3, we survey the existing literature, summarizing prior work related to 2D and 3D video error concealment, and other concepts that inspired our design. In Ch. 4, we provide a high-level overview of our proposed pipeline framework. For each stage in the pipeline, we propose alternative solutions to accommodate different conditions or usage scenarios. We evaluate the performance of each solution on a synthetic dataset and analyze their time complexity. In Ch. 5, we combine the proposed methods into four distinct pipeline combinations, each aiming to achieve different goals. We describe the implementation of our system prototype and evaluate its performance in terms of view quality. Objective tests are conducted to compare our pipeline with 3D Frame Copy (3DFC). In Ch. 5.4, a user study is conducted to compare our pipelines with 3DFC and two learning-based methods, providing insights into user preferences and experiences. Finally, we summarize the whole thesis in Ch. 6 and present the possible future work.

Chapter 2

Background

In this chapter, we begin by exploring commonly used 3D representation formats, then delve into compression techniques and discuss the challenges posed by missing or delayed packets during streaming.

2.1 3D Representations

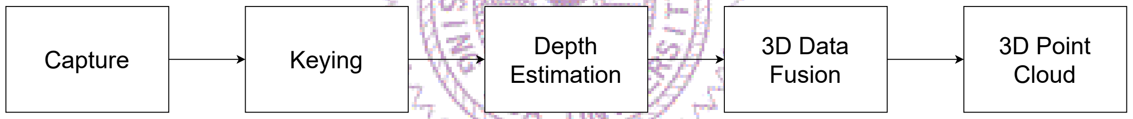


Figure 2.1: Point cloud generation steps.

Point cloud representation is a pivotal format for volumetric streaming, enabling high levels of interaction such as voxelization, manipulation, and easy semantic segmentation. Fig. 2.1 illustrates the generation and production process of point clouds [40]. To acquire high-resolution volumetric videos, Fig. 2.2 presents an exemplary camera setup employed by Sky Studio, involving over a hundred depth cameras placed strategically around the subject to capture exhaustive data from various angles. After capturing images, the next step involves segmenting the subject from the background through image capture from different perspectives. This is followed by depth estimation, where each pixel's depth information is accurately determined using stereo pairs to derive 3D data. The final step involves 3D fusion, where depth data from each camera pair is integrated to form a comprehensive 3D point cloud.

3D Mesh is another 3D representation format that efficiently approximates the surfaces of objects through polygons, typically triangles or quadrilaterals. The flexibility and efficiency of 3D meshes allow for detailed and complex object modeling, enabling the creation of highly realistic virtual environments. Fig. 2.3 compares the resolution be-

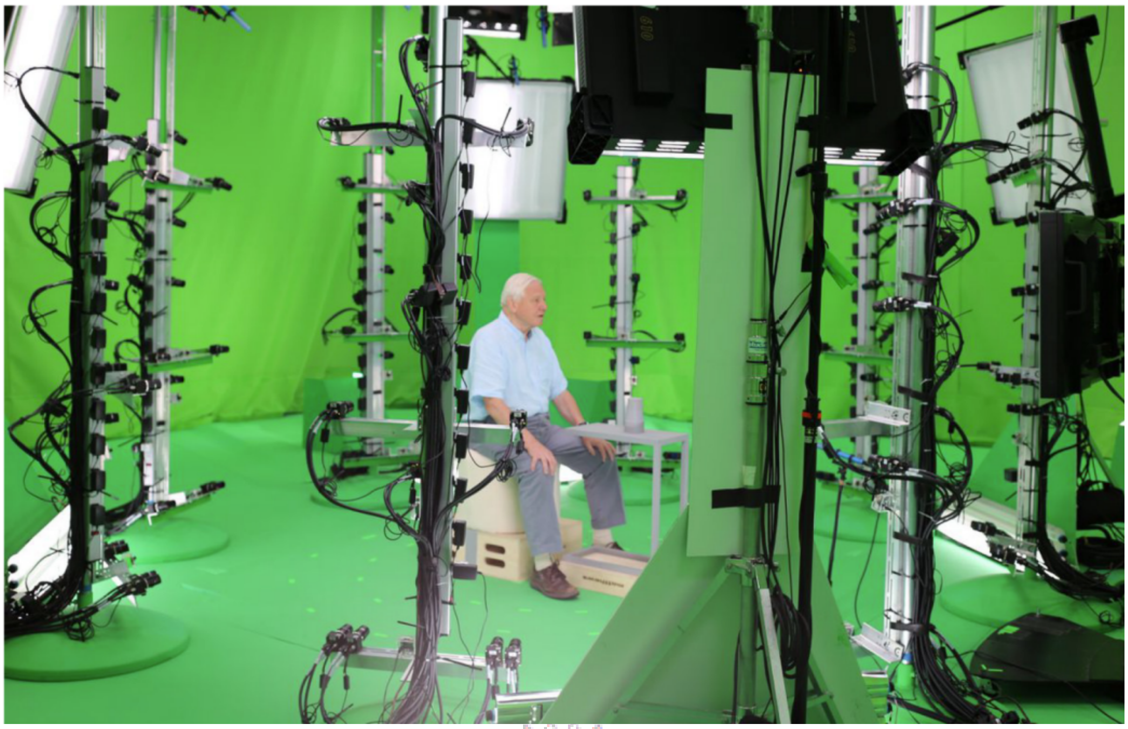


Figure 2.2: Example cameras setup to capture volumetric video by Sky Studios [38].

tween different triangle densities of 3D mesh. Fig. 2.4 shows the generation process of 3D mesh, following which we get the point cloud from the depth camera [40]. Initially, the process involves a depth-based surface reconstruction from the point cloud data, leading to the creation of a dense mesh composed of numerous vertices and faces. Subsequently, a process of geometric simplification is applied to this high-density mesh. This step, often referred to as mesh reduction, streamlines the mesh into a more manageable and consistent form by reducing its complexity. After simplification, the meshes undergo a texturing phase, where they are mapped with 2D texture images, converting them into recognizable, standard 2D image file formats. The culmination of this process is the temporal registration of the meshes, which aligns them over time to produce animated meshes, adding a dynamic dimension to the static models.

Neural Radiance Fields (NeRF) have emerged as a 3D representation format that leverages deep learning to synthesize highly realistic images from novel viewpoints. NeRF models the volumetric scene function using a fully connected deep neural network. This function maps a spatial location (x, y, z) and a viewing direction (θ, ϕ) to a color (R, G, B) and a density value. The density is indicative of the presence of material at each location, while the color represents the appearance of that material from the given direction. To train a NeRF model, people need to collect multiple pictures of a scene from different viewing angles. The process of image synthesis in NeRF involves casting rays from the camera into the scene, sampling points along these rays, and using the neural network to

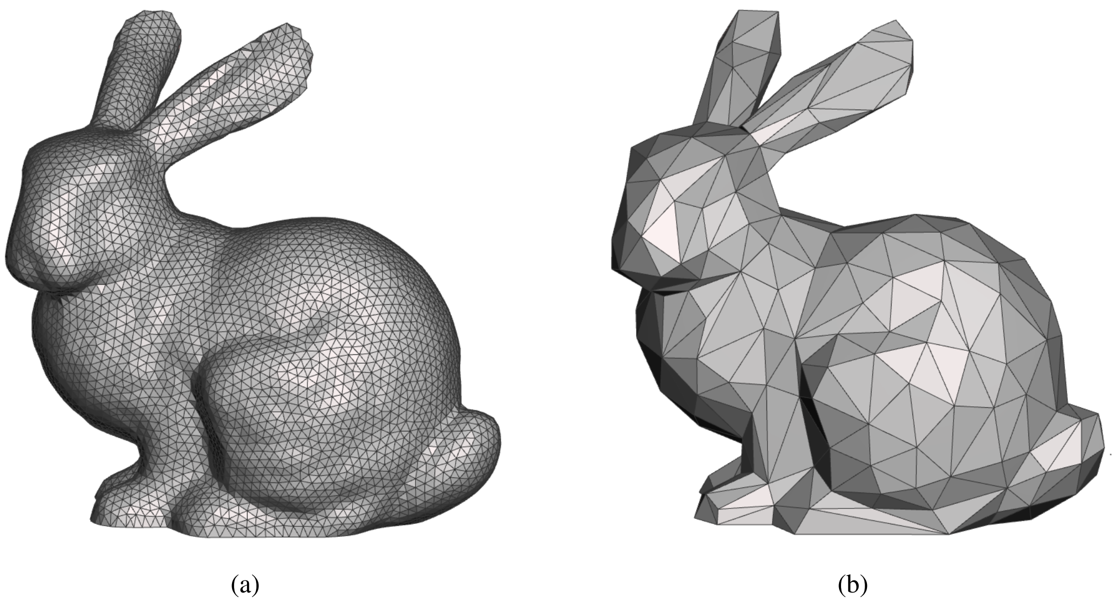


Figure 2.3: Example 3D mesh representations with different levels of resolution: (a) represent with more triangles and (b) represent with less triangles.

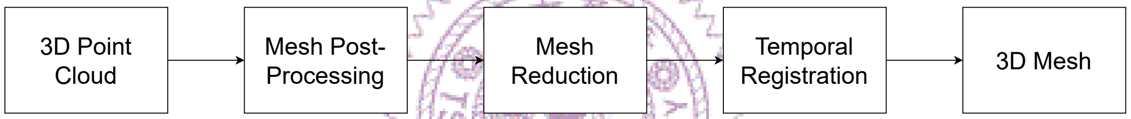


Figure 2.4: 3D mesh generation steps.

predict the color and density at each point. These predictions are then composited back along the rays using volume rendering techniques to produce the final image, which allows for the generation of photorealistic images from viewpoints that were not seen during training, making NeRF a powerful tool for applications in virtual reality, augmented reality, and visual effects. Unlike traditional 3D representation formats that rely on explicit geometry (e.g., meshes or point clouds), NeRF captures both geometry and appearance in a continuous, implicit form, enabling more detailed and nuanced renderings of complex scenes.

3D Gaussian Splatting (3DGS) is a volumetric rendering technique used in visualization to represent three-dimensional data. This method involves projecting 3D points onto a discrete voxel grid, where each point is represented as a Gaussian distribution (splat). The essence of this approach is to model the influence of each 3D point over a range of voxels based on the Gaussian function, which is characterized by its mean (the point's location) and its standard deviation (controlling the spread of the splat). The intensity or weight of each voxel is computed by summing the contributions of all Gaussians affecting that voxel, effectively blending overlapping splats into a smooth, continuous field.

This technique is particularly useful in medical imaging, scientific visualization, and point cloud rendering, where it can be employed to create more natural and interpretable visualizations of sparse or irregularly spaced data. By adjusting the Gaussian parameters, users can control the smoothness and level of detail in the rendered output. 3DGS thus offers a flexible and powerful means for the effective visualization of volumetric data, facilitating the exploration and analysis of complex 3D structures.

In this thesis, our focus is on streaming dynamic 3D content; however, 3D content typically requires a significant amount of storage space, necessitating compression before streaming. Among all the 3D representation formats, point clouds offer better interaction between frames and are much easier to process further (such as voxelization or motion estimation). Additionally, research on point cloud compression has been extensively studied. As a result, we have selected dynamic 3D point clouds as the representation format for error concealment during volumetric streaming.

2.2 Point Cloud Compression

The evolution of point cloud compression (PCC) standards, guided by the Moving Picture Experts Group (MPEG), marks a significant advancement in immersive technology applications. Recognizing the potential of point clouds for immersive representations in tele-immersive applications in 2013, MPEG has played a pivotal role in standardizing efficient coding solutions for this purpose. By 2017, MPEG launched a call for proposals to identify innovative coding technologies for PCC [33], benchmarked against a baseline derived from the point cloud library (PCL2). The initiative focused on three Test Model Categories (TMCs): TMC1 for static objects and scenes, TMC2 for dynamic point clouds, and TMC3 for dynamically acquired point clouds. Following a comprehensive evaluation in 2018, TMC1 and TMC3 merged into TMC13, adopting octree coding for geometry-based PCC (G-PCC), while TMC2, emphasizing 2D projection for video-based PCC (V-PCC), leveraged existing video codecs like HEVC. The formulation of the V-PCC and G-PCC standards was completed in 2020 and 2021, respectively. Currently, MPEG is enhancing G-PCC with advanced features, such as scalability and temporal coding, aiming to revolutionize lossy and lossless point cloud geometry/attributes compression.

Geometry-based point cloud compression (G-PCC) codes the geometry (x, y, and z axis) and attributes (color, reflectance) separately. Fig. 2.5 shows the high-level overview. The coding of attributes utilizes the decoded geometry to minimize discrepancies between the original and reconstructed attributes. G-PCC standard mandates the use of integer coordinates, necessitating the normalization and voxelization of input point cloud coordinates to a specified bit depth. This initial step transforms point cloud coordinates into a frame coordinate system, delineating a 3D bounding box that encompasses the entire

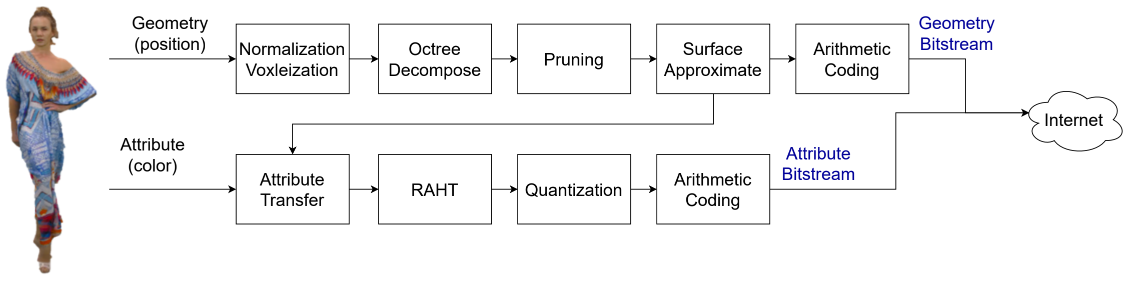


Figure 2.5: High-level overview of geometry-based point cloud compression pipeline.

point cloud. The dimensions of this bounding box are then quantized based on the chosen voxelization precision, a process that may result in duplicate points, which the codec subsequently resolves.

G-PCC employs two primary encoding modes for geometry compression: octree coding, designed for high compression efficiency with various optimization modes, and predictive coding, a simpler method intended for low-latency applications like LiDAR dynamic acquisition, which sequentially predicts the current point's value from previous samples. In octree coding, following voxelization, the 3D space undergoes octree decomposition. The compression becomes lossy when the octree is pruned at a specific depth, significantly reducing the number of points at lower bitrates and impacting visual quality. To mitigate density loss, G-PCC introduces an advanced tool known as triangle soup (Trisoup) coding, which employs surface approximation. At a particular octree level, a leaf node, representing a 3D cube, may be filled or empty. The Trisoup method approximates the cube's surface by estimating intersections with the cube's edges, resulting in vertices that are entropy coded and included in the bitstream. The surface is then approximated by a polygon through triangulation, further detailed in the literature. The decoded geometry, represented by a point cloud, is obtained by rasterizing the triangles at a resolution determined by the voxel size, allowing for the reconstruction of smooth surfaces with minimal information transmission. The detail of the triangulation algorithm is described in [32].

G-PCC standard currently facilitates the encoding of color and reflectance attributes. Color information is encoded using the YCbCr color space, necessitating a conversion process for point clouds originally in different color spaces. Reflectance is encoded using a single color channel. Attribute encoding is performed subsequent to the geometric decoding phase. In instances of lossy geometric encoding, discrepancies between the original and coded point coordinates necessitate the transfer of color attributes to the newly encoded geometry, executed exclusively at the encoder level, thus falling outside the standardization framework.

Additionally, G-PCC incorporates the Region-Adaptive Hierarchical Transform (RAHT), a wavelet-like hierarchical spatial transform applied to the octree structure from its leaves upwards. Each step of the RAHT generates approximation and detail coefficients, effectively concentrating the attribute signal's energy into a minimal set of low-frequency coefficients, thereby achieving a coding gain. For densely occupied voxel blocks, RAHT is analogous to a 3D Haar transform, modified to account for empty voxels and variable point cloud density by adjusting the Haar filter coefficients to reflect local density variations. Recent advancements have introduced a region-adaptive technique that forms a hierarchical graph Fourier transform, with extensions including intra-prediction, offering significant performance improvements over the traditional RAHT approach.

G-PCC concludes with context-based arithmetic coding, where the contexts are determined by the utilized coding tools such as octree, triangle soup (Trisoup), or predictive coding. The Direct Coding Mode (DCM) bypasses this step due to the challenge of identifying statistical dependencies among isolated points. For octree occupancy coding, contexts are primarily based on the Neighbor Configuration (NC), which refers to the occupancy status of adjacent nodes to the parent node. Moreover, compression efficiency can be enhanced by considering the occupancy of encoded child nodes neighboring the current node. G-PCC employs a diverse array of contexts, which can be streamlined by recognizing and applying symmetries and rotations to invariant cases. For further insights into arithmetic coding within G-PCC, readers are directed to consult the detailed codec documentation.

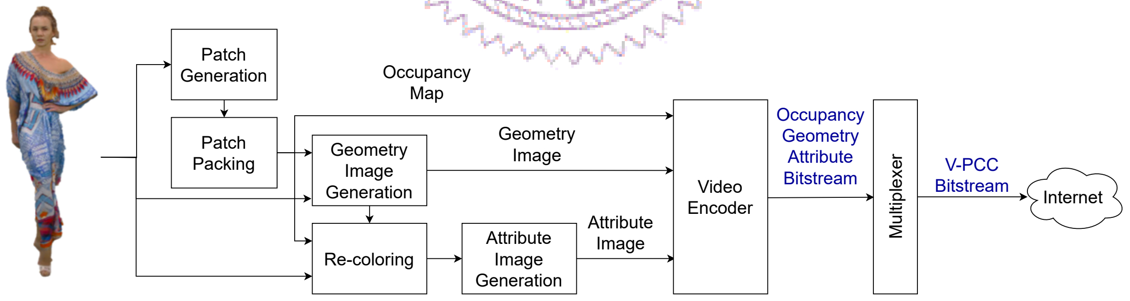


Figure 2.6: High-level overview of video-based point cloud compression pipeline.

Video-based Point Cloud Compression (V-PCC) utilizes the 2D projection principle to transform the 3D geometry and attributes of point clouds into a sequence of optimized 2D patches for coding, as outlined in the simplified model of the V-PCC codec. Fig. 2.6 gives a high-level overview of the V-PCC pipeline. The initial phase involves dividing the 3D point cloud into patches through a non-standardized method. For instance, the heuristic used in the V-PCC reference framework involves calculating local normals for each point and assigning these points to one of the six faces of a bounding box based on normal similarity. This method aims to ensure projections retain maximum visible surface detail. Subsequently, points are grouped by their normals and plane associations in a cyclical manner, resulting in three patch types: attribute projections that map initial attributes, geometry projections detailing point distances and their 3D bounding box locations, and occupancy maps that denote the presence of voxels within the 2D projections, as shown in Fig. 2.7.

After the patch packing process in V-PCC, attribute, geometry, and occupancy images are created, collectively known as the atlas. The coding process might alter the geometry of the decoded point cloud, leading to changes in point positions or their removal. To address this, V-PCC incorporates a recoloring process, where the color of each point in the decoded point cloud is adjusted based on the closest original point and its surrounding neighbors. Additionally, an optional attribute smoothing step can be applied to minimize seam artifacts at patch boundaries, executed as a post-process in the 3D domain to utilize accurate neighborhood information.

The packed images present coding challenges due to the disjoint nature of the patches and the presence of sharp edges and high frequencies. To make these images more conducive to coding, both texture and geometry images undergo filtering processes. These processes aim to mitigate discontinuities between patches and soften edges while retaining object contours. Various methods have been suggested for padding attributes/textures and geometry. For example, the reference G-PCC implementation uses a multi-resolution guided filtering for textures, producing a smoother image that is easier to encode. Geometry images are improved by filling the spaces between patches with a padding function to create a piece-wise smooth surface, facilitating efficient encoding, such as with directional spatial prediction techniques in HEVC. This enhancement step is referred to as geometry dilation.

Following the construction and padding of atlas images in V-PCC, they are subjected to video compression, utilizing the HEVC standard. Notably, this framework is adaptable to newer compression standards like Versatile Video Coding. Occupancy maps are encoded in the codec's lossless mode to preserve detail. Bitrate adjustment is achievable by packing occupancy images at block sizes larger than one pixel, effectively implementing spatial downsampling. The individual streams of the atlas, along with crucial metadata—such as patch positions, orientations, and packing block sizes—are multiplexed to form the comprehensive V-PCC bitstream.

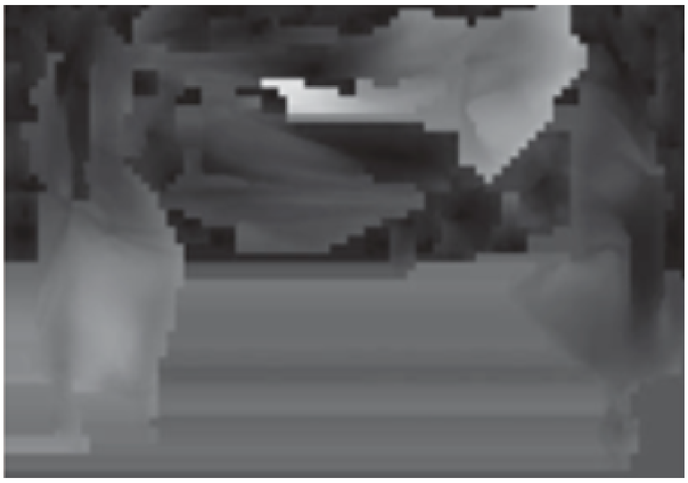
In summary, G-PCC efficiently compresses point clouds within a three-dimensional space, preserving the intricate 3D structures, and is capable of lossless compression. Conversely, V-PCC transforms point clouds into a two-dimensional space. This process compromises the structural integrity of the point cloud, but leverages the advanced development of 2D video codecs, facilitating more efficient compression ratios and enabling real-time streaming capabilities. The focus of this thesis is on V-PCC, given its superior compression performance and streaming potential.

2.3 Error Concealment Problem

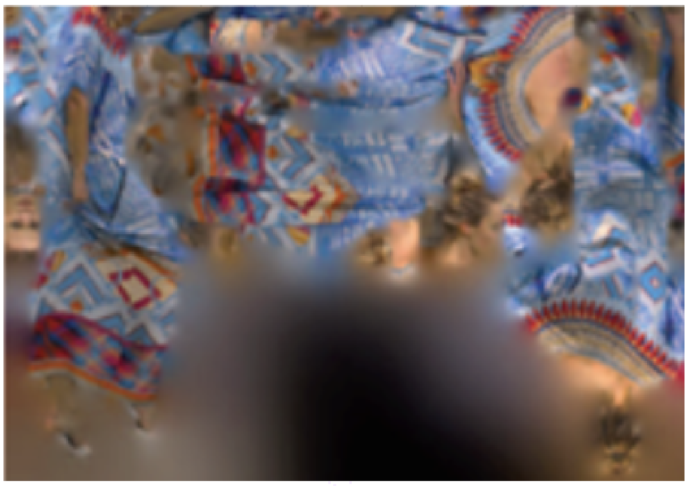
The error concealment problem of dynamic 3D point cloud streaming aims to conceal distorted 3D point cloud frames due to lost or late packets. It has been shown [18, 19, 49] that the frame distortion could be classified into: (i) attribute distortion, where point colors are corrupted, and (ii) geometry distortion, where point coordinates are corrupted. Attribute distortion is easier to conceal, as long as the geometry data remain accurate. Specifically, Hung et al. [18,19] proposed to copy the attribute (color) of the nearest point in the preceding frame to the distorted frame, which led to acceptable concealed quality. In contrast, geometry distortion is typically catastrophic and can only be concealed from scratch [49]. While the three error concealment algorithms proposed by Hung et al. [18, 19] improve the quality over the naive "copy-over" method, referred to as 3D Frame Copy (3DFC), there exists some room for enhancement. Hence, in the rest of this thesis, we use geometry error concealment and error concealment interchangeably. The distorted frame can lose all structural information or even fail.

Wu et al. [49] and Hung et al. [18, 19] performed a thorough investigation on the impact of dropping individual Network Abstraction Layer Units from all possible permutations of the bitstreams in V-PCC. By observing a total of 43 corrupted V-PCC bitstream permutations, Hung et al. [18, 19] classified them into *attribute distortion*, which affects color or *geometry distortion*, which affects the point position of 3D point cloud.

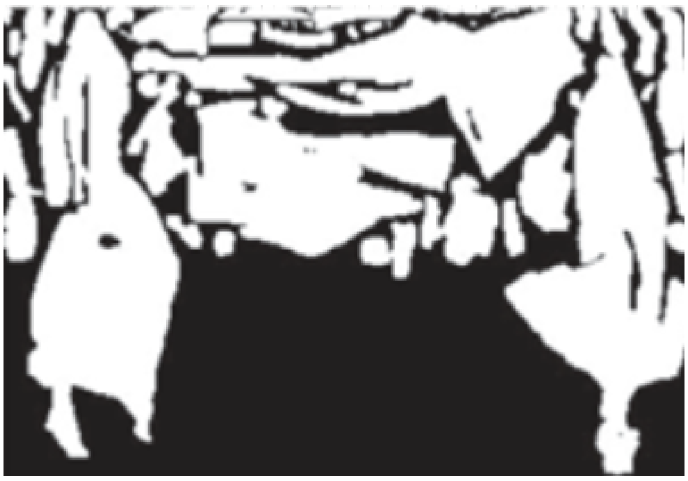
The current naive error concealment method in V-PCC codec, which is a 2D frame copy method (2DFC) at the pixel level, could suffer from serious quality drops. This is because the intrinsic 3D object's motion continuity is destroyed after the 2D projections, as mentioned in [28]. Therefore, the projected patches of nearby frames can be placed in totally different positions. Hung et al. [18, 19] showed that 2DFC suffers from a serious quality drop in both 3D and 2D quality (as high as 25.08 dB in GPSNR and 0.45 in SSIM compared to the correct decode frame) under 5% packet loss. *This demonstrates the necessity of 3D error concealment algorithms.* For further experiment details and results, please refer to [18, 19, 49].



(a)



(b)



(c)

Figure 2.7: Example projection map of V-PCC: (a) geometry map, (b) attribute map, and (c) occupancy map.

Chapter 3

Related Work

In this chapter, 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.

3.1 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 [23]. 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 that are 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 [22]. *That said, it has been shown that concealing missing 3D point clouds in the 2D domain leads to unacceptable concealed quality [18, 19, 49].*

3.2 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 [54], videos [27,51], meshes [5, 29,41], and point clouds [8,15,30,31,44]. For 2D content, Yu et al. [54] 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. [27] 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. [51] 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. [41] 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 [29] 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. [15] 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.*

3.3 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 [17], meshes [3, 36], and point clouds [4, 47, 50]. For 2D content, Huang et al. [17] 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. [36] 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. [47] 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.*

3.4 Error Concealment of 3D Point Clouds

The error concealment problem addressed in this thesis 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. [53] 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 cannot solve the error concealment problem studied in this thesis.* Another cluster of prior works performed temporal interpolation [2,43,55] of dynamic 3D point clouds using deep neural networks mainly for increasing the frame rate. Particularly, Viola et al. [43] 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. [55] 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 may 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, 55] in Sec. 5.4. 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. [49] 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 thesis 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 thesis were given in Hung et al. [18, 19].

Chapter 4

Error Concealment Pipeline Framework

This chapter introduces our five-stage pipeline framework, outlining the design goals for the algorithms at each stage. Following this overview, we delve into detailed discussions of each individual stage, presenting and comparing alternative solutions for each phase.

4.1 Overview

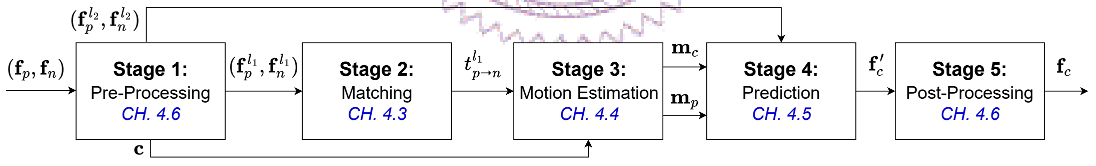

$(\mathbf{f}_p^{l_2}, \mathbf{f}_n^{l_2})$
$(\mathbf{f}_p, \mathbf{f}_n)$
Stage 1: Pre-Processing CH. 4.6
$(\mathbf{f}_p^{l_1}, \mathbf{f}_n^{l_1})$
Stage 2: Matching CH. 4.3
$t_{p\to n}^{l_1}$
Stage 3: Motion Estimation CH. 4.4
$\mathbf{m}_c$
$\mathbf{m}_p$
Stage 4: Prediction CH. 4.5
$\mathbf{f}_c'$
Stage 5: Post-Processing CH. 4.6
$\mathbf{f}_c$
$\mathbf{c}$

Figure 4.1: The proposed error concealment pipeline framework for dynamic 3D point cloud streaming.

Fig. 4.1 illustrates the framework of our error concealment pipelines. It generates the 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 frame drops. 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 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 $t_{p\rightarrow n}^{l_1}$ 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 $t_{p\rightarrow n}^{l_1}$. 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 as: (i) stages are optional and can be bypassed, and (ii) each stage can be instantiated by alternative algorithms.

4.2 Design Objective

The eventual goal of error concealment algorithms is generating $\mathbf{f}_c$ with high visual quality, which can be quantified by 3D and 2D quality metrics [48]. 3D metrics include: (i) *GPSNR* is the PSNR of the Chamfer distance, which is the average distance of pair-wise closest points of two frames. (ii) *Hausdorff distance* is the maximal shortest distance between the closest points of two frames. (iii) *CPSNR (color PSNR)* is a quality function of the luminance distortion between the closest points of two frames.

2D metrics include (i) the average *PSNR* of the foreground object in the point cloud sequence; (ii) the average Structural SIMilarity [46] (*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. [56] reported that most VR users view 3D avatars on their horizontal planes. Hence, to compute the 2D metrics, we render eight inward 2D images at 45° intervals with Open3D [57], and report the average 2D metrics across these eight images. We employ a rendering point size of 8 with sparser sequences [39] and that of 3 with denser sequences [6]. For convenience, we align the range of the point coordinates of the sparser sequences [39] to that of the denser sequences [6].

As point matching is a common and essential stage 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:

$$d_g(t_{p\to n}, t^{*}_{p\to n}, p_i) = \Delta_{cor}\Big(t_{p\to n}(p_i), t^{*}_{p\to n}(p_i)\Big), \tag{4.1}$$

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:

$$d_r(t_{p\to n}, t^{*}_{p\to n}, p_i) = \cos^{-1}\frac{\overrightarrow{p_i t_{p\to n}(p_i)} \cdot \overrightarrow{p_i t^{*}_{p\to n}(p_i)}}{\left|\overrightarrow{p_i t_{p\to n}(p_i)}\right|\left|\overrightarrow{p_i t^{*}_{p\to n}(p_i)}\right|}, \tag{4.2}$$

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

$$d_g(t_{p\to n}) = \Sigma_{p_i\in\mathbf{f}_p} d_g(t_{p\to n}, t^{*}_{p\to n}, p_i)/|\mathbf{f}_p|; \tag{4.3}$$

$$d_r(t_{p\to n}) = \Sigma_{p_i\in\mathbf{f}_p} d_r(t_{p\to n}, t^{*}_{p\to n}, p_i)/|\mathbf{f}_p|. \tag{4.4}$$

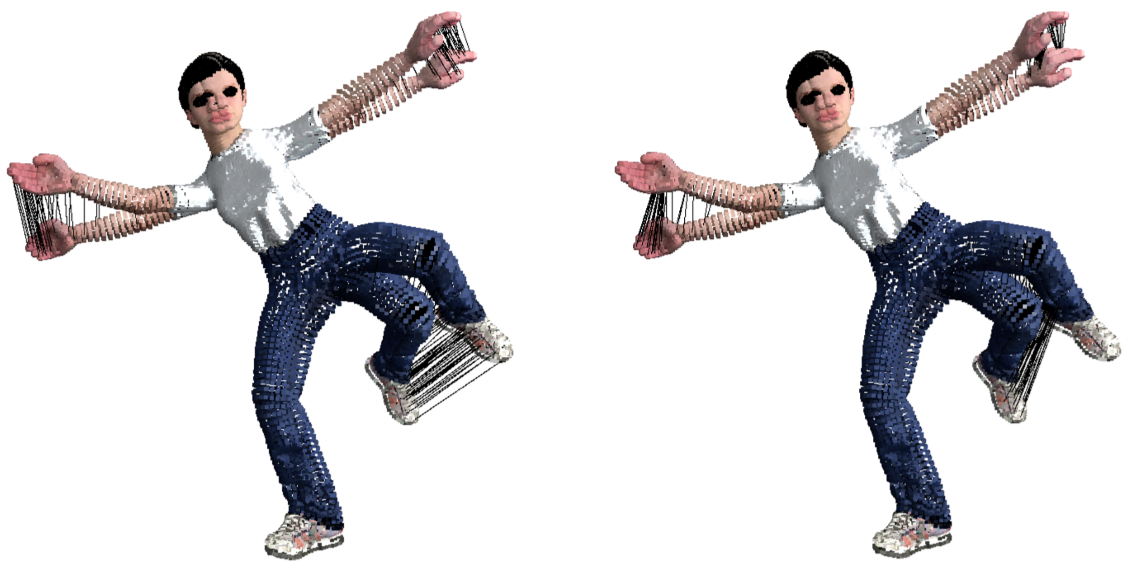
Smaller $d_g(\cdot)$ and $d_r(\cdot)$ values lead to higher spatial and temporal smoothness, respectively.

4.3 Matching Stage

In this section, we elaborate on the goal of finding good matching between the previous and next frames. Then we introduce three solutions approaches and compare the tradeoff among all three methods between time and quality.

4.3.1 Alternative Solutions

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 $t^{l_1}_{p\to n}$ is challenging for real-world dynamic point cloud sequences since each frame is composed of different numbers of unordered points. As an illustrative example, we implement the **Nearest-Neighbor (NN)** algorithm to find the matching table: $t^{l_1}_{p\to n} = \Big\{ \arg\min_{q\in\mathbf{f}^{l_1}_n} \Delta_{cor}(p, q) \Big| p \in \mathbf{f}^{l_1}_p \Big\}$. We use the sequences with ground-truth matched points in Sun et al. [39] in our experiments. Fig. 4.2 visualizes sample results from two frames of a dancing woman, in which matched points are connected by lines. Fig. 4.2(b) reveals that NN matches some points of the avatar's



(a) (b)

Figure 4.2: Point matching results between overlaid $(\mathbf{f}_p, \mathbf{f}_n)$: (a) ground-truth and (b) NN matching. Only 2% of samples' matched points are shown for clarity.

fingertips to those of her palms and some points of her 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. We present two options for the 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 $p \in \mathbf{f}_p^{l_1}$, we define $\mathrm{s}(p, \tau) = \left\{ q \in \mathbf{f}_n^{l_1} \middle| \Delta_{cor}(p, q) < \tau \right\}$ as the candidate point set. Next, we find the most similar point q^* from $\mathrm{s}(p, \tau)$ by: $q^* = \arg\min_{q \in \mathrm{s}(p,\tau)} \Delta(p, q)$, with a utility function:

$$\Delta(p, q) = \alpha \Delta_{cor}(p, q) + (1 - \alpha)\Delta_{rgb}(p, q) + \beta t(q). \tag{4.5}$$

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 $p \in \mathbf{f}_p^{l_1}$ with $\mathrm{s}(p, \tau) = \emptyset$, we match p to $\emptyset$ in the resulting $t_{p \to n}^{l_1}$.

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 the 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 $\mathbf{f}_n^{l_1}$ by their colors in two passes. The first pass scans through $\mathbf{f}_n^{l_1}$ and incrementally builds a set of point clusters $\mathbf{u}_n$. More specifically, for each $p \in \mathbf{f}_n^{l_1}$, we add $\{p\}$ to $\mathbf{u}_n$ iff $\max_{u_j \in \mathbf{u}_n} \Big\{ \min_{q_j \in u_j} \Delta_r(p, q_j), \min_{q_j \in u_j} \Delta_g(p, q_j), \min_{q_j \in u_j} \Delta_b(p, q_j) \Big\} \geq 32$, where 32 is empirically selected. After the first pass, we get a set of clusters, where each cluster 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 $q_i = \arg\min_{q \in u_j} \Delta_{cor}(p_i, q)$. 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:

$$\tau_k = \gamma \max_{p_i \in u_k} \Delta_{cor}(p_i, q_i), \tag{4.6}$$

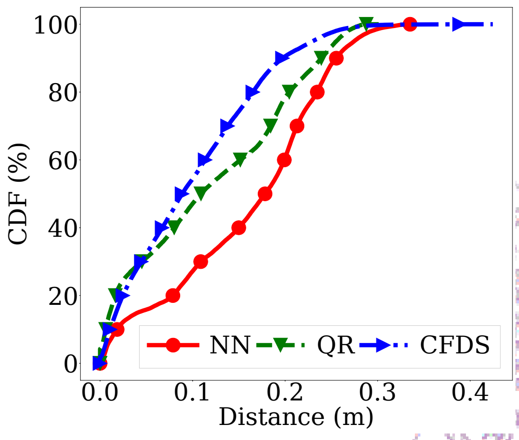
where the scaling factor $\gamma \in \mathbb{R}^+$ is a system parameter. Notice that we go with the largest search radius to accommodate the point with the largest motion vector.

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 a 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).

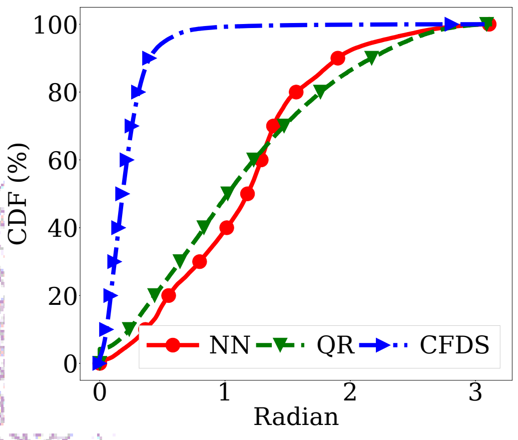
AQR can be performed with either forward ($t_{p \to n}^{l_1}$) or backward ($t_{n \to p}^{l_1}$) point matching. We found that by performing the AQR algorithm in the direction with a larger average magnitude of matching tables as computed by the NN algorithm, we can significantly improve the smoothness. For example, with a dancing man sequence [39], 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.

4.3.2 Comparison

We compared NN, QR, and AQR using a dancing man sequence [39] 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 $s(p, \tau)$. We sampled an $\mathbf{f}_p$ frame every 10 frames, leading to 50 frames in total. The voxel length l_1 was set to 1[1]. The following system parameters were empirically chosen: $\alpha = 0.9$, $\beta = 0.1$, and $\gamma = 1.5$.



(a)



(b)

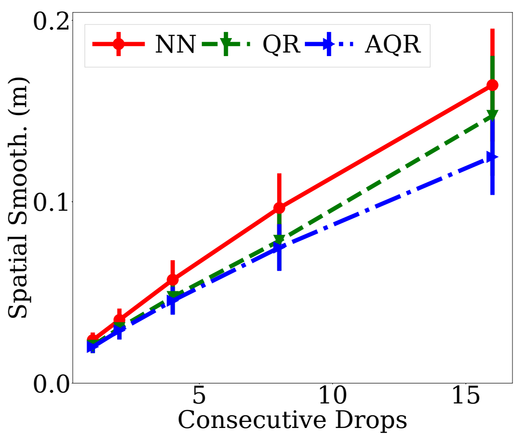
Figure 4.3: Sample smoothness comparison of matching algorithms with 4 consecutive frame drops. Sample results from a sample frame of the dancing man sequence (a) spatial and (b) temporal.

We first plot the Cumulative Distribution Function (CDF) of d_g and d_r from a sample frame with four consecutive drops in Fig. 4.3. We observe that AQR can pick a matching point closer to the ground-truth than QR and NN, and the direction will also be much more correct. In this sample frame, AQR can achieve 80% point matching accuracy with errors of less than 0.16 meters in d_g and 0.3 radians in d_r. On the other hand, QR and NN have lower accuracy with errors of 0.20 and 0.23 meters in d_g and 1.76 and 1.56 radians in d_r, respectively. at least 0.001 m in d_g and 0.12 radian in d_r

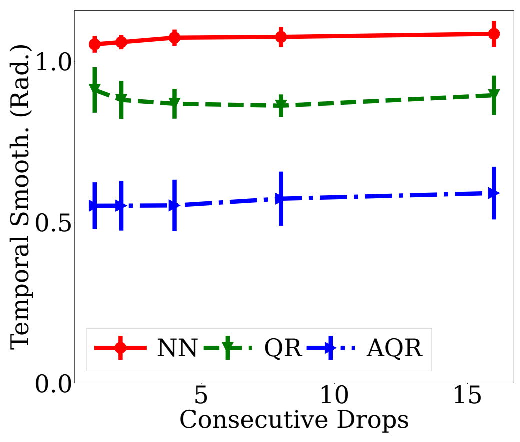
Fig. 4.4 gives the average smoothness results with 95% confidence intervals[2]. We observe that as the number of consecutive frame drops increases, the spatial smoothness

[1]In this thesis, we assume coordinates are integers. Therefore, downsampling with a voxel length of 1 means skipping a downsample.

[2]Throughout this thesis, we plot 95% confidence intervals as errorbars, if not otherwise specified.



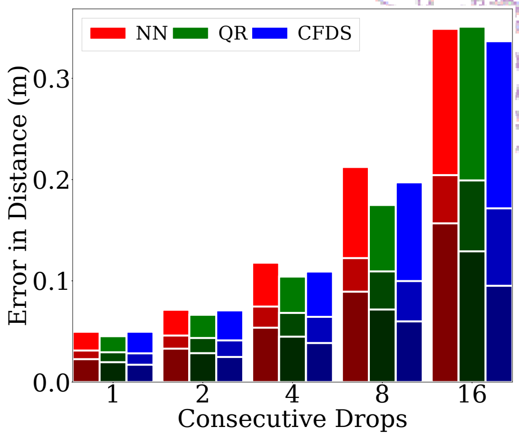
(a)



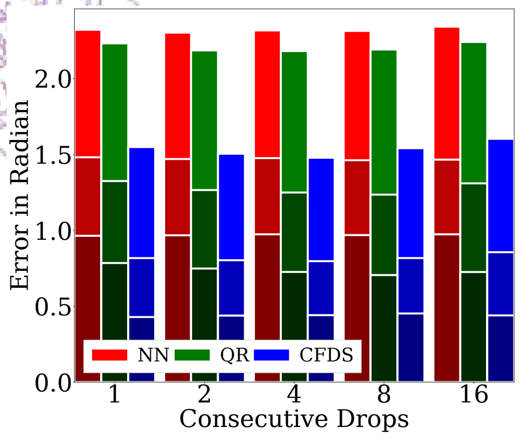
(b)

Figure 4.4: Comparison of matching algorithms: (a) spatial and (b) temporal 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.



(a)



(b)

Figure 4.5: Detailed smoothness comparison of matching algorithms with a diverse number of consecutive drops across the dancing man sequence: (a) spatial and (b) temporal. The three segments of each bar represent 95, 75, and 50 percentiles.

We plot the 95th, 75th, and 50th percentile d_g and d_r error distribution for each method at different consecutive frame drops in Fig. 4.5. It was found that AQR consistently outperforms QR, and QR consistently outperforms NN in the 75th and 50th percentile for d_g and d_r. However, QR achieves the best 95th percentile in d_g. This is because QR uses a fixed search radius τ, and the point is skipped if there are no candidates within this radius.

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 $\mathcal{O}(|\mathbf{f}|\log|\mathbf{f}|)$ on average. As τ increases, the time complexity approaches $\mathcal{O}(|\mathbf{f}|^2)$ 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, users have to manually find proper τ values in a trial-and-error fashion. AQR achieves the best smoothness levels but takes longer to compute τ values than QR.

4.4 Motion Estimation Stage

This section starts with proposing two motion estimation algorithms, examining both their targets. Following this, we evaluate the performance of these algorithms through the use of a synthetic dataset, which provides ground truth matches for assessment purposes.

4.4.1 Alternative Solutions

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

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

$$\mathbf{m}_p = \left\{ \overrightarrow{p_i t_{p\to n}(p_i)} \middle| p_i \in \mathbf{f}_p^{l_1} \right\}. \tag{4.7}$$

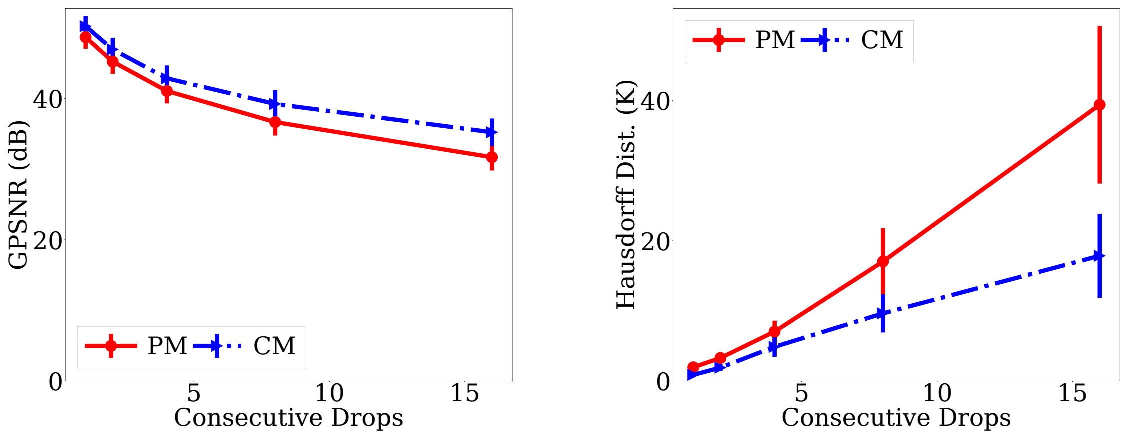
whereas **Cube Motion (CM)** generates cube-based motion vectors by:

$$\mathbf{m}_c = \left\{ \Sigma_{p_i \in c_j} \overrightarrow{p_i t_{p\to n}(p_i)} / |c_j| \middle| c_j \in \mathbf{c} \right\}. \tag{4.8}$$

Cube-based motion estimation 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, point-based motion vectors have a chance to be more accurate if their smoothness levels are small.

4.4.2 Comparison

We take the matching table produced by NN in Sec. 4.3 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 the estimated current frame $\mathbf{f}'_n$ 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.



(a) (b)

Figure 4.6: 3D quality comparison of motion estimation algorithms: (a) GPSNR and (b) Hausdorff distance.

Fig. 4.6 gives the average 3D point cloud quality under different numbers of consecutive frame drops. This figure reveals that, with a dancing man sequence [39], CM improves the 3D quality by up to 3.54 dB in GPSNR 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 $t_{p \to n}$ and runs in $\mathcal{O}(|\mathbf{f}|)$ time.

4.5 Prediction Stage

This section further explains the objective of using the previous stage's motion estimation to predict new frames. We then introduce three prediction methods that utilize interpolation techniques and evaluate their performance in terms of quality.

4.5.1 Alternative Solutions

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. Our earlier work [18, 19] revealed that ill-designed prediction algorithms often lead to severe *cracks*, which dramatically degrade the visual quality and can turn users away from the streaming services. Hence, we propose several prediction algorithms to avoid cracks in the following.

For the sake of presentation, we generalize $\mathbf{m}_c$ into functions returning the corresponding motion vector, 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 $\mathbf{f}_p^{l_1}$.

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

$$\mathbf{f}'_c = \{p_i + \mathbf{f}_c.r \times \mathbf{m}_p(p_i) | p_i \in \mathbf{f}_p^{l_2}\}. \tag{4.9}$$

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*:

$$\mathbf{c}_c = \{c_j + \mathbf{f}_c.r \times \mathbf{m}_c(c_j) | c_j \in \mathbf{c}\} \tag{4.10}$$

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 sorted 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 iterated through all $c_j \in \mathbf{c}_c$, and computed 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 integrate it into 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. (4.8). Last, with the updated $\mathbf{m}_c$, we generate $\mathbf{f}'_c$ by:

$$\mathbf{f}'_c = \{p_i + \mathbf{f}_c.r \times \mathbf{m}_c(c_j) \Big| p_i \in \mathbf{f}_p^{l_2}\}. \tag{4.11}$$

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 an alternative 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:

$$e_{i,k} = 1/\Delta_v(p_i, c_k), \tag{4.12}$$

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:

$$\mathbf{f}'_c = \left\{ p_i + \mathbf{f}_c.r \times \frac{\sum_{c_k \in \mathbf{n}_j \cup \{c_j\}} e_{i,k} \mathbf{m}_c(c_k)}{\sum_{c_k \in \mathbf{n}_j \cup \{c_j\}} e_{i,k}} \middle| p_i \in \mathbf{f}_p^{l_2} \right\}. \tag{4.13}$$

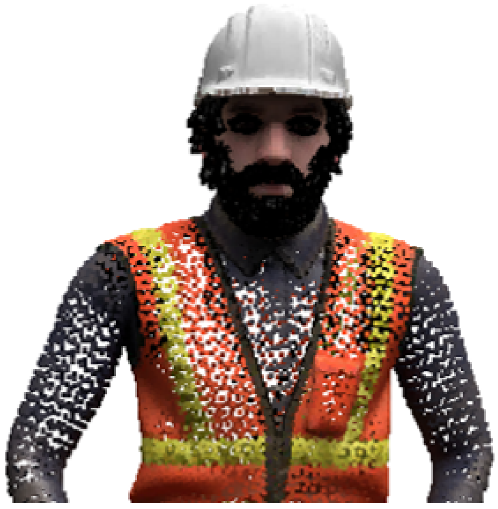
4.5.2 Comparison

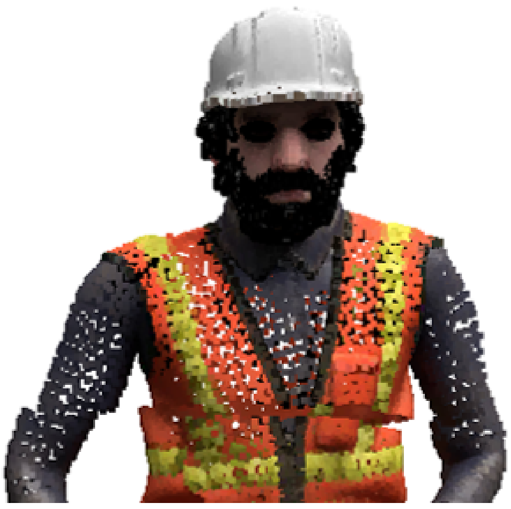
To compare PP, CP, and NP, we let $l_1, l_2 = 1$. We conceal the most challenging middle frame in different numbers of consecutive frame drops in $\{1, 3, 5, 7, 9\}$. We predict $\mathbf{f}'_c$ by running PP, CP, and NP on ground-truth motion vectors. Other settings are consistent with those used in Secs. 4.3 and 4.4.

Fig. 4.7 presents sample 2D-rendered views of a dancing man sequence [39]. Among the three algorithms, we observe that CP (Fig. 4.7(b)) moves points in each cube toward a single direction, leading to irregular surfaces on the avatar's body and arms, while NP (Fig. 4.7(c)) mitigates this issue. To better quantify their performance, Fig. 4.8 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 0.01 m, which is negligible (0.6%) to the height of the avatar (1.75 m). Similarly, the temporal smoothness difference is also small (figures not shown for brevity). Fig. 4.9 reports the 2D quality of rendered views (the figure on SSIM is omitted), 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 quite different in end-to-end settings.

Next, we render the predicted frame into 2D images and compare the PSNR in Fig. 4.9 (we omit the figure for SSIM due to space limit). The results show that PP performs better than CP, with a maximum difference of 1.25 in PSNR and 0.06 in SSIM. The visual quality of CP and NP in 2D images did not show much difference, with a maximum difference of 0.26 in PSNR and 0.009 in SSIM.

PP iterates through all points in Eq. (4.9) and has a time complexity of $\mathcal{O}(|\mathbf{f}|)$. As for CP, the motion vectors $\mathbf{m}_c$ are firstly applied to the cubes $\mathbf{c}$, whose time complexity is $\mathcal{O}(|\mathbf{c}|)$. Subsequently, the cubes in $\mathbf{c}$ are sorted according to their motion quantified by $|\mathbf{m}_c|$, which takes $\mathcal{O}(|\mathbf{c}| \log |\mathbf{c}|)$. For each cube $c_j \in \mathbf{c}$, we compute l_m from its six adjacent cubes in $\mathcal{O}(1)$ using their coordinates and l_c, so the complexity for all cubes is $\mathcal{O}(|\mathbf{c}|)$. The final step recalculates the motions of the updated cubes and applies them to points in $\mathbf{f}$, whose time complexity is $\mathcal{O}(|\mathbf{f}|)$. Thus, the overall time complexity of CP is


(a)


(b)

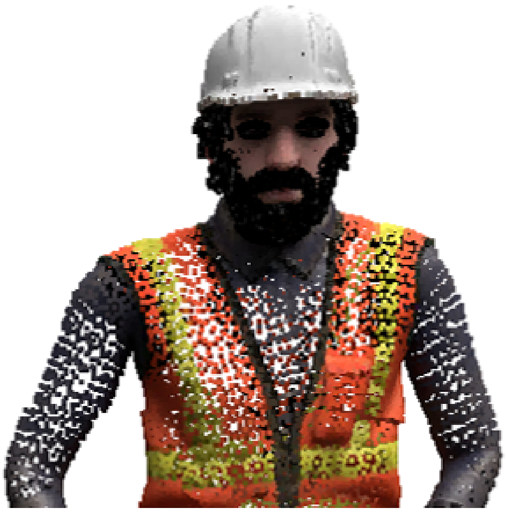
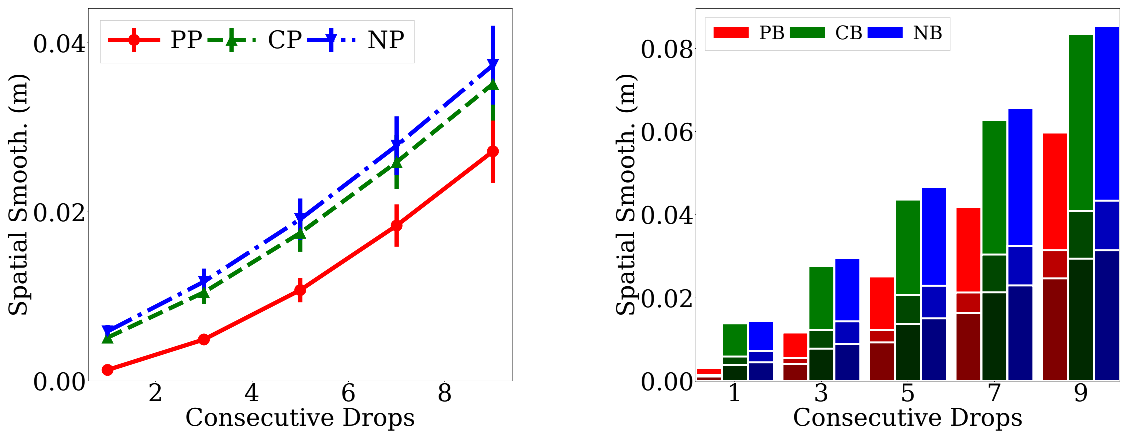

(c)

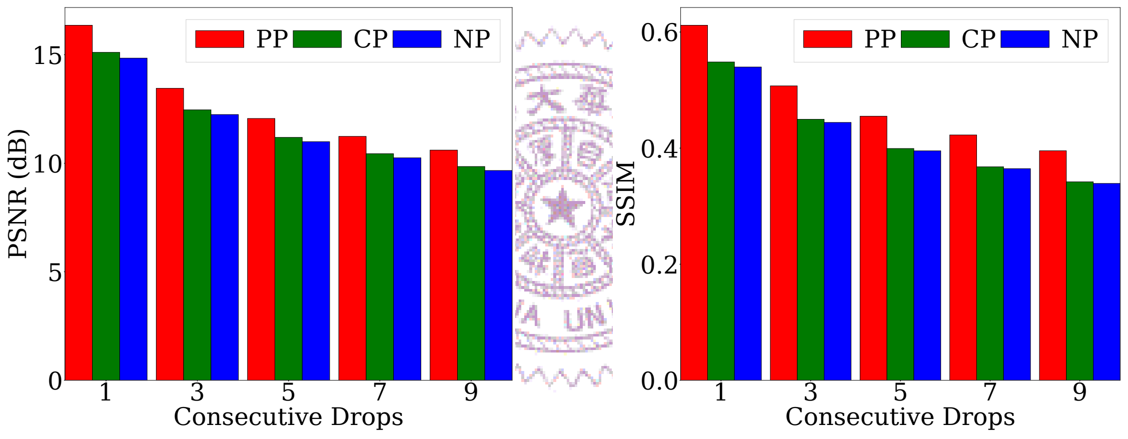
Figure 4.7: Sample 2D-rendered views from different prediction algorithms: (a) PP, (b) CP, and (c) NP.

$\mathcal{O}(|\mathbf{c}| \log |\mathbf{c}| + |\mathbf{f}|)$. For NP, initially, 26 neighbors are identified for each cube, yielding a time complexity of $\mathcal{O}(|\mathbf{c}|)$. Then, for every point in $\mathbf{f}_p^{l_2}$, the calculation of Eq. (4.13) takes a time complexity of $\mathcal{O}(|\mathbf{f}|)$. Hence, the overall time complexity of NP is $\mathcal{O}(|\mathbf{c}| + |\mathbf{f}|)$. 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 $|\mathbf{c}|$.



(a) (b)

Figure 4.8: Smoothness comparison of prediction algorithms: (a) average and (b) 95, 75, 50 percentile.



(a) (b)

Figure 4.9: 2D quality comparison of prediction algorithms: (a) PSNR and (b) SSIM.

4.6 Pre-Processing and Post-Processing Stages

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 spatial correlation among filled voxels [52].

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.

A simple yet effective surface reconstruction method, i.e., screened Poisson Surface

Reconstruction (PSR) [24], is chosen 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 [26]) 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 [14].

To understand its potential, we compared the concealed frame quality with and without PSR on the first 200 frames of MPEG Red&Blk sequence [6] with 3 consecutive frame drops using AQR, CM, and NB as middle stages, and performed normal estimation on $\mathbf{f}'_c$ using Open3D [57] for PSR. The PSR improves PSNR, SSIM, and VMAF by 0.59 dB, 1.42%, and 0.96 on average. To visualize the difference, Fig. 4.10 provides an illustration of the rendered point clouds with and without PSR, demonstrating its effectiveness on crack filling. PSR consumes considerable computational resources, even in situations where the normals are pre-computed and are part of the attributes. Alleviating this overhead is our future work.

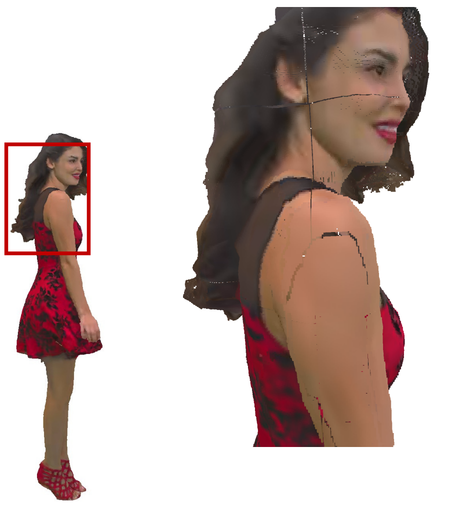
To evaluate the tradeoff between 2D quality and running time, we tested different downsample ratios $s \in \{0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ by adjusting l_2 so that $s \times |\mathbf{f}_p| = |\mathbf{f}_p^{l_2}|$. We used PSR for surface refinement. We picked 20 random frames from Red&Blk [6] and evaluated them for 10 rounds to acquire the average quality and running time.

Observation. Table 4.1 lists the average of PSNR, SSIM, and VMAF among the first 200 frames with three consecutive drops. The dynamic point clouds with surface refinement achieve 0.59 dB, 1.42%, and 0.96 improvements in PSNR, SSIM, and VMAF. Fig. 4.10 provides an illustation of the rendered point cloud with and without surface refinement. We can observe in Fig. 4.10(a) that large cracks appear among the surface of point cloud. As discussed in Sec. 4.3, the surface distortion is caused by an unsatisfied matching method. However, as shown in Fig. 4.10(b), the proposed surface refinement method effectively alleviates the distortion, resulting in a smoother surface.

Table 4.1: Average 2D Quality among 200 Point Cloud Frames

	PSNR (dB)	SSIM (%)	VMAF
Without PSR	22.06	64.92	41.31
With PSR	22.65	66.34	42.27

Fig. 4.11 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


(a)

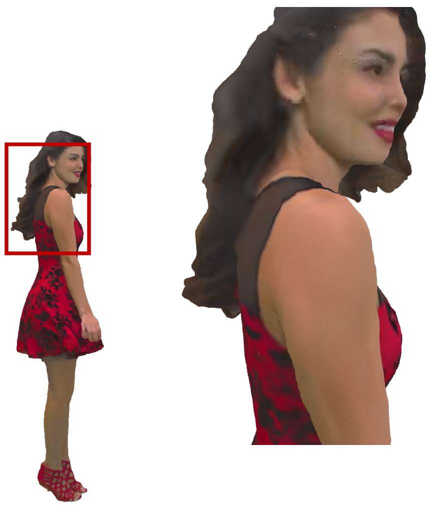

(b)

Figure 4.10: Visual results of sample point clouds: (a) without and (b) with PSR.

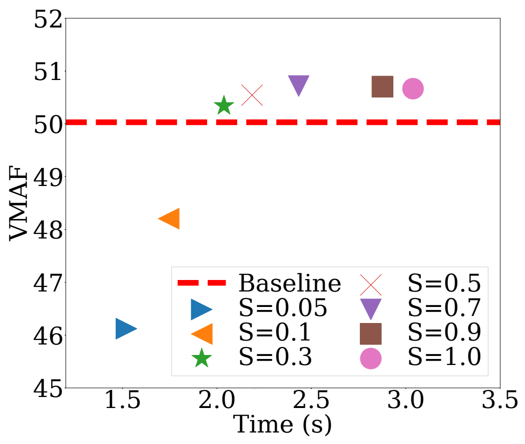


Figure 4.11: Tradeoff between quality and running time of the post-processing stage with PSR as the surface refinement algorithm.

$s > 0.3$ 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 the point cloud, thus improving the quality of concealed point cloud. Fig. 4.11 shows that optimal downsample ratios may be chosen based on the application requirements.

Chapter 5

Experiments

In this chapter, we first summarize and combine the proposed stage algorithms into different pipelines with different targets. Next, we evaluate the end-to-end performance of our pipeline over baseline 3DFC in objective experiments, and also conduct a user study to compare against state-of-the-art learning-based methods [2, 55].

5.1 Our Proposed Representative Pipelines

Table 5.1: Alternative Algorithms in Individual Stages

Stage / Algorithm	Pre-Proc.	Matching	M. Est.	Pred.	Post-Proc.
Adopted	Downs. [52]	NN (baseline)	-	-	Ups. + PSR [24]
Proposed	-	QP, AQR	PM, CM	PP, CP, NP	-

We have implemented error concealment framework and algorithms in C++ and Python using libraries including PCL [37] and Eigen [10]. Our implementation [16] includes the alternative algorithms proposed in Secs. 4.3–4.6, as summarized in Table 5.1. In addition, we concatenate the proposed algorithms in different stages to form representative error concealment pipelines for diverse usage scenarios, as listed in Table 5.2. More details are given below.

- **Fast (F)**: We combined QR with PP for minimum time complexity. We opted not to use 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 5.2: Our Representative Error Concealment Pipelines

Stage / **Pipeline**	**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 5.3: Dynamic 3D Point Cloud Sequences

Sequence / **Property**	**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]

5.2 Setup

We have found that the simplistic 2DFC suffers from significant quality degradation, as briefly illustrated in Fig. 1.1(b) and detailed in the preliminary conference version [18,19] of this thesis. For brevity, in this thesis, 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 5.3 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 [35] as the codec with the default settings defined in Common Test Conditions (CTC) [34]. 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.

In this thesis, we first use the Gilbert-Elliot (G-E) model [13] to simulate 5%, 7.5%, 10%, 12.5%, and 15% packet loss rate on geometry drop. Furthermore, to better understand the impact under more severe conditions, we simulate drops of one to five frames in our subsequent experiments.

5.3 Results

Fig. 5.1–5.5 show the overall quality of the concealed frame while simulating over G-E Model under 5%, 7.5%, 10%, 12.5%, and 15% packet loss. Next, Fig. 5.6–5.10 show the overall quality of concealed frame under 1, 2, 3, 4, and 5 consecutive frame drops. Fig. 5.11–5.16 show the improvement of our pipeline over baseline 3DFC under 5%, 7.5%, 10%, 12.5%, and 15% packet loss. Finally, Fig. 5.17–5.22 show the improvement of our pipeline over baseline 3DFC under 1, 2, 3, 4, and 5 consecutive frame drops. We pick some of the representative result figures for the following observation.

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. 5.6 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.

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. 5.19 and 5.22 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. 5.6–5.22. Fig. 5.23 depicts the same frame concealed by Q+ from Queen sequence in which the avatar is holding a pad in her hands. We observe that before applying PSR, the pad in the avatar's hand retains a square shape. However, once PSR is applied, the pad becomes uneven. This is because the pad is thin, making it challenging to obtain a precise normal estimation, significantly impacting the PSR results. Consequently, the quality of the Queen sequence with PSR is significantly decreased. Fig. 5.6 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 look at why Q+ performs badly with Queen reveals that it is largely caused by imperfect normal estimation done by Open3D [57]. For example, normals of the tablet held by the avatar are clearly ill-estimated (by Open3D), which significantly affects the performance of PSR. Furthermore, Figs. 5.19 and 5.22 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 5.4 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

Table 5.4: Per-frame Running Time Break-down from Loot/LongDress/Dancer: Second (Fraction)

Stage / Pipeline	Pre-Proc.	Matching	M. Est.	Pred.	Post-Proc.	Total
Fast (F)	-	**QR** 1.05/0.96/2.66 (93%/93%/92%)	**PM** 0.02/0.02/0.03 (2%/2%/1%)	**PP** 0.06/0.06/0.19 (6%/6%/6%)	-	1.13/1.03/2.88
Balance (B)	-	**NN** 0.99/1.34/14.64 (24%/30%/54%)	**CM** 1.13/1.11/4.52 (27%/25%/17%)	**CP** 1.98/1.98/7.93 (48%/45%/29%)	-	4.11/4.44/27.08
Quality (Q)	**Downs.(l_1, 1)** 0.04/0.01/0.15 (<1%/<1%/<1%)	**AQR** 1.94/2.17/18.81 (18%/18%/34%)	**CM** 0.01/0.01/0.03 (<1%/<1%/<1%)	**NP** 8.69/9.49/36.10 (81%/81%/66%)	-	10.67/11.71/55.09
Quality+ (Q+)	**Downs.(l_2, 1)** 0.11/0.11/0.36 (2%/2%/2%)	**AQR** 1.77/1.89/9.53 (36%/36%/43%)	**CM** 0.01/0.01/0.03 (<1%/<1%/<1%)	**NP** 2.80/3.07/11.48 (58%/58%/52%)	**Ups. + PSR** 0.17/0.18/0.52 (4%/3%/2%)	4.86/5.27/21.92

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 [57]. Table 5.4 also gives the fraction of running times consumed by individual stages, shedding some light on the strategies for optimizing 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 shorter running time.

5.4 User Study

This section starts with a discussion on the design of our user study, which evaluates our pipeline in comparison to 3DFC and learning-based approaches [2, 55]. Following this, we present the outcomes and delve into a discussion, quantifying insights gathered from user feedback.

5.4.1 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 [21].

We considered two tests with different baseline algorithms: (i) 3DFC and (ii) state-of-the-art learning-based interpolation algorithms [2, 55][1][2]. 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, 55] 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, instead of using a scaler of 1 - 5 as BT.500-13 standard [21] suggest, we simplified the question to a binary choice by asking 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 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.

5.4.2 Results

Fig. 5.24 presents the sample winning rates: 3DFC test results from the best- and worst-performing sequences in Figs. 5.24(a) and 5.24(b), respectively; average 3DFC results in Fig. 5.24(c); and overall results against the state-of-the-art learning-based interpolation algorithms in Fig. 5.24(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 made the following observations. First, *our representative pipelines resulted in better temporal smoothness.* Fig. 5.24(c) shows that compared to 3DFC, our Q, B, and F pipelines achieved average winning rates of 91.43%, 75.24%, and 57.14%, respectively. Fig. 5.24(b) reveals that even with the worst-performing sequence, Q and B still resulted in 50+% winning rates. Fig. 5.24(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, 55].

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

[2]Because Zeng et al. [55] only take point clouds with 1024 point as input, we divide each point cloud frame into multiple sub point clouds with $\leq$ 1024 points before sending them into the neural network.

Second, *our representative pipelines lead to better spatial smoothness compared to the state-of-the-art learning-based interpolation algorithms.* Fig. 5.24(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. 5.24(a) reveals that the B, Q, and Q+ pipelines achieve as high as 80% winning rate. However, Fig. 5.24(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. 5.24(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. 5.25(a) gives a sample concealed frame 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.

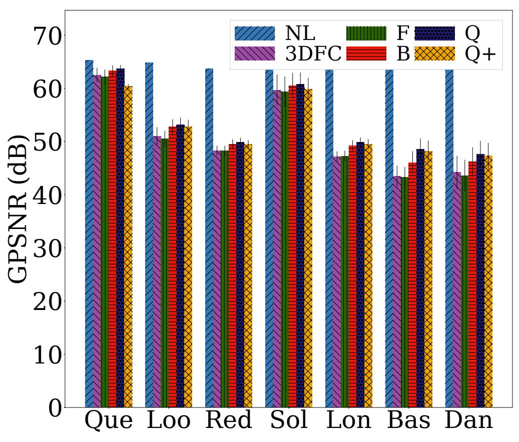
Last, our representative pipeline Q subjectively outperforms the state-of-the-art learning-based interpolation algorithms. Fig. 5.25(b) shows a sample frame concealed from a state-of-the-art learning-based interpolation algorithm under a single frame drop. Compared to Fig. 5.25(a), it is not hard to see why subjects prefer our pipeline Q. In fact, Fig. 5.24(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 5.5 compares the mean (maximum) see-through pixels. Across the seven sequences, we found that Akhtar et al. [2] and Zeng et al. [55] suffer from on average 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, 55] 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. [55] took 102.22–182.20 times. *Hence, the state-of-the-art learning-based interpolation algorithm [2, 55] cannot solve our considered error concealment problem.*

In summary, our best-quality pipeline Q outperforms 3DFC in temporal smoothness

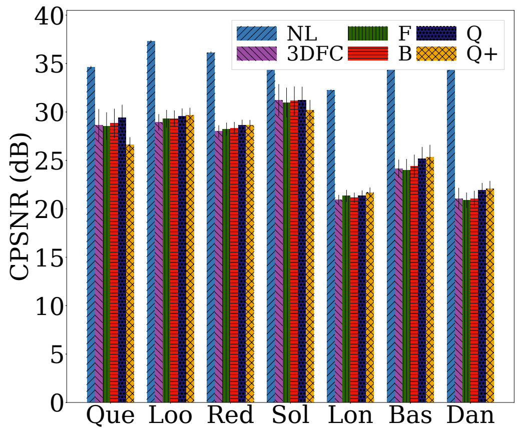
Table 5.5: Number of Continuous See-through Pixels: Mean (Max)

Sequence / Algorithm	**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. [55]	18 (764)	55 (1118)	89 (1417)	56 (1010)	79 (1086)	1325 (29097)	286 (5321)

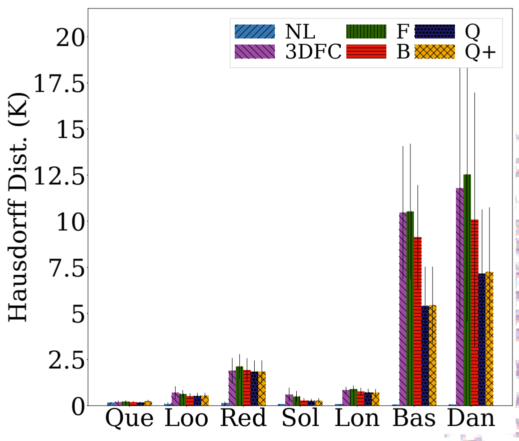
(91.43% average winning rate) and preference (64.76%); it also outperforms the state-of-the-art learning-based interpolation algorithms [2, 55] in temporal smoothness (97.14%), spatial smoothness (100%), and preference (100%), as well as running time ($< 1/11.70 = 8.55\%$).



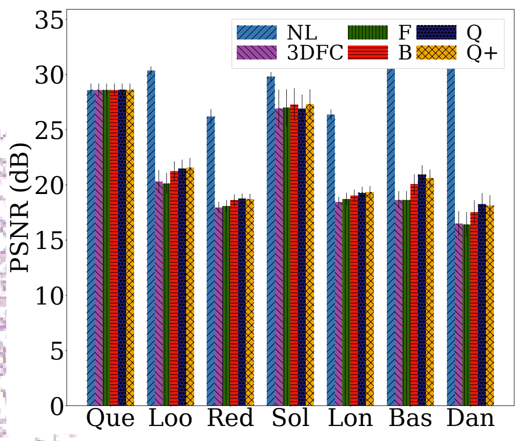
(a)



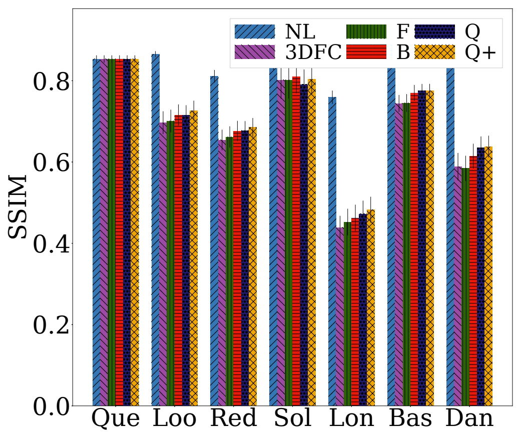
(b)



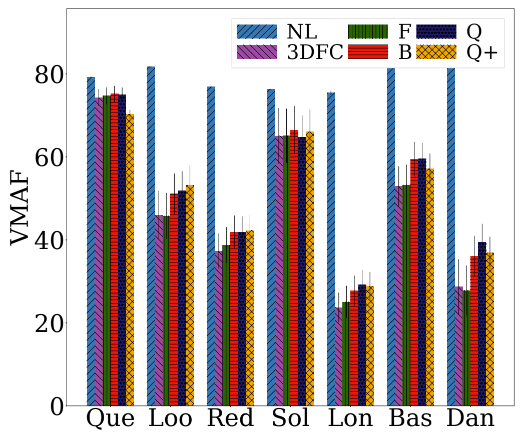
(c)



(d)

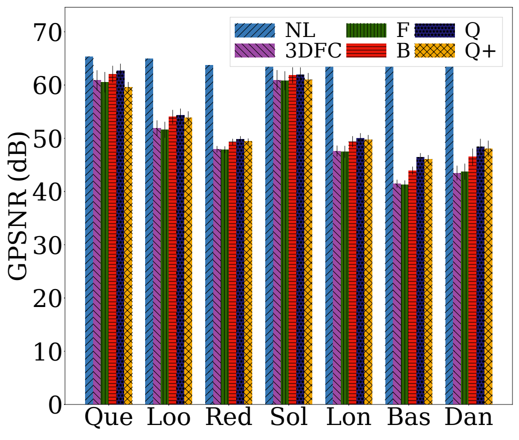


(e)

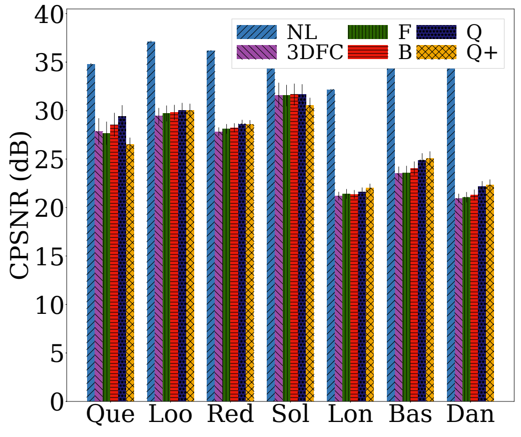


(f)

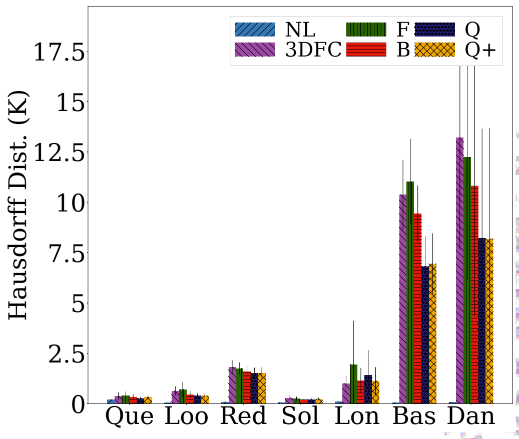
Figure 5.1: Overall quality of concealed frames from individual sequences with 5% packet loss rate: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



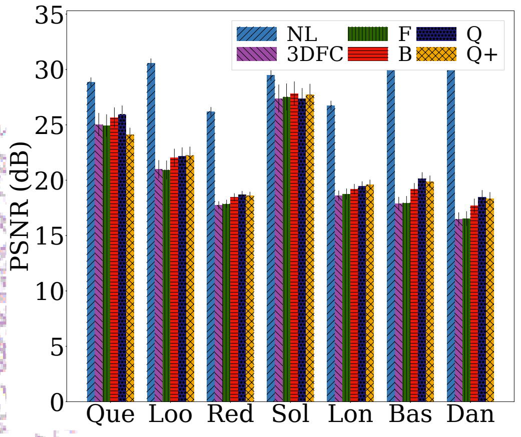
(a)



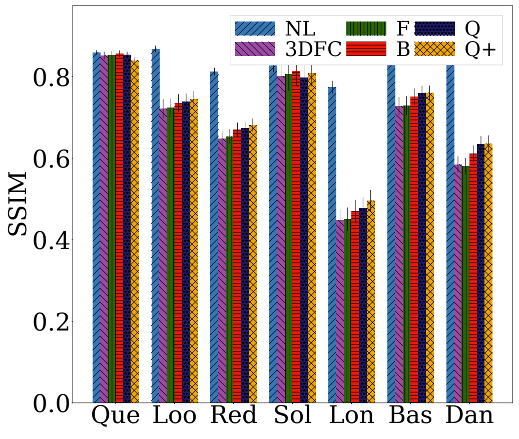
(b)



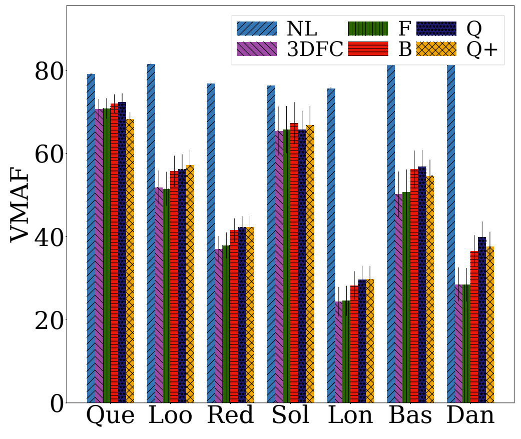
(c)



(d)

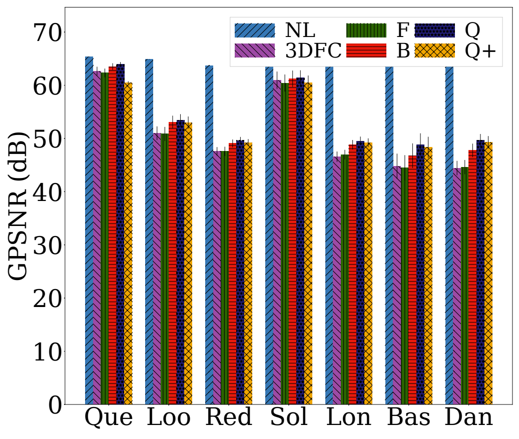


(e)

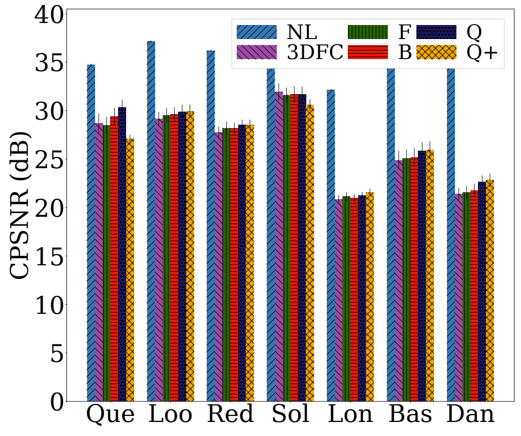


(f)

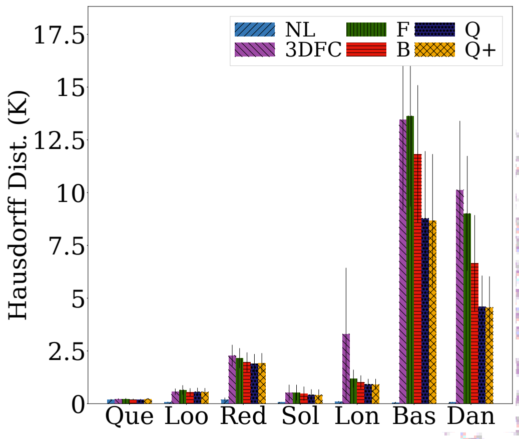
Figure 5.2: Overall quality of concealed frames from individual sequences with 7.5% packet loss rate: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



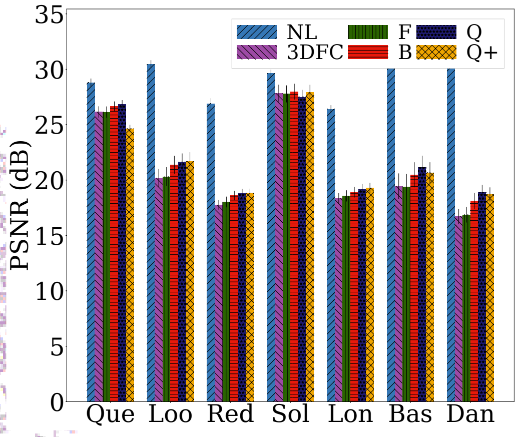
(a)



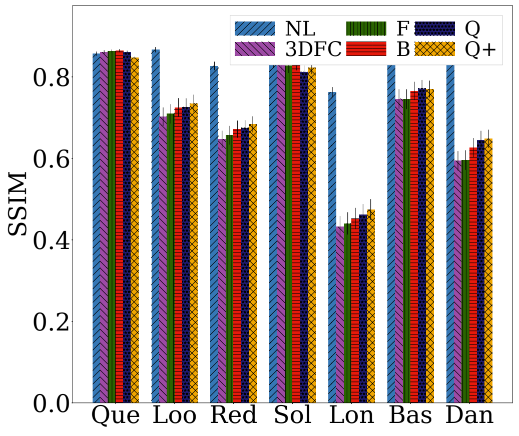
(b)



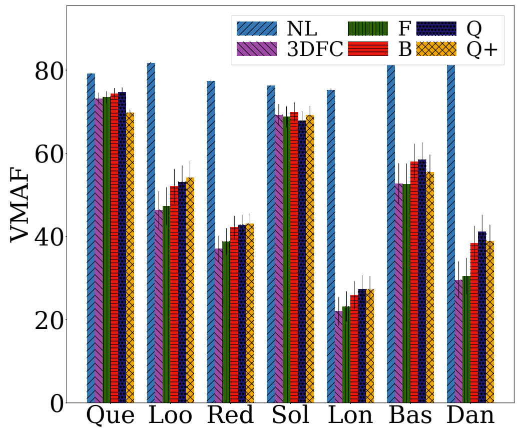
(c)



(d)

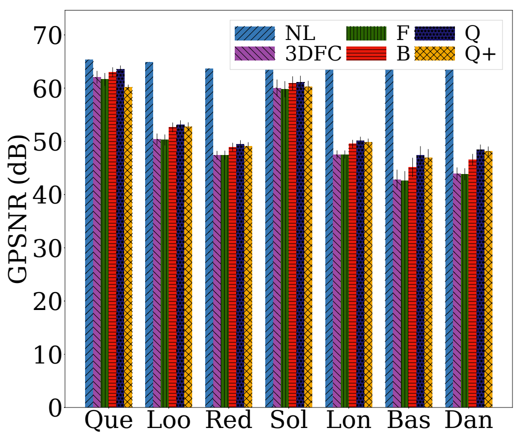


(e)

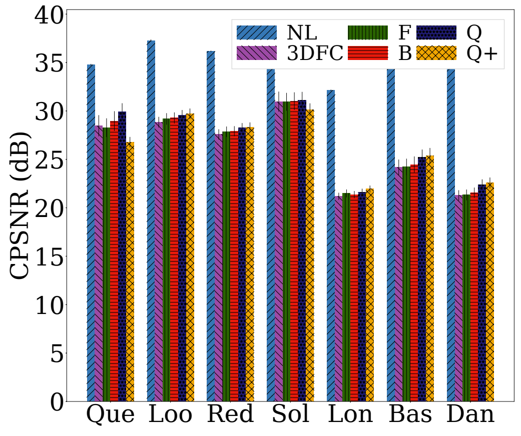


(f)

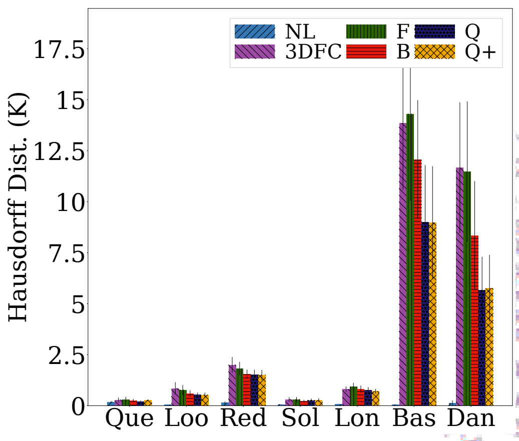
Figure 5.3: Overall quality of concealed frames from individual sequences with 10% packet loss rate: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



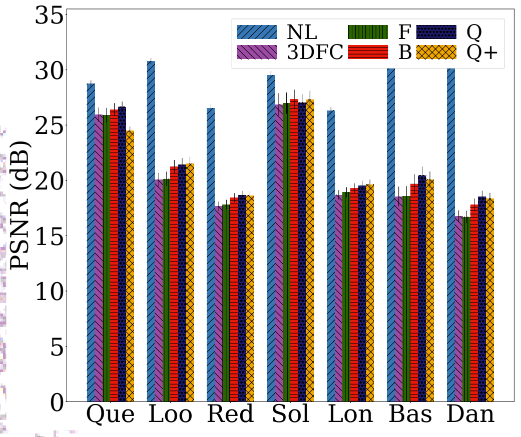
(a)



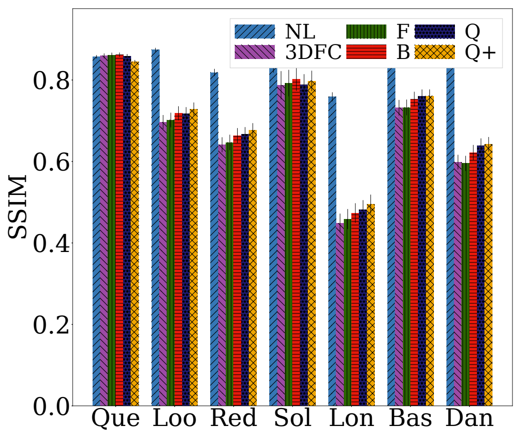
(b)



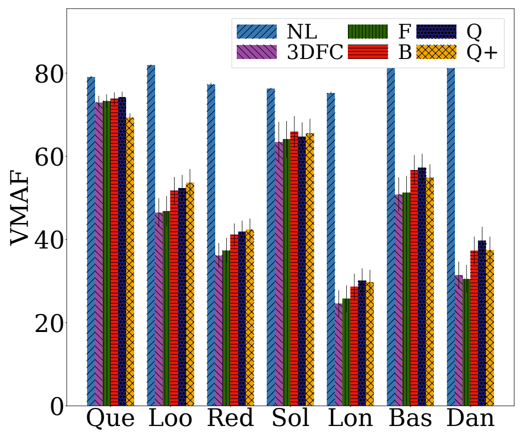
(c)



(d)

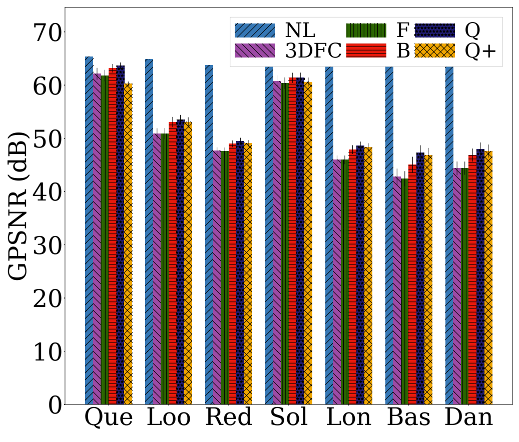


(e)

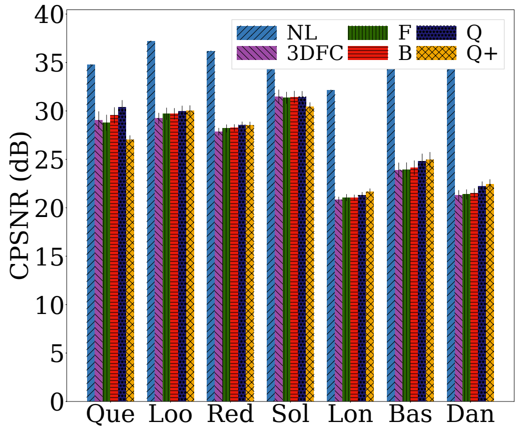


(f)

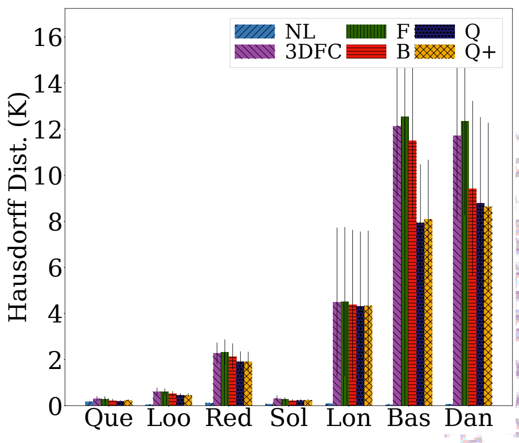
Figure 5.4: Overall quality of concealed frames from individual sequences with 12.5% packet loss rate: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



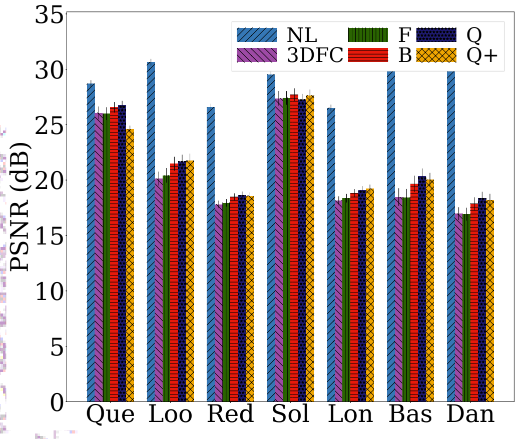
(a)



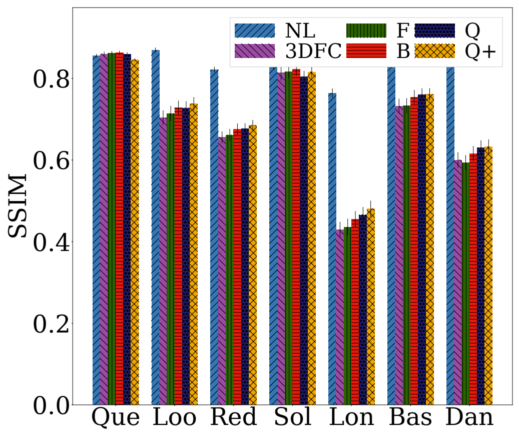
(b)



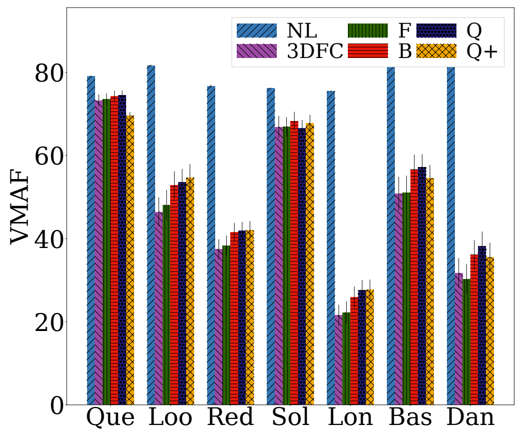
(c)



(d)

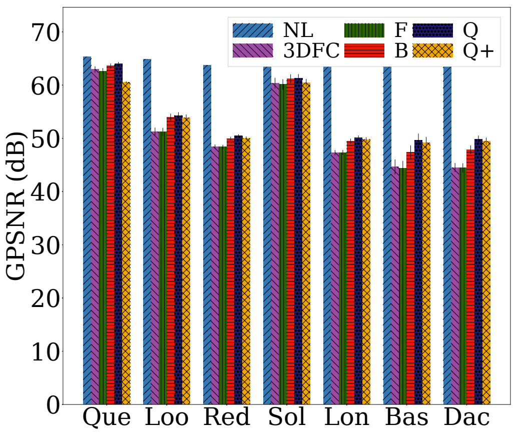


(e)

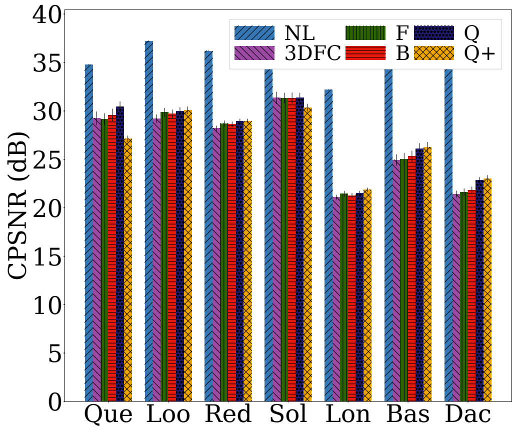


(f)

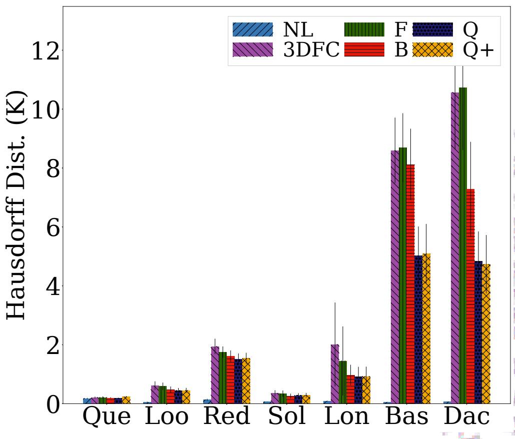
Figure 5.5: Overall quality of concealed frames from individual sequences with 15% packet loss rate: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



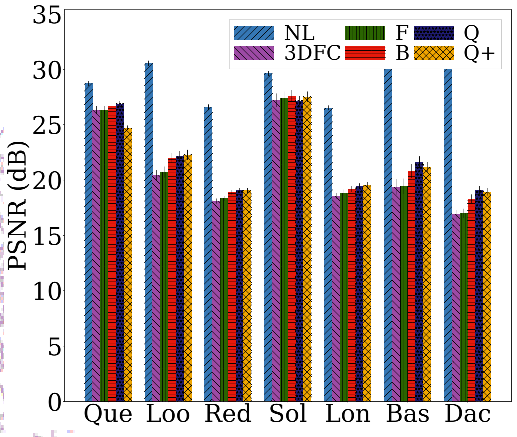
(a)



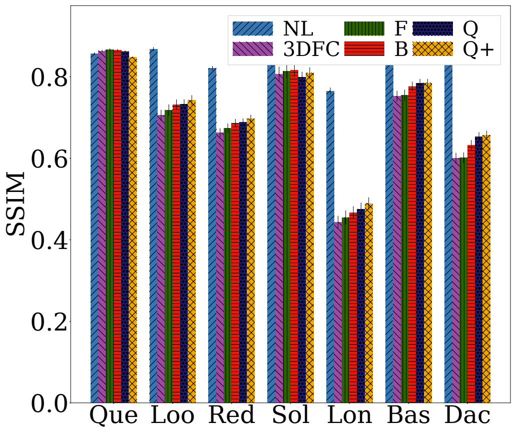
(b)



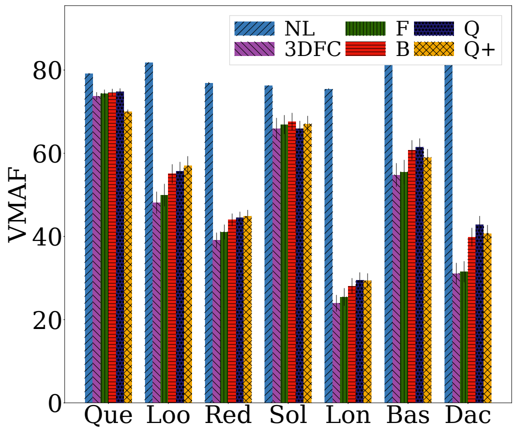
(c)



(d)

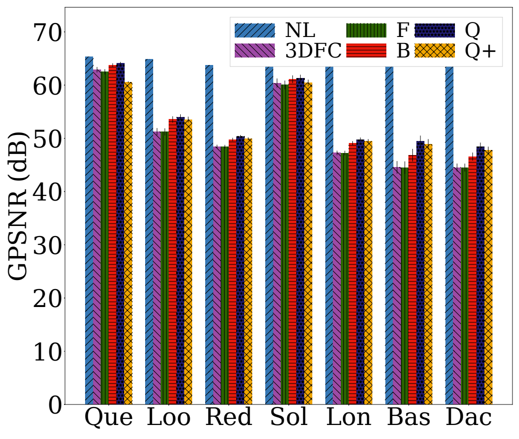


(e)

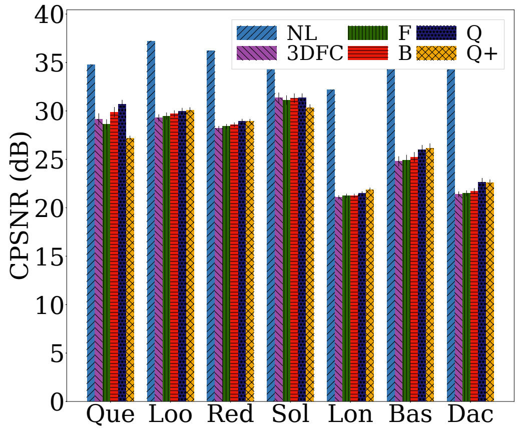


(f)

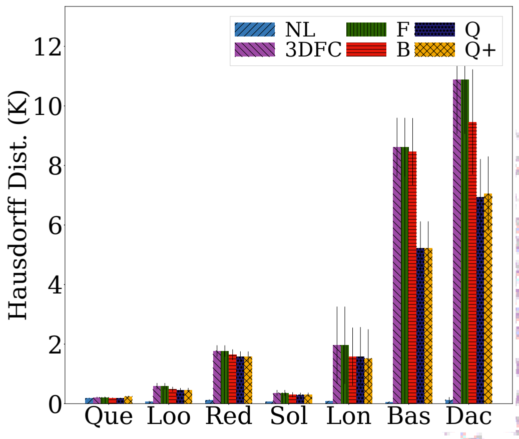
Figure 5.6: Overall quality of concealed frames from individual sequences with a single frame drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



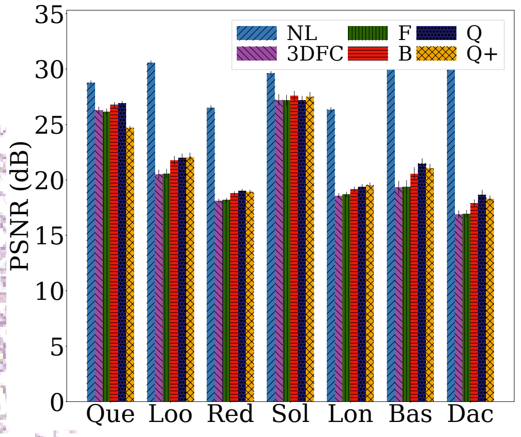
(a)



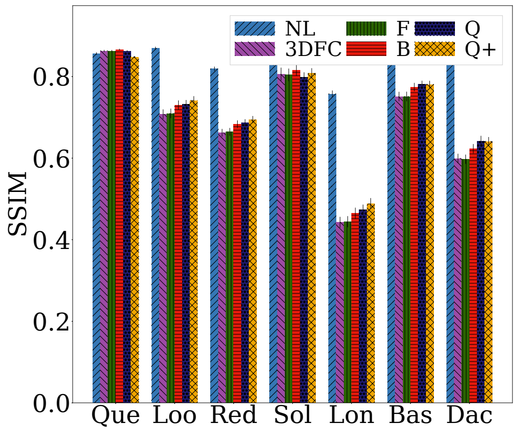
(b)



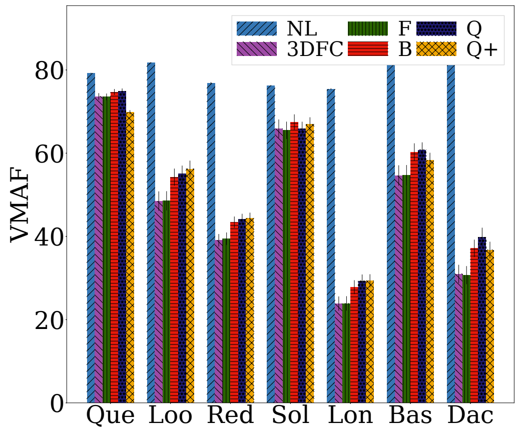
(c)



(d)

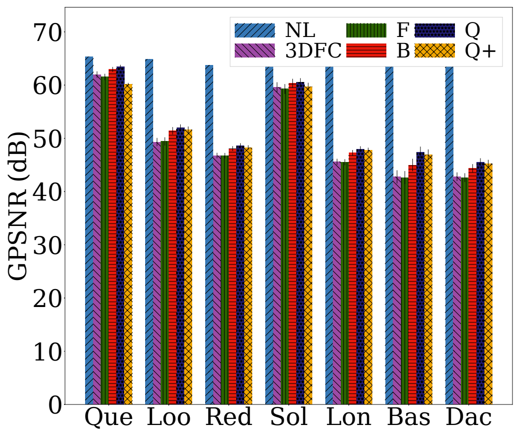


(e)

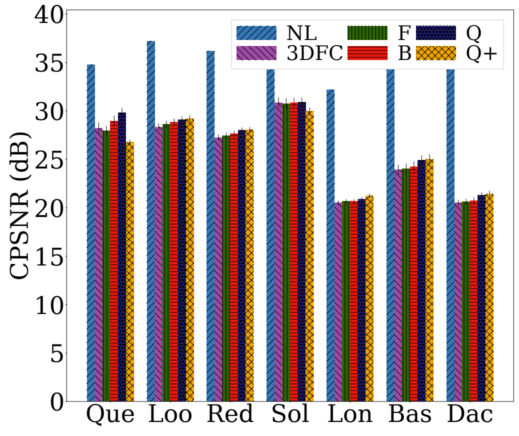


(f)

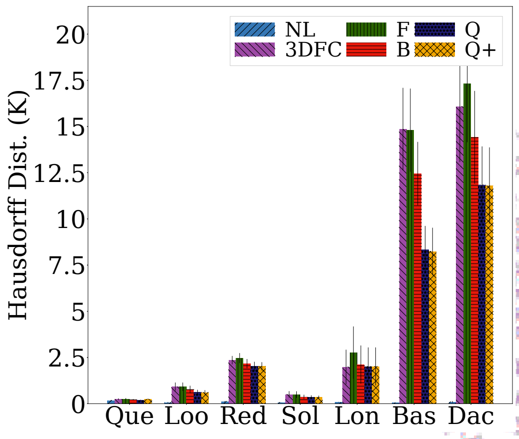
Figure 5.7: Overall quality of concealed frames from individual sequences with a two frames drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



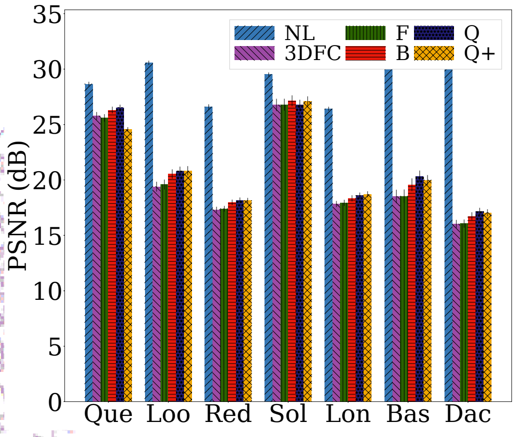
(a)



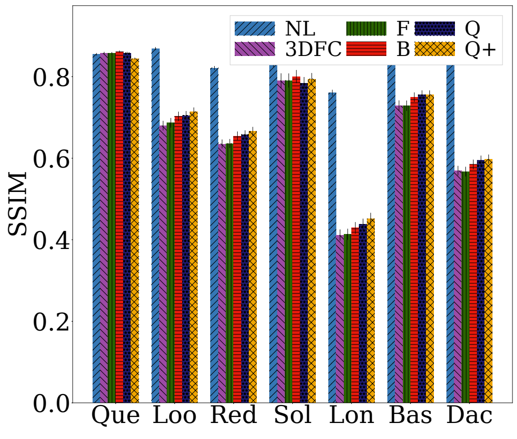
(b)



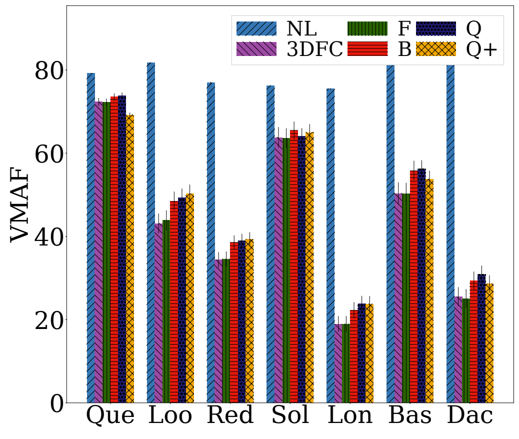
(c)



(d)

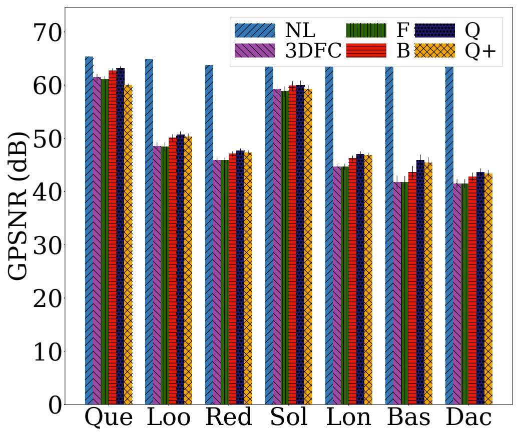


(e)

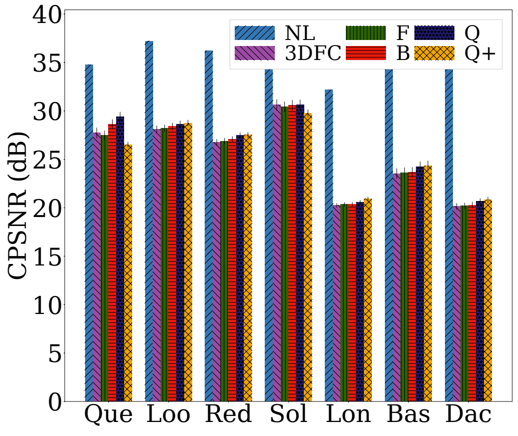


(f)

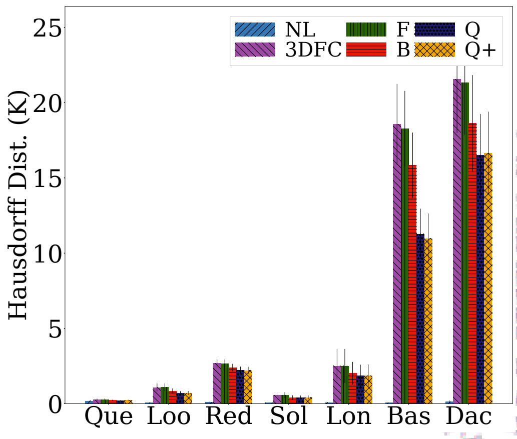
Figure 5.8: Overall quality of concealed frames from individual sequences with a three frames drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



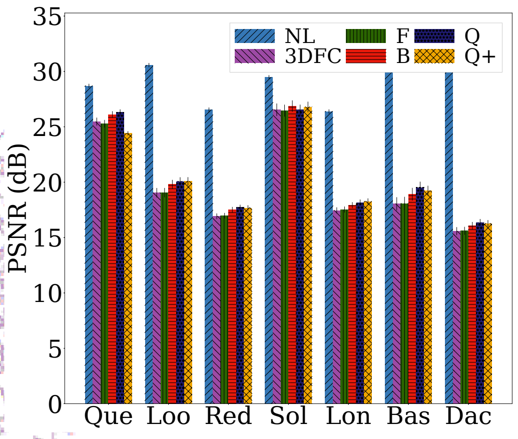
(a)



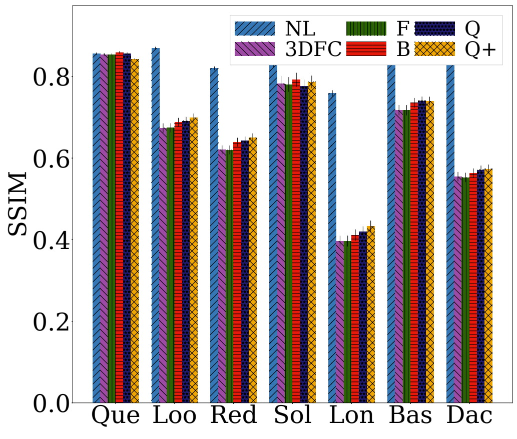
(b)



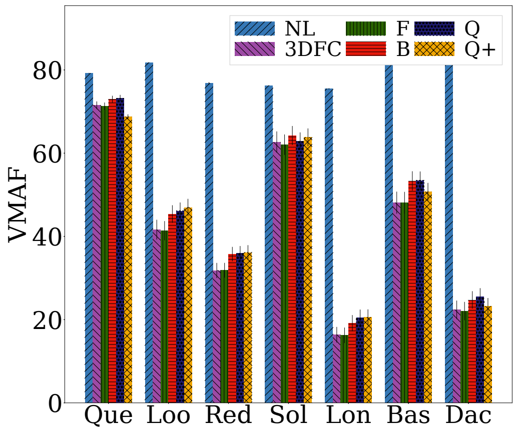
(c)



(d)

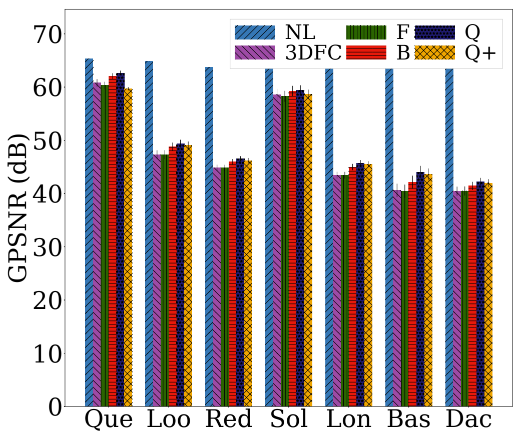


(e)

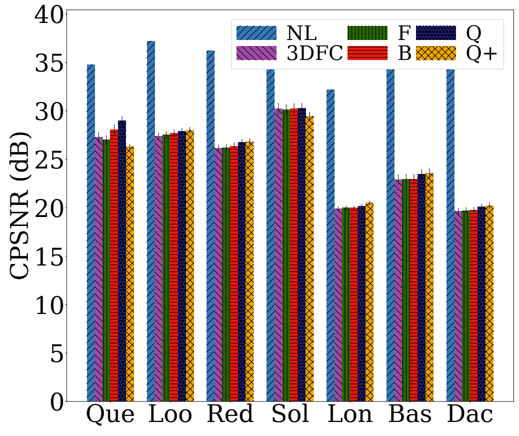


(f)

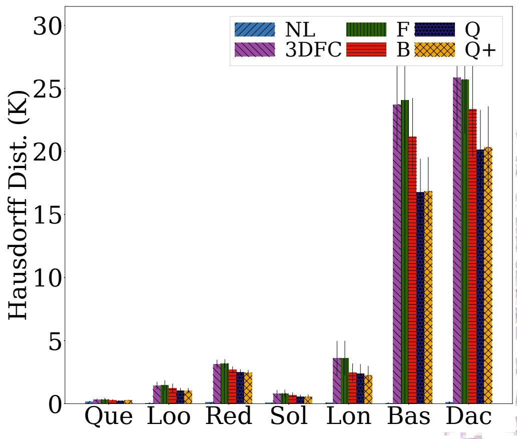
Figure 5.9: Overall quality of concealed frames from individual sequences with a four frames drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



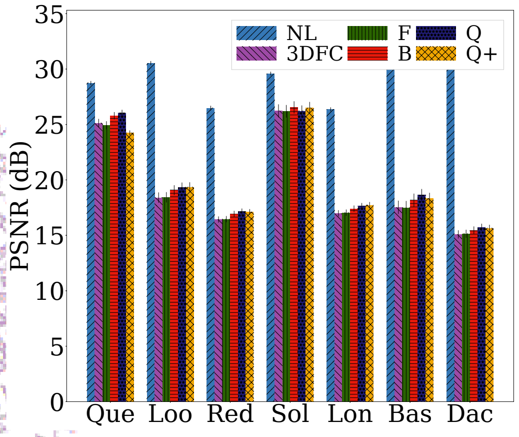
(a)



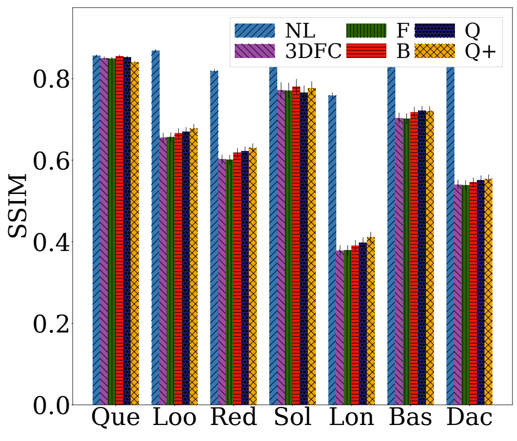
(b)



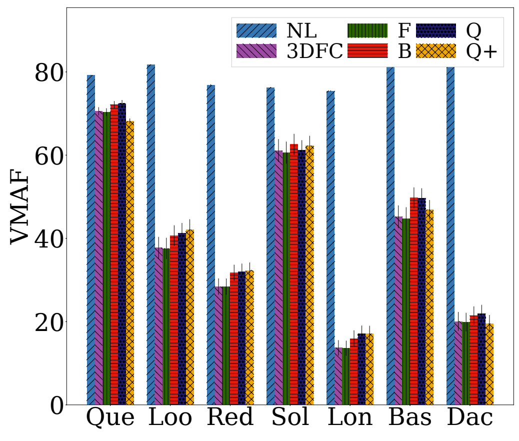
(c)



(d)

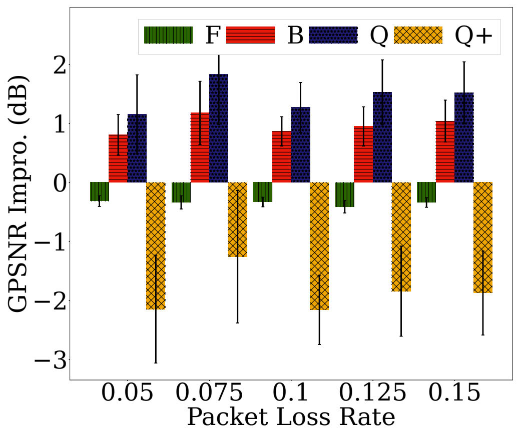


(e)

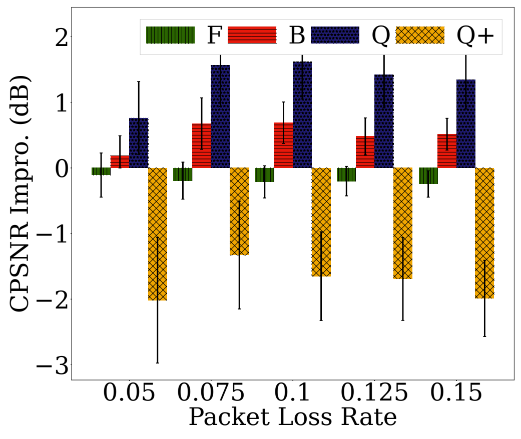


(f)

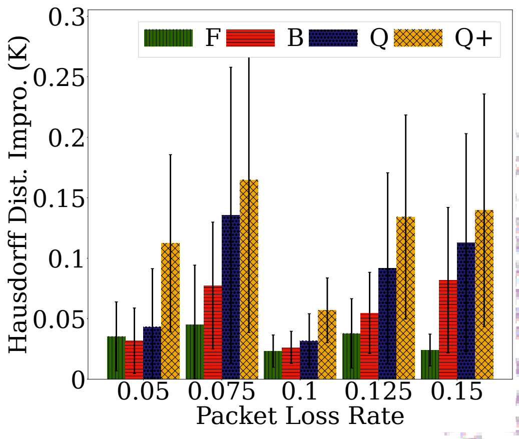
Figure 5.10: Overall quality of concealed frames from individual sequences with a five frames drop: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



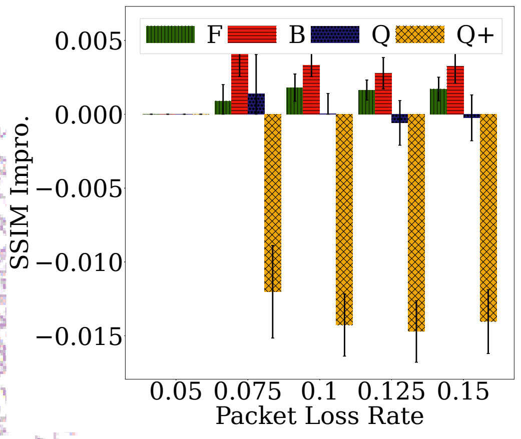
(a)



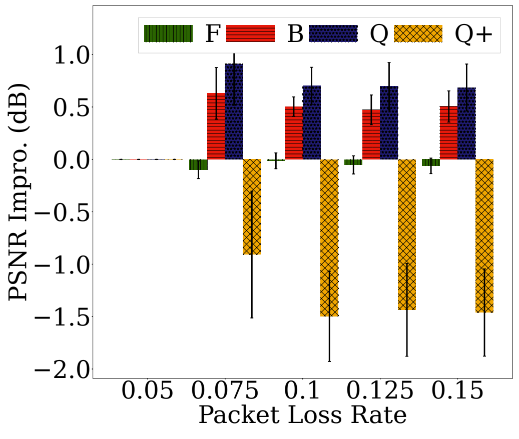
(b)



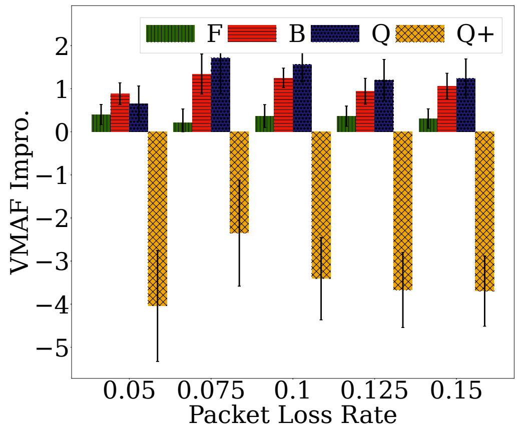
(c)



(d)

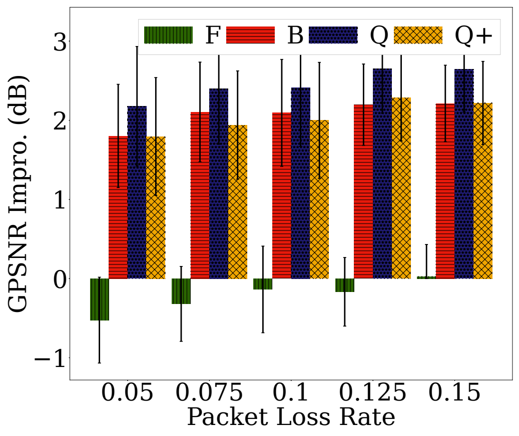


(e)

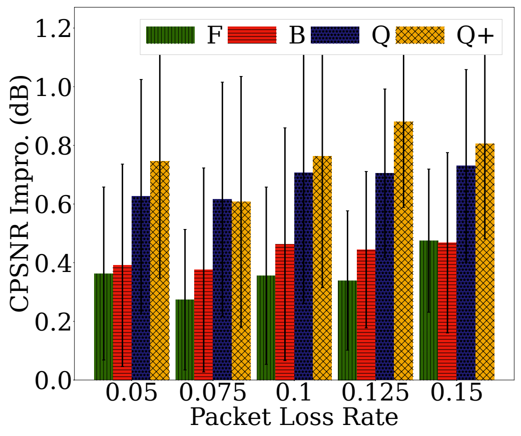


(f)

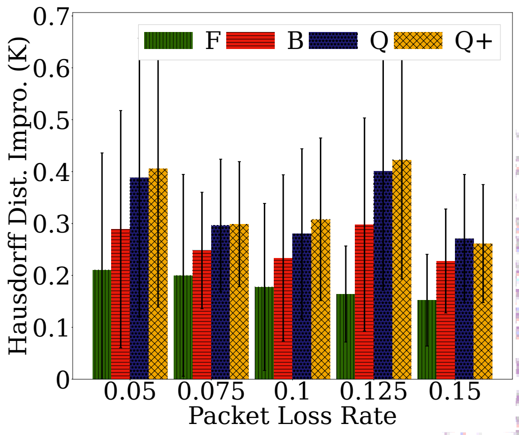
Figure 5.11: Overall quality improvement over 3DFC for different packet loss: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Queen are shown.



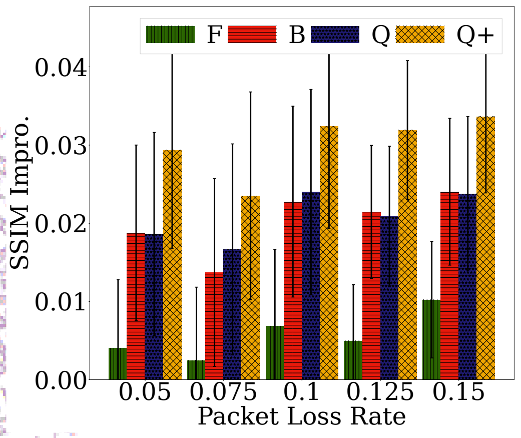
(a)



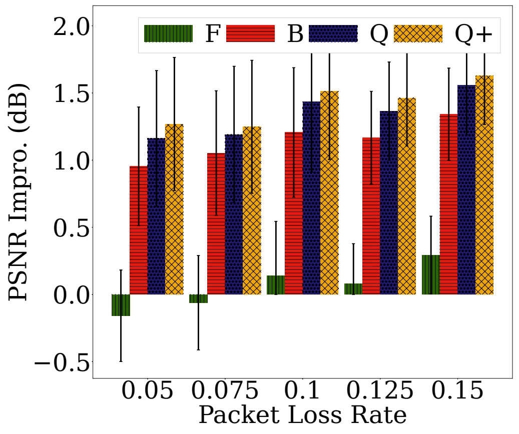
(b)



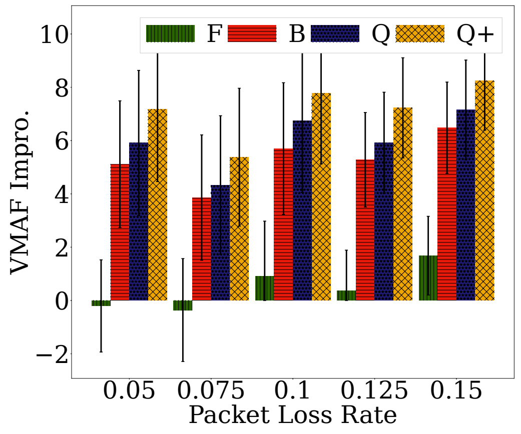
(c)



(d)

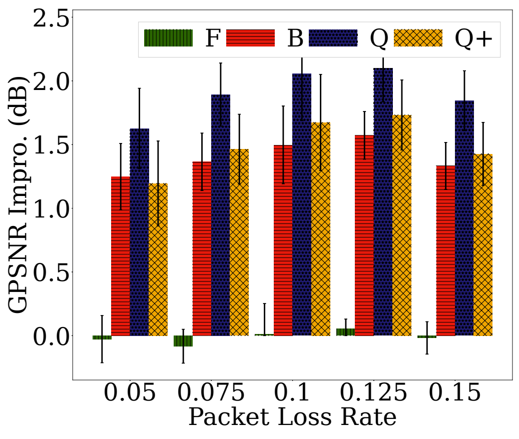


(e)

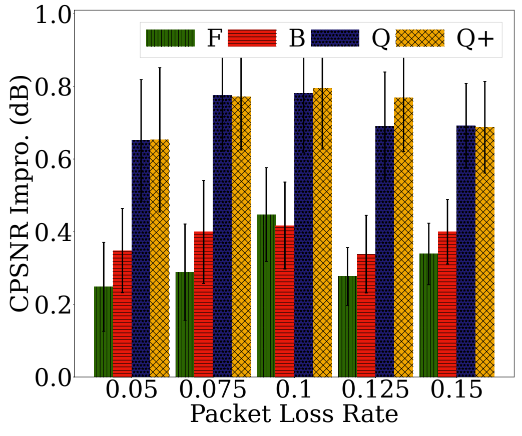


(f)

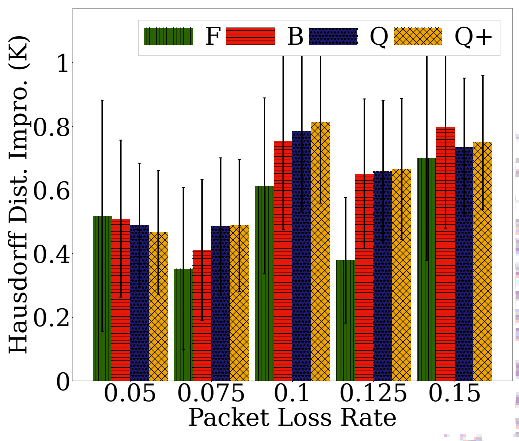
Figure 5.12: Overall quality improvement over 3DFC for different packet loss: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Loot are shown.



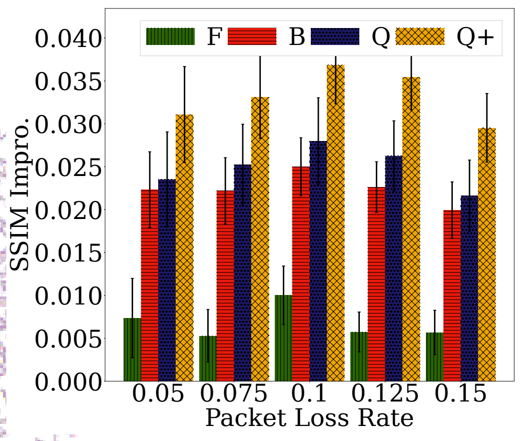
(a)



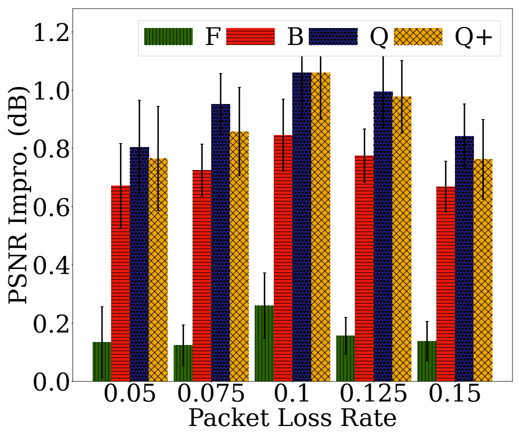
(b)



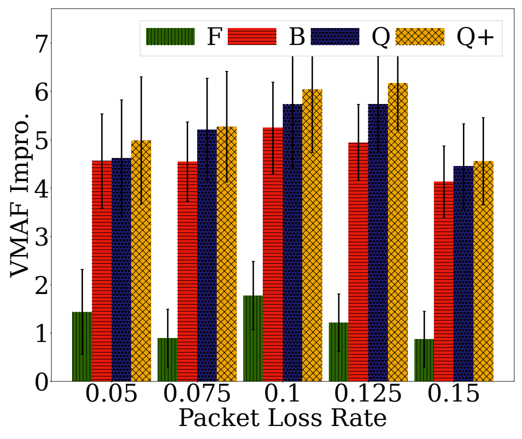
(c)



(d)

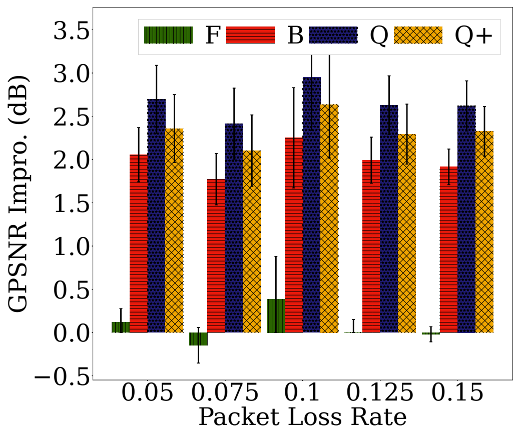


(e)

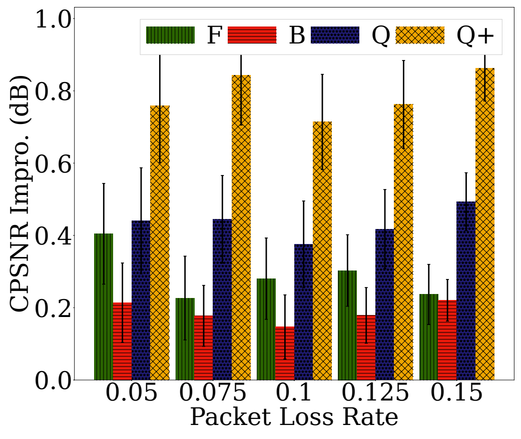


(f)

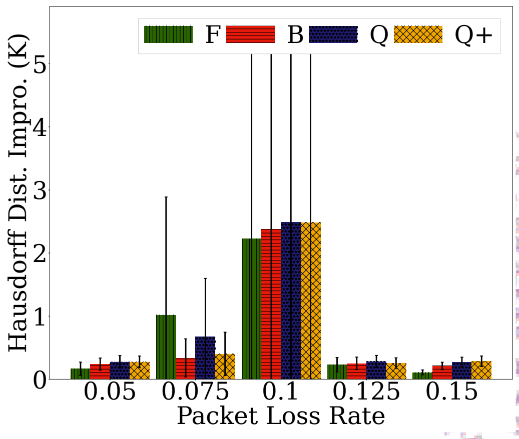
Figure 5.13: Overall quality improvement over 3DFC for different packet loss: (a) GP-SNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Red&Blk are shown.



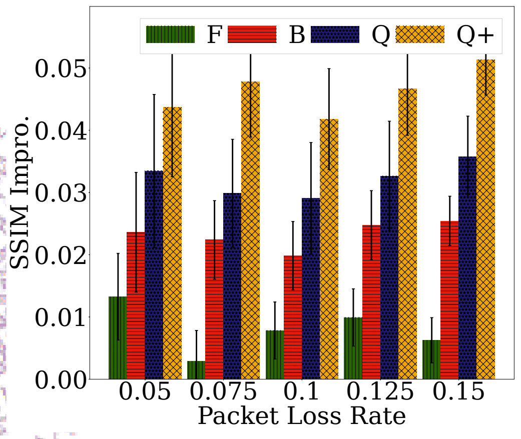
(a)



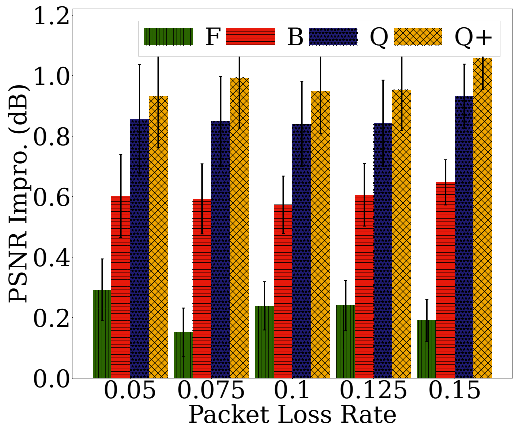
(b)



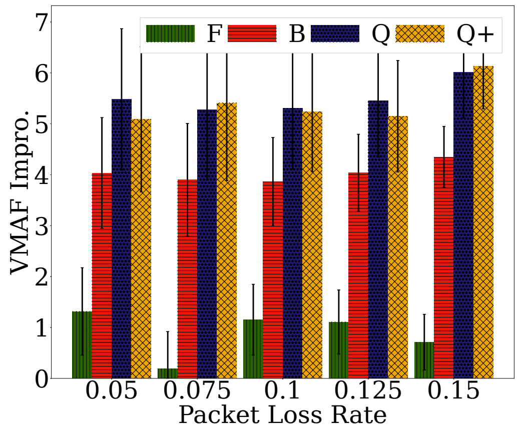
(c)



(d)

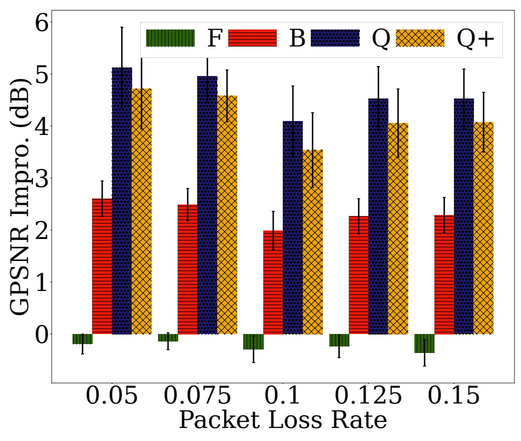


(e)

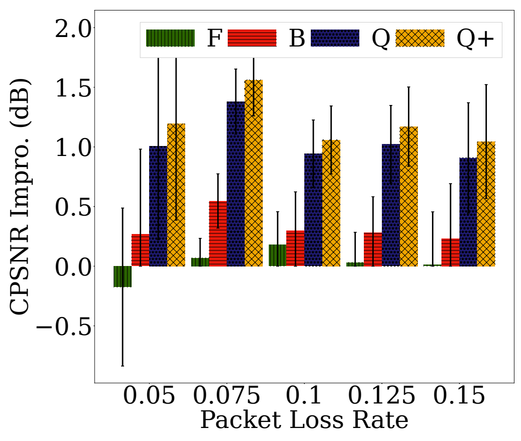


(f)

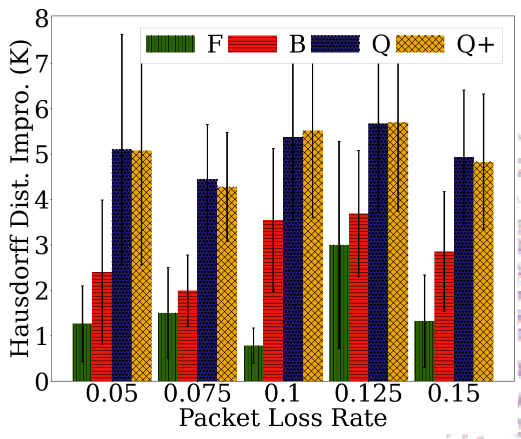
Figure 5.14: Overall quality improvement over 3DFC for different packet loss: (a) GP-SNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Longdress are shown.



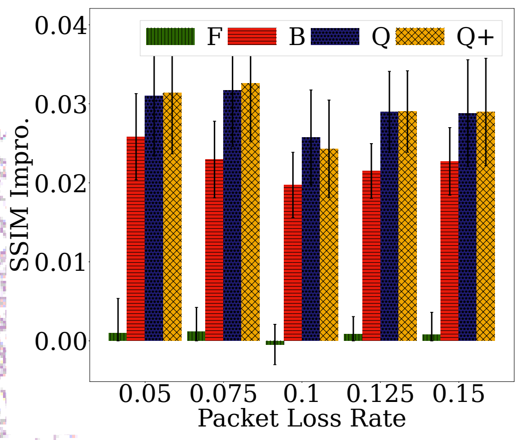
(a)



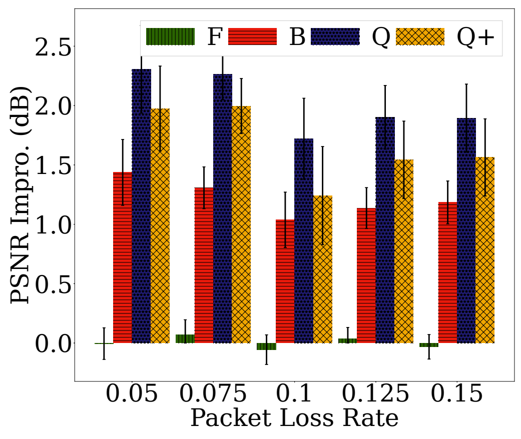
(b)



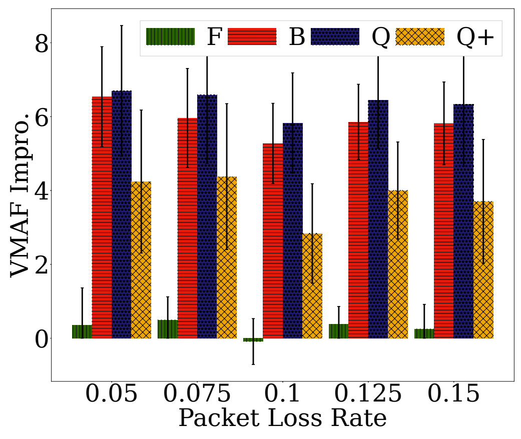
(c)



(d)

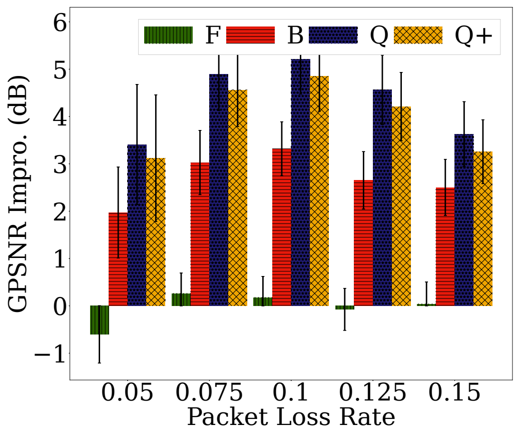


(e)

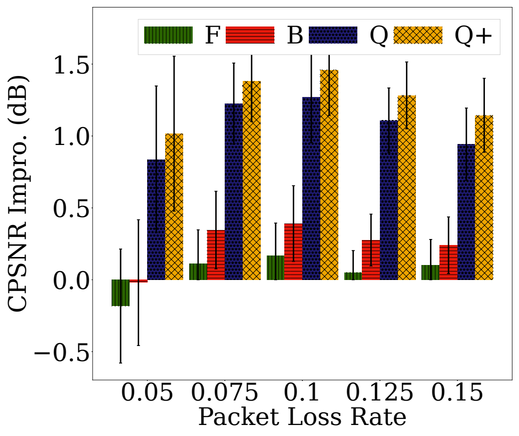


(f)

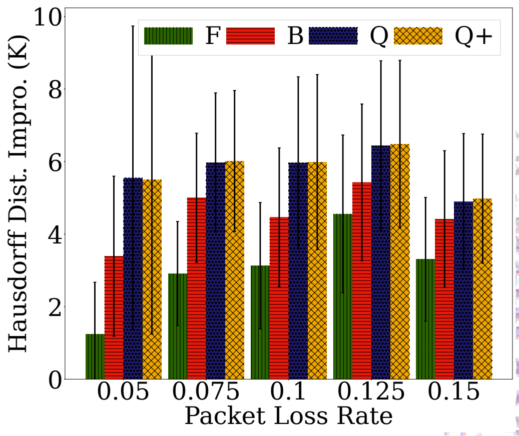
Figure 5.15: Overall quality improvement over 3DFC for different packet loss: (a) GP-SNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Basketball are shown.



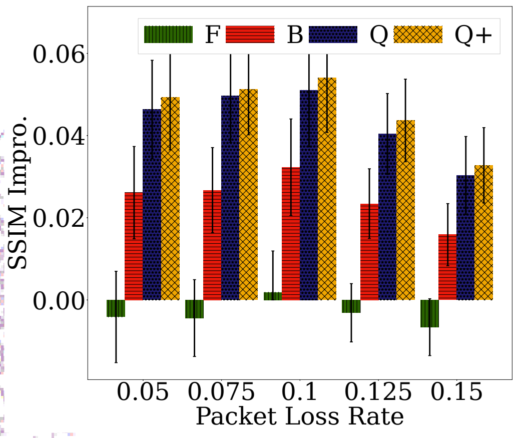
(a)



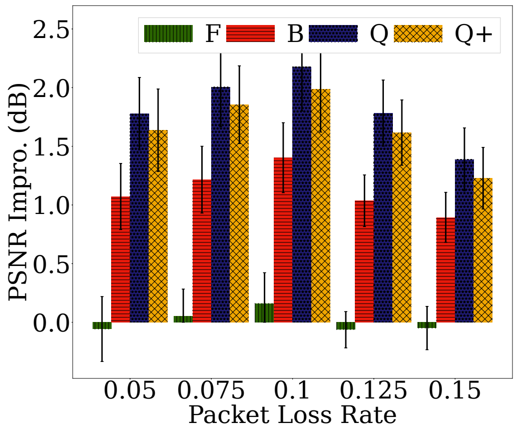
(b)



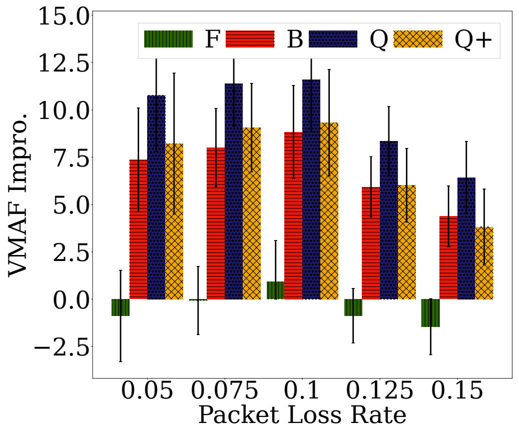
(c)



(d)

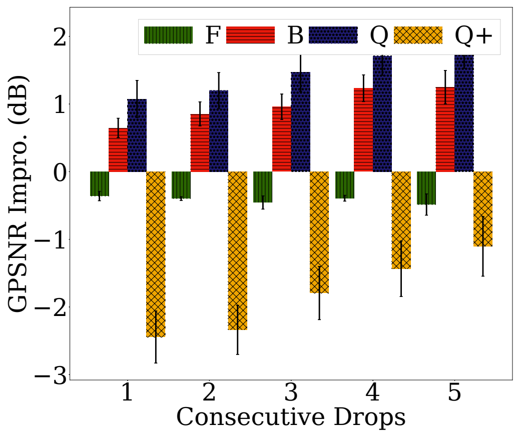


(e)

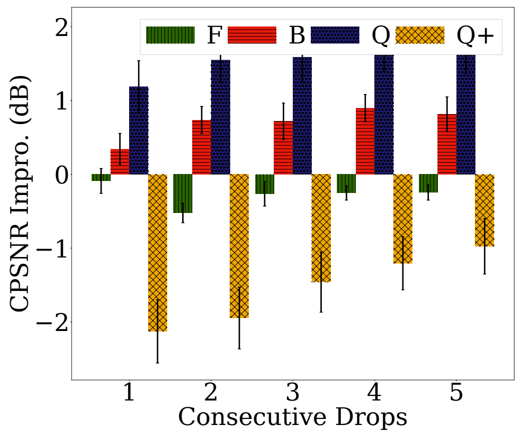


(f)

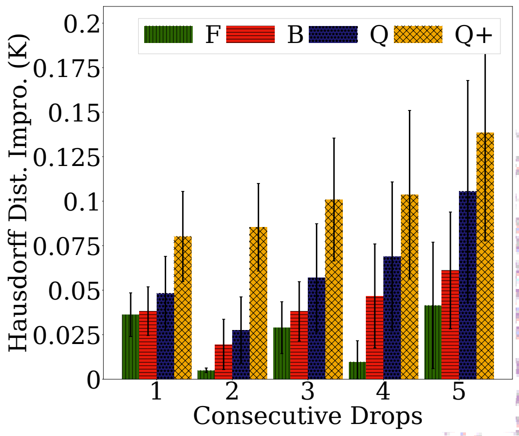
Figure 5.16: Overall quality improvement over 3DFC for different packet loss: (a) GP-SNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Dancer are shown.



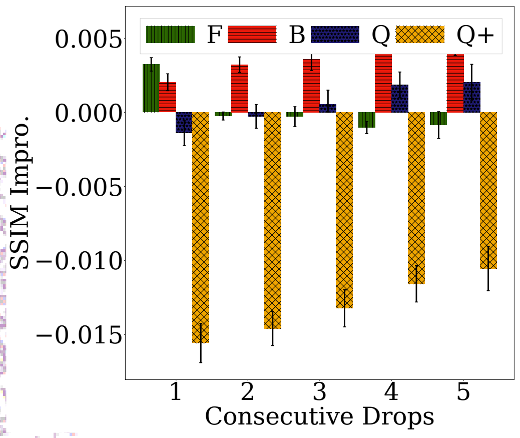
(a)



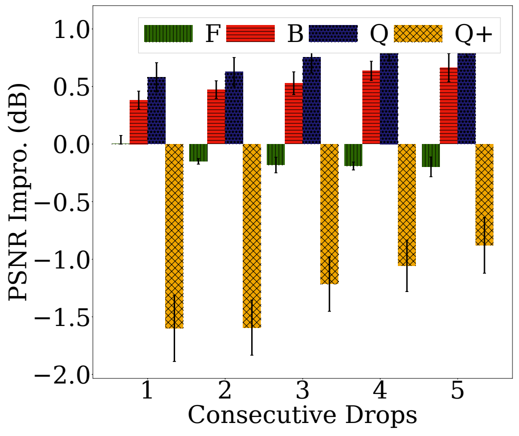
(b)



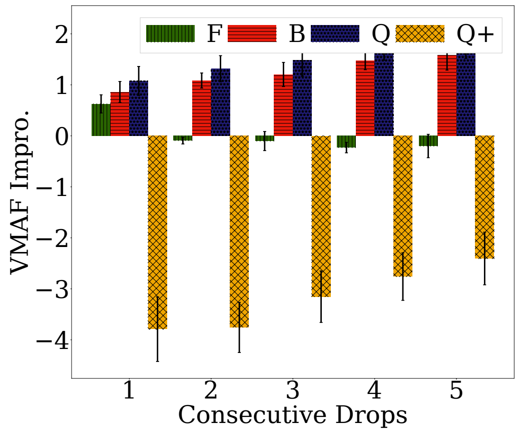
(c)



(d)

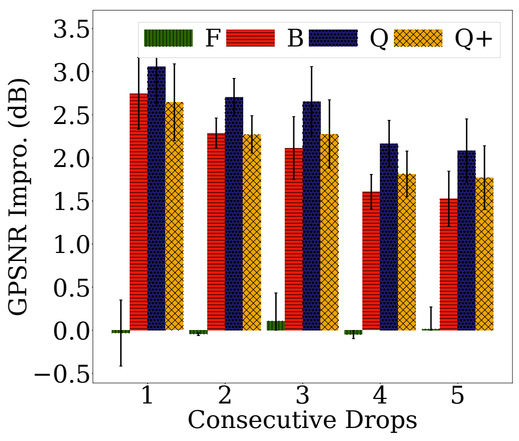


(e)

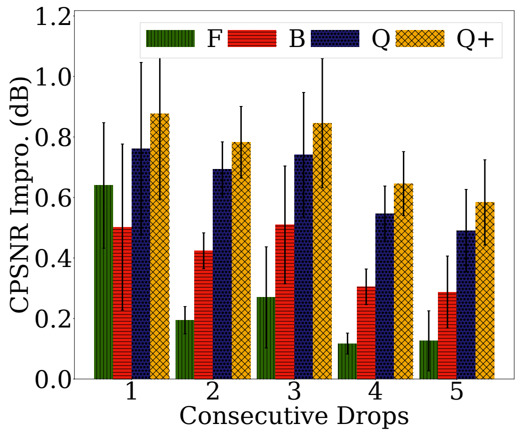


(f)

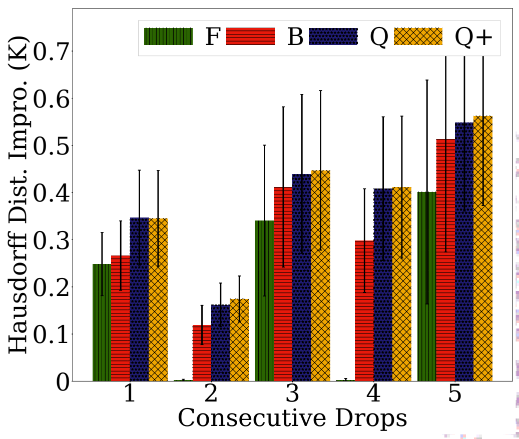
Figure 5.17: Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Queen are shown.



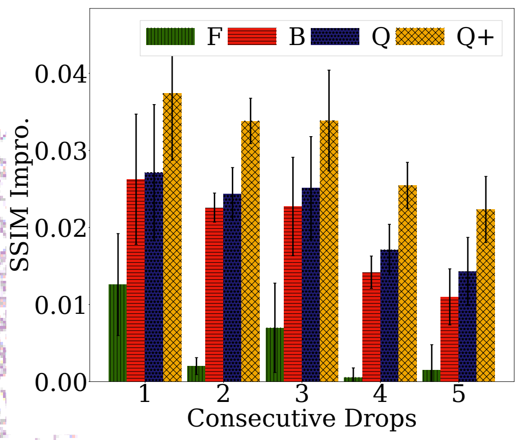
(a)



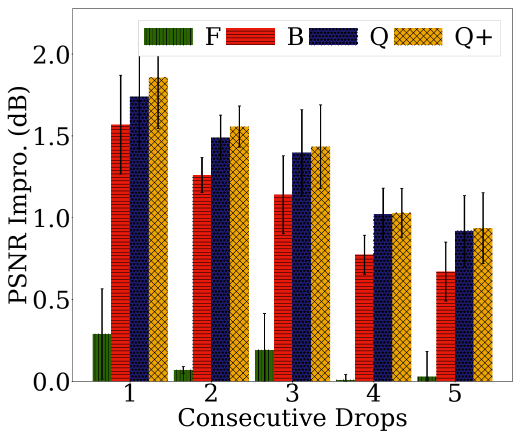
(b)



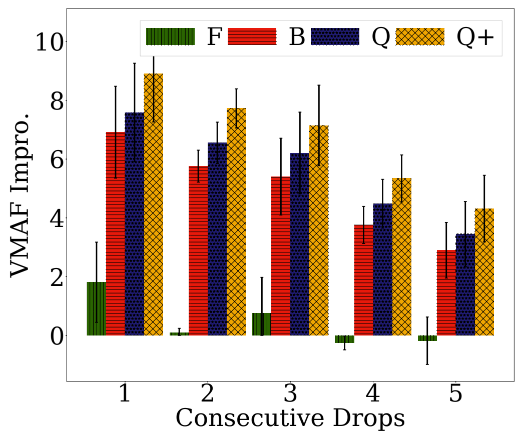
(c)



(d)

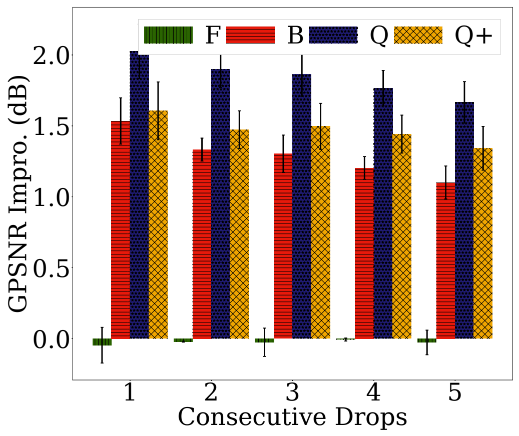


(e)

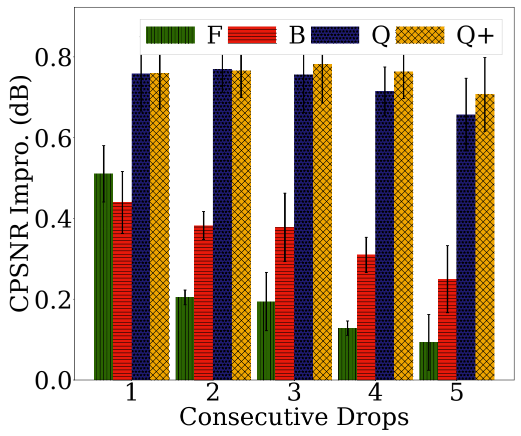


(f)

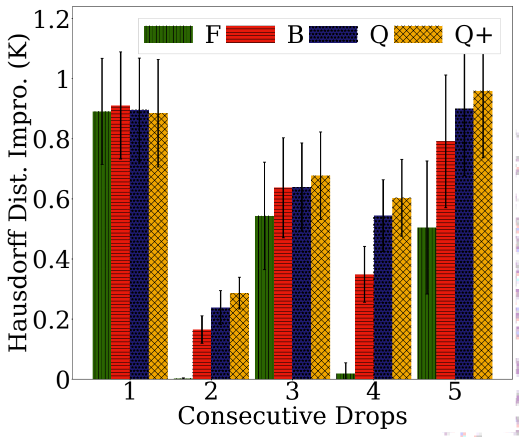
Figure 5.18: Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Loot are shown.



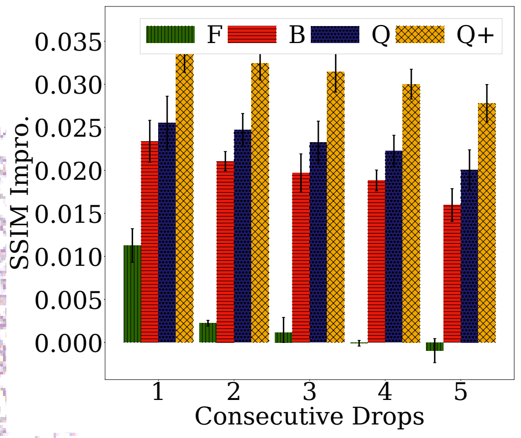
(a)



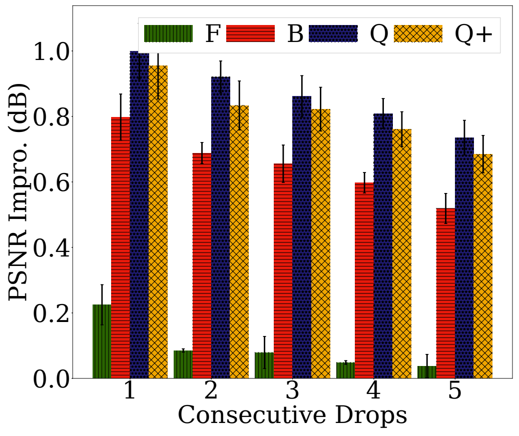
(b)



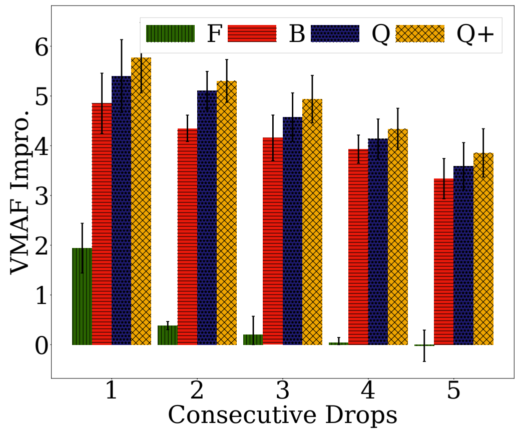
(c)



(d)

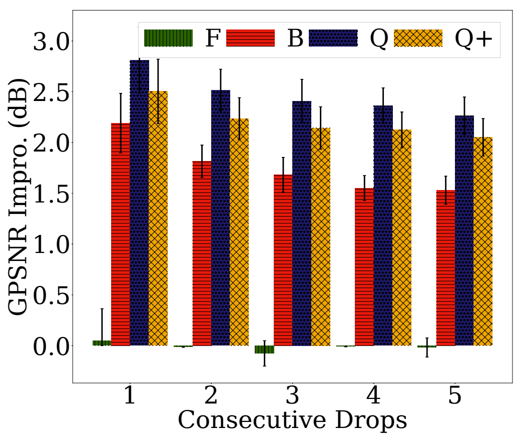


(e)

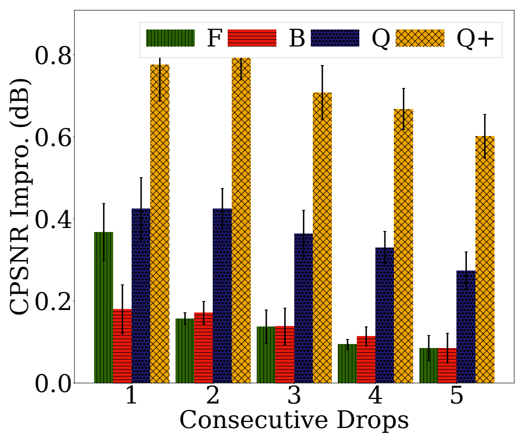


(f)

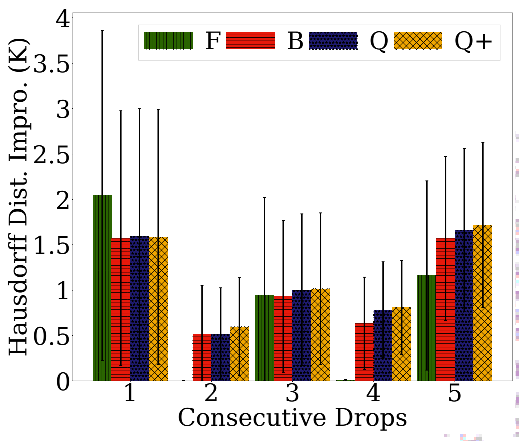
Figure 5.19: Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Red&Blk are shown.



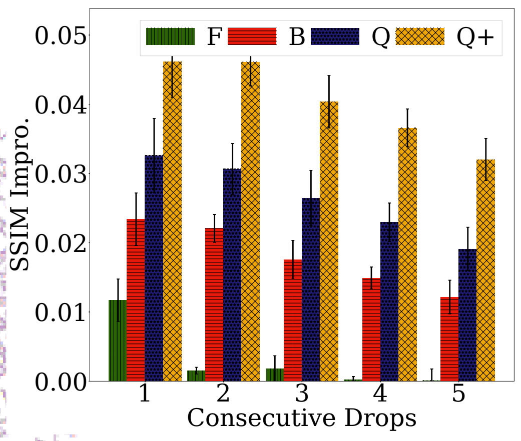
(a)



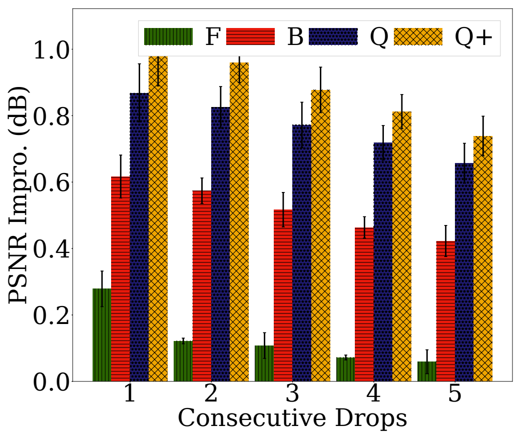
(b)



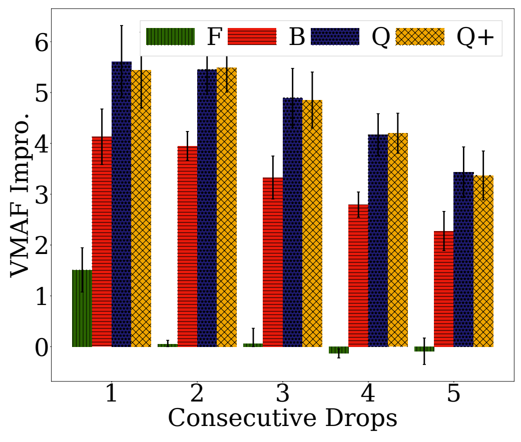
(c)



(d)

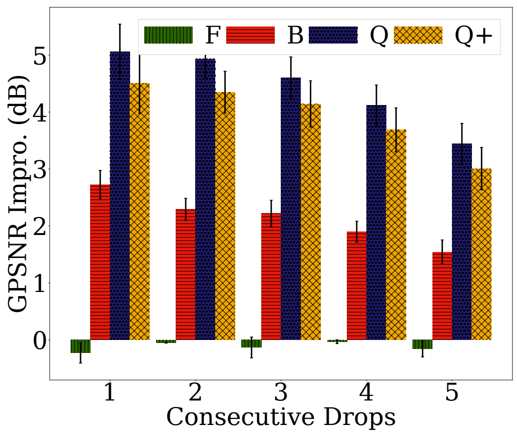


(e)

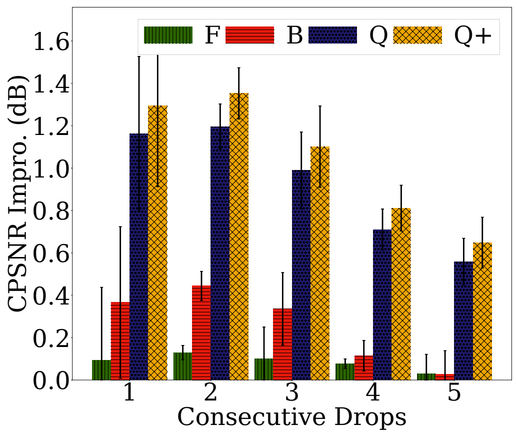


(f)

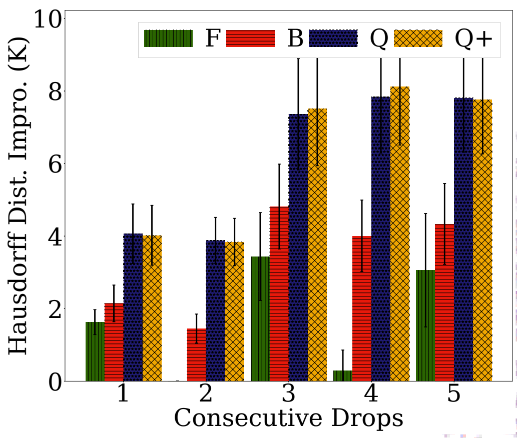
Figure 5.20: Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Longdress are shown.



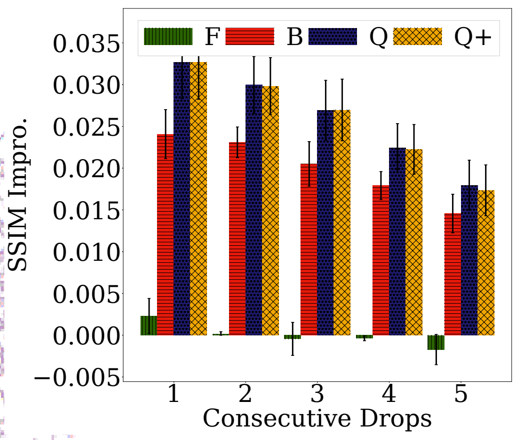
(a)



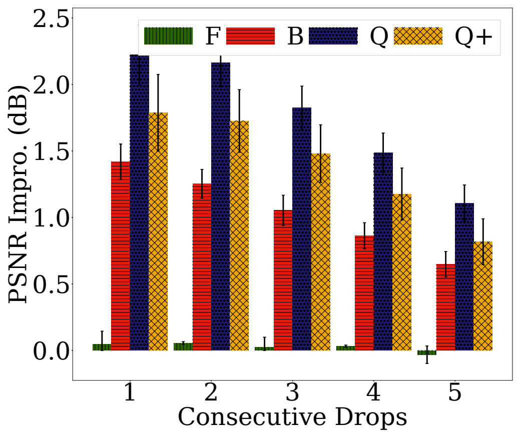
(b)



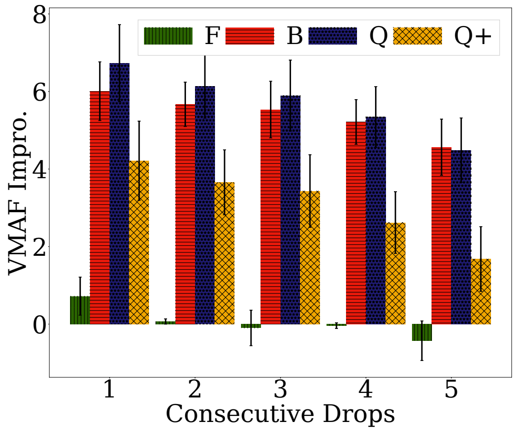
(c)



(d)

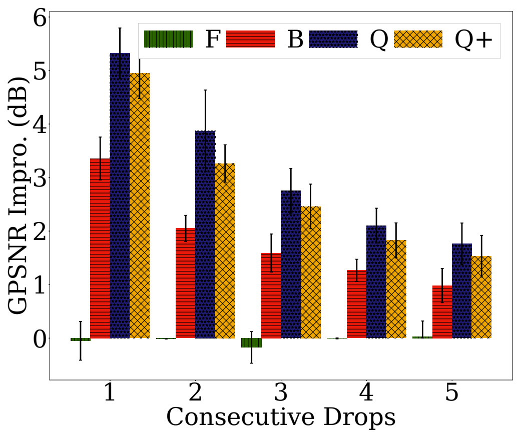


(e)

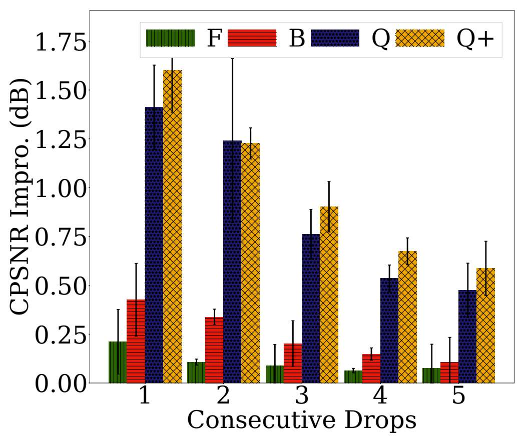


(f)

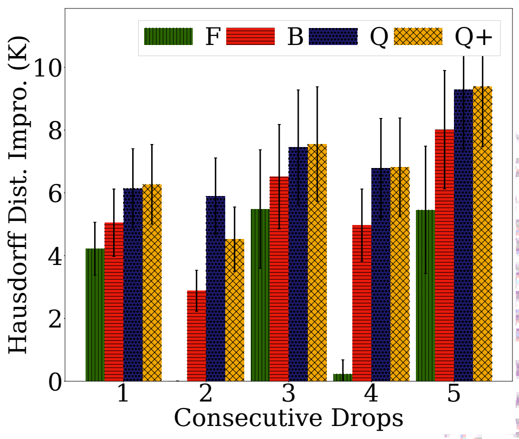
Figure 5.21: Overall quality improvement over 3DFC for different numbers of consecutive frame drops: (a) GPSNR, (b) CPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF. Sample results from Basketball are shown.



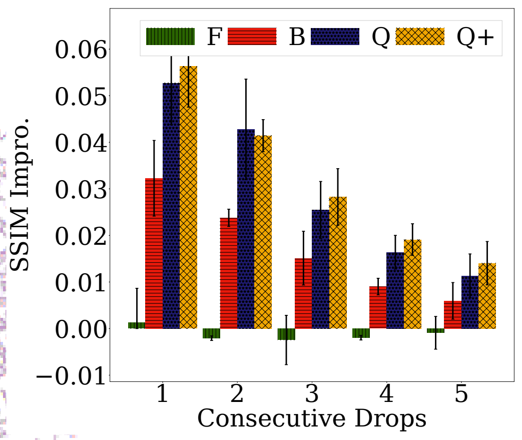
(a)



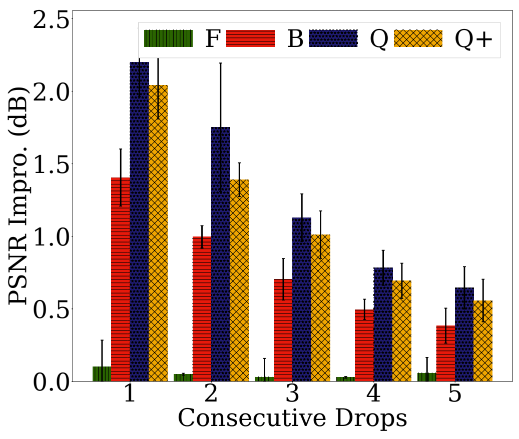
(b)



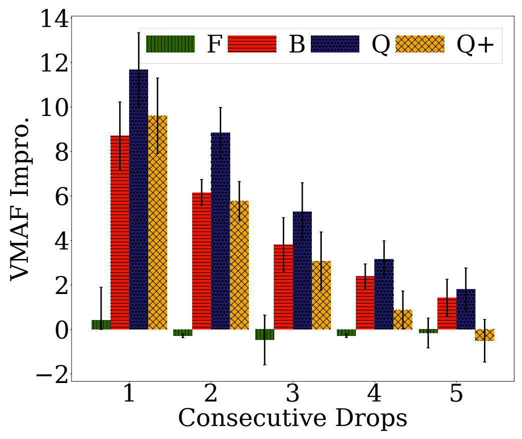
(c)



(d)

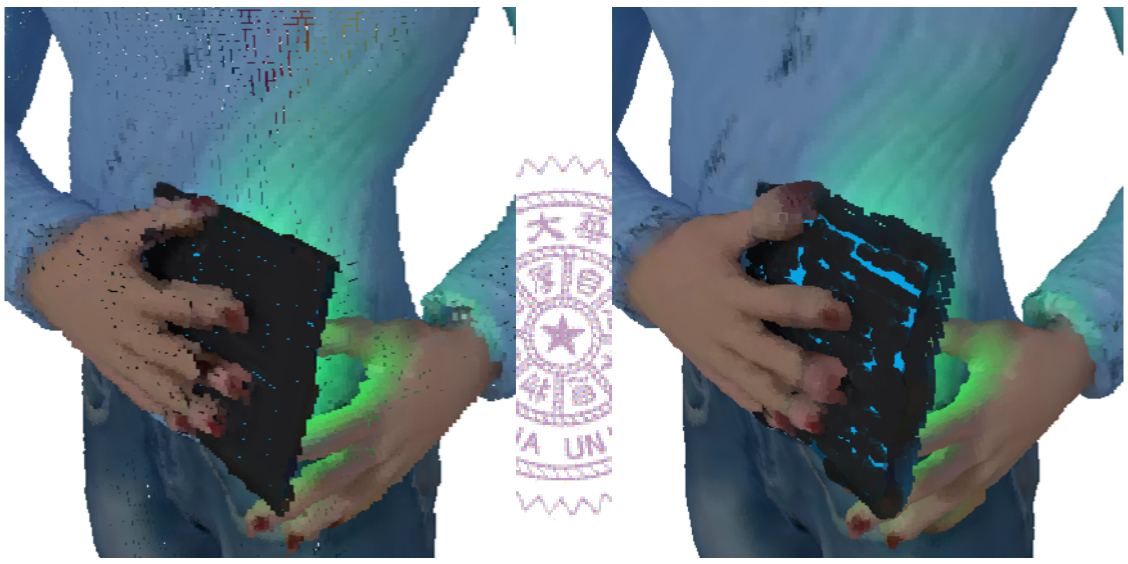


(e)



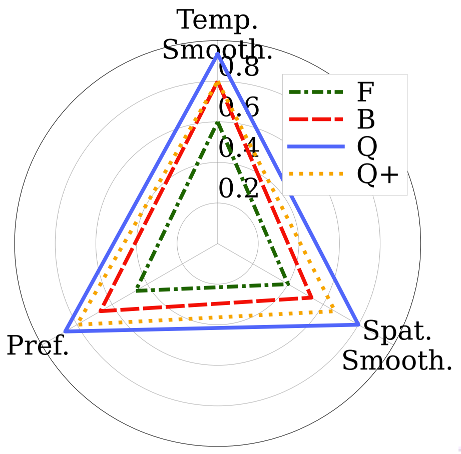
(f)

Figure 5.22: 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.

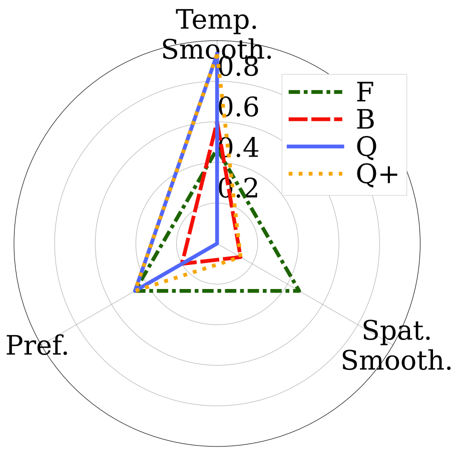


(a) (b)

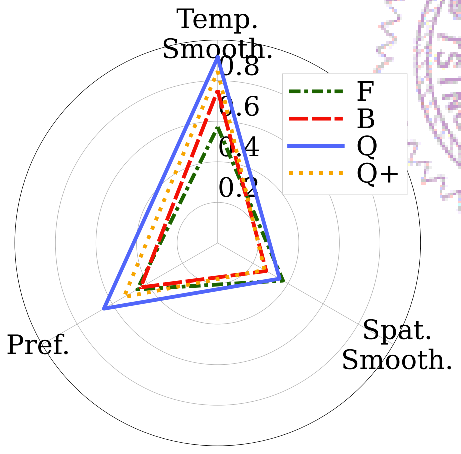
Figure 5.23: Sample 3D point cloud frame: (a) before PSR and (b) after PSR, when the normal estimation is bad.



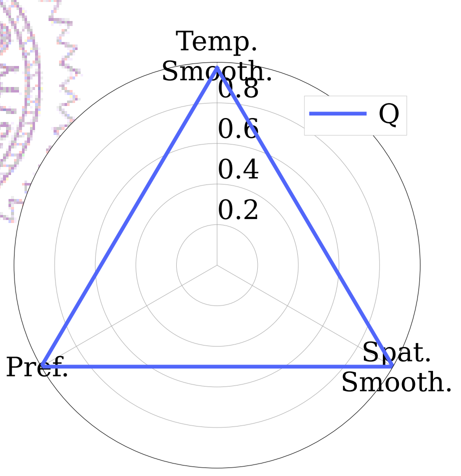
(a)



(b)

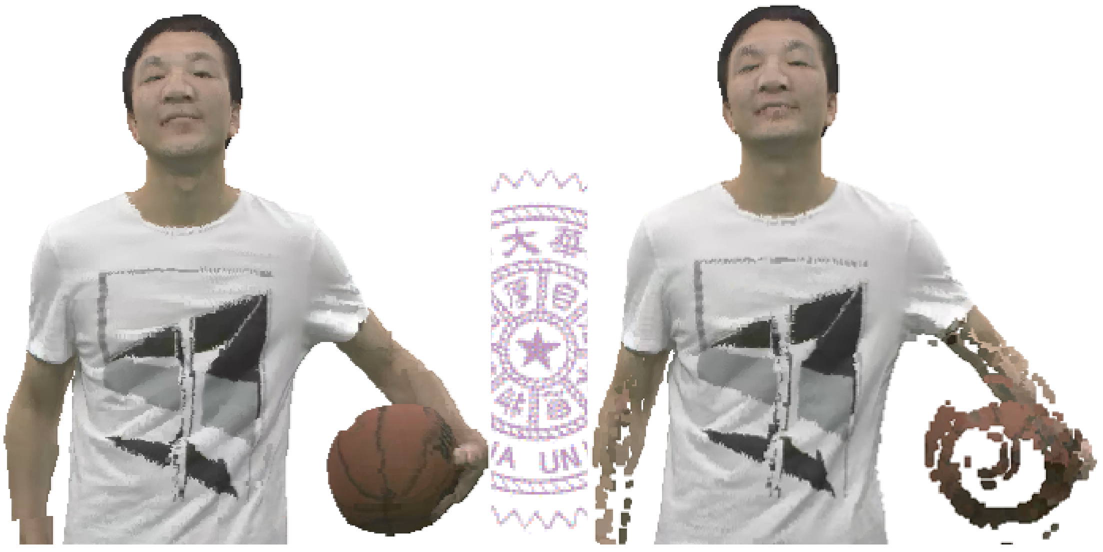


(c)



(d)

Figure 5.24: 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.



(a) (b)

Figure 5.25: 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.

Chapter 6

Concluding Remarks

This chapter starts by summarizing both the objective and subjective findings, and introduces a table that outlines a recommended pipeline under various conditions. It concludes with an exploration of potential future research directions that could be pursued by upcoming scholars.

6.1 Conclusion

In this thesis, 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 6.1: Recommended Error Concealment Pipelines

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

Our proposed reference pipelines have different pros and cons. Table 6.1 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 now, but believe there is room for innovation for such very challenging scenarios. Indeed, researchers are more than welcome to experiment with existing or to-be-developed algorithms using our framework.

6.2 Future Works

This thesis can be extended into the following directions, including but not limited to:

- *Adaptively selecting which error concealment pipeline to apply.* In our study, we noticed that point cloud sequences exhibit varying degrees of motion magnitude; thus, dropping an identical number of frames across different sequences can result in varying extents of motion. Currently, our experimental approach employs the same pipeline methods across all sequences without considering the variability in frame drops and motion magnitudes. To enhance efficiency and improve visual quality, our framework could benefit from a more adaptable strategy, selectively applying different pipeline techniques tailored to the unique attributes of each sequence and the specific frames dropped.
- *Better method to deal with large motion frames.* During our objective experiments, we noticed a trend where an increase in consecutive frame drops leads to a diminished enhancement in quality from our proposed pipeline. Similarly, in subjective evaluations, we observed that our approach might induce noticeable distortions, particularly a squeezing effect on the avatar's surface, when the motion between two frames is excessively large. These findings highlight areas for potential refinement within our pipeline framework to better accommodate large motions.
- *Running Time Optimization.* Our implementation and assessment of the proposed methods were conducted on a single thread, without employing parallelization or optimization techniques. Nonetheless, it is clear that the algorithms for various stages can be executed in parallel, as the computations for each point are independent. Leveraging Single Instruction Multiple Data and General-Purpose Graphics Processing Unit technologies could significantly reduce execution times. With the application of effective optimization methods for runtime, we are confident that our pipeline could attain real-time processing capabilities.

Bibliography

[1] A. Aaron, Z. Li, M. Manohara, J. Y. Lin, E. C.-H. Wu, and C.-C. J. Kuo. Challenges in cloud based ingest and encoding for high quality streaming media. In *Proc. of the IEEE International Conference on Image Processing (ICIP'15)*, pages 1732–1736, Quebec City, Canada, September 2015.

[2] A. Akhtar, Z. Li, G. Van der Auwera, and J. Chen. Dynamic point cloud interpolation. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'22)*, pages 2574–2578, Singapore, May 2022.

[3] T. Breckon and R. Fisher. Three-dimensional surface relief completion via nonparametric techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2249–2255, December 2008.

[4] J. Chen, J. Yi, M. Kahoush, E. Cho, and Y. Cho. Point cloud scene completion of obstructed building facades with generative adversarial inpainting. *Sensors*, 20(18):5029:1–5029:27, September 2020.

[5] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proc. of the First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 428–441, Padova, Italy, June 2002.

[6] E. d'Eon, B. Harrison, T. Myers, and P. Chou. 8i voxelized full bodies–a voxelized point cloud dataset, 2017. http://plenodb.jpeg.org/pc/8ilabs/.

[7] E. Dumic and L. da Silva Cruz. Subjective quality assessment of V-PCC compressed dynamic point clouds degraded by packet losses. *Sensors*, 23(12):5623:1–5623:28, June 2023.

[8] Z. Fu and W. Hu. Dynamic point cloud inpainting via spatial-temporal graph learning. *IEEE Transaction on Multimedia*, 23:3022–3034, March 2021.

[9] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. 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:e13:1–e13:17, April 2020.

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

[11] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, December 2021.

[12] M. S. Hamid, N. Abd Manap, R. A. Hamzah, and A. F. Kadmin. Stereo matching algorithm based on deep learning: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(5):1663–1673, May 2022.

[13] G. Haßlinger and O. Hohlfeld. The Gilbert-Elliott model for packet loss in real time services on the Internet. In *Proc. of the GI/ITG Conference-Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB'08)*, pages 1–15, Dortmund, Germany, March 2008.

[14] H. Hoppe. Poisson surface reconstruction and its applications. In *Proc. of the ACM Symposium on Solid and Physical Modeling (SPM'08)*, pages 10–10, Stony Brook, NY, June 2008.

[15] 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*, 28(8):4087–4100, March 2019.

[16] I.-C. Huang. Composing error concealment pipelines for dynamic 3D point cloud streaming, 2023. https://github.com/Huang-I-Chun/error_concealment_pipeline.

[17] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Transactions on graphics*, 33(4):1–10, July 2014.

[18] T.-K. Hung. On error concealment of dynamic 3D point cloud streaming. Master's thesis, National Tsing Hua University, 2022.

[19] T.-K. Hung, I.-C. Huang, S. R. Cox, W. T. Ooi, and C.-H. Hsu. Error concealment of dynamic 3D point cloud streaming. In *Proc. of the ACM Multimedia (MM'22)*, pages 3134–3142, Lisboa, Portugal, October 2022.

[20] A. K. Ingale and D. U. J. Real-time 3D reconstruction techniques applied in dynamic scenes: A systematic literature review. *Computer Science Review*, 39:100338:1–100338:13, February 2021.

[21] Methodology for the subjective assessment of the quality of television pictures ITU-R Recommendation BT.500-13. Document, International Telecommunication Union, 2012.

[22] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi. Video-based point-cloud-compression standard in MPEG: From evidence collection to committee draft. *IEEE Signal Process Mag*, 36(3):118–123, May 2019.

[23] M. Kazemi, M. Ghanbari, and S. Shirmohammadi. A review of temporal video error concealment techniques and their suitability for HEVC and VVC. *Multimedia Tools and Applications*, 80:12685–12730, January 2021.

[24] M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):(29):1–(29):13, June 2013.

[25] D. Kumari and K. Kaur. A survey on stereo matching techniques for 3D vision in image processing. *Int. J. Eng. Manuf*, 4:40–49, July 2016.

[26] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158, July 1981.

[27] S. Lee, S. W. Oh, D. Won, and S. J. Kim. Copy-and-paste networks for deep video inpainting. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*, pages 4413–4421, Seoul, Korea, October 2019.

[28] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li. Advanced 3D motion prediction for video-based dynamic point cloud compression. *IEEE Transactions on Image Processing*, 29:289–302, August 2020.

[29] P. Liepa. Filling holes in meshes. In *Proc. of the Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP'03)*, pages 200–205, Goslar, Germany, June 2003.

[30] F. Lozes, A. Elmoataz, and O. Lézoray. Partial difference operators on weighted graphs for image processing on surfaces and point clouds. *IEEE Transactions on Image Processing*, 23(9):3896–3909, July 2014.

[31] F. Lozes, A. Elmoataz, and O. Lézoray. PDE-based graph signal processing for 3-D color point clouds: Opportunities for cultural heritage. *IEEE Signal Process Mag*, 32(4):103–111, June 2015.

[32] MPEG. G-PCC codec description v12. Technical Report N00151, ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document, 2021.

[33] MPEG 3DG. Call for proposals for point cloud compression v2. Technical Report N16763, ISO/IEC, JTC 1/SC 29/WG 11, 2017.

[34] Common test conditions for V3C and V-PCC. Document, ISO/IEC JTC1/SC29/WG7 MPEG 3D Graphics Coding, 2020.

[35] V-PCC test model v11. Document, ISO/IEC JTC1/SC29/WG11 MPEG 3DG, 2020.

[36] M. Pauly, N. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3D scan completion. In *Proc. of the Symposium on Geometry Processing (SGP'05)*, pages 23–32, Vienna, Austria, July 2005.

[37] R. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proc. of the ICRA'11*, May 2011.

[38] S. Studio. Factory 42 and hold the world., 2023. https://www.immerseuk.org/case-study/factory-42-and-hold-the-world.

[39] Y.-C. Sun, I.-C. Huang, Y. Shi, W. T. Ooi, C.-Y. Huang, and C.-H. Hsu. A dynamic 3D point cloud dataset for immersive applications. In *Proc. of the ACM Multimedia Systems Conference (MMsys'23)*, pages 376–383, Vancouver, Canada, June 2023.

[40] J. van der Hooft, H. Amirpour, M. T. Vega, Y. Sanchez, R. Schatz, T. Schierl, and C. Timmerer. A tutorial on immersive video delivery: From omnidirectional video to holography. *IEEE Communications Surveys & Tutorials*, 25(2):1336–1375, March 2023.

[41] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. In *Proc. of the International Conference on Image Processing (ICIP'03)*, pages 903–906, Barcelona, Spain, September 2003.

[42] Verified Market Research. Immersive technology market size and forecast, 2023. https://www.verifiedmarketresearch.com/product/immersive-technology-market/.

[43] I. Viola, J. Mulder, F. De Simone, and P. Cesar. 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)*, pages 90–97, San Diego, CA, December 2019.

[44] J. Wang and M. Oliveira. Filling holes on locally smooth surfaces reconstructed from point clouds. *Image Vis Comput*, 25(1):103–113, January 2007.

[45] J. Wang, S. Tan, X. Zhen, S. Xu, F. Zheng, Z. He, and L. Shao. Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225:1–103225:21, September 2021.

[46] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[47] X. Wen, T. Li, Z. Han, and Y.-S. Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proc. of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR'20)*, pages 1939–1948, Seattle, WA, June 2020.

[48] C.-H. Wu, C.-F. Hsu, T.-K. Hung, C. Griwodz, W. T. Ooi, and C.-H. Hsu. Quantitative comparison of point cloud compression algorithms with PCC Arena. *IEEE Transaction on Multimedia*, 25:3073–3088, February 2023.

[49] C.-H. Wu, X. Li, R. Rajesh, W. T. Ooi, and C.-H. Hsu. 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)*, pages 9–14, Istanbul, Turkey, September 2021.

[50] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. GRnet: Gridding residual network for dense point cloud completion. In *Proc. of the European Conference on Computer Vision (ECCV'20)*, pages 365–381, Virtual, August 2020.

[51] R. Xu, X. Li, B. Zhou, and C. C. Loy. Deep flow-guided video inpainting. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*, pages 3723–3732, Long Beach, CA, June 2019.

[52] Y. Xu, X. Tong, and U. Stilla. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 126:103675:1–103675:26, June 2021.

[53] M. Yang, D. Wu, Z. Wang, M. Hu, and Y. Zhou. Understanding and improving perceptual quality of volumetric video streaming. In *Proc. of the IEEE International Conference on Multimedia and Expo (ICME'23)*, pages 1979–1984, Brisbane, Australia, July 2023.

[54] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*, pages 5505–5514, Salt Lake City, UT, June 2018.

[55] Y. Zeng, Y. Qian, Q. Zhang, J. Hou, Y. Yuan, and Y. He. 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)*, pages 6338–6347, New Orleans, LA, June 2022.

[56] E. Zerman, R. Kulkarni, and A. Smolic. User behavior analysis of volumetric video in augmented reality. In *Proc. of the International Conference on Quality of Multimedia Experience (QoMEX'21)*, pages 129–132, Virtual, June 2021.

[57] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing, 2023. http://www.open3d.org/.