國立清華大學電機資訊學院資訊工程學系
碩士論文

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

動態點雲串流的錯誤隱藏研究
On Error Concealment of Dynamic 3D Point Cloud Streaming

洪梓寬
Tzu-Kuan Hung

學號：109062680
Student ID:109062680

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

中華民國 111 年 五 月
May, 2022

國立清華大學
資訊工程學系

碩士論文

動態點雲串流的錯誤隱藏研究

洪梓寬

110

Abstract

Recently standardized MPEG Video-based Point Cloud Compression (V-PCC) codec has shown promise in achieving a good rate-distortion ratio of dynamic 3D point cloud compression by building on top of state-of-the-art techniques for 2D video compression. Current error concealment methods of V-PCC, however, lead to significantly distorted 3D point cloud frames under imperfect network conditions. To address this problem, we propose a general framework for concealing distorted and lost 3D point cloud frames due to packet loss. We also design, implement, and evaluate a suite of tools for each stage of our framework, which can be combined into multiple variants of error concealment algorithms. We propose five error concealment algorithms, four of which can conceal the geometry losses. These algorithms span from point-to-point, triangular, and cube-based matching methods, which offer wide variant of tradeoff between computational complexity and visual quality. We conduct extensive experiments using seven dynamic 3D point cloud sequences with diverse characteristics to understand the pros/cons of our proposed error concealment algorithms. Our experiment results show that our error concealment algorithms outperform: (i) the method employed by V-PCC by at least 3.58 dB in Geometry Peak Signal-to-Noise Ratio (GPSNR) and 10.68 in Video Multi-Method Assessment Fusion (VMAF) and (ii) point cloud frame copy method by at most 5.8 dB in (3D) GPSNR and 12.0 in (2D) VMAF. This work can both be broadened and deepen by: (i) accelerating the running time exploiting the parallelization ability of graphics processing units (GPUs). (ii) looking deeper into better matching of motion cubes and store residual values in the metadata of the codec. (iii) applying real streaming system with adaptive bitrate mechanism. (iv) make use of profound spatial info remaining in the distorted 3D point cloud and conduct spatial concealment.

中文摘要

近來動態影像專家小組 (MPEG) 所制定的V-PCC編解碼器在點雲壓縮率上展現超凡壓縮率，它將三 維的幾何資訊投影至二維，並使用了舊有且成熟的二維影片壓縮技術。但在錯誤隱藏方面，V-PCC 可說是表現得很差。在不穩定的網際網路情況下，當有錯誤的位元流被傳送，接收端收到後解碼的 三維點雲將會有嚴重的扭曲。為了解決點雲傳輸錯誤的這個問題，我們提出一個通用的錯誤隱藏架構。 我們也提出一系列包含設計，實作與比較的錯誤隱藏工具與演算法。在我們提出的五個錯誤隱藏 演算法中，其中四個可以處理錯誤的幾何資訊。這些演算法囊括了點對點，三角形法，與方塊配對法。 對上述方法，我們除了比較各種表現量尺之外，也提供了程式執行時間，讓開發者得以在模組品質 與執行時間進行權衡比較。為了分析我們提出的演算法的優點與缺點，我們在錯誤隱藏實驗中比較 了七組擁有不同特徵的點雲人像影片。我們的實驗得出以下結論：(一)我們的演算法在GPSNR量尺 中，至少贏過了V-PCC達3.58分貝。在VMAF量尺中則至少多出10.68。(二)我們的演算法在GPSNR 量尺中，至少贏過了三圍複製法達5.8分貝。在VMAF量尺中則至少多出12。這項工作可以往下列幾點 作延伸：(一)以通用圖形處理器與平行運算加速錯誤隱藏程式。(二)更深入的研究與改善運動向量 預測並預先在編解碼器的元資料中儲存運動向量殘值。(三)實作帶有自動位元率調整的點雲影片 串流系統。(四)更佳地運用幀內尚餘的幾何資訊，在少部分資訊流失時實現幀間錯誤隱藏。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dynamic Three-Dimensional (3D) point cloud is gaining prominence as a representation format for capturing and rendering volumetric videos, where 3D information (including *geometry*, coordinates, and *attributes*, such as colors) of real-world scenes are recorded and can be displayed as a free-viewpoint video for the viewers to consume with 6 Degree-of-Freedom (6DoF). The rise of 3D point clouds as a media data format is fueled by consumer-grade Light Detection and Ranging (LiDARs) as well as Red Green Blue-Depth (RGB-D) cameras, and recent emerging study for point cloud compression algorithms. In

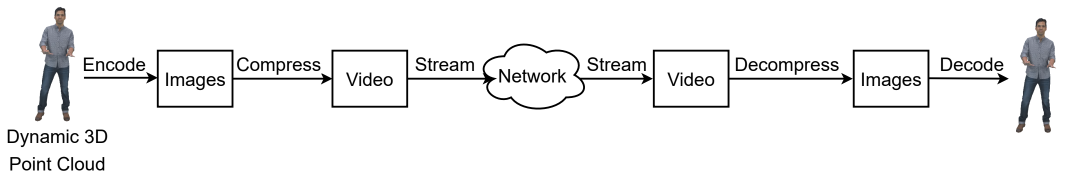


Figure 1.1: (Simplified) V-PCC pipeline.

particular, the MPEG standardization activities represent two traditional signal-processing (SP) based algorithms [22]. Both of them have exhibited a good rate-distortion ratio [22, 35], which may be an enabler of many distributed multimedia applications, in which point clouds need to be encoded, streamed, and decoded in real-time, as depicted in Figure 1.1.

We envision that streaming of volumetric video with 6DoF free-viewpoint support will soon become a killer application of dynamic 3D point clouds, partly due to the unfortunate COVID-19 pandemic. Such type of video for social gathering, remote collaborations, and distance educations provides much more immersive experience than existing Two-Dimensional (2D) video-based solutions. For real-time communications, dynamic 3D point clouds are likely streamed over Real-Time Transport Protocol (RTP) for smooth interactions. Different from reliable Dynamic Adaptive HTTP Streaming (DASH), RTP trades reliability for shorter latency. Unfortunately, lost (or late) packets of transmitted bitstreams could lead to staggering negative impacts on the quality of the reconstructed 3D point clouds, which may dramatically degrade the user experience. To illustrate this, Fig. 1.2 shows sample rendered images encoded and decoded by the referenced codec V-PCC (Video-based Point Cloud Compression) [35] when: (a) the attribute (color) data are lost, (b) the geometry (coordinate) and attribute data are lost, and (c) the error propagates to a subsequent point cloud frame. More details on this figure will be explained in Chapter 4.

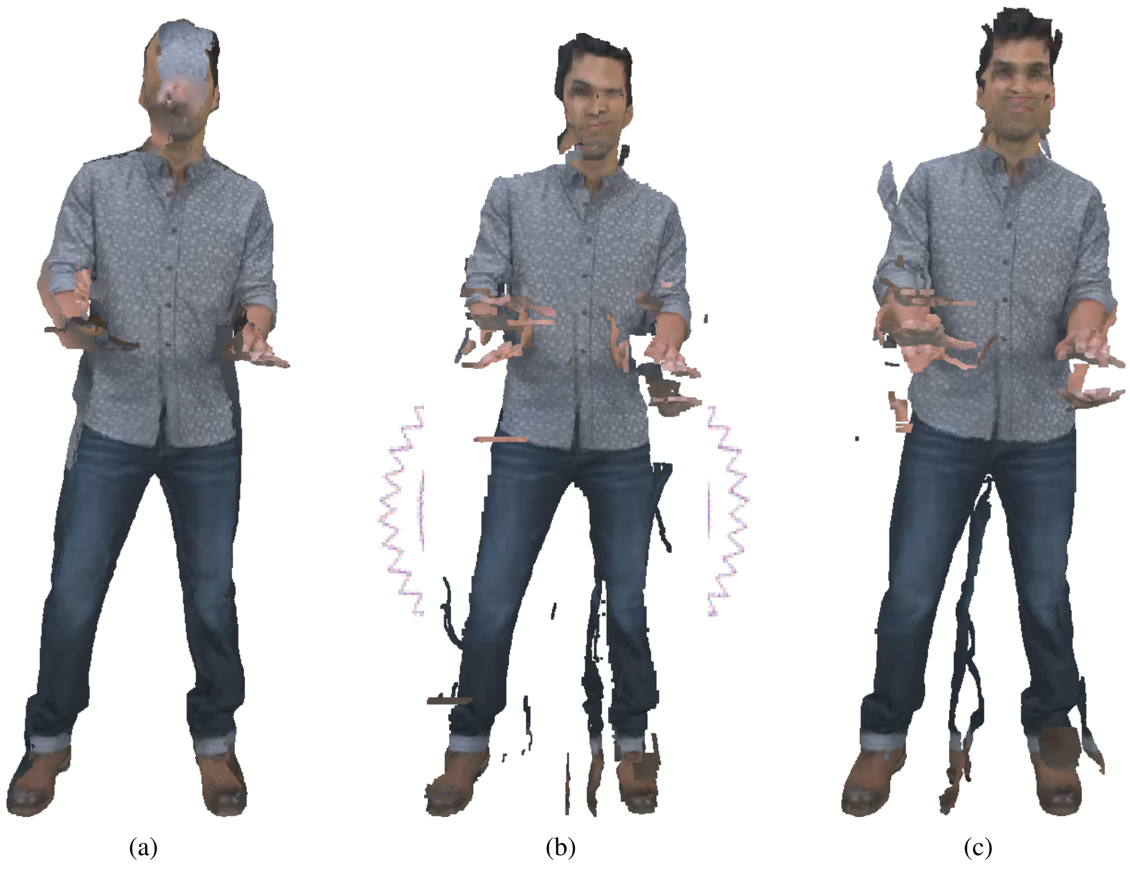


Figure 1.2: Sample reconstructed point clouds under packet losses: (a) distorted attributes (colors), (b) distorted geometry and attributes (colors), and (c) distortion due to error propagation across point cloud frames.

In this scenario, to deliver immersive user experience over the best-effort Internet, where the network bandwidth is diverse and dynamic, while high network latency and packet loss are both possible, we develop several techniques to conceal transmission distortion due to lost or late packets in dynamic 3D point clouds streaming. Instead of concealing the error in the video decoding process, we generate a new point cloud in the place of the missing one. We exercised and compared several techniques when designing the point cloud generation algorithms through real experiments. The techniques we present have applications beyond error concealment. These techniques essentially generate new frames for point clouds, given existing and nearby frames. For instance, given two frames, we wish to generate an intermediate frame between the two. Such intermediate frame generation can be used to up-sample the point cloud sequence in the temporal dimension, leading to smoother playback.

1.1 Limitations

We have the following assumptions and limitations for our work.

- **Loss whole frames.** We overwrite the whole packet of geometry information of each frame. Then we consider the all geometry information in the exact frame is unreliable as well as color information and completely depend on the temporal concealment.

- **All-Intra coding mode.** We only use the all-intra mode from three modes provided by V-PCC codec. We don't use any metadata and 2D motions because we think the back projection from 2D motions to 3D motions is not suitable for our current concealment algorithms; thus, we can ignore the metadata packet loss.

1.2 Organizations

We organize the rest of this paper as follows. Chapter 3 presents the related work on streaming 3D content. Chapter 4 analyzes the impact of losses on dynamic 3D point clouds. Chapter 5 presents our error concealment techniques. The experiment results are presented in Chapter 6. Chapter 8 concludes the paper.

1.3 Contributions

This paper makes the following contributions:

- We develop a suite of techniques to generate intermediate point clouds to conceal transmission distortion in dynamic 3D point cloud streaming. To the best of our knowledge, the only prior art is Wu et al. [66], which only presented a naive and straightforward interpolation method for error concealment for 3D point cloud streaming.

- We conduct extensive experiments with real dynamic point cloud sequences to demonstrate the merits of our error concealment techniques and study the implications of different system parameters. For example, our error concealment techniques outperform the current 2D frame copy (2DFC) error concealment in terms

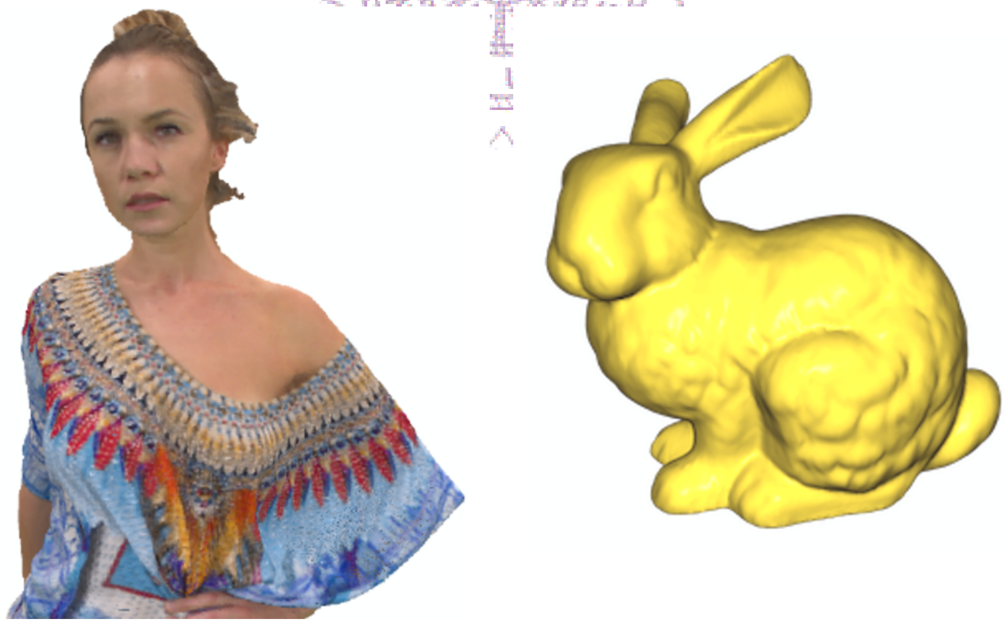
of rendered quality: by up to 11.09 dB in PSNR (Peak Signal-to-Noise Ratio) and up to 0.3 in SSIM (Structural Similarity Index).

Chapter 2

Background

We summarize the backgrounds of relative data representations and applications in this section. We first list popular 3D Representations. What comes next is the detailed characteristics analysis of 3D point clouds. Finally, we conclude the heterogeneity and applications for different kinds of point clouds.

2.1 3D Representations



(a) (b)

Figure 2.1: 3D representations of (a) 3D point clouds and (b) 3D meshes.

- **Point Clouds.** In point clouds, each point is composed of mandatory geometry attributes, i.e., (x, y, z) coordinates with the optional attributes such as RGB colors, reflectances, and normals. Fig 2.1(a) shows a point cloud avatar from 8i [16] dataset. Traditional RGB cameras can capture (x, y) coordinates and RGB colors. The third coordinate z can be obtained from either stereo images, which derive depth value by epipolar geometry [45] or commodity RGB-D cameras, which estimate depth by at least two infrared (IR) cameras and capture colors by a normal RGB camera [36]. Because of the simplicity of the data format, point clouds are easier to be manipulated, rendered [5], and stitched [4, 68]. Point clouds are also more suitable for real-time interactive applications compared to the other 3D representations. On the other hand, the primitiveness of the point cloud results in high storage usage and high bandwidth consumption over the Internet if the data is uncompressed. Fortunately, the emerging numbers of compression researches [7], open-source codecs [65], and coding standards abound [6, 17, 53].

- **Meshes.** Meshes are composed of points, polygons, and textures. The polygons not only form the surfaces but also provide connectivity information among vertices for further applications. On the other hand, the connectivity makes 3D meshes harder to be modified compared to point clouds. Fig 2.1(a) shows a mesh model. The rendering [3] and compression [41] techniques for meshed are quite mature, probably due to the popularity of video games. One drawback of 3D meshed is that none of the output data types of any 3D scene acquisition sensors come in the format of meshes. Nonetheless, 3D meshes can be generated by point clouds following the process of normal estimation, downsampling, surface reconstruction, and texture generation. Although there are open-sourced software for such conversion process [11, 49, 71], the time-consuming generation process prevent meshes from real-time applications.

- **Light Fields.** Light Fields (LF) can faithfully reproduce 3D scenes captured in the past. It's an ultimate way to present the real-world scenes with parallax and color density. If LF can be projected in the air, it becomes the hologram and the user can consume 6DoF scenes without any wearable devices. The current state-of-the-art method of LF acquisition is to capture either micro-images by LF camera [21] or image arrays, as depicted in Fig 2.2, captured by camera arrays and gantries [37, 50, 72] with a slight angle and viewing position tweaking among neighbor images. For micro-images, the refocusing applications which utilize the wide depth of field abound [46]. On the other hand, image arrays acquired from the camera array or the gantry are suitable for multi-view applications. The light field video frames composed of 4D (with horizontal and vertical parallaxes) or 3D (horizontal parallaxes only) image arrays can benefit from sophisticated 2D video Codec

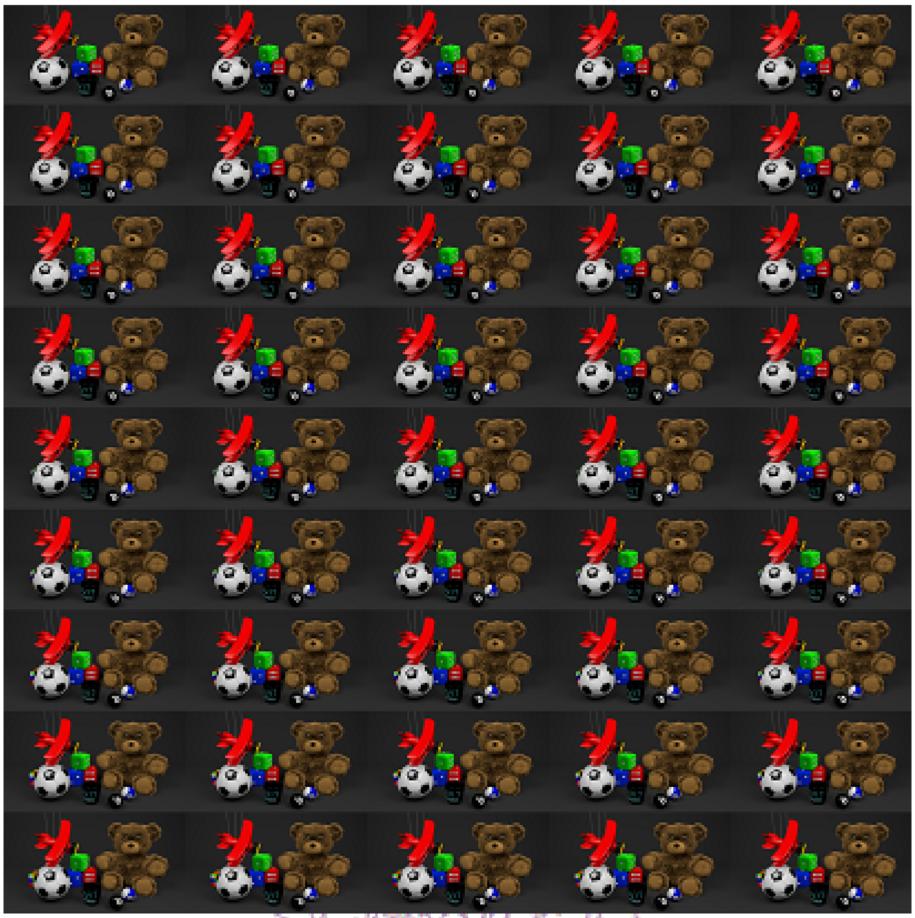


Figure 2.2: 5x9 image array created by 45 different images from the TUT dataset [59].

and Multiview Video Codec (MVC) [14] with affordable display devices [40]. For example, the image array in Fig 2.2 can be displayed by Looking Glass Liquid Crystal Display (LCD) with more than 45 horizontal parallaxes. Furthermore, with the dihedral corner reflector devices, the light field images can be displayed as a hologram [42]. For example, Fig 2.3 shows the holographic scenes displayed by combining the Looking Glass LCD and the ASKA3D plate [2].

We use point clouds as the data type for 3D representations because of their ease of data acquisition, vertex manipulation, scene registration, rendering, and interactions for teleconferences. Further characteristics of point clouds will be analyzed in the following section.

2.2 Point Clouds

Compared to 3D meshes, which cannot be natively output by any existing sensors, the point cloud is a light-weight data format with the following characteristics:

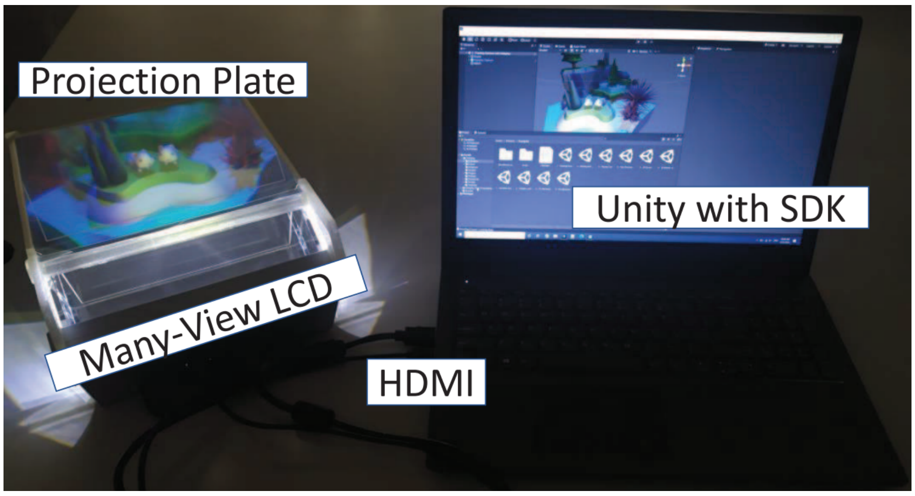


Figure 2.3: A prototype holographic display driven by a laptop in our lab.

- **No Edge or Face Information.** Unlike polygon meshes, point clouds do not have any information of connectivity which can form edges and faces. Thus, point clouds can be directly captured by Time-of-Flight (TOF) sensors and are more suitable for real-time applications. The lack of connectivity also enables point clouds to be more easily edited. On the other hand, much more vertices are required for point clouds to render an object with similar quality in comparison with 3D meshed, which consume much more storage spaces.

- **Heterogeneity.** Depending on the applications, a variety of kinds of point clouds are suitable for different applications. A wide spectrum of densities of point numbers from tens to millions by either directly acquired (sensors that output the point cloud natively) or efficiently converted (generated from stereo images) can be applicable to applications with different budget (in terms of bandwidth or money) concerns. Tens to hundreds of points acquired by low-cost sensors can be used for human activity recognition [54] while prevent users from identity revealing; moreover, this kind of point clouds favors resource-constrained embedded systems. On the opposite, to form a human avatar with acceptable visual quality, more than a half-million vertices are required [16]. The optional attributes can also be appended as metadata or critical ingredients in addition to mandatory geometry coordinates for each point of point clouds.

- **Unordered.** The point cloud is an unordered set, which means, for the dynamic point clouds, each vertex of which contains no correlations to any other point within the same frame (intra-frame). Here, the Dynamic 3D point clouds are the sequences of 3D point clouds captured over a period of time, where each point cloud captured at a specific time is referred to as a point cloud frame. The vertex number among point cloud frames is also different. In other words, there is no one-to-one correlation among points in the different frames (inter-frame).

Given the aforementioned characteristics of point clouds, transmitting uncompressed dynamic point clouds may cause catastrophic network congestions. Unfortunately, dynamic point clouds contain non-trivial spatial and temporal redundancy, which implies difficulty and challenges for advanced applications like compression and error concealment.

2.3 Variants of Point Clouds

The diverse properties of sparseness levels and optional attributes demonstrate heterogeneity and flexibility of point clouds. Point clouds acquired by different sensors are applied for different usage scenarios across healthcare, traffic safety, educations, and entertainments.

- **Sparse Point Clouds.** Capturing tens of vertices without color attributes in each

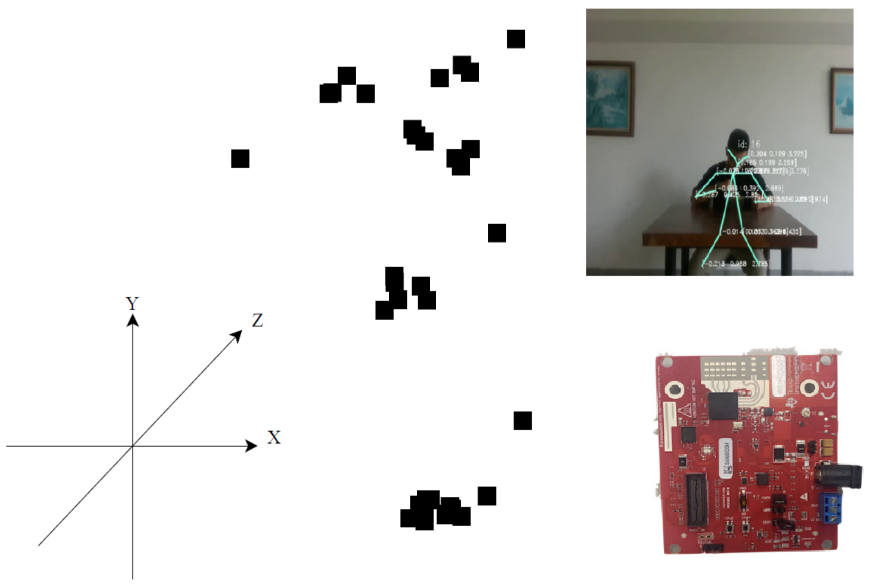


Figure 2.4: Sample point clouds from mmWave radars.

frame, sensors that don't reveal human identities, as Fig. 2.4 shows, can be installed in places with privacy concern like bathrooms and bedrooms. For example, fall detections can not only achieve by RGB cameras [15] but also by millimeter wave (mmWave) radars [58] with tens of vertices [55] for healthcare services in nursing homes. The bandwidth-friendly characteristic of this kind of sensors acquiring points with such sparseness level favors real-time applications. For example, Human activities can also be monitored and recognized with such sparseness level in real-time [54, 70].

- **Cylindrical Point Clouds.** Firing tens of lasers cylindrically, the LiDARs [61]

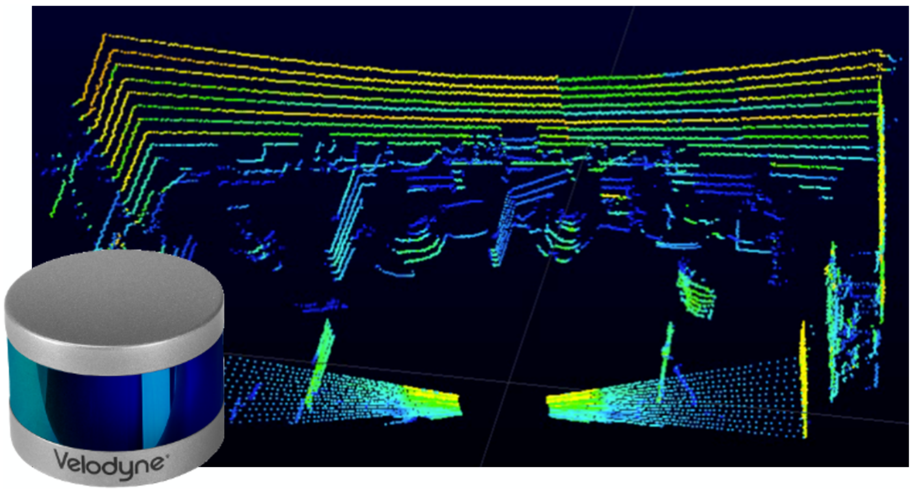


Figure 2.5: Sample point clouds from LiDARs.

output point clouds with high vertical resolution but sparse in horizontal resolution as depicted in Fig. 2.5. Along with the high accuracy characteristic, this kind of point cloud can be used for 3D urban models [62] and inspection for civil engineering [47, 56]. Pedestrian and obstacle detection is also an emerging applications due to the popularity of self-driving cars.

- **Dense Point Clouds.** This kind of dense point cloud can be efficiently generated

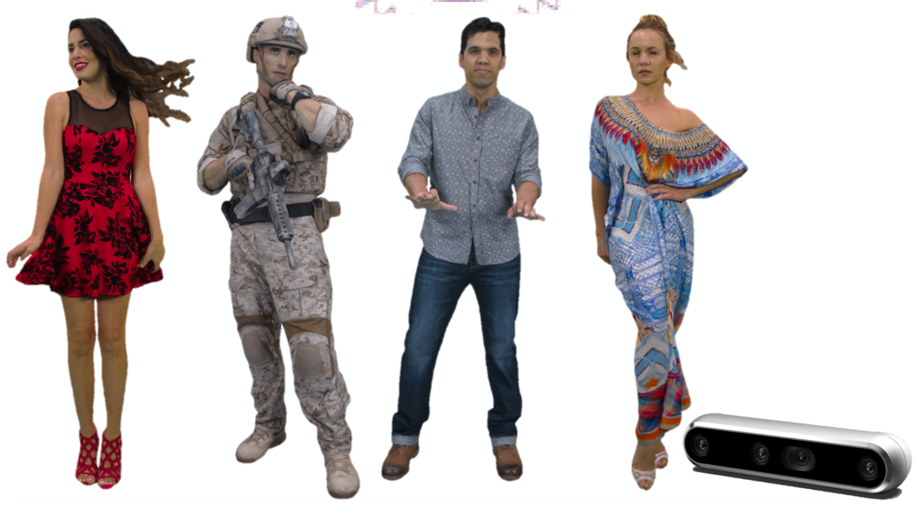


Figure 2.6: Sample point clouds from RGB-D cameras.

by RGB-D cameras with color attributes. Fig. 2.6 shows an commercial Intel RealSense RGB-D camera [33] and point cloud avatars of the 8i dataset [16] generated by logical RGB-D cameras. The dense and colored point clouds can be utilized in immersive communications among remote participants wearing a Head-Mounted

Display (HMD). Dense point clouds generated from RGB-D cameras, where each point contains color attributes, can be used in many XR applications including gaming, entertainments, remote collaborations, and distance educations [64]. Here, XR refers to HMD-enabled interaction modes, including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), where users can freely move (in x, y, and z coordinates) and rotate (in yaw, row, and pitch angles) to achieve 6-DOF in a virtual world with immersive experience.

Though heterogeneity exists in point clouds, recent point cloud compression are mostly focusing on the dense point cloud for entertainments and tele-conferences. As well as our usage scenario, our work is mainly applicable to dense 3D point cloud streaming.

Chapter 3

Related Work

3.1 Dynamic 3D Point Cloud Streaming

Existing work on *streaming of static 3D content adopts diverse 3D representations*. Researchers have studied 3D content streaming with multi-resolution point-based representations [48], progressive meshes [8, 10, 23], geometry images [30], and polygon soup [19]. For static 3D content, since there is no temporal component, any packet loss can be recovered through retransmission. Such research focused on error-resilient coding to minimize the retransmission effort instead of error concealment. *Dynamic 3D content* can be streamed with representations like multiple RGB-D video streams [69], free-viewpoint videos [13], and dynamic 3D meshes [60]. Both RGB-D video streams and free-viewpoint videos capture the 3D content into 2D videos, and thus video-based error concealment still applies. Recently, the increased availability of LiDARs and RGB-D cameras has spurred interests in the streaming of dynamic 3D point clouds. Hosseini et al. [29] adapted MPEG DASH for streaming dynamic point clouds; Bo et al. [24] and Lee et al. [38] presented two separate systems for the streaming dynamic 3D point clouds to mobile phones. These papers employed KD-tree-based representations for 3D point clouds, rather than more comprehensive codecs, such as V-PCC [35], which are more fragile to packet loss. Further, the existing work targeted non-interactive applications, where latency is less crucial and lost packets could be recovered through retransmission.

3.2 Point Cloud Compression

Streaming uncompressed avatars in the form of dynamic point clouds consume up to 4Gbps of bandwidth [7], not to mention a whole scene, which is not supported by contemporary network infrastructures [12]. For dynamic point clouds and point cloud videos, MPEG has the standardized V-PCC [35], which projects each frame of 3D point clouds into several 2D images. The resulting 2D images can then be encoded using state-of-the-art video codecs which exploits years of research and advances in video codecs. However, the V-PCC reference decoder applies naive error concealment to recover any lost frame, by copying the lost data from the previous 2D video frame over as a replacement. Doing so leads to catastrophic distortion, because color or geometry information from the previous 2D video frame does not align with that of the current 2D video frame, which significantly distorts the reconstructed 3D point cloud frame.

3.3 Error Concealment

There are many advanced error concealment methods on video proposed in the literature. For example, Muhammad et al. [44] proposed a multi-threaded algorithm for frame-level error concealment. Through parallel computations, they successfully reduced the overall running time and outperformed the naive frame-copy and block-matching approaches. Block-level error concealment algorithms [28, 32] attempts to conceal the distortion using both spatial and temporal information from surrounding blocks and frames, typically guided by the motion vectors. For instance, Hwang et al. [32] proposed an improved multi-directional interpolation algorithm for spatial error concealment, which gives higher weights to the boundary pixels on the edges parallel to the interpolation direction. Hojati et al. [28] gave an algorithm to conceal the errors in parallelogram partitions via recovering the motion vectors therein. More recently, neural networks have been applied for 2D error concealment [51, 52]. For example, Sankisa et al. [51] developed a deep neural network to predict future optical flows, which are then used to conceal the damaged blocks in individual frames. In their other work [52], the authors presented a temporal capsule network to encode motion vectors as parameters, which are then extracted for motion-compensated error concealment. Their network comes with an initial feature extraction layer and a recurrent layer for spatio-temporal features. The capsule output is combined with the most recent frames before being sent through a fully connected network.

3.4 Inpainting

Several recent papers have proposed methods to *conceal errors in 3D point clouds due to imperfection in data capture by inpainting.* He et al. [26, 31] proposed several efficient inpainting methods to conceal the attributes of point clouds, which exploit smoothness and self-similarity in graph spectral domain. They represent irregular point clouds naturally on graphs, split a point cloud into fixed-sized cubes as the processing unit, and search for the most similar cubes to the target cube with holes inside. Fu et al. [20] inpainted dynamic 3D point clouds by leveraging both intra-frame self-similarity and inter-frame consistency. Other studies focused on point cloud completion, which estimated the complete geometry of the objects and scenes, based on partial observations. Recent work in this area mostly employed deep learning techniques. Chen et al [9] propose a method that perform point cloud scene completion of building facades using orthographic projection and generative adversarial inpainting methods. Xie et al. [67] take 3D grids as intermediate representations to regularize unordered point clouds and propose a novel Gridding Residual Network (GRNet) for point cloud completion. They devise two novel differentiable layers, Gridding and Gridding Reverse, to convert between point clouds and 3D grids without losing structural information. Wen et al. [63] propose Skip-Attention Network (SANet) for 3D point cloud completion, which effectively exploit the local structure details of incomplete point clouds during the inference of missing parts. Addresses missing data during capture which could lead to large holes in point clouds, they also propose a structure-preserving decoder with hierarchical folding for complete shape generation. In our work, however, the artifacts are mostly (much smaller) cracks due to imperfect error concealment from the previous (and next) point cloud frames. These inpainting and scene completion techniques are excessive for these cracks, and could be too heavy for real-time dynamic 3D point cloud streaming. Our error concealment techniques for dynamic 3D point clouds are inspired by error concealment for video streaming. Compared to spatial concealment, temporal concealment is more challenging, because the encoded motion vectors are the motion vectors of the 2D video frames and may not correlate to motion vectors of 3D point clouds unless the encoder specifically takes both 2D and 3D data into consideration [39]. Hence, our work focuses on temporal error concealment.

Chapter 4

Error Concealment Problem

4.1 System Overview

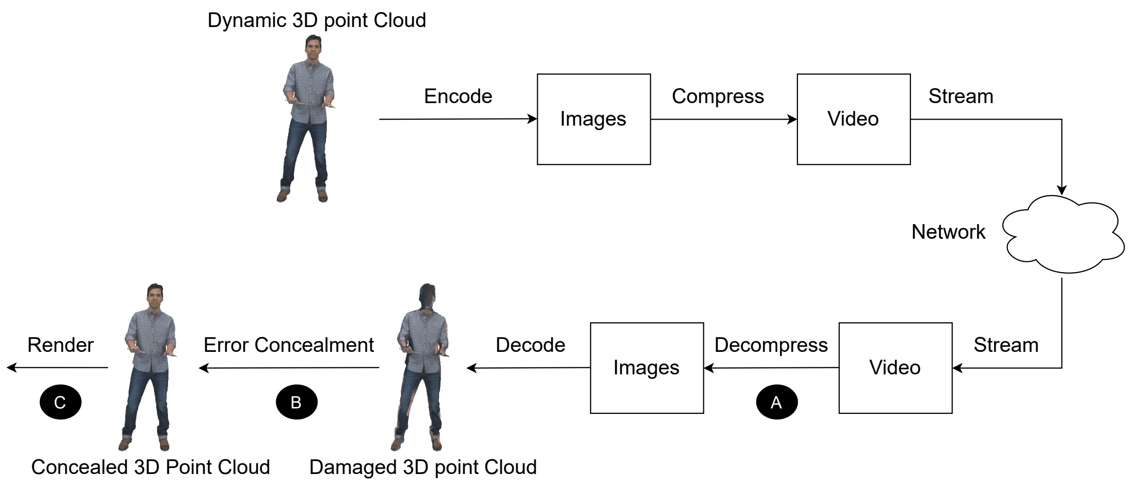


Figure 4.1: Sample error concealment pipeline of dense point clouds, where missing frames are concealed with neighboring ones.

We give a high level pipeline that conceals missing geometry (coordinates) and attributes (colors) in Fig 4.1. We use V-PCC as the reference codec to encode and decode the dynamic point clouds, where parts of the bitstreams are damaged. We locate the damaged point cloud frame and consider the whole frame is missing and perform temporal error concealment. In particular, we conceal the missing frame at B by matching and interpolation from neighbor frames. Finally, we render the concealed point clouds and adopt the similar method with PCC Arena [65] to evaluate the 2D and 3D metrics at C. Note that, this figure was created with V-PCC in mind. For instance, the decompress step at A is done by traditional 2D video codecs, such as HEVC [27]. Nonetheless, our proposed error concealment algorithms at B operate on corrupted point clouds in 3D domain and, hence, are agnostic to how the point clouds are compressed and streamed. The only assumption we make about the representation of the point clouds is that the geometry and attribute data are encoded separately. Our approach is thus general and is applicable beyond the MPEG V-PCC codec.

4.2 Problem

In this section, we briefly introduce the structure of V-PCC bitstreams. We then carry out experiments with staged packet losses to understand how different packet loss patterns affect the reconstructed 3D point clouds. Each V-PCC bitstream comprises three (sub) bitstreams: Occupancy Video Data (OVD), Geometry Video Data (GVD), and Attribute Video Data (AVD). Similar to Wu et al. [66], we parsed encoded V-PCC bitstreams and encoder logs to identify individual NALUs (Network Abstraction Layer Units) in OVD, GVD, and AVD bitstreams, where NALUs are essentially network packets. To emulate lossy networks, we first manually marked the NALUs to drop. We then parsed the V-PCC bitstream again and replaced the marked NALUs with zeros to create the corrupted V-PCC bitstream. The resulting bitstream was subsequently decoded by the V-PCC reference decoder for a reconstructed dynamic point cloud sequence. We observed that the V-PCC reference decoder performs no error concealment in the 3D domain, but relies on the 2D video codec to do so.

Table 4.1: Decoding Outcomes Under Diverse Loss Patterns

Pattern	**I**	**P**	**S**	**I+P**	**I+S**	**P+S**	**I+P+S**
O	C_G	-	-	-	-	-	-
G	C_G	C_G-End	N	C_G-End	C_G	C_G-End	C_G-End
A	C_A	X	N	X	C_A	X	X
O+G	C_G	C_G-End	N	C_G-End	C_G	C_G-End	C_G-End
O+A	C_G	X	N	X	C_G	X	X
G+A	C_G	C_G-End	N	C_G-End	C_G	C_G-End	C_G-End
O+G+A	C_G	C_G-End	N	C_G-End	C_G	C_G-End	C_G-End

According to previous research results [66], we observe that dropping NALUs from OVD and GVD leads to similar distortion on geometry structures while dropping NALUs from AVD maintains the same geometry structures but with incorrect colors. In previous research we have found that the V-PCC reference software performs no error concealment in 3D space, but relies on 2D codecs to do so. However, only the NALUs of either I or P frames were dropped in their study, and the implications of dropping different frames were not thoroughly analyzed. Some of our pilot tests lead to more insights beyond their reported results. However, only the NALUs of I frames were dropped in their study, while the NALUs of P frames and Supplemental Enhancement Information (SEI) were assumed to be reliably delivered.

Different from Wu et al. [66]We study the impacts of packet loss of I frames, P frames, and Supplemental Enhancement Information (SEI) NALUs from OVD, GVD, and AVD in this paper. More precisely, we encode each point cloud sequence of 5 point cloud frames with a single Group of Frame (GoF). We then drop some NALUs of the third point cloud frame. We consider all permutations of dropping I frame, P frame, and SEI NALUs, which leads to seven patterns. Besides, we consider dropping these NALUs from all permutations of OVD (O), GVD (G) and AVD (A) bitstreams, which results in 43 corrupted V-PCC bitstreams in total[1]. We decode these V-PCC bitstreams and report the decoding outcomes in Table 4.1[2]. There are five different outcomes:

- **N**: No clear visual impairment is observed.
- **C**$_G$: Point cloud frame 3 is distorted in both geometry and attributes.
- **C**$_A$: Point cloud frame 3 is distorted in attributes only.
- **C$_\mathbf{G}$-End**: Point cloud frames 3–5 (end of the GoF) are distorted due to error propagation of distorted frame 3 in both geometry and attributes.
- **X**: Point cloud frames 3–5 (end of the sequence) are not decoded due to an assertion error of the reference decoder.

We make the following observations on this table:

- *The reference decoder is not robust against inconsistent attribute and geometry data.* We see that when dropping P frame from AVD (A or O+A) without dropping P frame from GVD results in assertion errors.
- *Error propagation across point cloud frames happens whenever the P frame is dropped.* This can be seen in columns 2, 4, 6, and 7.
- *Dropping SEI NALUs does not impose significant impact on the decoding outcomes.* For example, when only SEI NALUs are dropped, we see no clear visual impairments in column 3. In addition, dropping additional SEI NALUs doesn't make the decoding outcome worse, e.g., rendered images from columns 2 (P) and 6 (P+S) are visually identical.

[1]The OVD bitstream of a point cloud frame only contains I frames; hence, we got 43 rather than 49 corrupted bitstreams.

[2]The precise dynamic point cloud sequence used in this motivative experiment is independent to our findings.

Fig. 1.2 gives three sample reconstructed point clouds with diverse distortion types. Fig. 1.2(a) is resulted from corrupted AVD and thus contains incorrect attribute (i.e., color) data. Fig. 1.2(b) is resulted from corrupted GVD and thus contains incorrect geometry data, in addition to incorrect attribute data. Last, Fig. 1.2(c) demonstrates the corrupted geometry data negatively affects the future frames' geometry data. *Because of cataclysmic distortion on the reconstructed 3D point clouds, novel error concealment techniques that are robust to error propagation caused by dynamic network conditions are needed.*

Chapter 5

Error Concealment Framework

In this section, we propose a suite of techniques to conceal errors for dynamic 3D point cloud streaming. Because the metadata (such as video, sequence, and picture parameter sets) of each bitstream is relatively small and critical, we assume proper error protection schemes are applied on them. We note that concealing loss of attribute data (but geometry data are received) is relatively easy – our algorithm simply copies the attribute of the closest point in the previously received frame over. In the rest of this section, we focus on the harder cases in our error concealment algorithm, where geometry data (in GVD) (and possible attribute data) is missing entirely. Furthermore, we consider the worst-case situation, where one or multiple 3D point cloud frames are lost. This kind of method is also the default behavior of the V-PCC reference decoder. That is, the decoder wouldn't reconstruct partial point cloud frames.

5.1 Design Space

Table 5.1: Error Concealment Schemes

Name	f_1	f_2	Motion Estimation	f'_2	Matching (m)	Prediction $P(\cdot,\cdot)$
3DFC	Closest frame	-	-	-	$m(p_1) = p_1$	$P(p_1, m(p_1)) = p_1$
PI	Previous frame	Next frame	-	$f'_2 = f_2$	most similar point in f_2	interpolates between p_1 and $m(p_1)$
TI	Previous frame	Next frame	-	$f'_2 = f_2$	most similar triangle in f_2	interpolates between p_1 and $m(p_1)$
CMI	Previous frame	Next frame	Cube-based motion	$f'_2 = f_1 + M$	-	$f_3 = f'_2$
NCI	Previous frame	Next frame	Cube-based motion	$f'_2 = \Sigma_{i=1}^{27}(M_i/V_i)/\Sigma_{i=1}^{27}(1/V_i)$ where $M_i = (x_i, y_i, z_i), V_i = \lvert x_i\rvert \times \lvert y_i\rvert \times \lvert z_i\rvert$	-	$f_3 = f'_2$

To ease the presentation of our proposed techniques, we first present a general framework for the design space of 3D point cloud error concealment techniques, as a sequence of steps. Each step can be performed with a different technique or skipped, leading to a combinatorial explosion of error concealment methods. As such, we intentionally keep our description of this design space vague and abstract in this section. Concrete realization of each step will be described in details in subsequent sections.

In our framework, the first step in error concealment is *motion estimation*. A major challenge for 3D point cloud error concealment is that there is no explicit correspondence between two point clouds – unlike video where there is a pixel-to-pixel correspondence. Taking inspiration from the literature in error concealment for 2D videos, the first step in error concealment for 3D point cloud is to estimate the motion between two 3D point clouds. Algebraically, we denote $M \approx f_1 - f_2$, where M is the motion, f_1 and f_2 as two frames in the point cloud sequence.

After the motion is estimated, the algorithm can compensate for the motion by translating the points according to the estimated motion. This *motion compensation* step aims to bring the points in one point cloud to the estimated position in the second point cloud, i.e., we compute $f_2' = f_2 + M$.

We then synthesize a point cloud f_3, by predicting each point's position and attributes, from f_1 and f_2'. Each point $p_1 \in f_1$ has a corresponding point $p_2 = m(p_1) \in f_2'$. We then predict a point $p_3 \in f_3$ with a function $P(p_1, m(p_1))$.

Table 5.1 outlines four methods that we will present in this paper and how they fit into this design space: 3DFC (3D Frame Copy) is a naïve scheme that copies the closest, non-loss, frame as the concealed frame. The prediction P simply copies a point over. No motion estimation nor compensation is done for 3DFC. PI is a Point-to-Point Interpolation scheme. Again, no motion estimation nor compensation is done. PI takes each point in the previous frame and finds the most similar point, in terms of geometry and attributes, in the next frame, and interpolates the concealed point between the two. TI (Triangular Interpolation) is similar to PI, except that it considers also the neighborhood information, by computing the similarity of three points in terms of geometry and attributes. Finally, CMI (Cube-based Motion Interpolation) performs block-based motion estimation between two frames, and then translates each block using part of the motion to generate the new frame.

Other possibilities within this design space include extrapolation-based schemes, where f_1 and f_2 are two previous frames and P extrapolates from p_1 and p_2. We elaborate on four methods below.

5.2 3D Frame Copy (3DFC)

We begin by presenting 3D Frame Copy (3DFC), a naïve scheme that simply copies the closest available 3D point cloud for concealment. Suppose that we have two frames f_p and f_n, and we wish to generate a missing frame f_c that falls between f_p and f_n. We denote the frame (between f_p and f_n) closest to f_c as the *source frame*, and the other as the *target frame*, breaking ties by preferring the previous frame.

3DFC simply replicates the points in the source frame as f_c. 3DFC is simple and fast,

Table 5.2: Symbols Used in This Paper

Sym.	Description	Sym.	Description
f_p	Previous point cloud frame	t_p	Triangle in f_p
f_c	Current point cloud frame	t_n	Triangle in f_n
f_n	Next point cloud frame	m_n	Motion vector for a point
a_p	Anchor point cloud frame	ta_p	Target point cloud frame
p_p	Point in f_p	l_t	List of matched triangle
p_c	Point in f_c		pairs (t_p, t_n)
p_n	Point in f_n	u	Edge length of motion
p_a	Point in a_p	p_o	Point in o_p
k_p	KD tree of f_p		estimation cubes
k_n	KD tree of f_n	i	Number of neighboring
$k_t a$	KD Tree of ta_p		
	l_s	length of a side	
l_p	List of matched point pairs		points in c_n
	(p_p, p_n)	d_n	Neighboring triangles in f_n
s	Rendering point size	α	Weight of coordinate
e_n	Estimated point in f_n		distance
	corresponding to $p_p \in f_p$	β	Weight of color distance
c_n	Neighboring points in f_n	τ	Searching radius for c_n

but it causes the playback to appear stutter, since the same frame appears multiple times. This effect is evident when there are multiple frames being interpolated between f_p and f_n.

5.3 Point-to-Point Interpolation (PI)

A better approach to generate a concealed frame is to perform temporal interpolation between f_p and f_n. The challenge for doing this is the lack of explicit point-to-point correspondence among the points in f_p and f_n. Furthermore, 3D motion vectors are not encoded in the bitstream.

Let f_s be the source frame and f_t be the target frame. To generate the interpolated frame f_c, the algorithm iterates through every point p in f_t, and looks for the most similar point $m(p)$ from among the points in f_s:

$$m(p) = \arg\min_{q \in f_p} \Delta(p, q), \tag{5.1}$$

where $\Delta(p, q) = \alpha\Delta_g(p, q) + (1-\alpha)\Delta_a(p, q)$, Δ_g is can be either Chebychev or Euclidean distance between the coordinates of two points; Δ_a is the Euclidean distance between the color (RGB) vectors of the two points. α is a parameter to tune the relative importance between the two terms of the utility function.

Naively iterating through all points in f_s is slow. In practice, for performance, the

Algorithm 1 Point-to-Point Interpolation (PI) Algorithm

```
Build KD trees $k_p, k_n$ for frames $f_p, f_n$
for point $p_p$ in $f_p$ do
    Use $k_n$ to find neighboring points $c_n$ within a radius $\tau$
    Find the best matching point $p_n$ in $c_n$ using Eq. (5.3)
    Interpolate a point $p_c$ using $(p_p, p_n)$
    $f_c = f_c \cup \{p_c\}$
Render $f_c$ with point size $s$
```

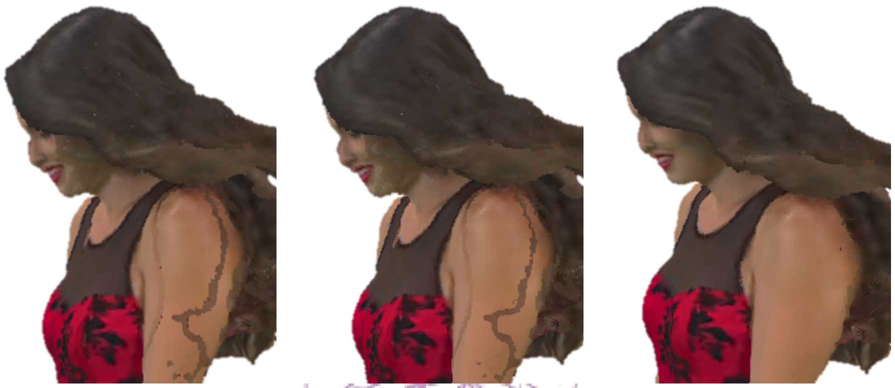


Figure 5.1: Sample concealed point cloud using: (a) PI, (b) TI, and (c) CMI.

algorithm builds a KD-tree containing the points in f_s, and uses the tree to query for a set of points $c(p, \tau) \subseteq f_s$ that is within Euclidean distance τ away from p. We only iterate through the points in $c(p, \tau)$ to find the match $m(p)$. For $c(p, \tau)$ is empty, then we let $m(p) = p$. In other words,

$$m(p) = \begin{cases} p, & \text{if } c(p, \tau) \text{ is } \phi; \\ \underset{q \in c(p,\tau)}{\arg\min}\, \Delta(p, q), & \text{otherwise.} \end{cases}$$

Here, the parameter τ is a control knob, where larger τ leads to potentially more optimized matched point at a higher computational cost. We found that $\tau = 2$ is a good setting for balancing between quality and speed in our experiments. Algorithm 1 gives the pseudocode of the PI algorithm. We have exercised several design variations for $m(p)$ in our pilot tests. For example, we have found that using *point-to-plane* distance function for Δ_g results in significantly worse quality of concealed point cloud frames. Hence, we stick with point-to-point distance throughout this paper. Moreover, we found that the two terms in Eq. (5.1) may differ a lot in scales and thus decided to normalize Δ_g and Δ_a with a factor $1/\sqrt{3m^2}$, where m is the maximum possible value (for coordinates and colors).

Algorithm 2 Triangular Interpolation (TI) Algorithm

```
Build KD trees $k_p$, $k_n$ for frames $f_p$, $f_n$
for point $p_p$ in $f_p$ do
    Find two closest points to form a triangle $t_p$ in $f_p$
    Use $k_n$ to find $i$ neighboring points $c_n$
    Permute 3 points from $c_n$ into a set of triangles $d_n$
    Find the best matching triangle $t_n$ in $d_n$ using Eq. (5.2)
    Interpolate each point of ($t_p$,$t_n$) into three $p_c$
    $f_c = f_c \cup \{p_c\}$
Render $f_c$ with point size $s$
```

5.4 Triangular Interpolation (TI)

Although our PI algorithm is fairly simple, it produces artifacts in the interpolated point cloud frame f_c, as shown in Fig. 5.1(a). One of the causes is that the PI algorithm matches each point independently, without considering its neighborhood, and thus a point might be matched incorrectly. We therefore considered extending PI to consider the geometry and attributes of nearby points when computing the similarity between points.

To consider the neighbors, we perform point matching in a group of k points ($k > 1$) instead of one-by-one. For each point p in f_s, we first find k points closest to it, forming a sequence of points $\mathcal{N}(p)$ (in arbitrary order). Similar to PI, we use the KD-tree to query for a set of points $c(p,\tau) \subseteq f_s$ that is within a distance τ away from p. We then iterate through all k-permutations of points in $c(p,\tau)$, finding the best sequence of k points using a utility function. If $c(p,\tau)$ has fewer than k points, then we copy the points from $\mathcal{N}(p)$ to the concealed frame.

When $k = 1$, this is just PI. We pick $k = 3$ to balance between computation complexity and the amount of neighbor information considered in matching. We call this method triangular interpolation (TI).

We extend the notation m, Δ, Δ_g and Δ_a to take in an ordered list of points as input. Δ_g and Δ_a sums the pair-wise differences between the geometry coordinates and colors of the points in the lists, respectively. In other words, TI computes $m(\mathcal{N}(p))$ as

$$m(\mathcal{N}(p)) = \begin{cases} \mathcal{N}(p), & \text{if } |c(p,\tau)| < k; \\ \underset{S \subseteq \pi_k(c(p,\tau))}{\arg\min} \ \Delta(\mathcal{N}(p), S), & \text{otherwise.} \end{cases} \tag{5.2}$$

Once TI finds the matching sequence of k points $m(\mathcal{N}(p))$, TI computes a pair-wise interpolation to produce k interpolated points for p. Note that each point can appear in the neighborhood of multiple points, and thus can be interpolated more than once. The resulting point cloud for TI will have more points than the original source point cloud.

5.5 Cube Motion Interpolation (CMI)

We found that, while TI algorithm can lead to smoother surfaces and remove many small holes in the model, for larger cracks in the surface of interpolated point clouds, TI is only able to reduce its size, and not "sealing" it entirely (see Fig. 5.1(b) for an example). We could increase the number of points in the neighborhood, k, to improve the quality, but as TI's computational complexity grows exponentially with k, we decided to consider another method using a different paradigm.

The insights to our next method, Cube Motion Interpolation (CMI), stem from the source of cracks appearing on the point cloud surfaces. A crack appears when points that are close to each other in the source frame get matched with points that are far apart in the target frame. Since the vector from p to $m(p)$ is supposed to approximate the motion, this means that two points that are close to each other in the source frame move in two different directions, which is rare for rigid objects.

CMI ensures that a block of the points moves in a consistent direction to avoid creating cracks in the generated frames. It works as follows. First, CMI uses Eq. (5.1) to find $m(p)$ for every point p in the source f_s. Then, it partitions the points in f_s into non-overlapping cubes, and treats the displacement between $m(p)$ and p as the motion vector, CMI computes the average motion vector $\bar{m}_C$ for each cube C. Finally, CMI generates the interpolated frame by adding the motion $r\bar{m}_C$ to each point $p \in C$, where $0 \leq r \leq 1$ is the ratio of temporal distance between the generated frame and the source frame. Let us denote this step as the interpolation step.

The advantage of CMI is that the rigid motion of cubes eliminates the possibility of cracks. There are, however, possibilities of gaps between two cubes after interpolation. To deal with this issue, we can enlarge the size of the cube (so that it now overlaps with the adjacent cubes). We use the following heuristic to find the enlargement of a cube to cover the gaps.

Let l be the length of each cube C. In the source frame, the distance between every center of every cube is exactly l. If, after interpolation, the center of two adjacent, interpolated, has a length l' larger than l, then there is a possibility of visible gaps in the 3D point cloud. We thus enlarge the size of C, from l to l', and redo the interpolation step above. To minimize the number of times CMI needs to repeat interpolation steps above, CMI always enlarge the cubes in decreasing order of the magnitude of their motion. Fig. 5.1(c) shows a concealed point cloud frame without cracks.

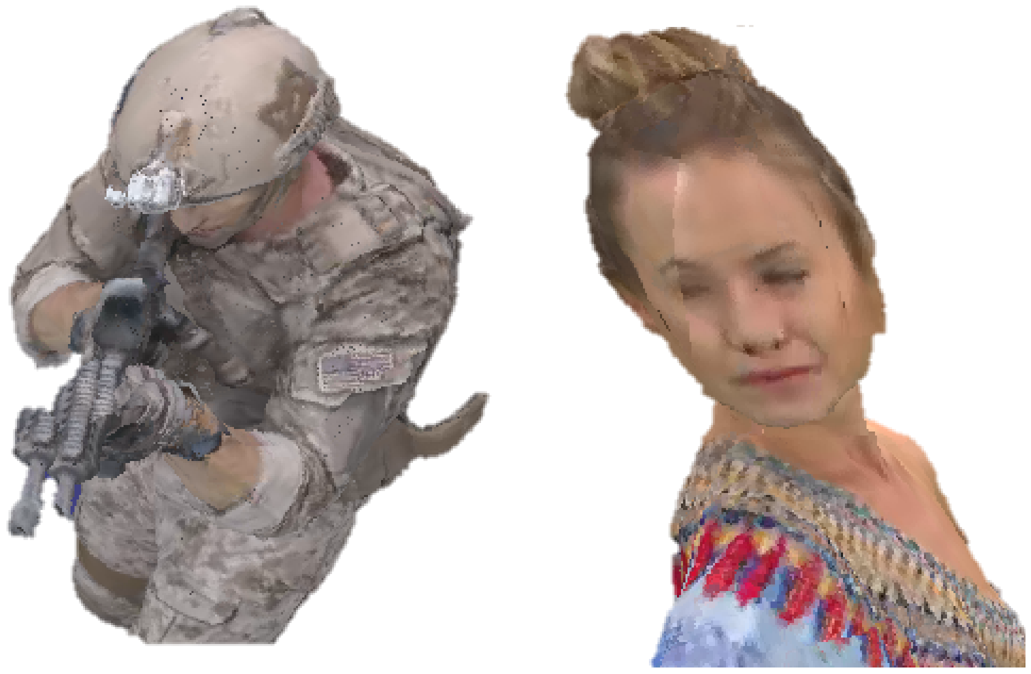


Figure 5.2: Less than perfect concealed point clouds due to: (a) rotation of the rifle and (b) extrusion of facial expression.

5.6 Neighbor-Cube Interpolation (NC)

Although we fix the issue of gap between each cube after the rigid transformation of each cube, we haven't addressed the issue of extrusion, as shown in Fig 5.2. We decided to consider the rigid motion from neighbor cubes and use the interpolated motion weighted by the distance to neighbor center as the motion vector for each point. This method eliminates the probability of that points near the boundaries of cubes move in the totally different direction which cause either gaps or extrusions.

5.7 Extrapolation

Each of the algorithm above, PI, TI, and CMI, have an extrapolation variant, which we termed PE, TE, and CME, respectively. The algorithms for works exactly the same way, except that: (i) instead of using the previous frame and the next frame, the extrapolation variants use the two previous frames, with the immediate previous frame f_p as the source, and (ii) instead of interpolating between points in the two frames, they extrapolate.

Algorithm 3 Cube-Based Motion Interpolation (CMI) Algorithm

```
Tar_iter = 2 // target iteration, iteration for while loop
Cur_iter = 0 // current iteration count
Uc_flag = 0 // stop flag if all scaler unchanged
Build KD trees $k_t a$ for frames $ta_p$
Divide $a_p$ into cubes with the same $l_s$ and store the center of non-empty cubes into list
C
Scaler = [1 * C.length]
// list of list storing neighbor cubes for each cube
Neighbors = [[] * C.length]
Search and record the neighbor cubes for each cube into Neighbors
while Uc_flag == 0 and Cur_iter ¡= Tar_iter do
    Cur_iter += 1 {/}/ list of motion for each cube
    Motions = []
    for i = 1 to C.length do
        // query points in cube with scaled cube size
        P = query_points(C[i], Scaler[i], $l_s$)
        M = [] // We will store motion the for whole block for avg
        for each point p in P do
            Use $k_t a$ to find the nearest neighboring point $ta_p$ by Eq. (5.1)
            $m_n$ = $ta_p$ - p
            M.append($m_n$)
        Motions.append(mean(M))
    sort the motion by magnitude in Motions and store the index by descending order in
    to M_idx
    Scaler_cur = [0 * C.length] // 0 means not update yet, we store the final scaler outside
    the while loop
    for i = 1 to M_idx.length do
        Old_Neighbors = Neighbor[i]
        Search and record the neighbor cubes for $C[i] + Motion[i]$ into New_Neighbors
        for j = 1 to Old_Neighbors.length do
            Candidate_list = []
            if not(Old_Neighbors[j] in New_Neighbor[j]) and Scaler_cur[Old_Neighbors[j]]
            == 0 then
                Candidate_list.append(Old_Neighbors[j])
            if Candidate_list.length == 0 then
                Scaler_cur[j] = 1
                continue
            S = the farthest center of candidate from current neighbor
            Scaler_cur[i] = S / $l_s$
    Uc_flag = 0
    for i = 1 to Scaler_cur.length do
        if Scaler_cur[i] ¿ Scaler[i] then
            Scaler[i] = Scaler_cur[i]
            Uc_flag = 1
```

Algorithm 4 Neighbor-Cube Interpolation (NCI) Algorithm

```
Tar_iter = 2 // target iteration, iteration for while loop
Cur_iter = 0 // current iteration count
Uc_flag = 0 // stop flag if all scaler unchanged
Build KD trees $k_t a$ for frames $ta_p$
Divide $a_p$ into cubes with the same $l_s$ and store the center of non-empty cubes into list
C
Scaler = [1 * C.length]
// list of list storing neighbor cubes for each cube
Neighbors = [[] * C.length]
Search and record the neighbor cubes for each cube into Neighbors
while Uc_flag == 0 and Cur_iter ¡= Tar_iter do
    Cur_iter += 1 {/}/ list of motion for each cube
    Motions = []
    for i = 1 to C.length do
        // query points in cube with scaled cube size
        P = query_points(C[i], Scaler[i], $l_s$)
        M = [] // We will store motion the for whole block for avg
        for each point p in P do
            Use $k_t a$ to find the nearest neighboring point $ta_p$ by Eq. (5.1)
            $m_n$ = $ta_p$ - p
            M.append($m_n$)
        Motions.append(mean(M))
    sort the motion by magnitude in Motions and store the index by descending order in
    to M_idx
    Scaler_cur = [0 * C.length] // 0 means not update yet, we store the final scaler outside
    the while loop
    for i = 1 to M_idx.length do
        Old_Neighbors = Neighbor[i]
        Search and record the neighbor cubes for $C[i] + Motion[i]$ into New_Neighbors
        for j = 1 to Old_Neighbors.length do
            Candidate_list = []
            if not(Old_Neighbors[j] in New_Neighbor[j]) and Scaler_cur[Old_Neighbors[j]]
            == 0 then
                Candidate_list.append(Old_Neighbors[j])
            if Candidate_list.length == 0 then
                Scaler_cur[j] = 1
                continue
            S = the farthest center of candidate from current neighbor
            Scaler_cur[i] = S / $l_s$
    Uc_flag = 0
    for i = 1 to Scaler_cur.length do
        if Scaler_cur[i] ¿ Scaler[i] then
            Scaler[i] = Scaler_cur[i]
            Uc_flag = 1
```

Chapter 6

Implementations and Experiments

6.1 Implementations

We have implemented our proposed error concealment algorithms in C++ using PCL [49] libraries and other system utility functions in Python. For comparisons, we have also realized 2DFC and 3DFC as the baselines since we are not aware of any other error concealment algorithms designed for 3D point cloud streaming in the literature. We use No Loss (NL) as a benchmark. We emphasize that while approaching NL is the ultimate goal of error concealment, achieving NL is virtually impossible under typical scenarios. We use V-PCC reference software version 11 [43] as our point cloud codec. To emulate dynamic 3D point clouds streamed over the Internet, we implement a Gilbert-Elliot (G-E) model [25] to generate packet (NALU) losses. We let q be the probability of transiting from the good to the bad state, while $1 - q$ be the that from the bad to the good state. The distorted V-PCC bitstreams are decoded by the V-PCC reference decoder, and then repaired by different error concealment algorithms.

6.2 Experiments Setup

Table 6.1: Dynamic 3D Point Cloud Sequences

	Queen	**Loot**	**Red&Blk**	**Soldier**	**LongDress**	**Basketball**	**Dancer**
Cplx.	Low	Low	Low	Low	Medium	High	High
Pt.#	1.00 M	0.78	0.70	1.50	0.80	2.90	2.60

We employ seven MPEG dynamic 3D point cloud sequences with different complexity levels [16] for our experiments, as listed in Table 6.1. Each sequence contains a human actor, which is a representative object class in tele-conferences. We select the first 250 point cloud frames from each sequence, and encode them with All-Intra mode (which is

Table 6.2: Statistics of GE drops

	Queen	Loot	Red&Blk	Soldier	LongDress	Basketball	Dancer
5% avg geo drop	22.67	24.33	20.33	22.67	26.33	23	19
5% avg color drop	7	6.0	6.33	5	6.67	4	5
5% avg drop length	1.07	1.06	1.06	1.08	1.07	1.1	1.07
5% max drop length	3	3	2	2	2	2	3
10% avg geo drop	44	46.33	47.67	47.67	41	50.33	46.67
10% avg color drop	8.6	10.66	9.67	8.67	12.67	8	10
10% avg drop length	1.12	1.12	1.21	1.1	1.11	1.16	1.2
10% max drop length	3	3	5	3	3	4	3
15% avg geo drop	60	69.67	67	75.33	69.33	66.33	65.67
15% avg color drop	11	7.33	12	12	10.33	10.33	13
15% avg drop length	1.27	1.31	1.22	1.23	1.3	1.3	1.21
15% max drop length	3	4	4	4	4	5	4

Table 6.3: No. Frames 2DFC Fails to Conceal (Out of 750)

PLR	Queen	Loot	Red&Blk	Soldier	LongDress	Basketball	Dancer
5%	37	42	28	33	35	36	30
10%	73	74	81	78	68	69	70
15%	88	109	111	108	113	109	98

more error resilient) of V-PCC at 20 frame-per-second. We analyze overall results from: (i) all frames (including the frames not affected by packet losses), (ii) distorted frames (including both color- and geometry-distorted ones), and (iii) geometry-distorted frames. We adopt PCC Arena [65] to compute the 3D quality of concealed point cloud frames. For 2D metrics, we also follow PCC Arena to use Open3D library [71] to render point clouds into six 2D images along the x, y, and z axes. The difference is that we upgrade Open3D to 0.14.1 with the new renderer class whose backend is filament [18] and we use ImageMagick [57] to generate masks to eliminate backgrounds and calculate 2D metrics. We report the average quality among them, if not otherwise specified. We consider the following performance metrics:

- *GPSNR*: The PSNR of Chamfer distance between pair-wise closest points in the target and reference frames.
- *Hausdorff distance*: The maximal shortest distance between the points in the target and reference frames.
- *CPSNR:* The luminance component of color distortion between the nearest points in the target and reference frames.
- *PSNR*: The PSNR of the foreground object (avatar) only, computed with ImageMagick [57]. We exclude the background pixels as they are identical in both reference and target rendered images.

- *SSIM*: The luminance SSIM of the foreground object using ImageMagick [57].
- *VMAF:* We compute Video Multi-Method Assessment Fusion [1] to quantify the perceived video quality.
- *Running time:* The per-frame execution time of each error concealment algorithm. We run our algorithms on a single core of an AMD Ryzen 7 5800X CPU at 3.8 GHz.

The first three metrics are 3D quality metrics, while the next three are 2D quality metrics. We carried out a grid search for the hyperparameters, such as the distance function and utility weights. We have found that: (i) the Euclidean distance and (ii) $\alpha = 0.9$ work the best. Moreover, we set the query radius to be 2, the maximal neighboring points to be 5, and the Open3D rendering point size to be 3, if not otherwise specified. We vary the G-E model parameter q to get packet loss rates of 5%, 10%, and 15%. For each packet loss rate, we generate three packet loss traces for 250 point cloud frames, to emulate three streaming runs. Based on our analysis in Sec. 4, we decide to exclude the following NALUs when dropping packets: (i) headers, (ii) the first I frame of a GoF, and (iii) the last GVD's I frame in each GoF. Without doing this, the V-PCC reference decoder would terminate prematurely due to assertion errors. In real-life scenarios, these crucial NALUs (packets) can be protected by, e.g., Forward Error Correction (FEC) or Automatic Repeat Request (ARQ).

6.3 Results

6.3.1 Limitations of The Current 2DFC method.

We first present per-frame visual quality using Queen, Red&Blk, and Dancer sequences, which represents low, intermediate, and high complexity under all G-E parameters packet loss rates. Note that: (i) We remove the frames that failed to concealed by 2DFC, which is in favor of 2DFC, and plot all algorithms for these frame at the value of No Loss. (ii) We select NL, 2DFC, 3DFC, PI and NC algorithms in this subsection. From Fig. 6.1 to Fig. 6.18, all figures clearly shows that the current practice, 2DFC, suffers from serious quality drop under packet losses comparing to our proposed error concealment algorithms: (i) For Queen, the quality drops as high as 33 dB in GPSNR, 12.5 dB in CPSNR, 20 dB in PSNR, 0.32 in SSIM, and surge up to 45 thousands in Hausdorff Distance. (ii) For Red&Blk, the quality drops as high as 17 dB in GPSNR, 13 dB in CPSNR, 12 dB in PSNR, 0.48 in SSIM, and surge up to 35 thousands in Hausdorff Distance. (iii) For Dancer, the quality drops as high as 14 dB in GPSNR, 10 dB in CPSNR, 10 dB in PSNR, 0.37 in SSIM, and surge up to 140 thousands in Hausdorff Distance. We then take a closer look at the error concealment results from 2DFC and find that it fails to conceal a significant number of frames, as illustrated in Table 6.3. *This demonstrates the necessity of 3D error concealment algorithms.*

6.3.2 Cumulative Distribution Function (CDF) Comparison.

We then plot CDF curves, each figure of which collect all three rounds of all distorted frame for each packet loss rate. From Fig. 6.19 to Fig. 6.39, we observe that our error concealment algorithms clearly outperform 2DFC and 3DFC at most of the time. Note that, for lines of 2DFC, we still exclude frames that cannot be concealed. From now on, for other algorithms except for 2DFC, we won't exclude frames that cannot be concealed by 2DFC. For all CDF figures, we can see that the better the quality is, except for Hausdorff distance, the lower and more right the line is. For example, except for Hausdorff distance, NL is always the lowest and the most right line. We also take the Red&Blk sequence who has intermediate complexity level as example. For 3D quality, Fig. 6.26 (a) reveals that, even after excluding the failed frames (see Table 6.3), 2DFC still results in a few very low GPSNR values, as low as less than 30 dB. We can also see all the lines of our algorithms are right to 3DFC algorithm in Fig. 6.26 (b) of CPSNR. For Hausdorff Distance in Fig. 6.26 (c), we can also see all the lines of our algorithms are left to 3DFC algorithm. In contrast, the 20% best performing frames achieved by our PI, TI, CMI, and NC algorithms are above 51 dB, and our CMI and NC algorithms are above 52.5 dB, while that of 3DFC is less than 50 dB. For 2D quality, Fig. 6.26 (d) reveals that, with 2DFC and 3DFC, the 20% best performing frames achieve 17.5 dB and 21 dB in PSNR, while our error concealment algorithms achieve at least 22 dB, and our CMI and NC algorithms can achieve 22.5. We can also see all the lines of our algorithms are right to 3DFC algorithm in Fig. 6.26 (e) of SSIM. For VMAF, Fig. 6.26 (f) reveals that, with 2DFC and 3DFC, the 20% best performing frames achieve 41.25 and 48.75, while our error concealment algorithms achieve at least 55, and lines of our CMI and NC are right to lines of other algorithms.

6.3.3 Bar Chart Analysis.

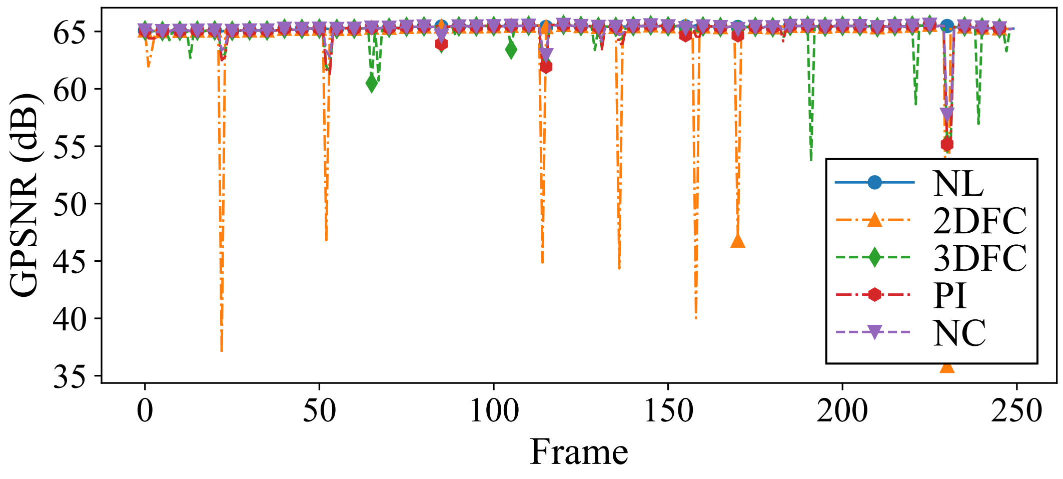
Next, we plot per-sequence overall visual quality results of distorted frames across all packet loss rates in Fig. 6.40 to Fig. 6.45. We give 95% confidence intervals as error bars in this figure. We make three key observations on this figure. First, our proposed error concealment algorithms mostly outperform 2DFC by far even though we remove the frames for 2DFC who cannot conceal some of the frames. Second, our proposed error concealment algorithms mostly outperform 3DFC in terms of visual quality. Third, among the four proposed error concealment algorithms, mostly, TI will outperform PI, and CMI will outperform TI. This demonstrates the effectiveness of better matching approach (of TI) and motion compensation (of CMI) developed when exploring the design space. However, between CMI and NC, there is no clear relationship between either one of them outperforming another.

6.3.4 Improvement Analysis.

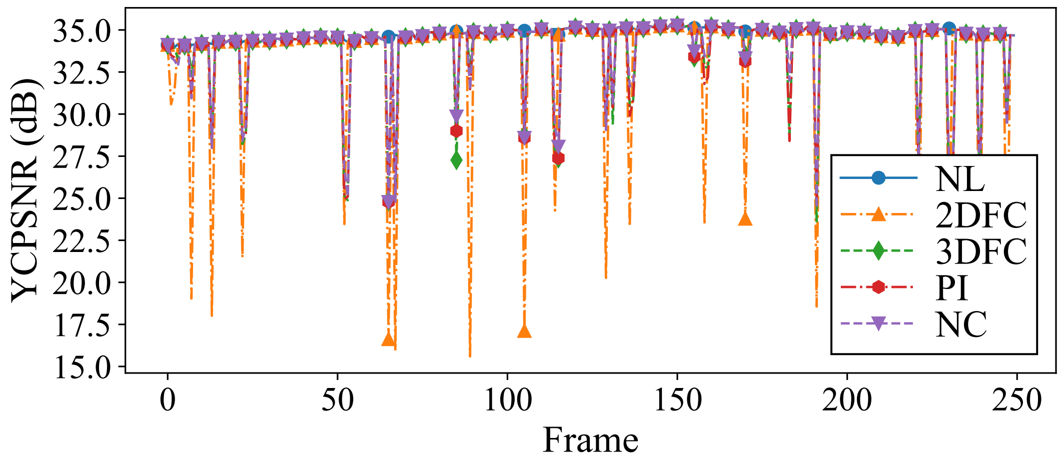
To better understand how much our error concealment algorithms outperform 3DFC, we plot the per-sequence average quality improvement in Fig. 6.46 to Fig. 6.51. We give 95% confidence intervals as error bars in this figure. For GPSNR and Hausdorff distance in Figs. 6.46(a), Figs. 6.46(c), Figs. 6.48(a) and, Figs. 6.48(c) and, Figs. 6.50(a), Figs. 6.50(c), we can see that all our proposed algorithms always outperform 3DFC. In addition, TI always outperform PI, and CMI as well as NC always outperform the other algorithms. Between CMI and NC, there is no clear relationship between either one of them outperforming another. For YCPSNR, mostly, our proposed algorithms outperform 3DFC. For all 2D quality metrics, except for SSIM of NC algorithm for Soldier sequence, all our proposed algorithms outperform 3DFC in all sequences. *In summary, figures of the chapter show that our proposed error concealment algorithms outperform the current practices, 2DFC and 3DFC, in terms of both 3D and 2D visual quality.*

6.3.5 Overhead of our error concealment algorithms.

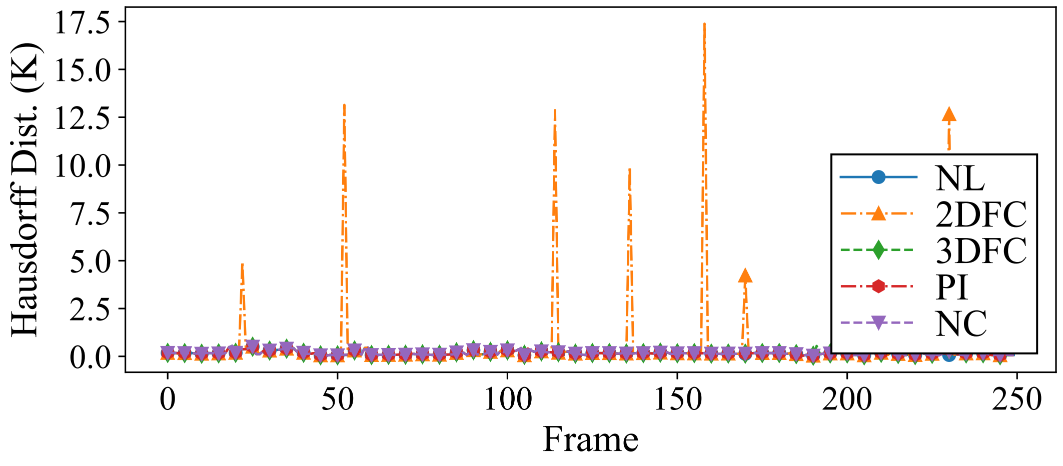
We select 24 random point cloud frames from each considered sequence and measure the running time of our algorithms. We report the average and 95% confidence intervals in Fig. 6.52. Among the three algorithms, PI terminates the fastest: between 0.84 and 3.34 seconds on average across seven sequences. The running time of CMI (with motion compensation) becomes longer compared to TI (considering all vertex permutations) when the complexity level of sequences increases. This shows the importance of efficient motion estimation algorithms, which is among our future tasks. In terms of memory consumption, our proposed PI, TI, and CMI consumes at most 886, 1110, and 840 MB memory (with Basketball sequence). We note that our C++ implementation is not well-optimized. Parallelization techniques, such as Single Instruction Multiple Data (SIMD), General-Purpose Graphics Processing Unit (GPGPU) can be adopted to reduce of execution and memory overhead of our error concealment algorithms. Such optimization is among our future tasks.



(a)

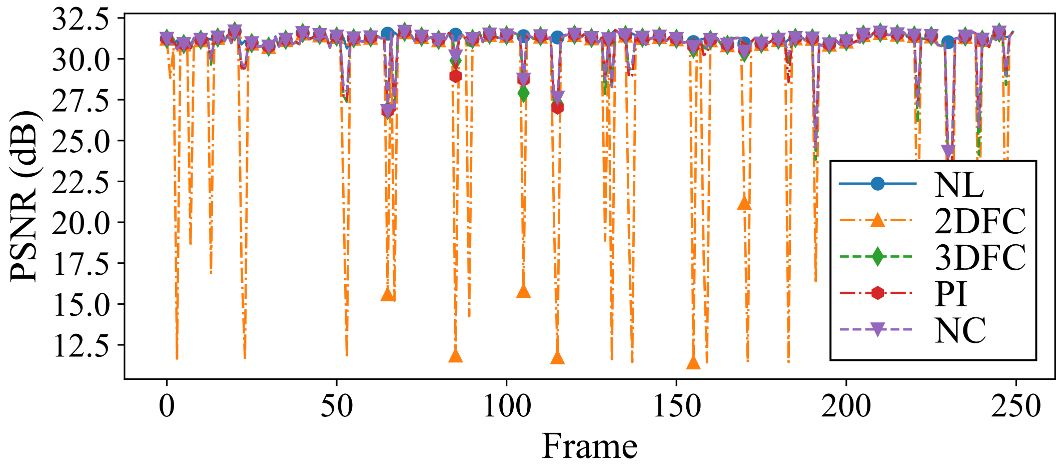


(b)

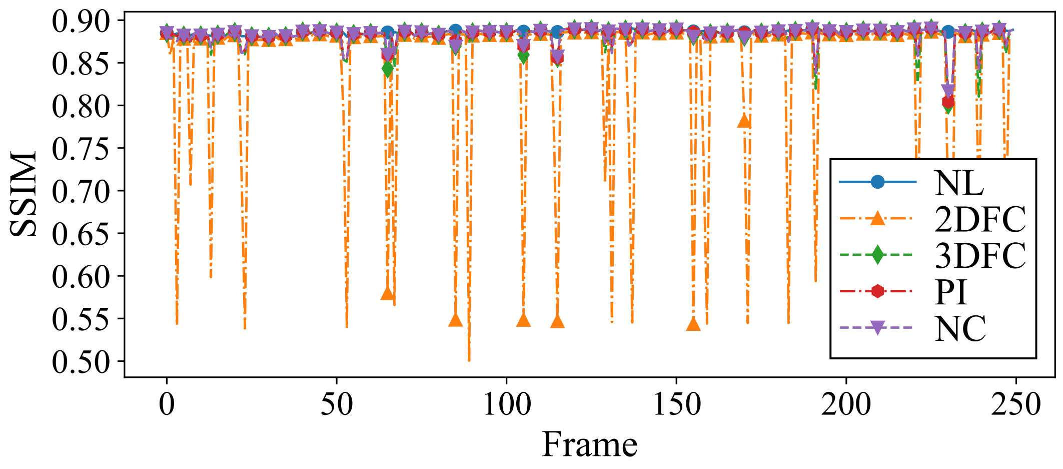


(c)

Figure 6.1: Per-frame 3D visual quality of 5% packet loss from Queen: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

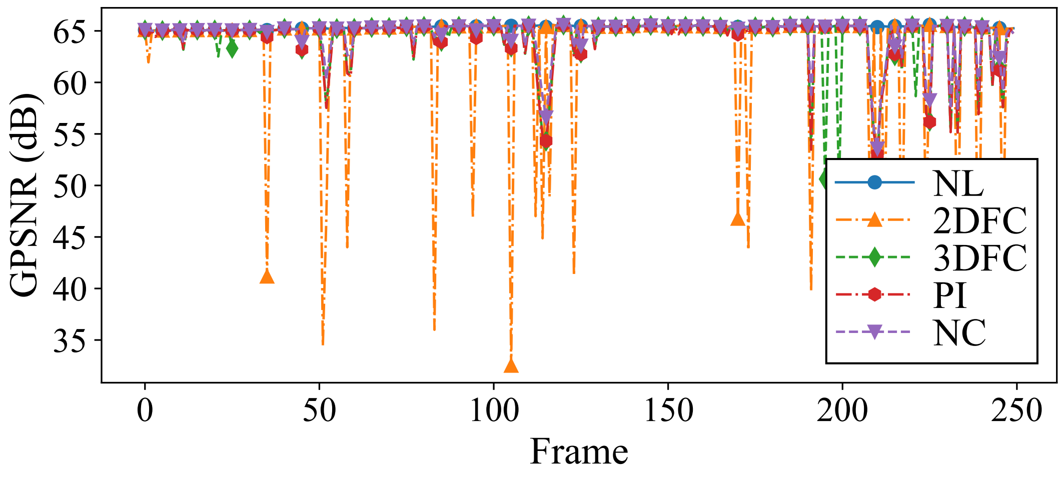


(a)

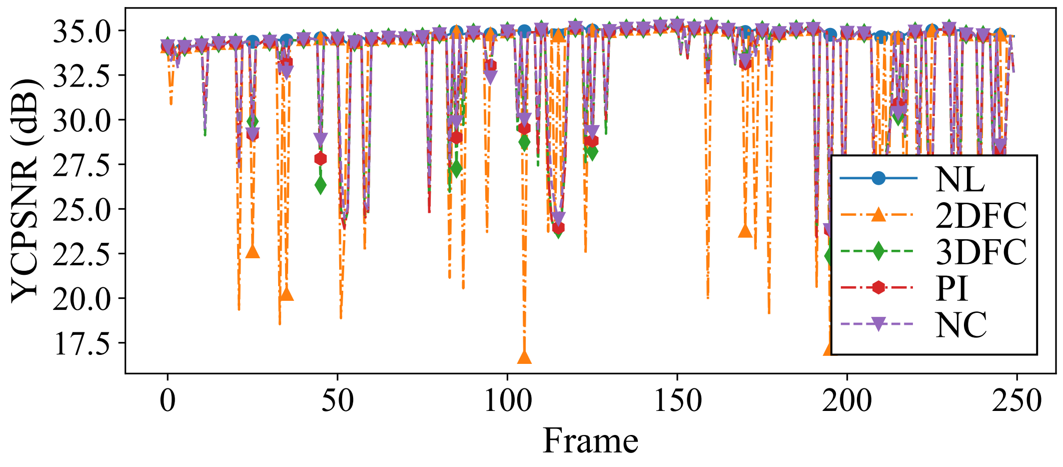


(b)

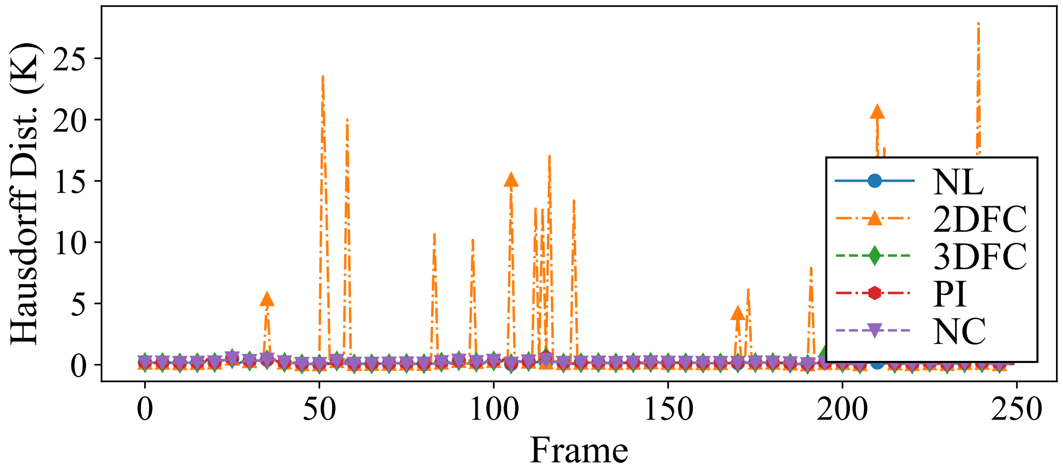
Figure 6.2: Per-frame 2D visual quality of 5% packet loss from Queen: (a) PSNR, and (b) SSIM.



(a)

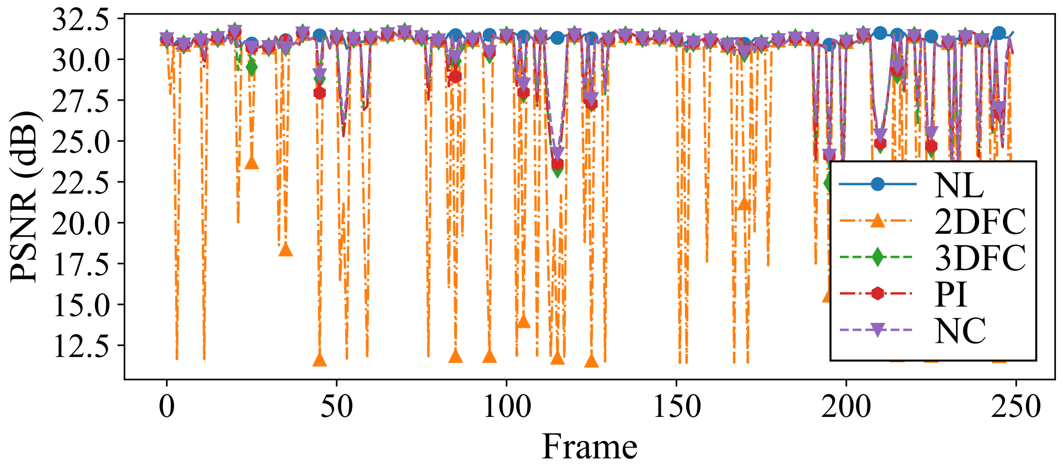


(b)

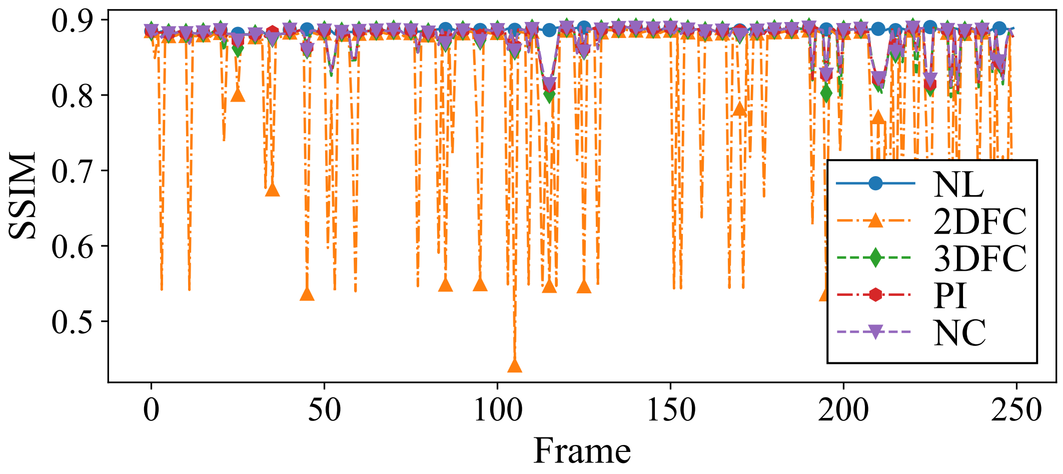


(c)

Figure 6.3: Per-frame 3D visual quality of 10% packet loss from Queen: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

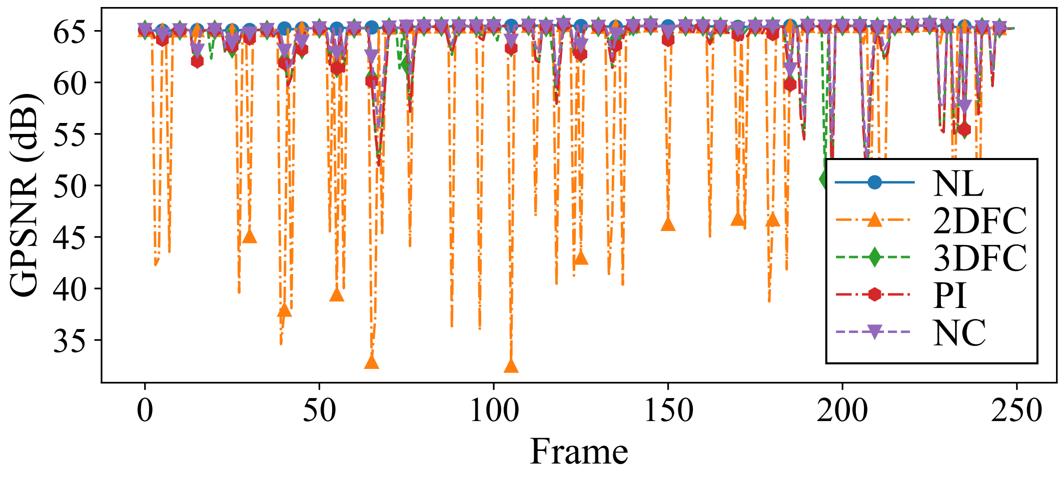


(a)

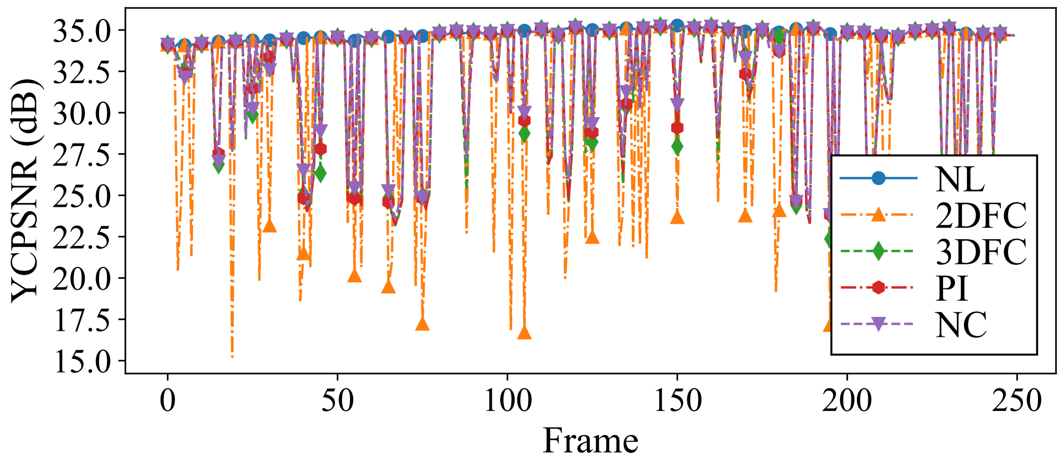


(b)

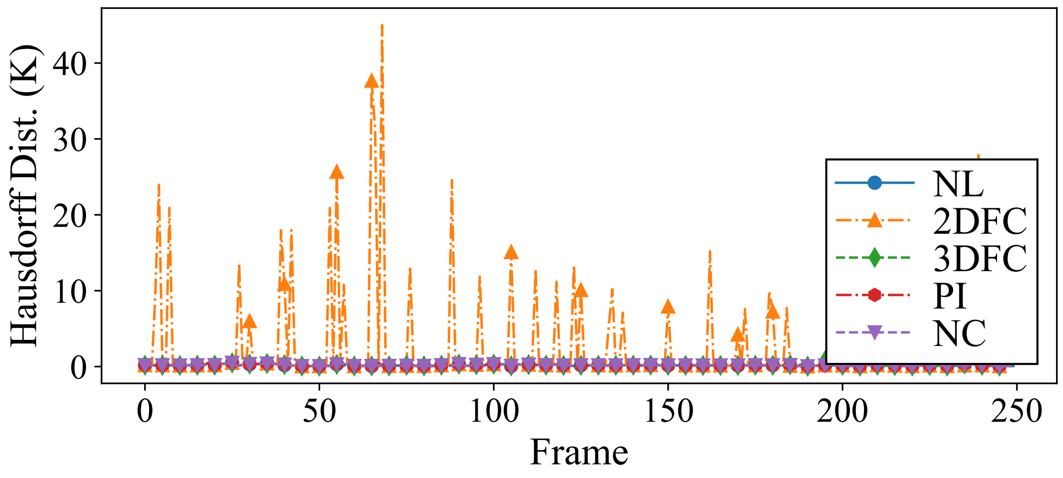
Figure 6.4: Per-frame 2D visual quality of 10% packet loss from Queen: (a) PSNR, and (b) SSIM.



(a)

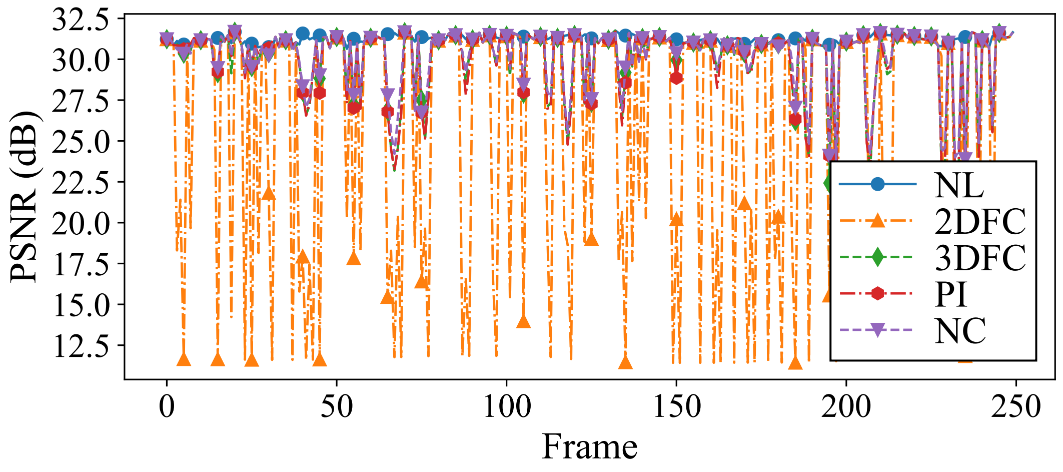


(b)

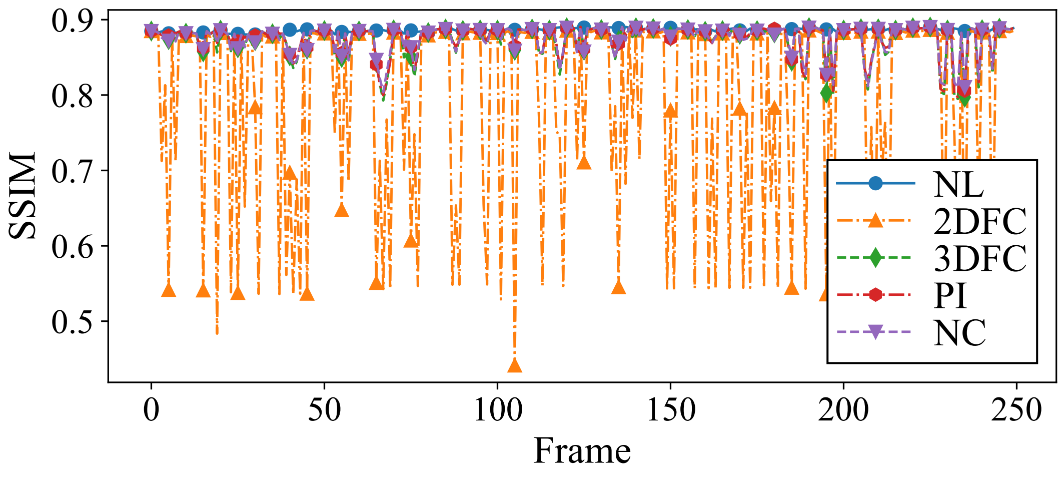


(c)

Figure 6.5: Per-frame 3D visual quality of 15% packet loss from Queen: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

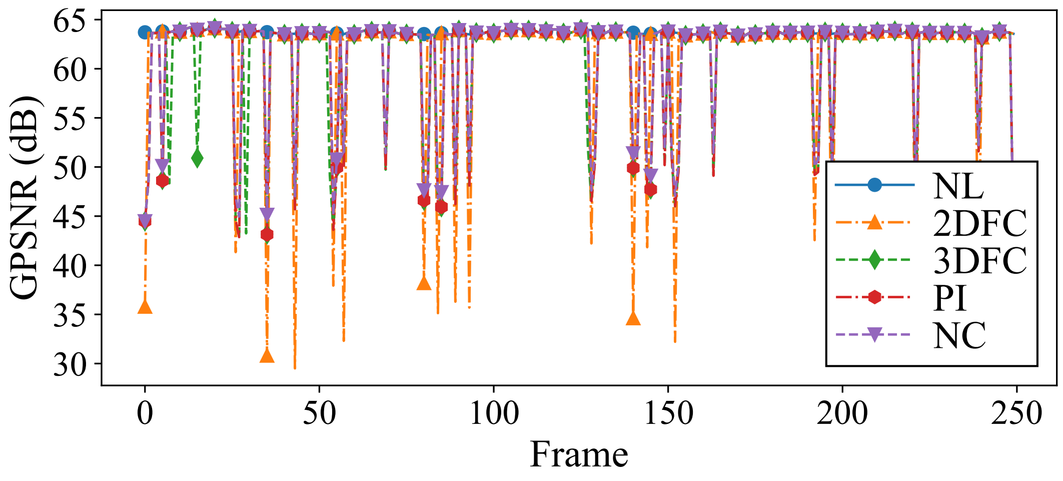


(a)

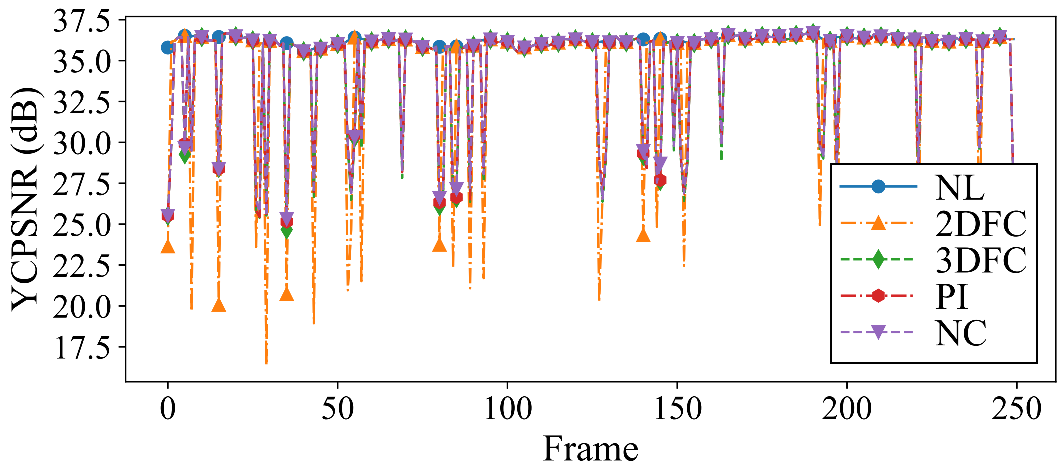


(b)

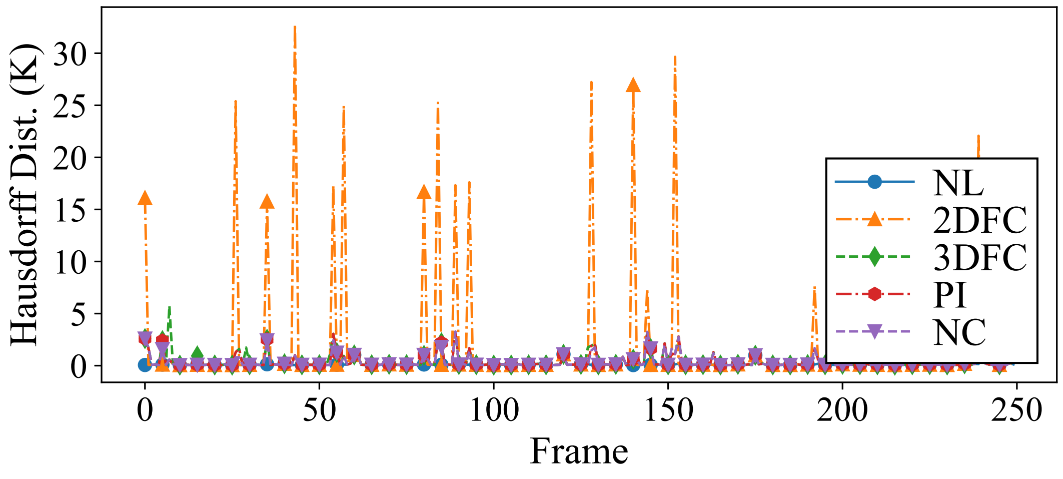
Figure 6.6: Per-frame 2D visual quality of 15% packet loss from Queen: (a) PSNR, and (b) SSIM.



(a)

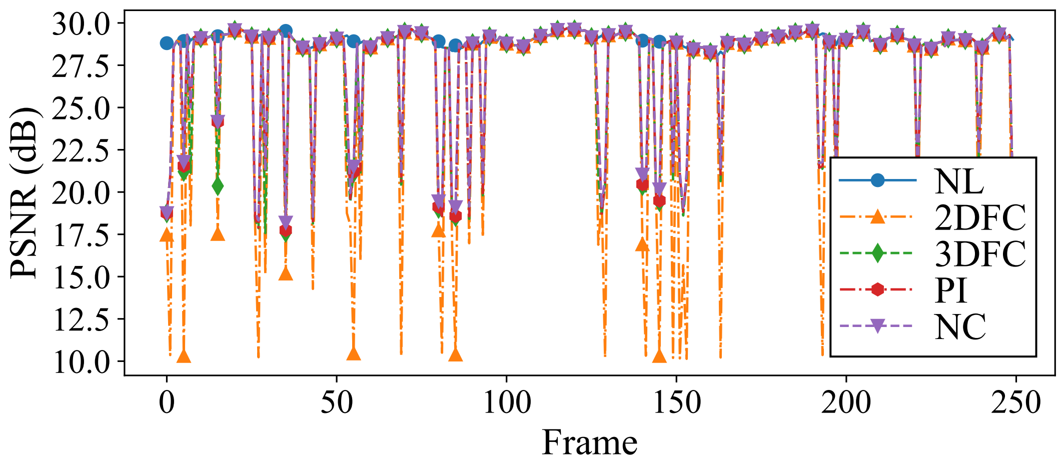


(b)

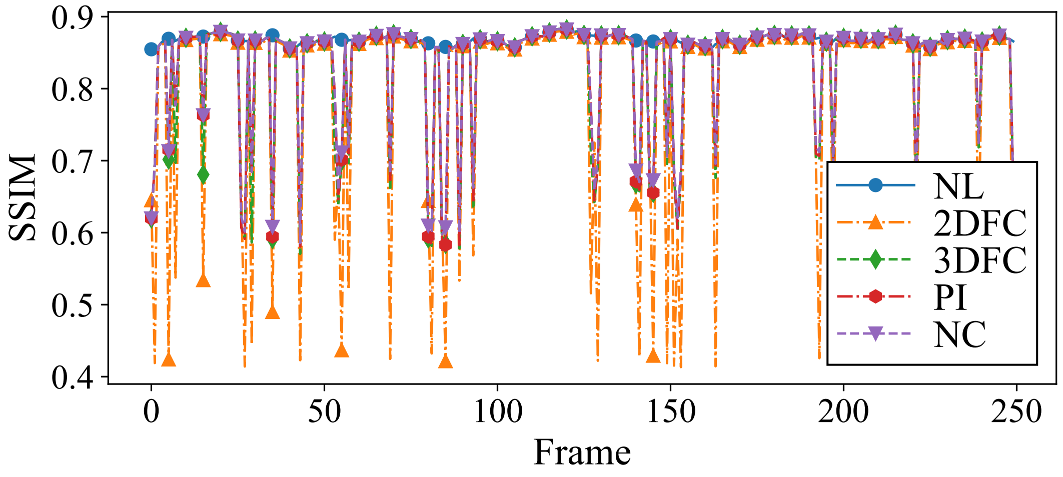


(c)

Figure 6.7: Per-frame 3D visual quality of 5% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

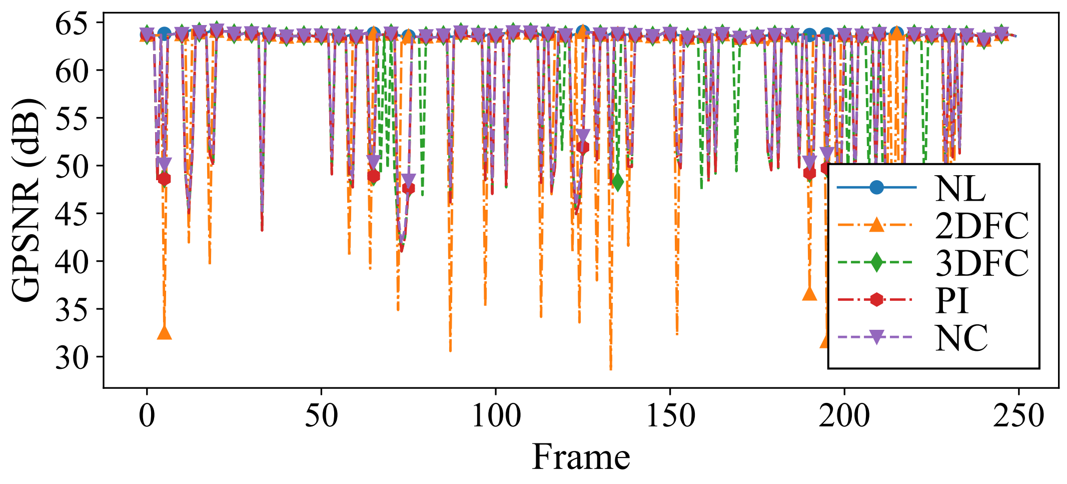


(a)

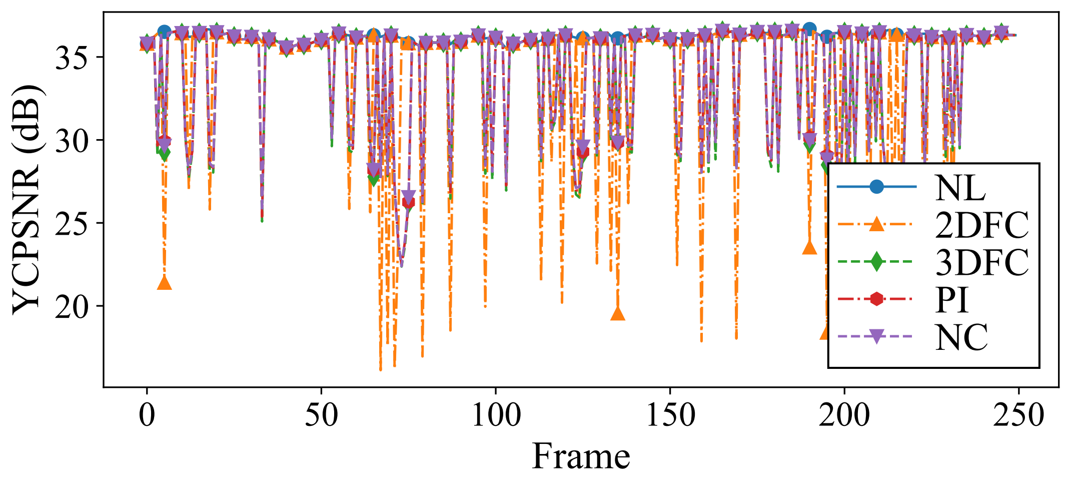


(b)

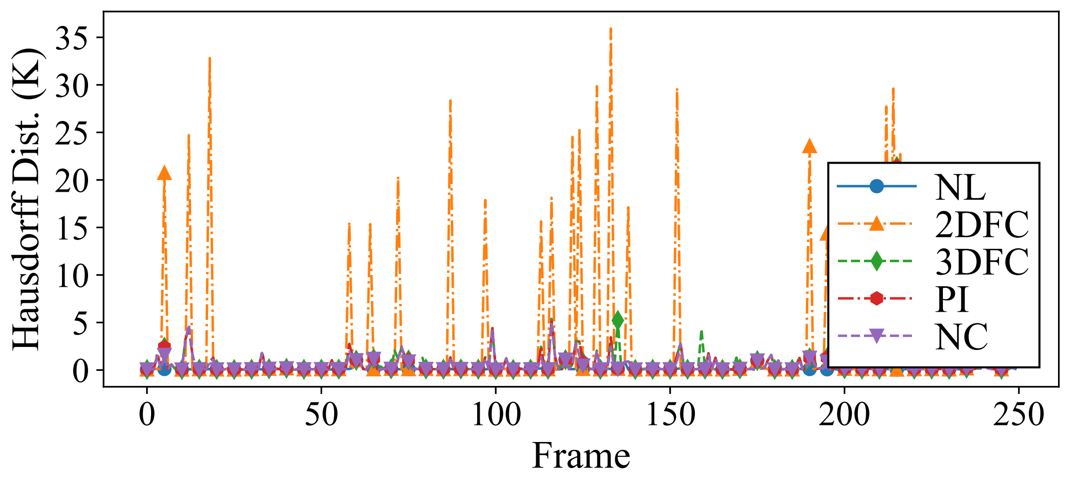
Figure 6.8: Per-frame 2D visual quality of 5% packet loss from Red&Blk: (a) PSNR, and (b) SSIM.



(a)

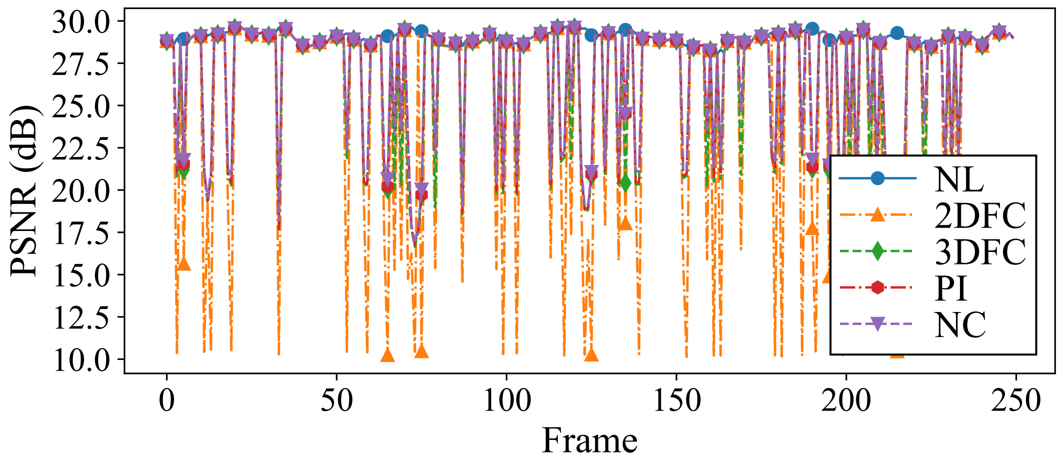


(b)

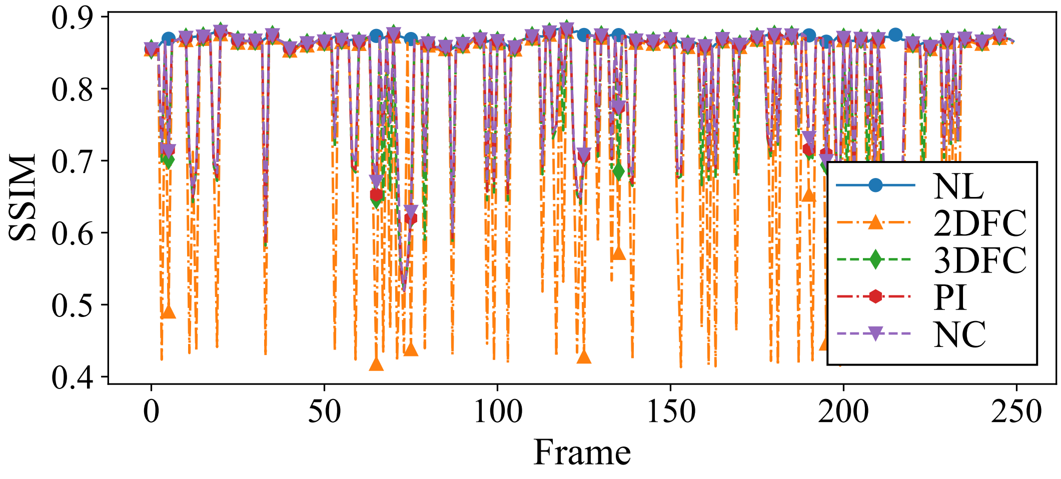


(c)

Figure 6.9: Per-frame 3D visual quality of 10% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

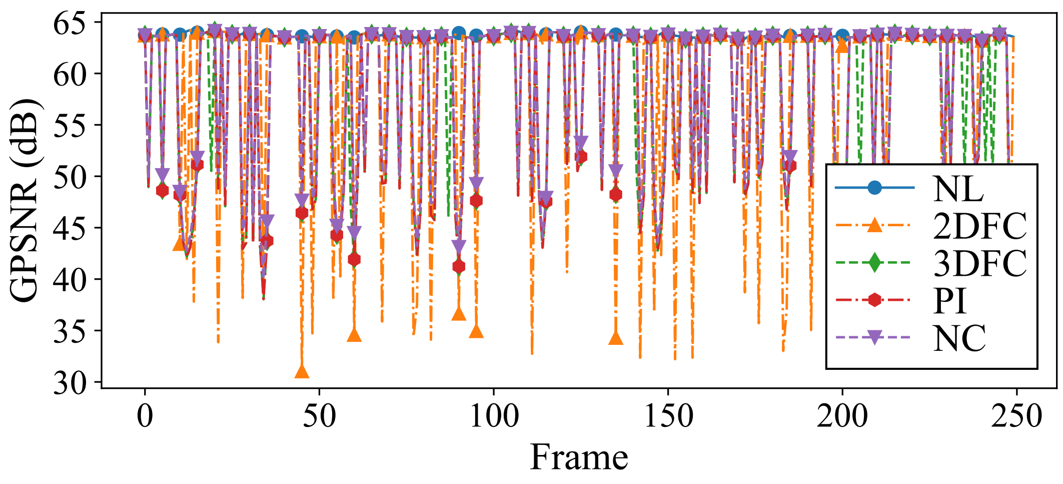


(a)

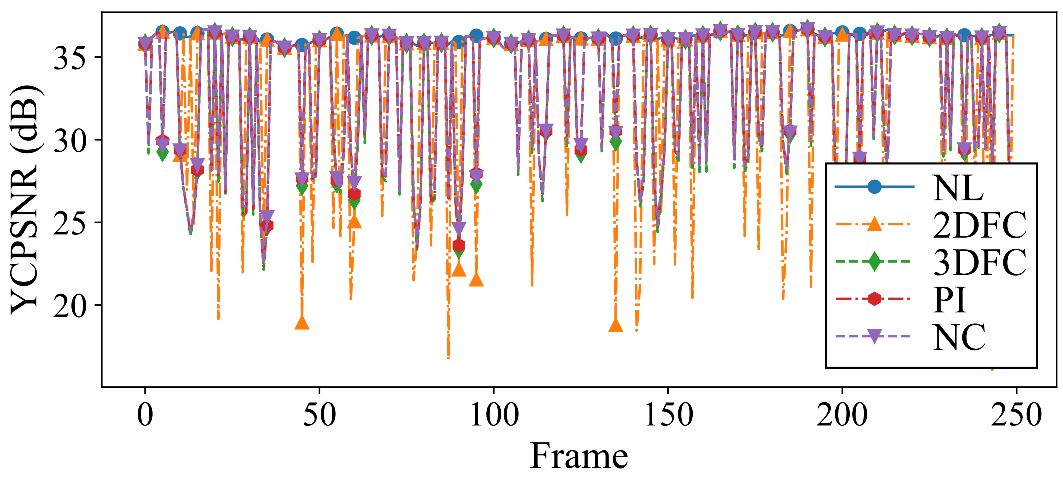


(b)

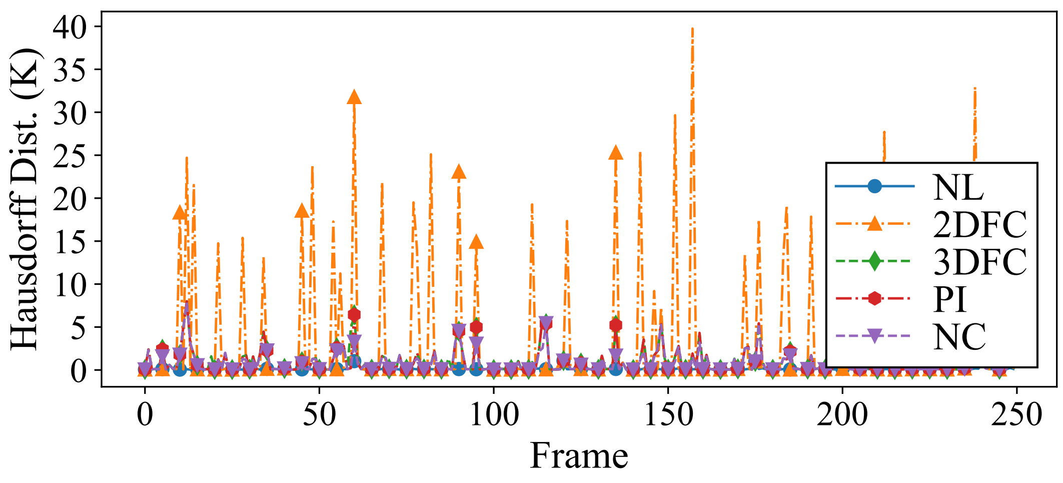
Figure 6.10: Per-frame 2D visual quality of 10% packet loss from Red&Blk: (a) PSNR, and (b) SSIM.



(a)

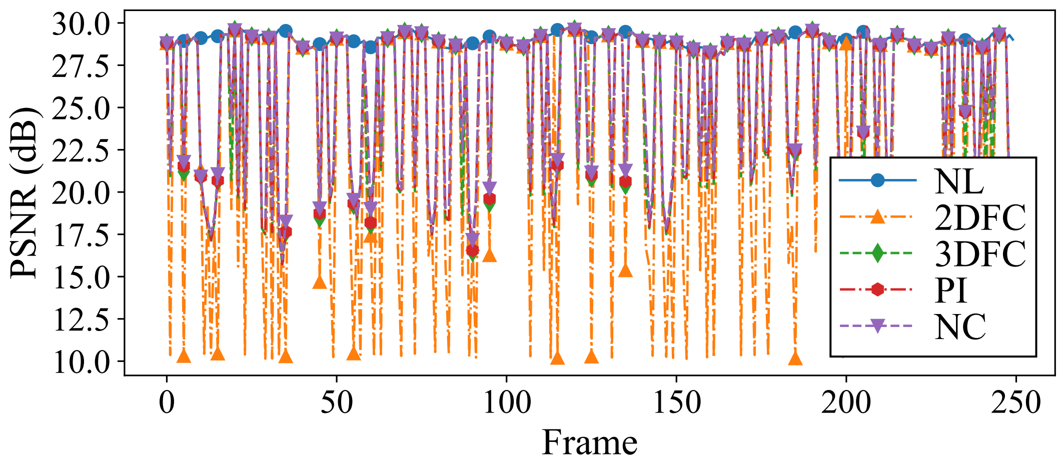


(b)

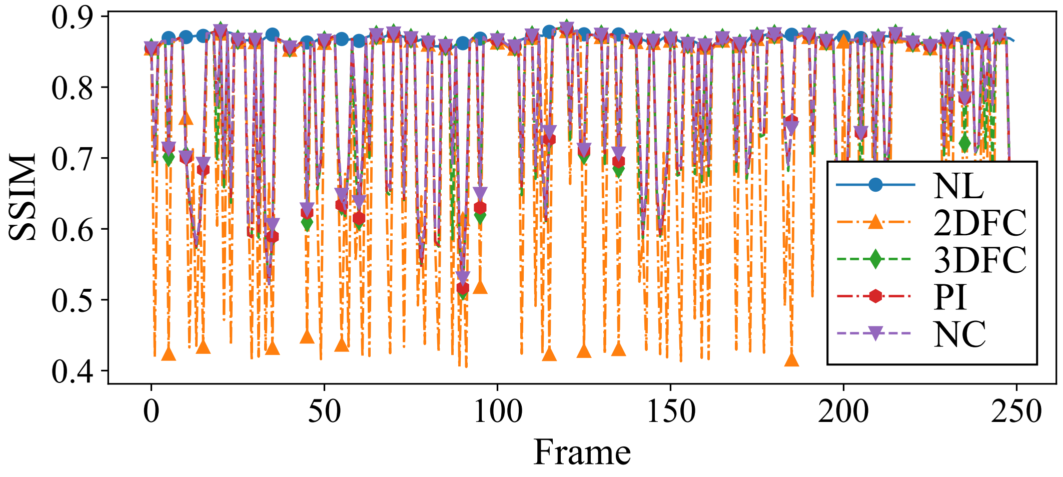


(c)

Figure 6.11: Per-frame 3D visual quality of 15% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

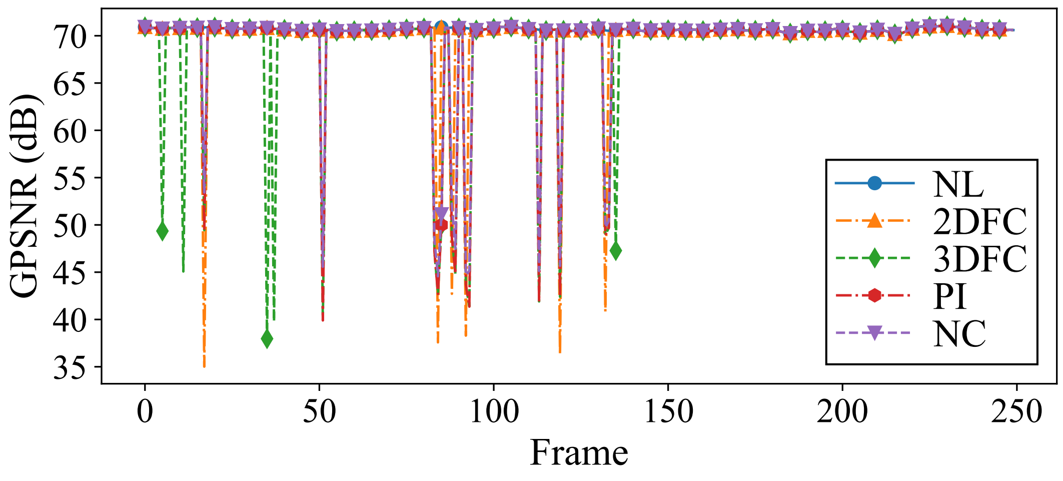


(a)

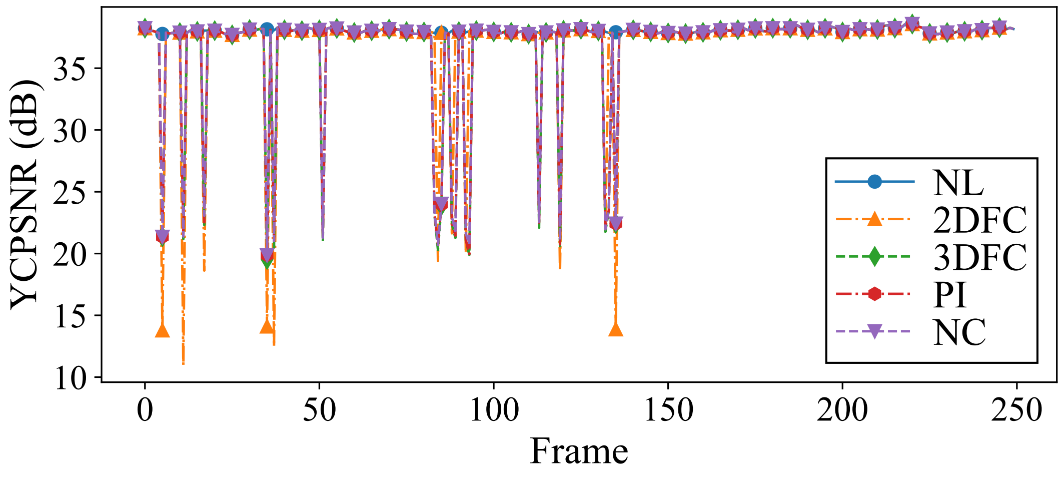


(b)

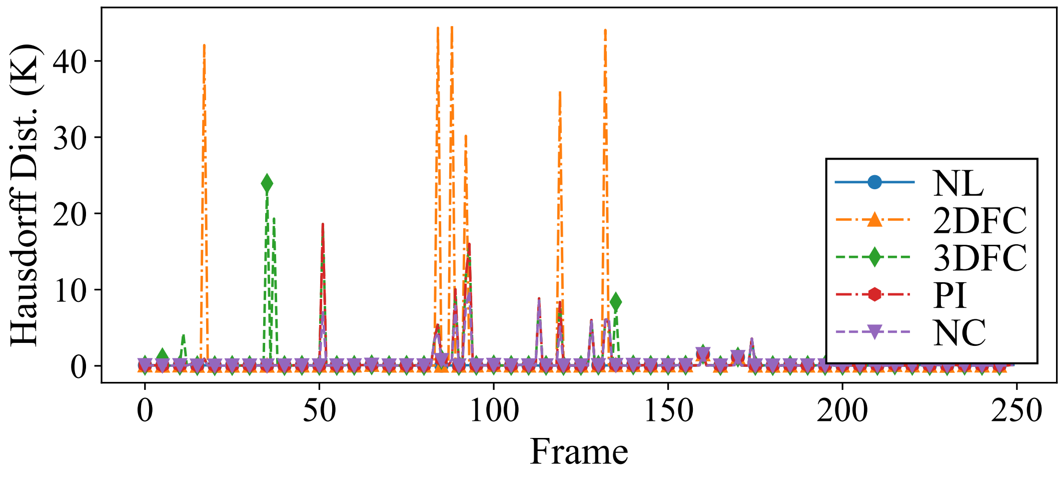
Figure 6.12: Per-frame 2D visual quality of 15% packet loss from Red&Blk: (a) PSNR, and (b) SSIM.



(a)

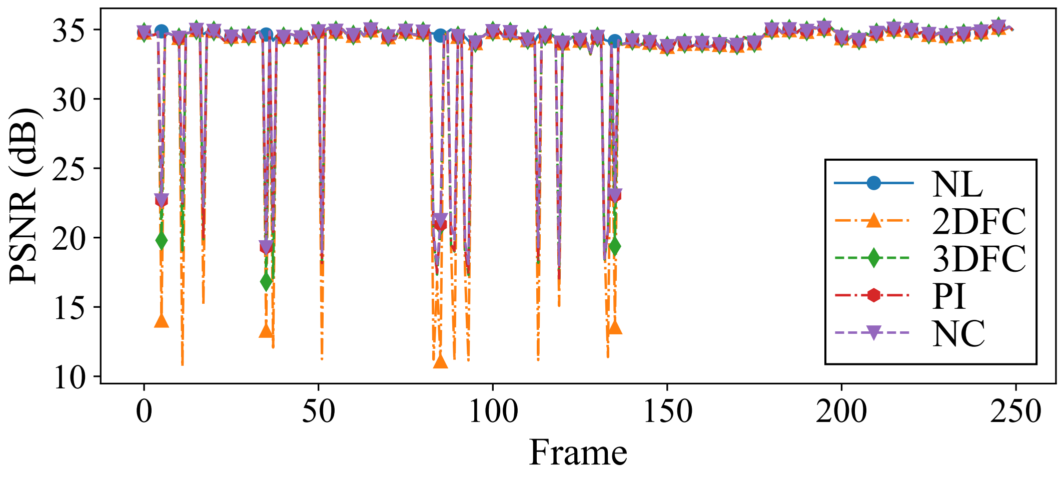


(b)

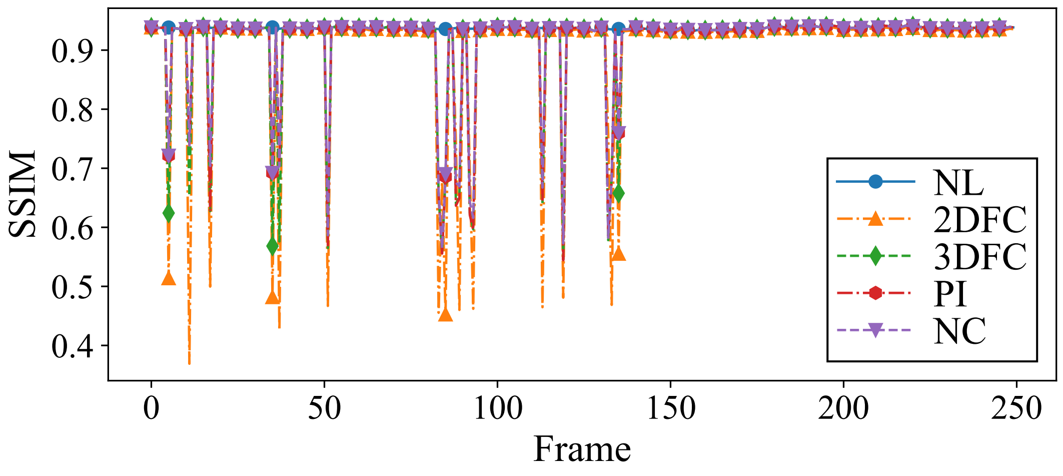


(c)

Figure 6.13: Per-frame 3D visual quality of 5% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

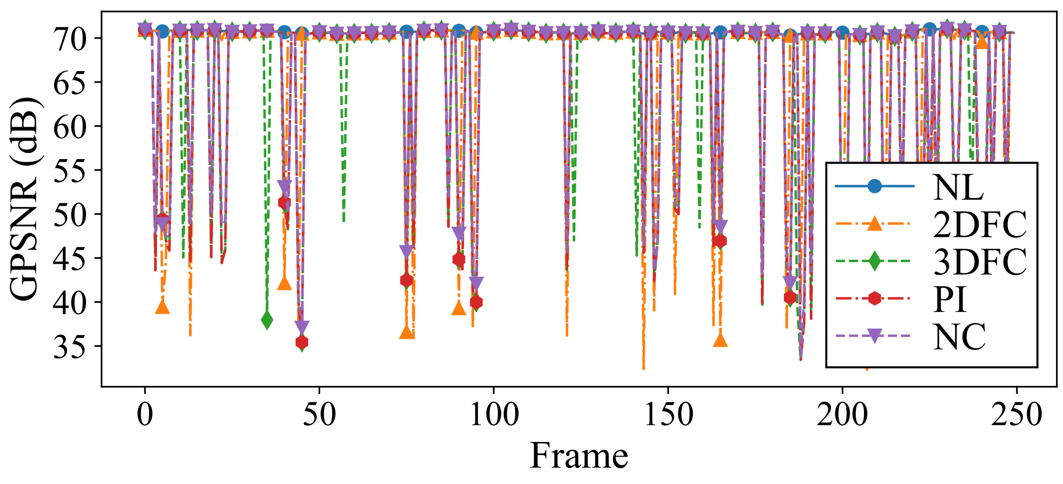


(a)

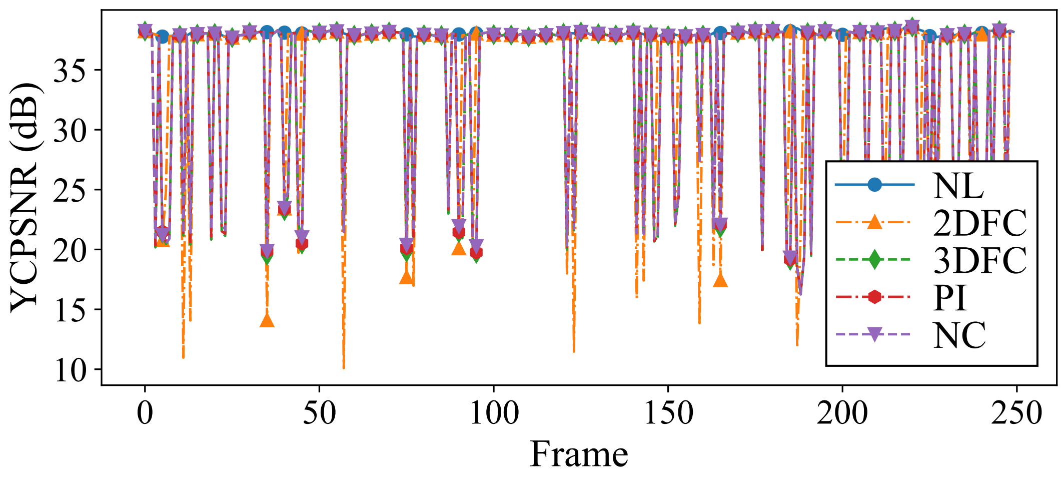


(b)

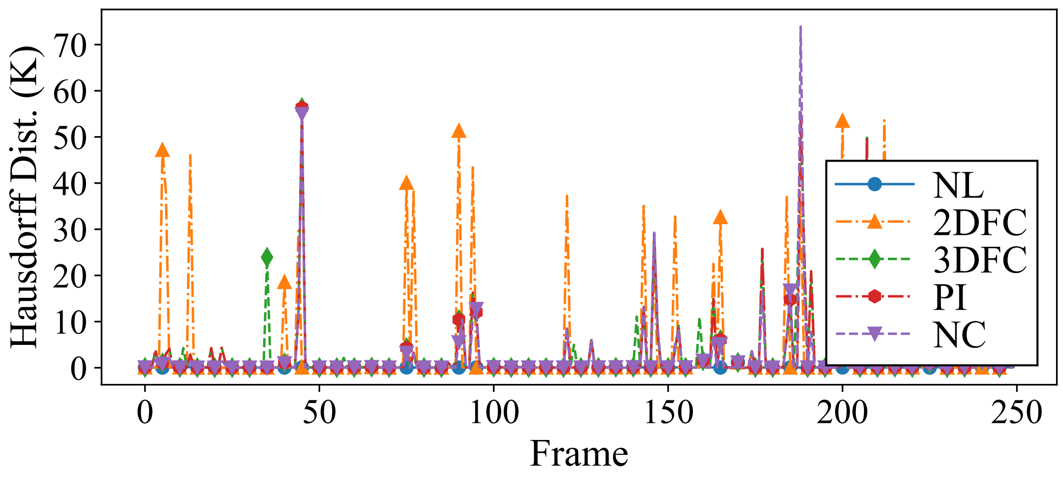
Figure 6.14: Per-frame 2D visual quality of 5% packet loss from Dancer: (a) PSNR, and (b) SSIM.



(a)

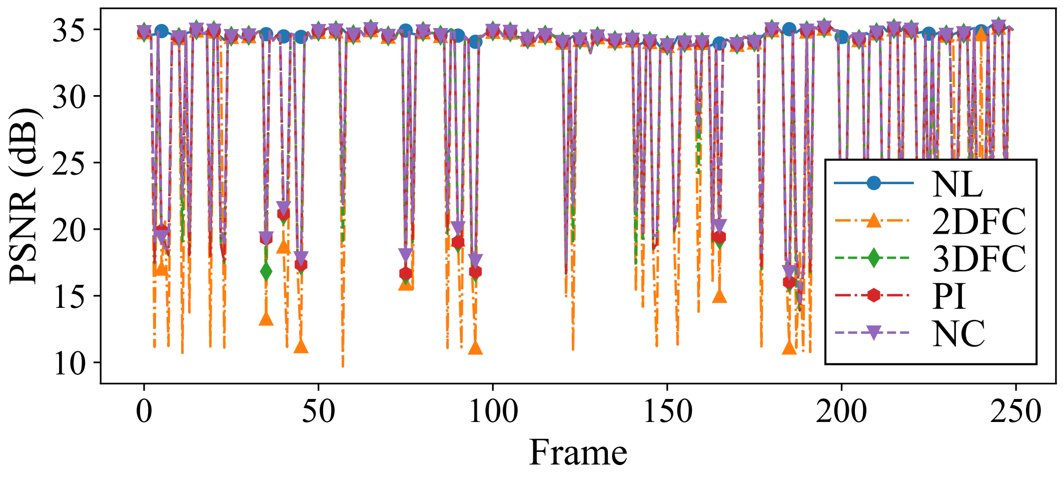


(b)

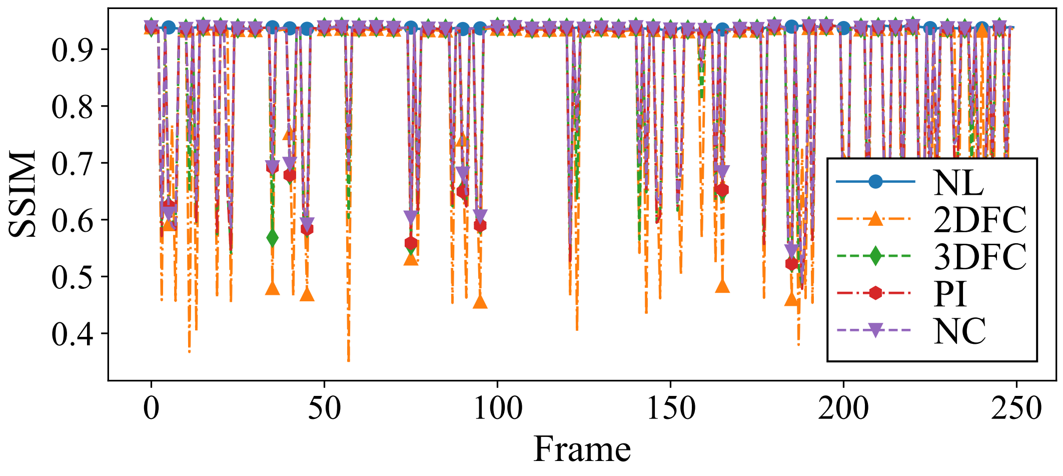


(c)

Figure 6.15: Per-frame 3D visual quality of 10% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.

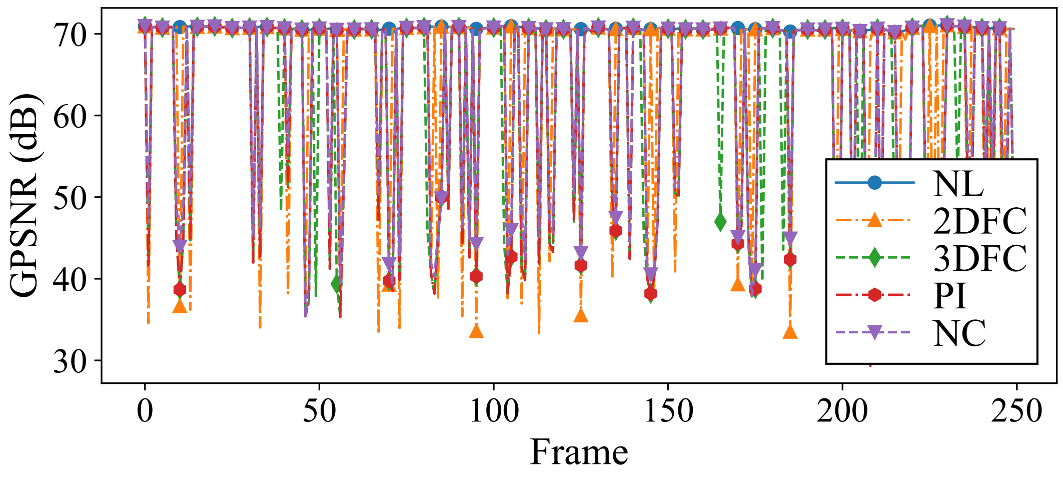


(a)

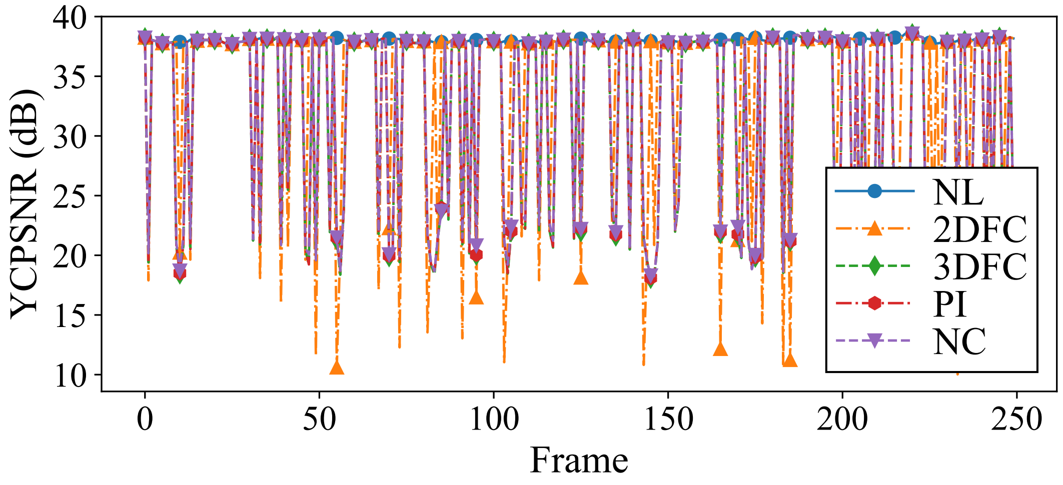


(b)

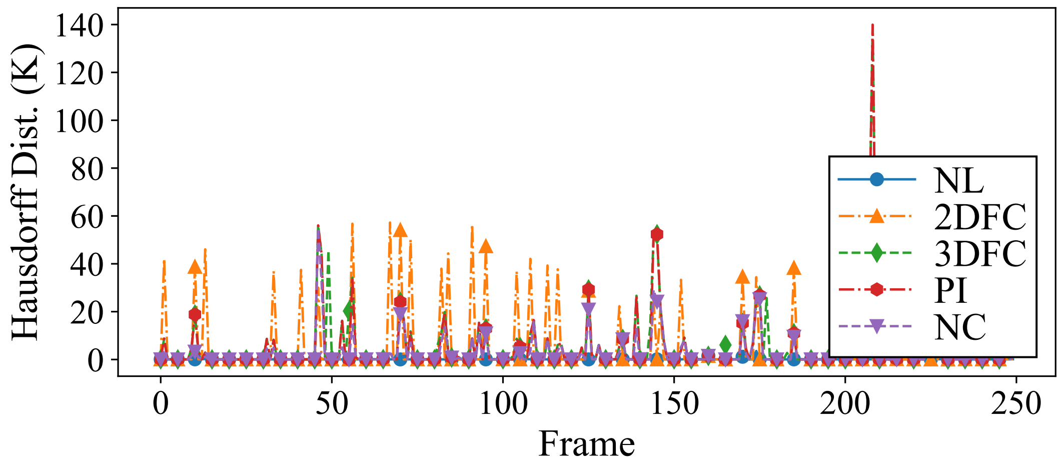
Figure 6.16: Per-frame 2D visual quality of 10% packet loss from Dancer: (a) PSNR, and (b) SSIM.



(a)

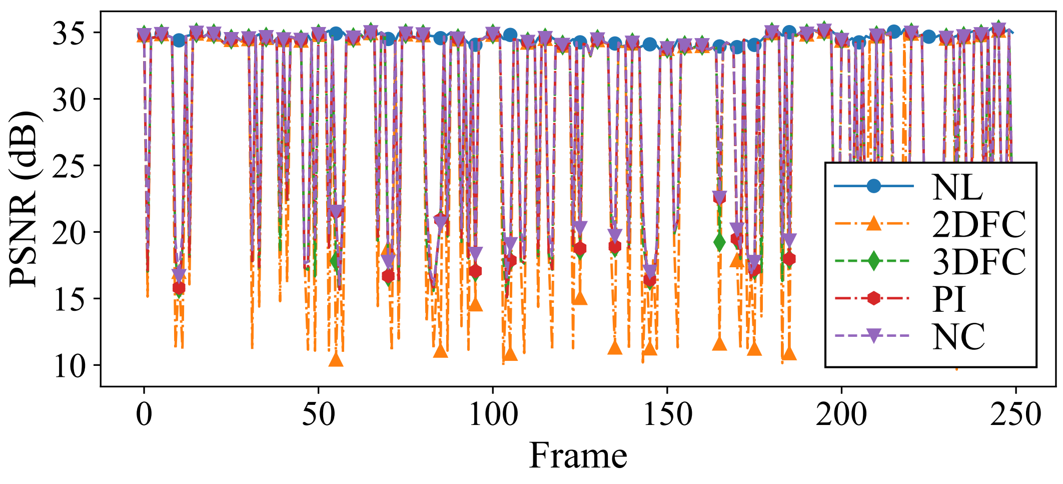


(b)

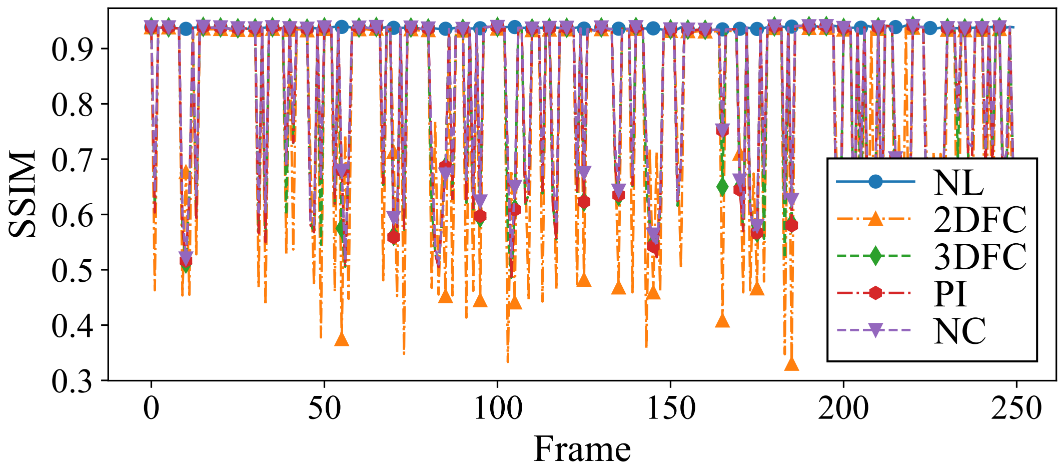


(c)

Figure 6.17: Per-frame 3D visual quality of 15% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.



(a)



(b)

Figure 6.18: Per-frame 2D visual quality of 15% packet loss from Dancer: (a) PSNR, and (b) SSIM.

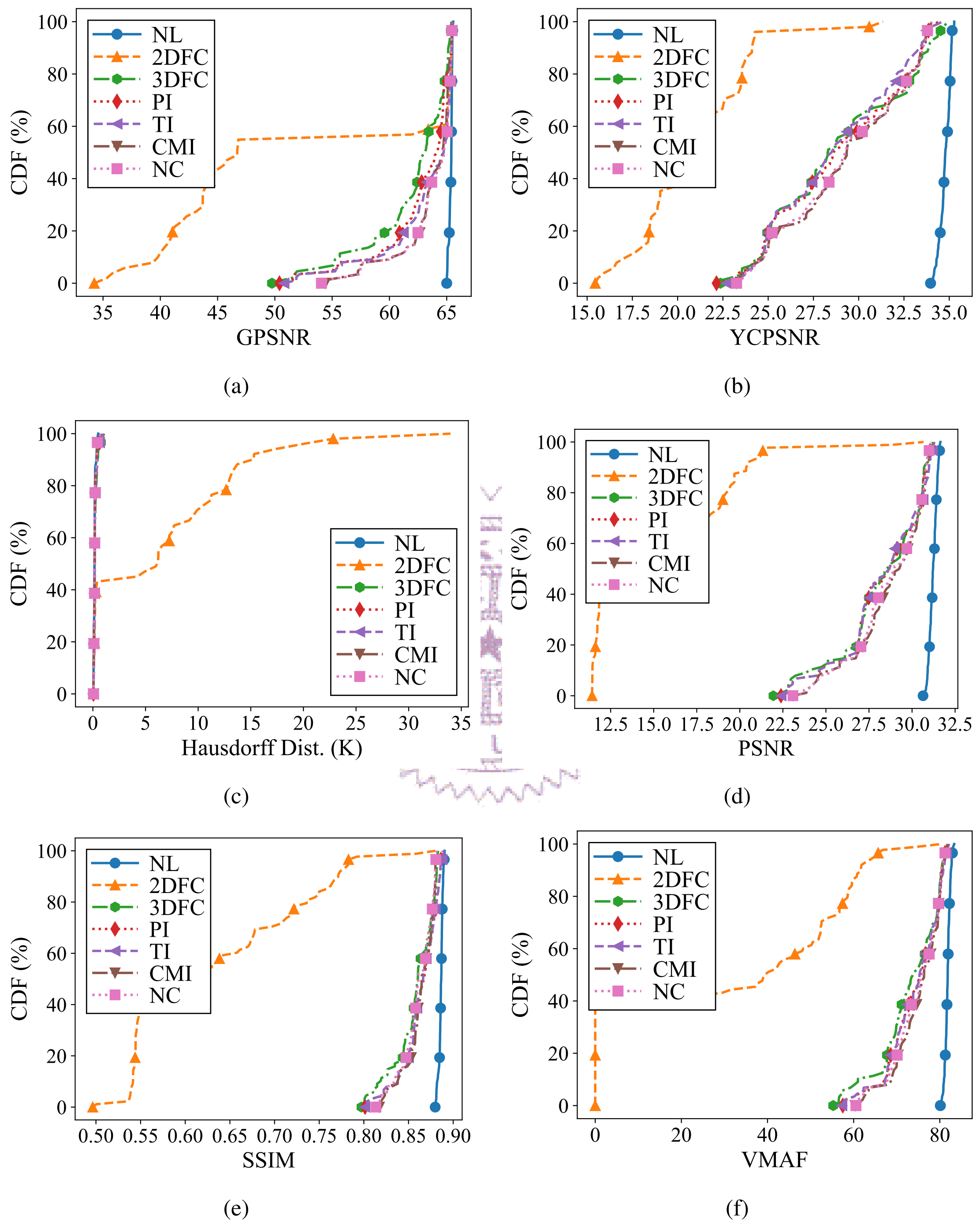


Figure 6.19: Per-frame visual quality distribution of distorted frames of 5% packet loss from Queen: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

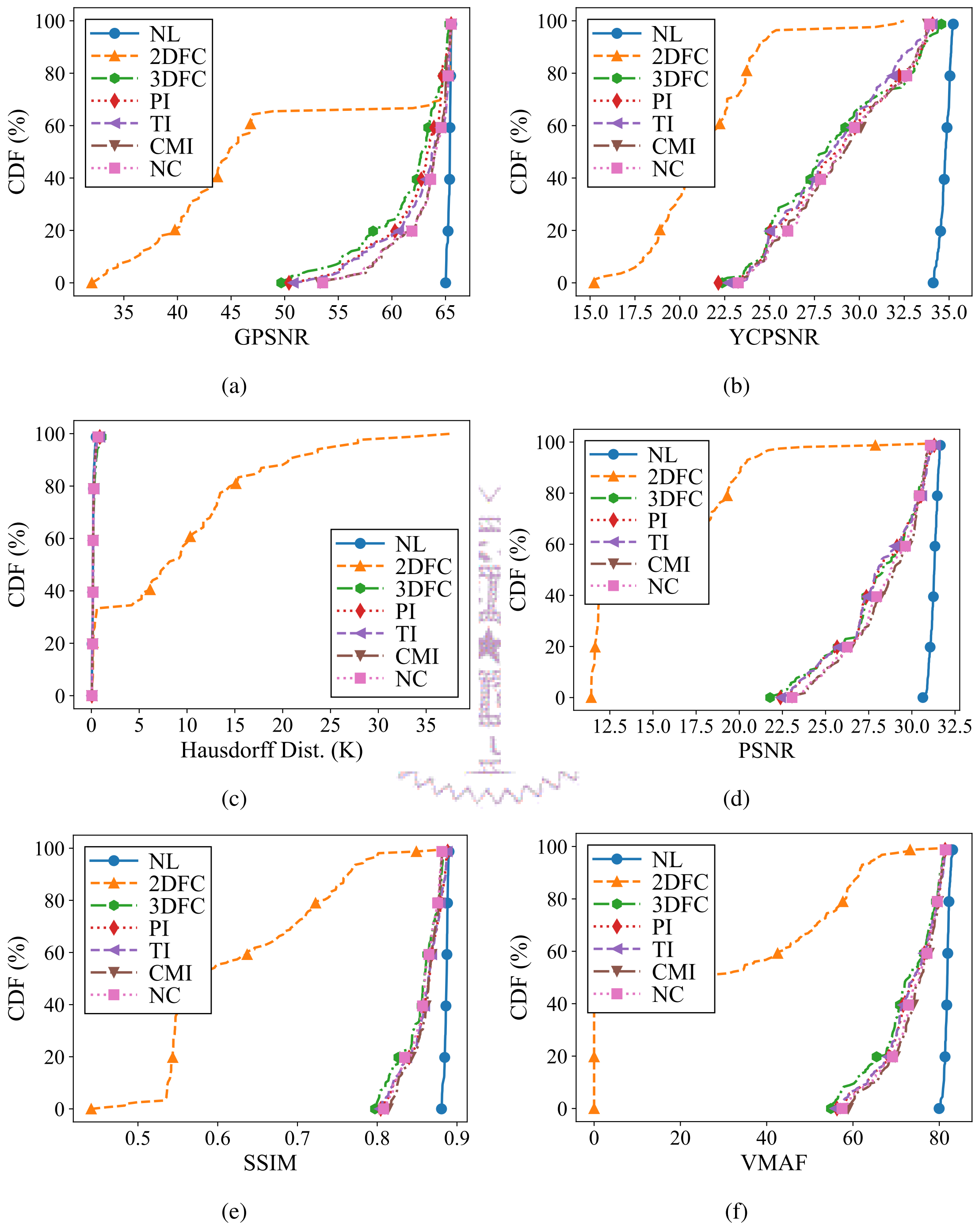
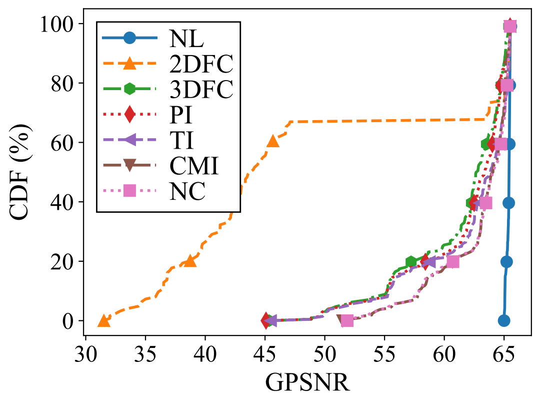
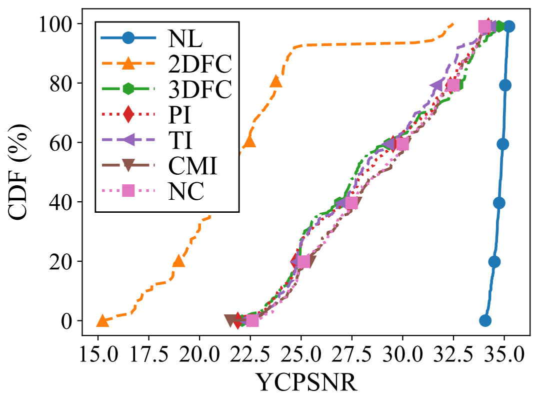


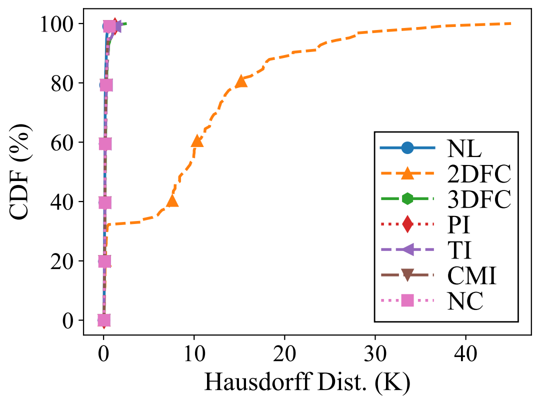
Figure 6.20: Per-frame visual quality distribution of distorted frames of 10% packet loss from Queen: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



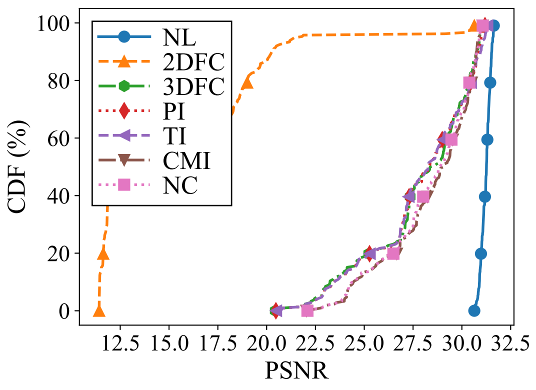
(a)



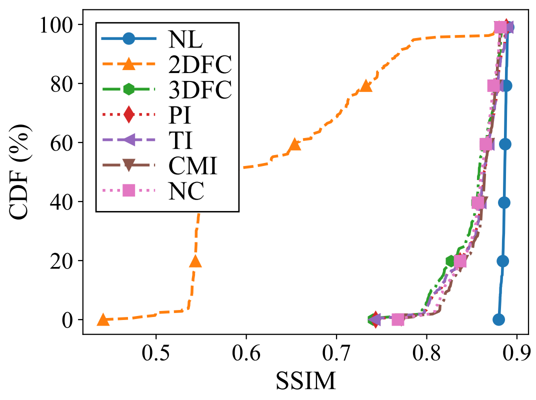
(b)



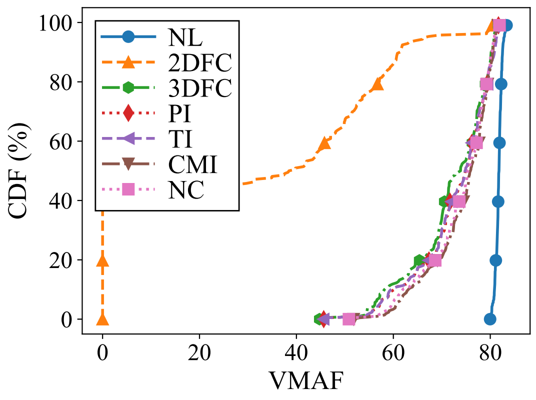
(c)



(d)



(e)



(f)

Figure 6.21: Per-frame visual quality distribution of distorted frames of 15% packet loss from Queen: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

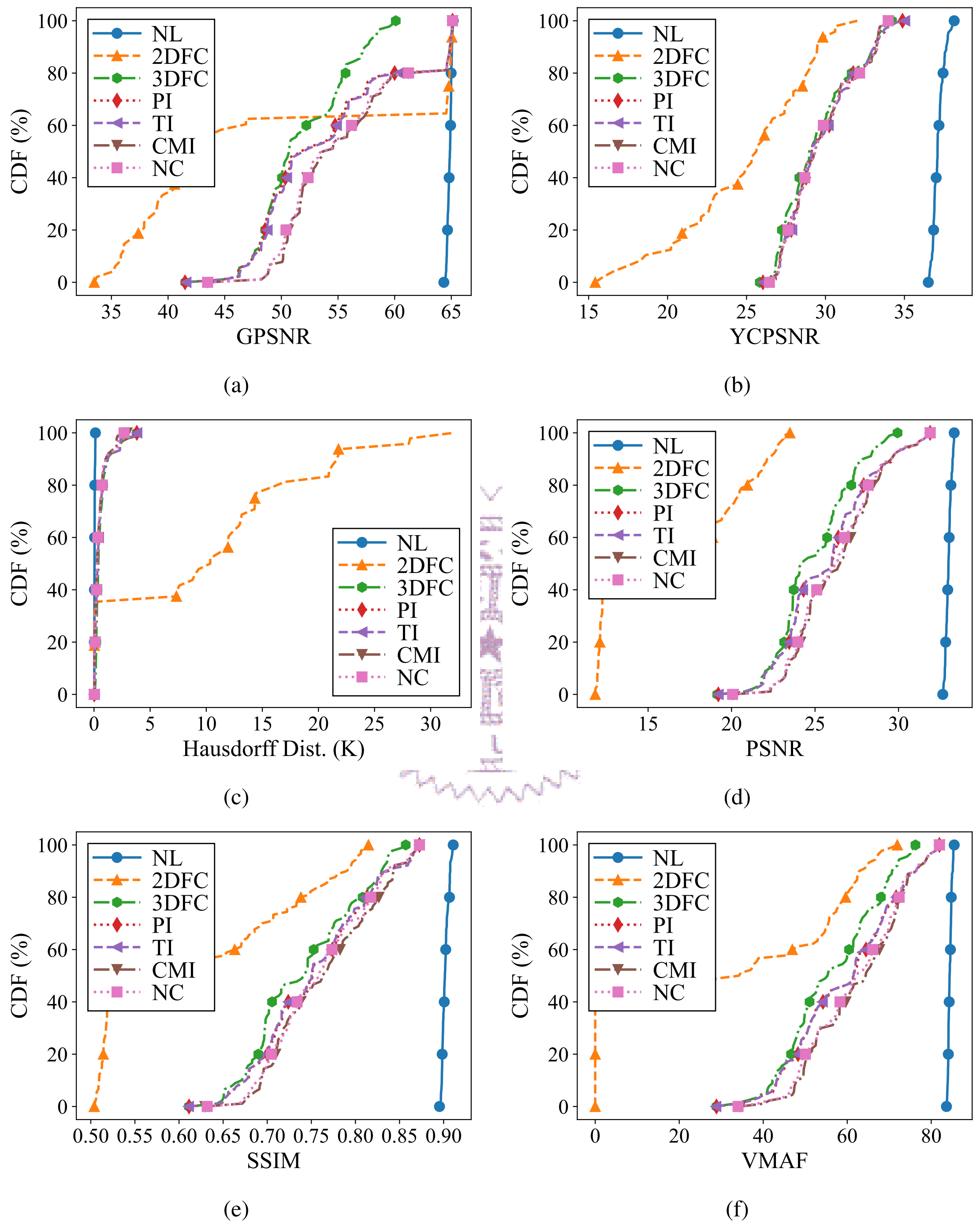


Figure 6.22: Per-frame visual quality distribution of distorted frames of 5% packet loss from Loot: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

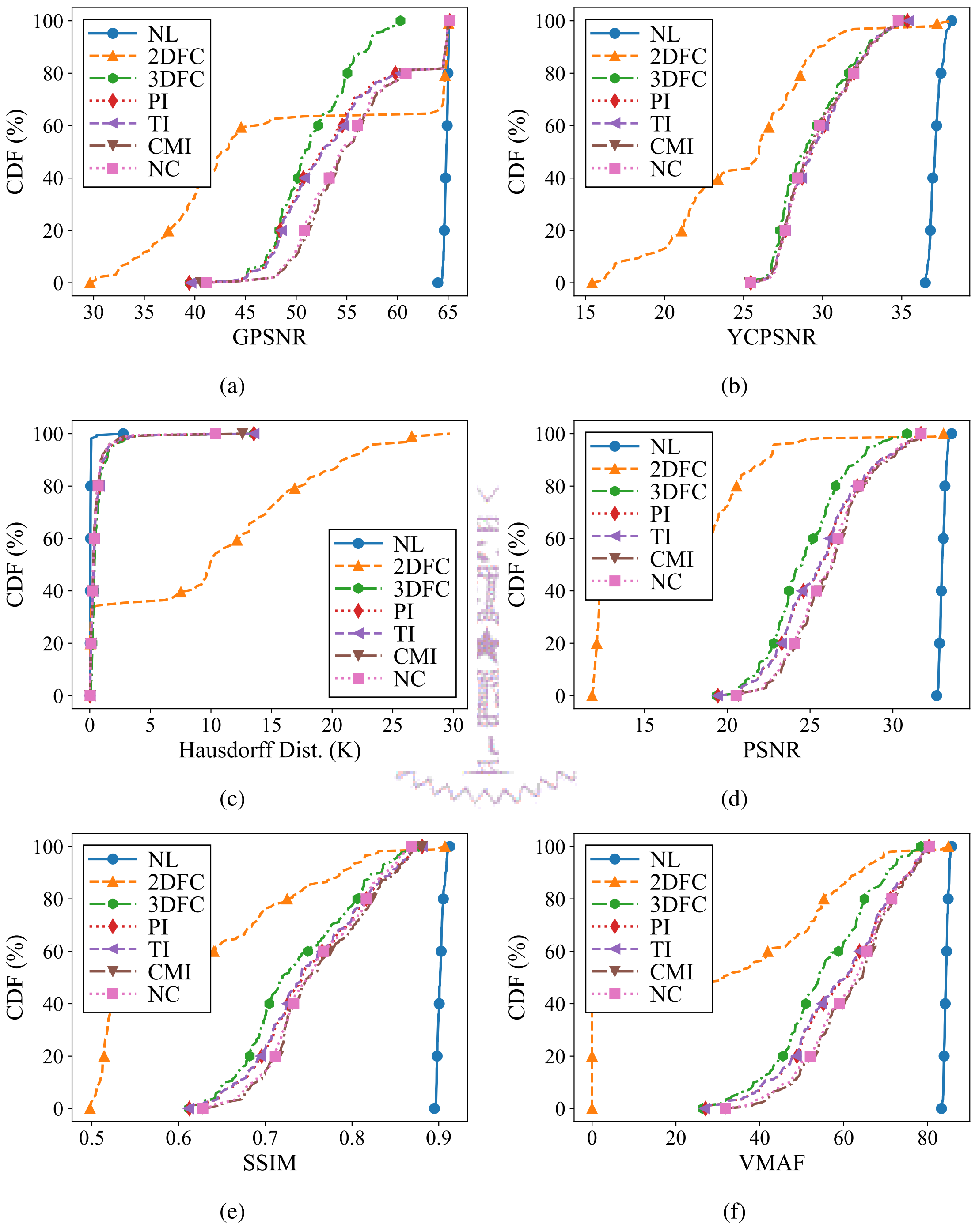
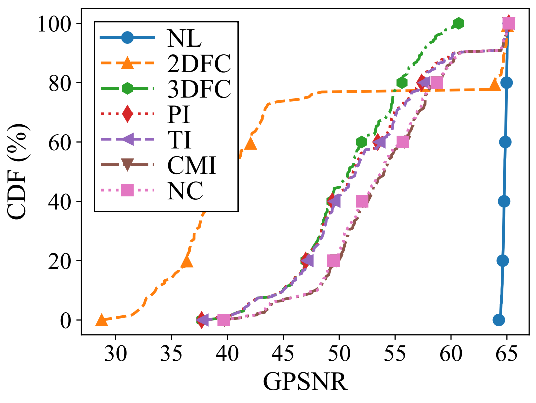
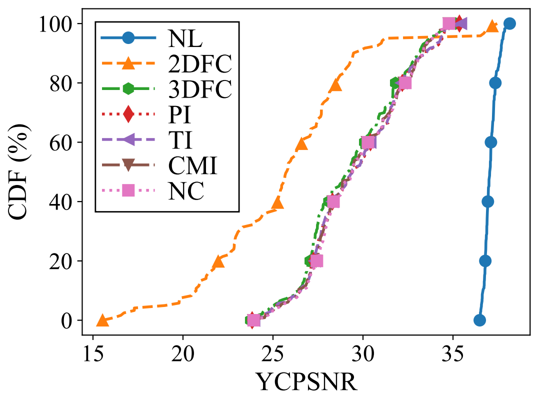


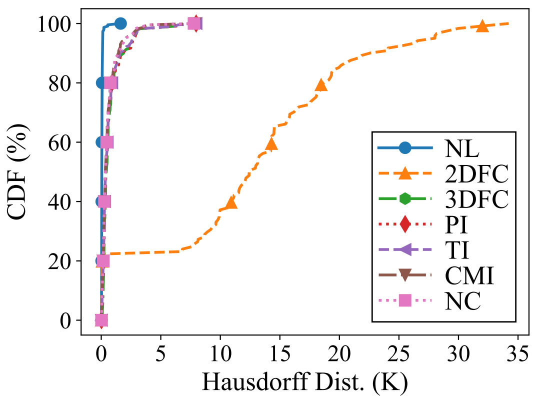
Figure 6.23: Per-frame visual quality distribution of distorted frames of 10% packet loss from Loot: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



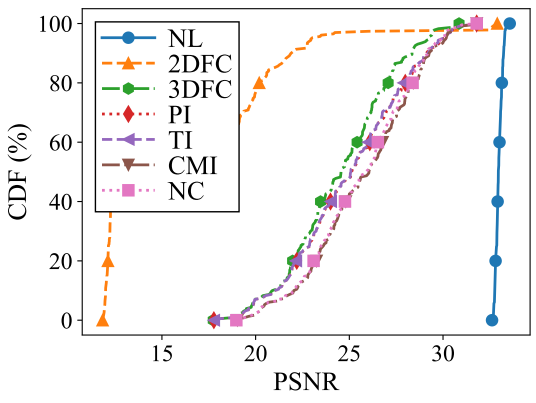
(a)



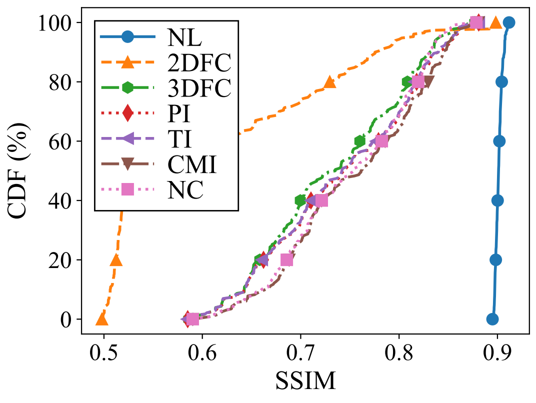
(b)



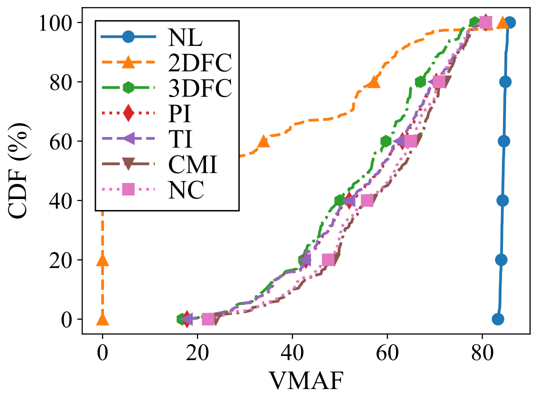
(c)



(d)



(e)



(f)

Figure 6.24: Per-frame visual quality distribution of distorted frames of 15% packet loss from Loot: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

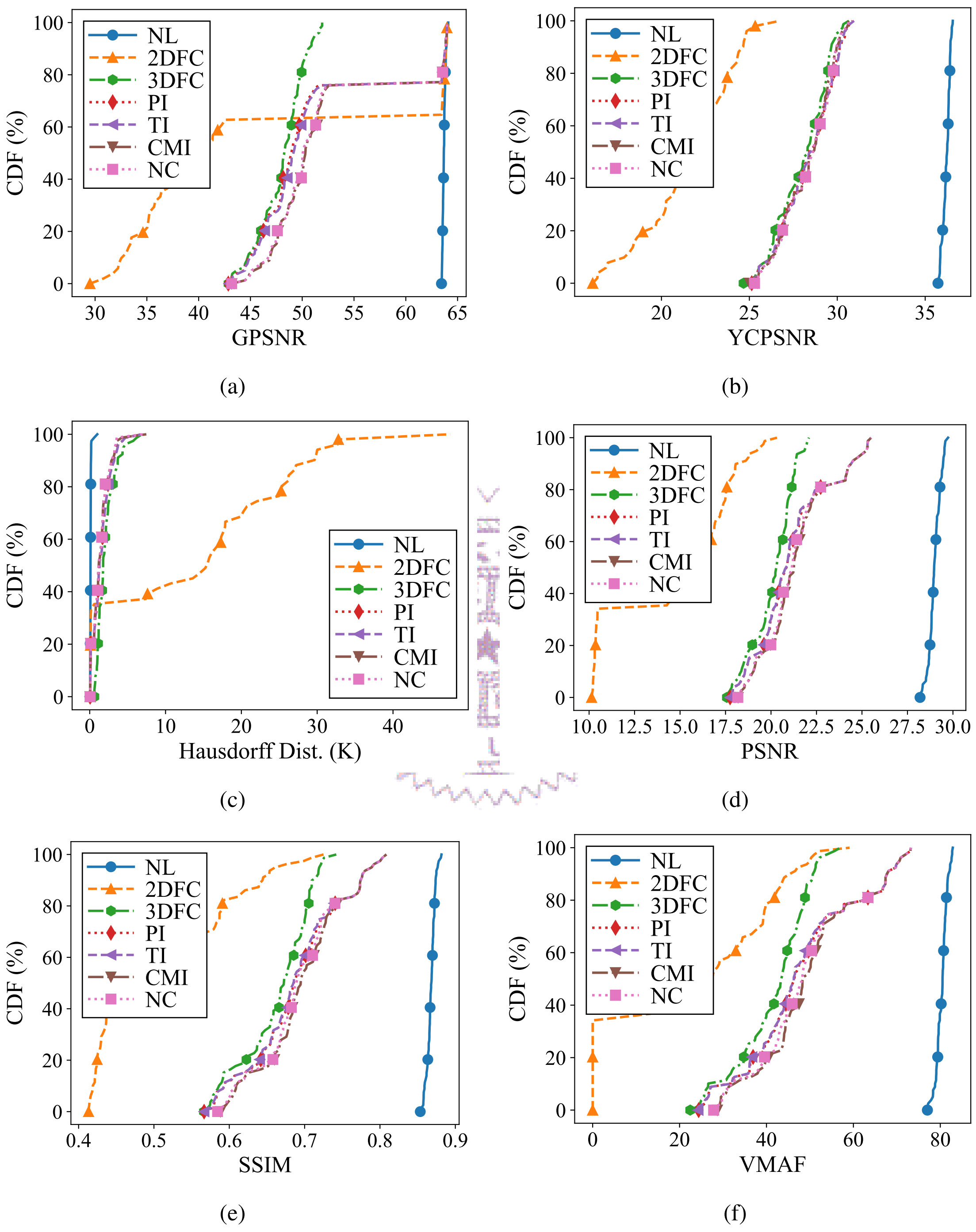


Figure 6.25: Per-frame visual quality distribution of distorted frames of 5% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

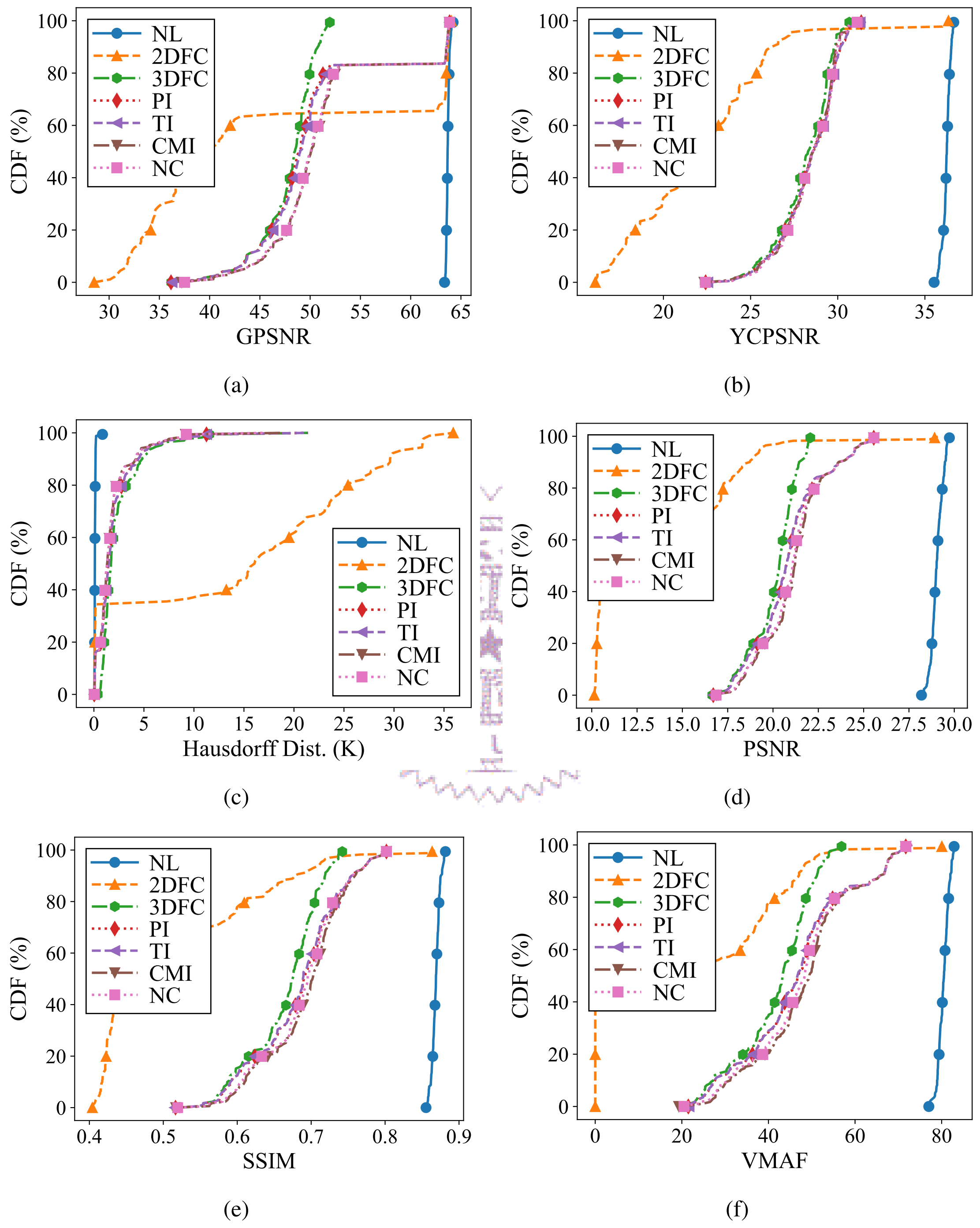


Figure 6.26: Per-frame visual quality distribution of distorted frames of 10% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

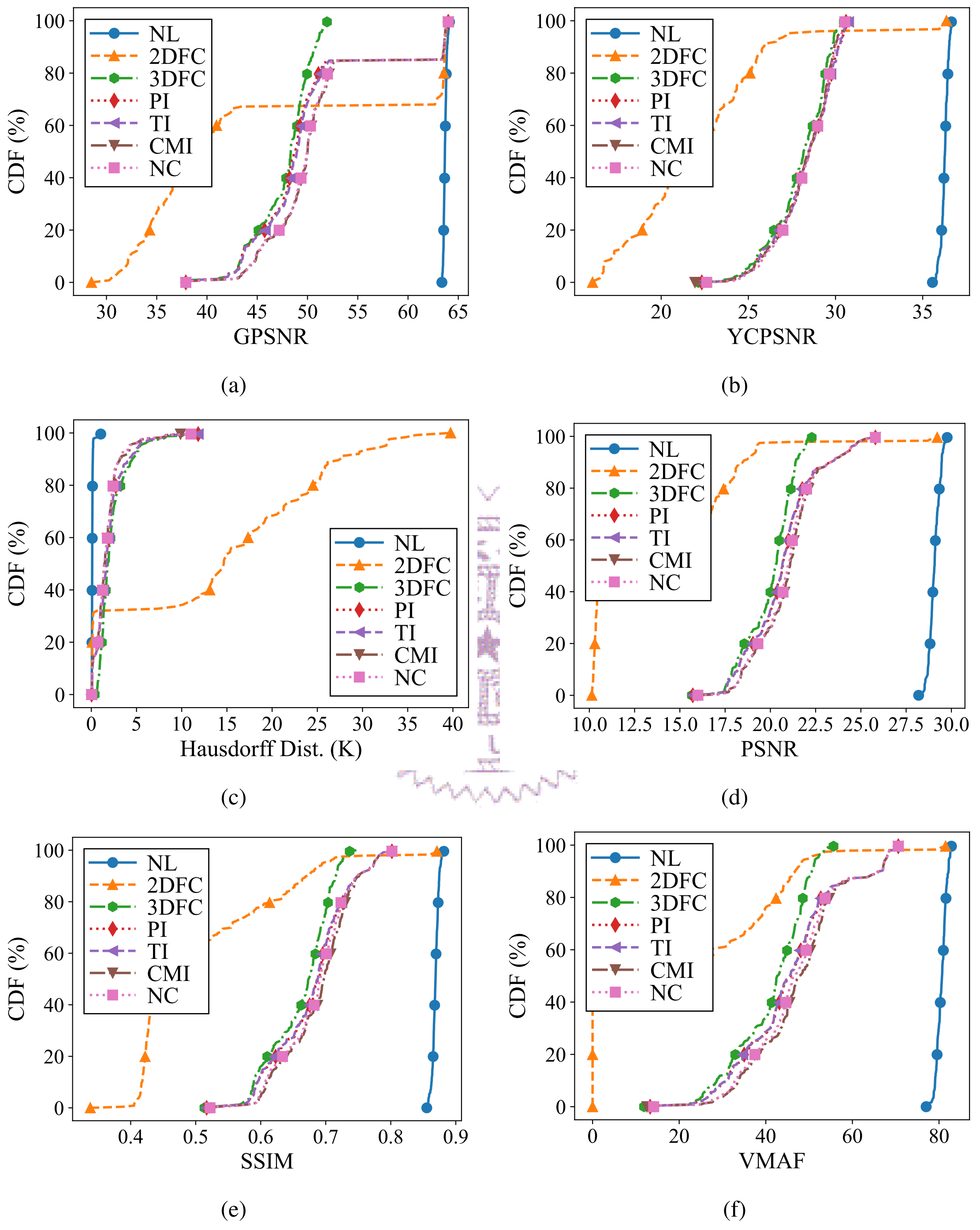
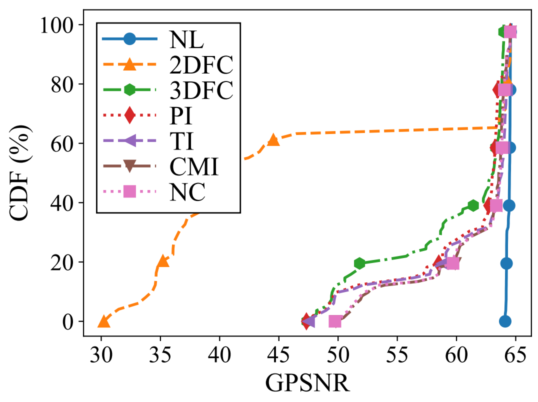
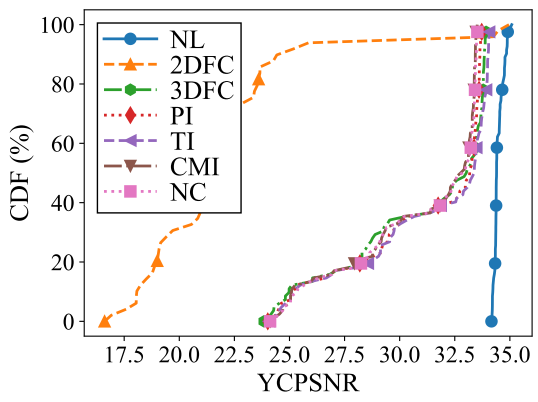


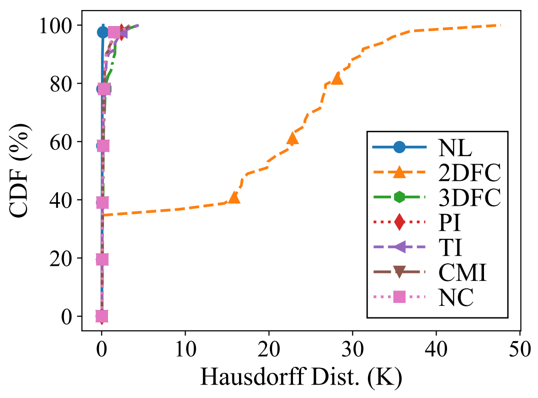
Figure 6.27: Per-frame visual quality distribution of distorted frames of 15% packet loss from Red&Blk: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



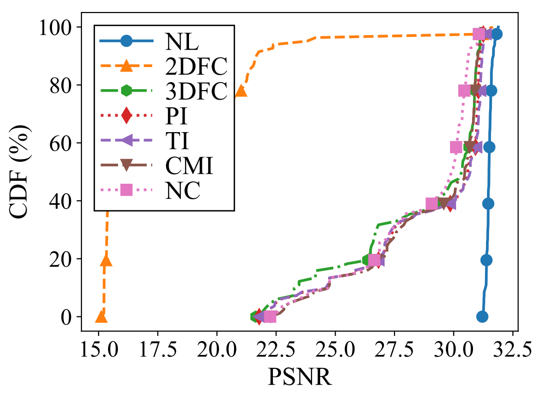
(a)



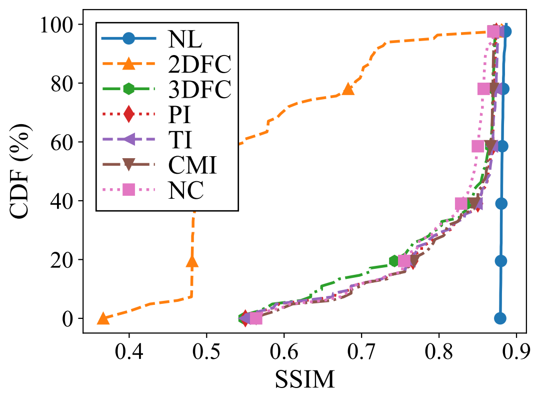
(b)



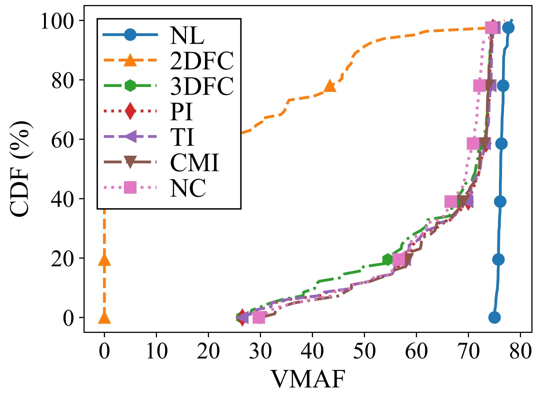
(c)



(d)



(e)



(f)

Figure 6.28: Per-frame visual quality distribution of distorted frames of 5% packet loss from Soldier: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

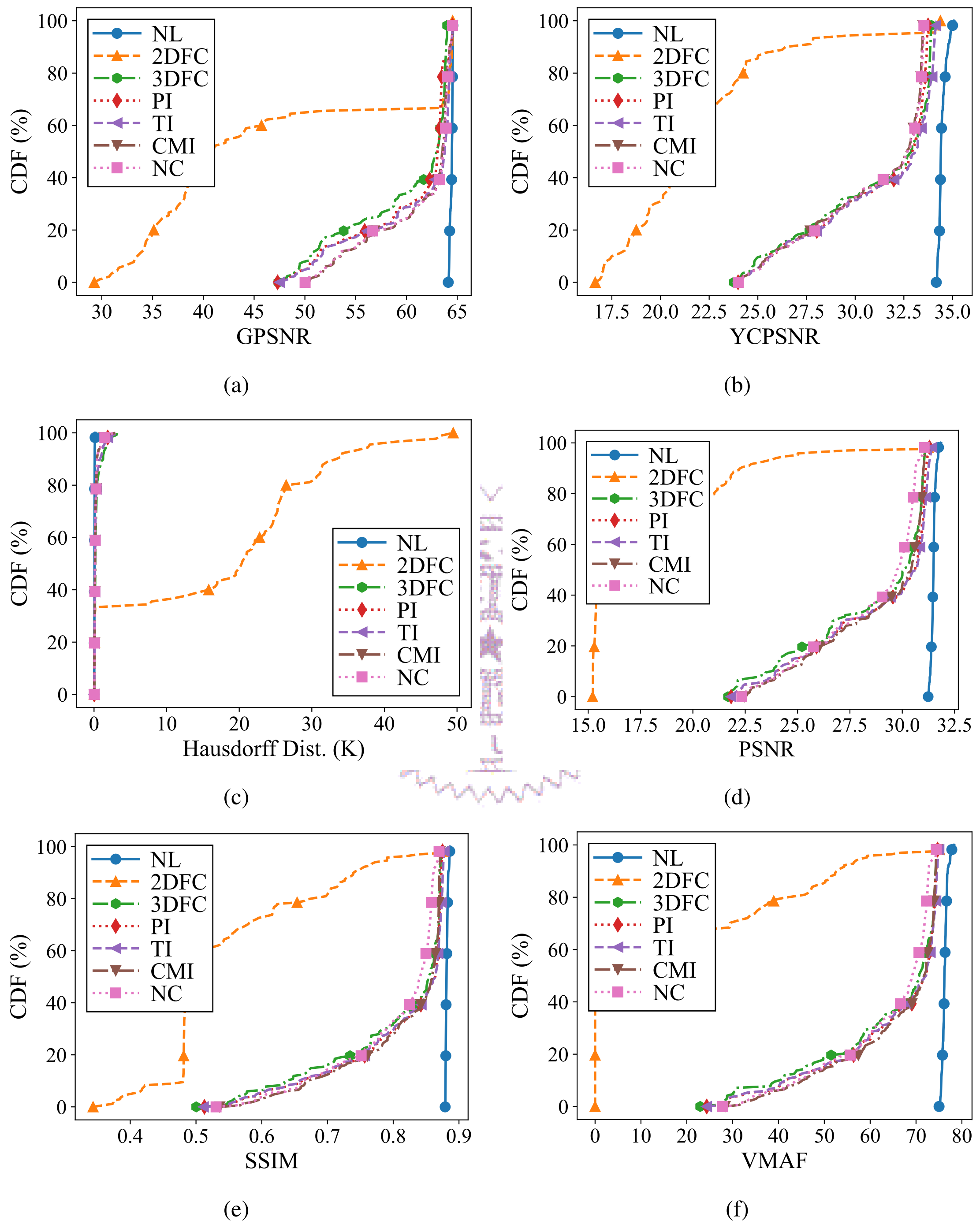
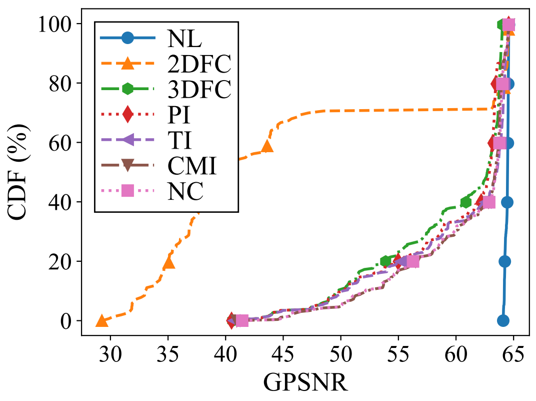
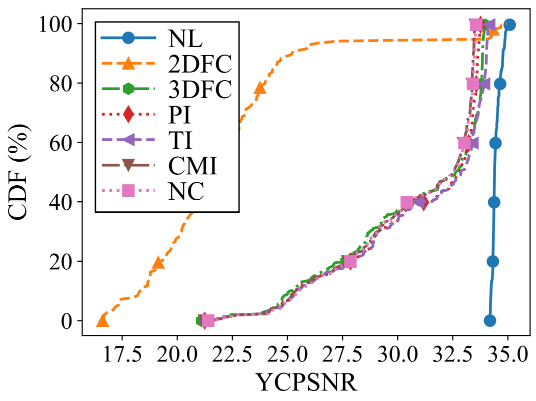


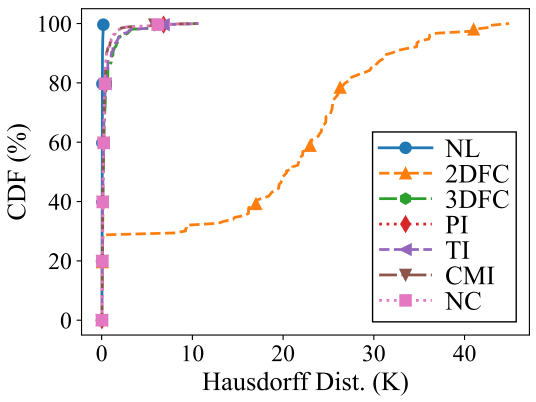
Figure 6.29: Per-frame visual quality distribution of distorted frames of 10% packet loss from Soldier: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



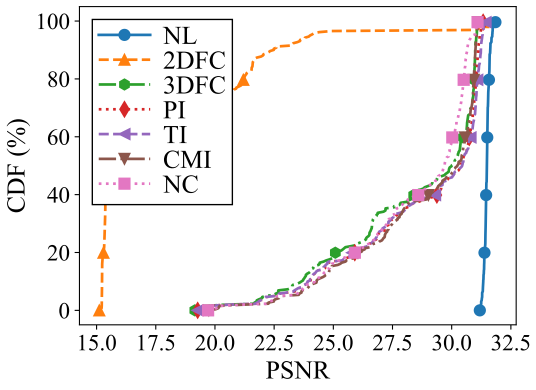
(a)



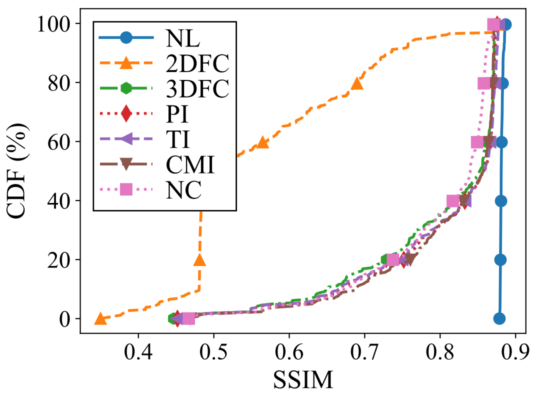
(b)



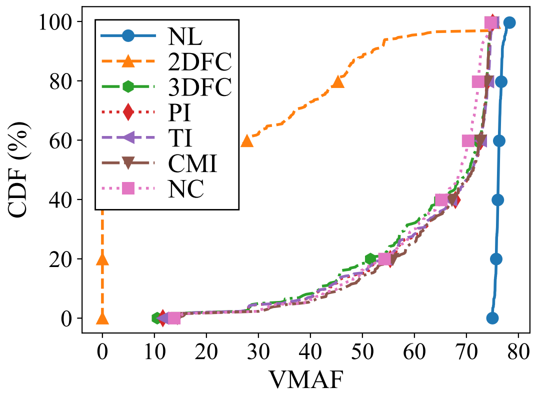
(c)



(d)



(e)



(f)

Figure 6.30: Per-frame visual quality distribution of distorted frames of 15% packet loss from Soldier: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

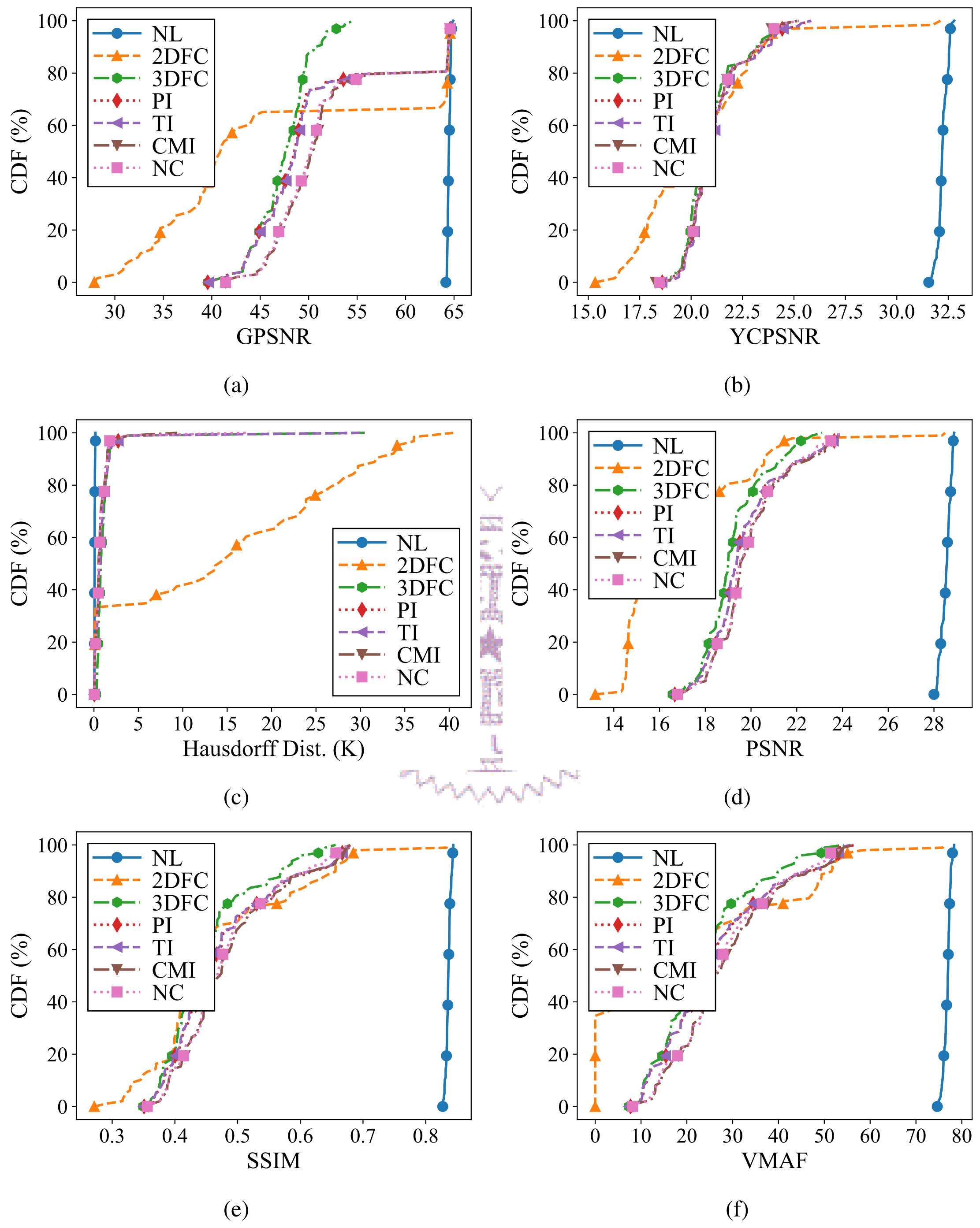
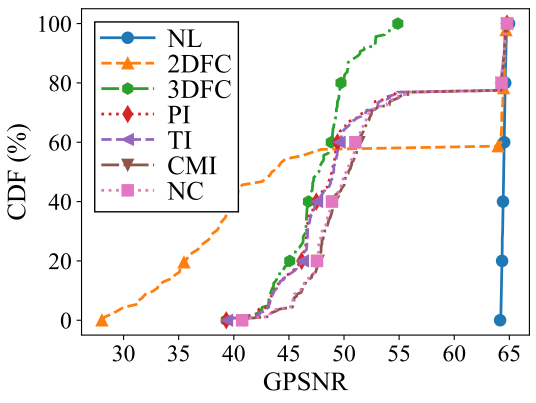
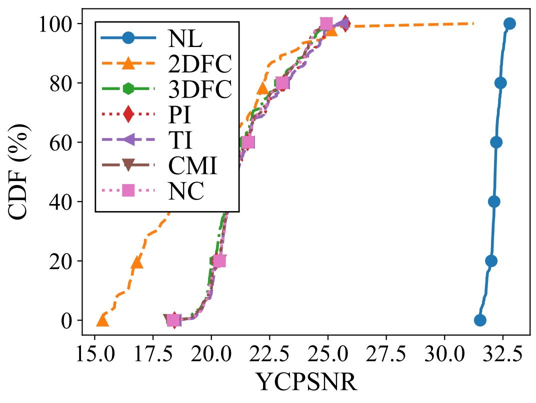


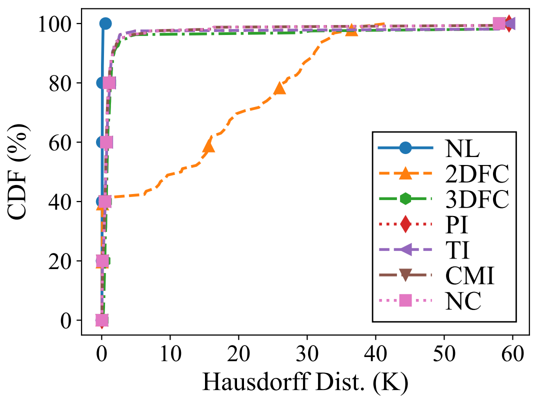
Figure 6.31: Per-frame visual quality distribution of distorted frames of 5% packet loss from LongDress: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



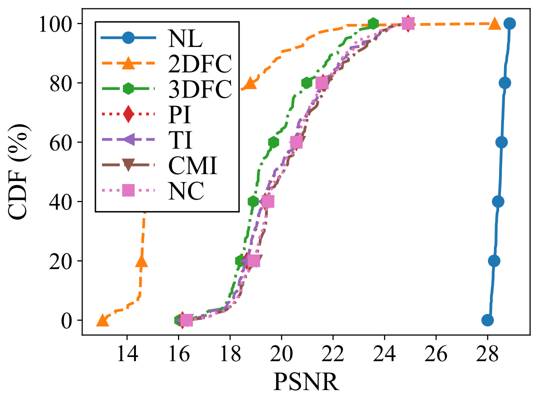
(a)



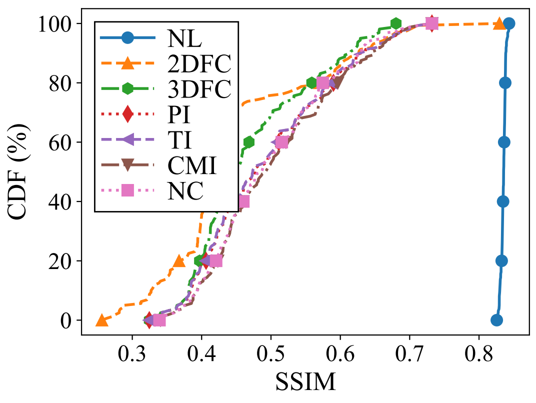
(b)



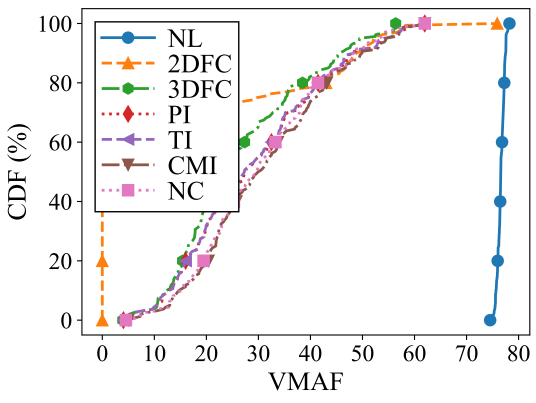
(c)



(d)



(e)



(f)

Figure 6.32: Per-frame visual quality distribution of distorted frames of 10% packet loss from LongDress: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

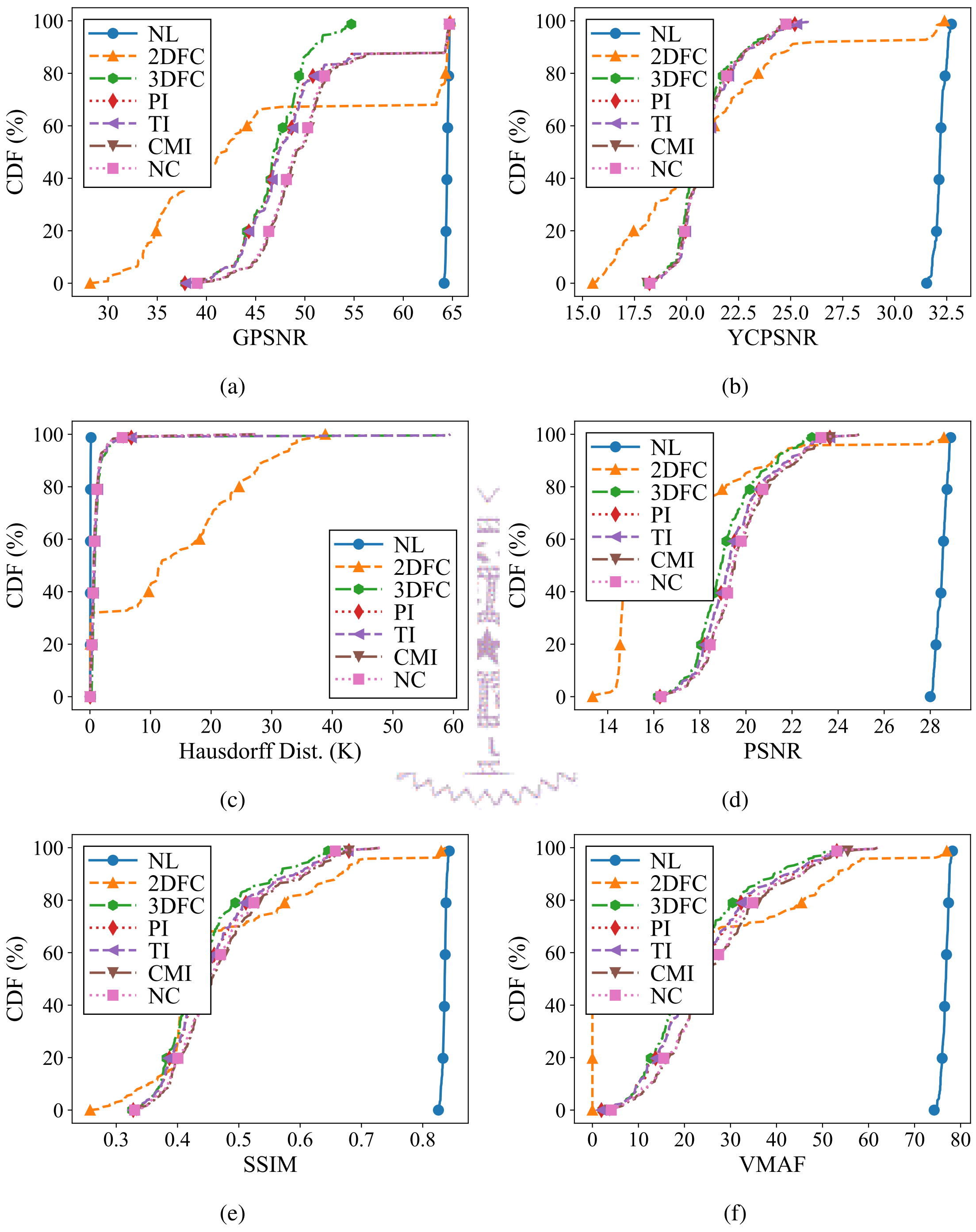
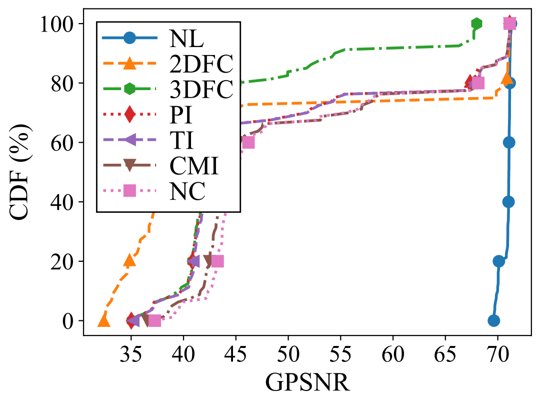
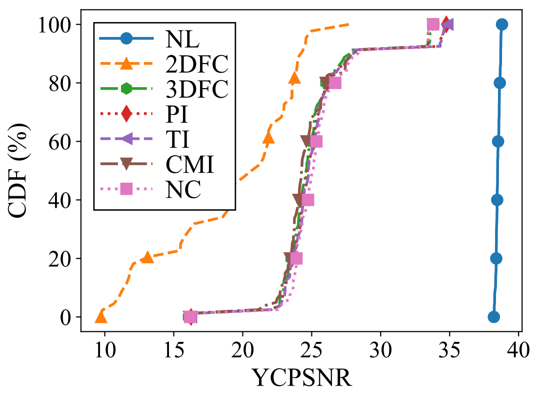


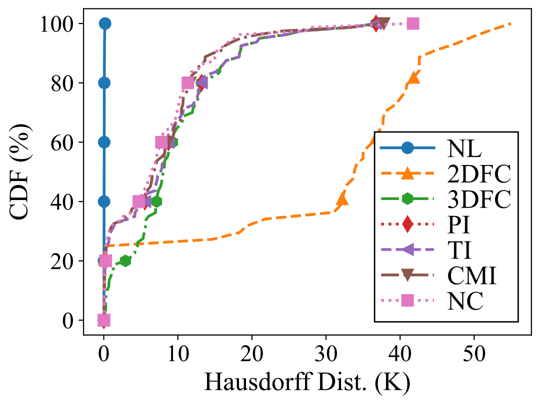
Figure 6.33: Per-frame visual quality distribution of distorted frames of 15% packet loss from LongDress: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



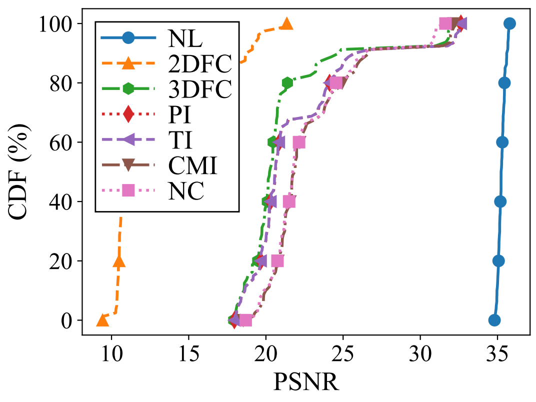
(a)



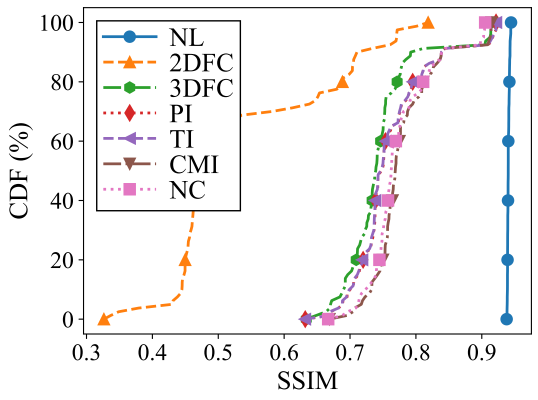
(b)



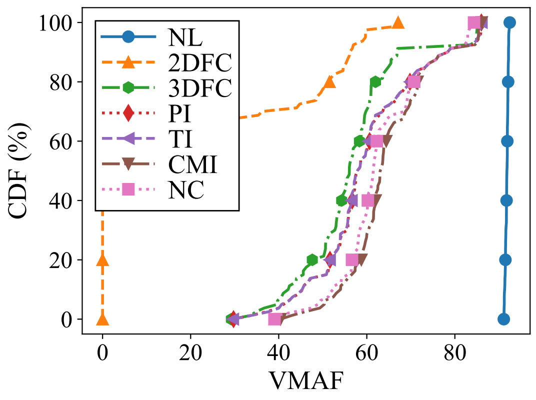
(c)



(d)

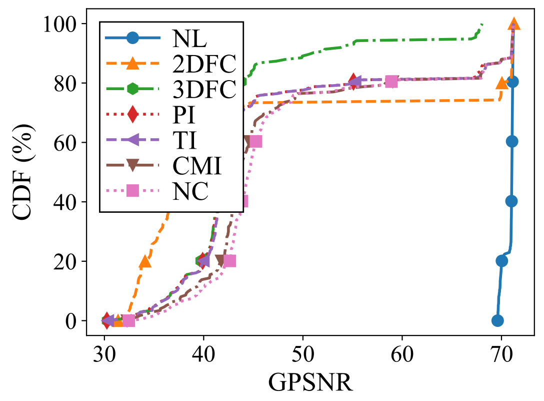


(e)

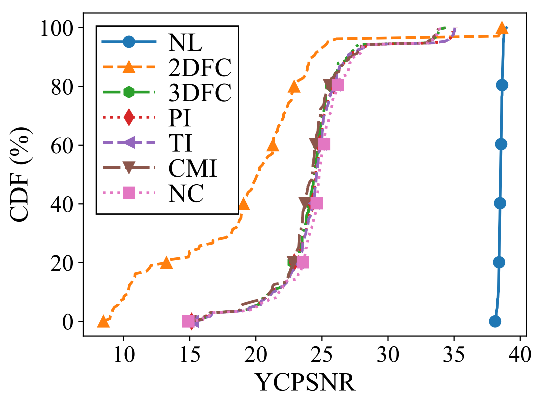


(f)

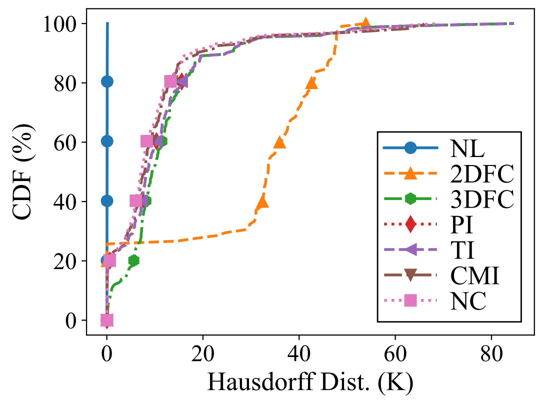
Figure 6.34: Per-frame visual quality distribution of distorted frames of 5% packet loss from Basketball: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



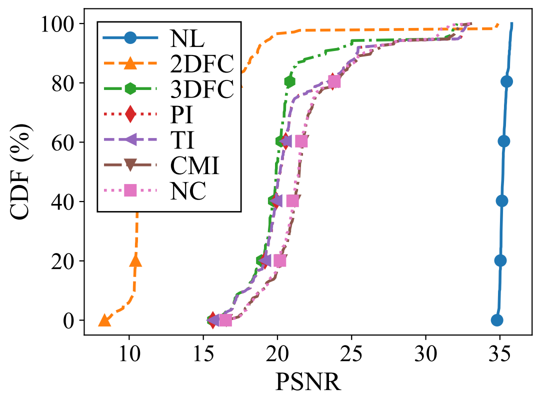
(a)



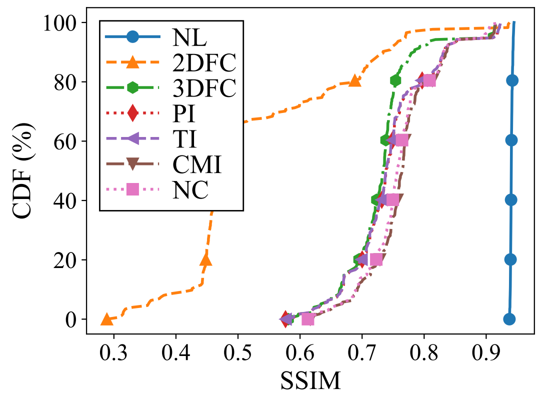
(b)



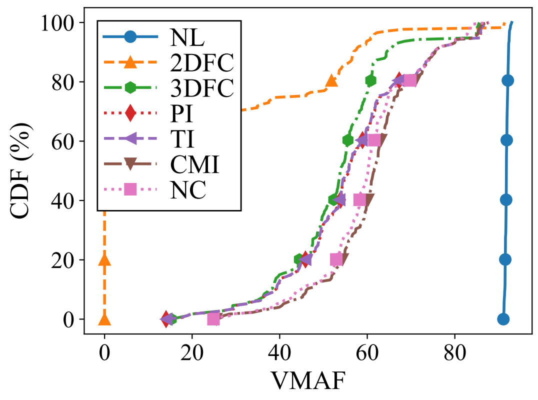
(c)



(d)



(e)



(f)

Figure 6.35: Per-frame visual quality distribution of distorted frames of 10% packet loss from Basketball: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.

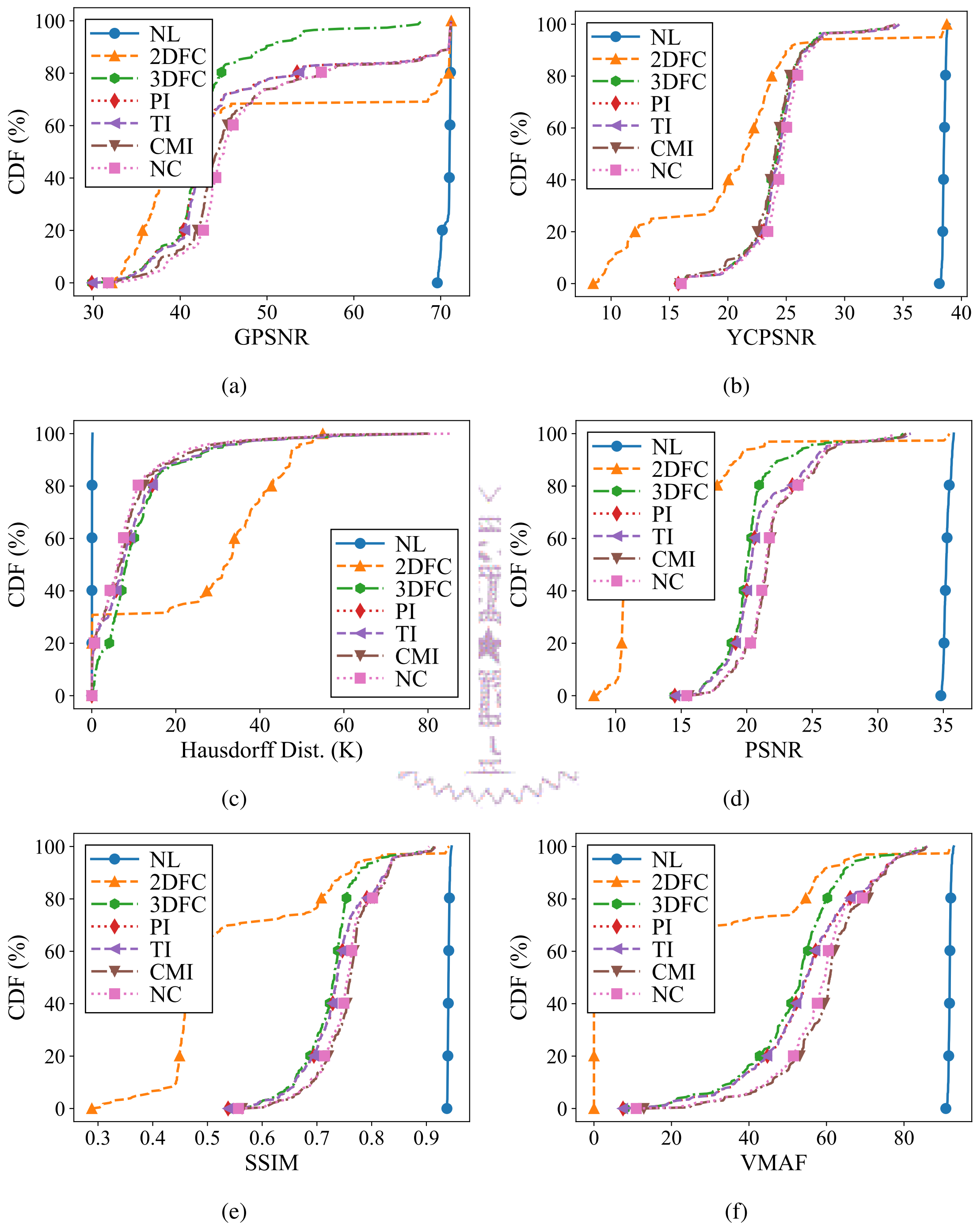
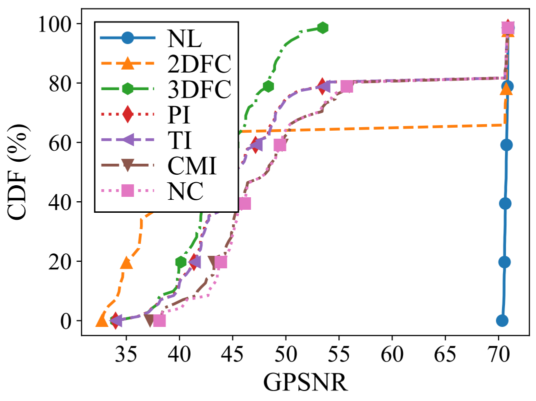
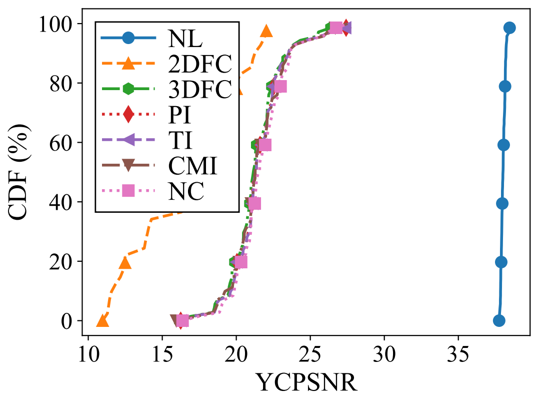


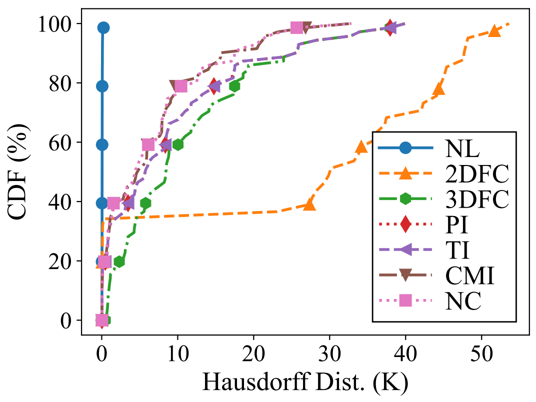
Figure 6.36: Per-frame visual quality distribution of distorted frames of 15% packet loss from Basketball: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



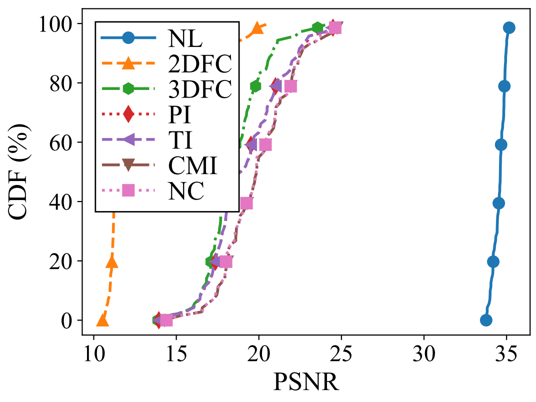
(a)



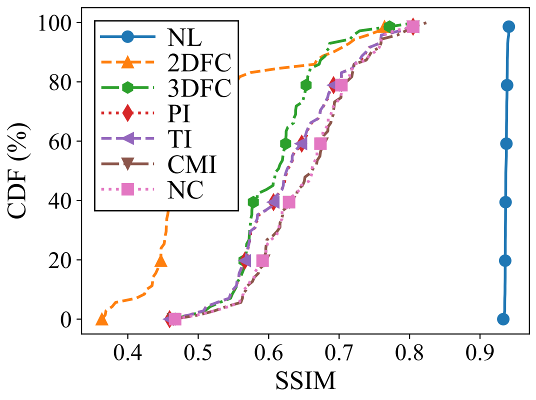
(b)



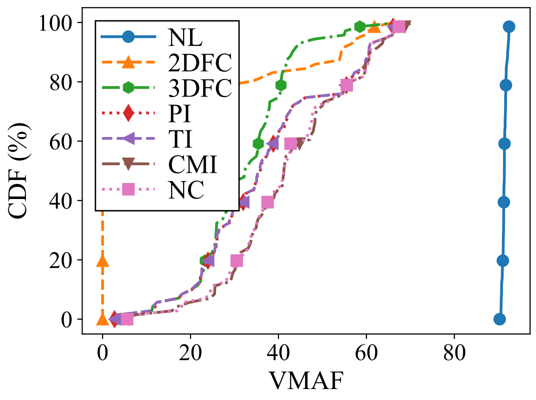
(c)



(d)

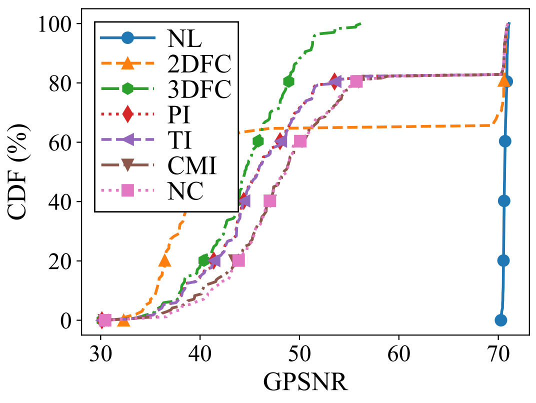


(e)

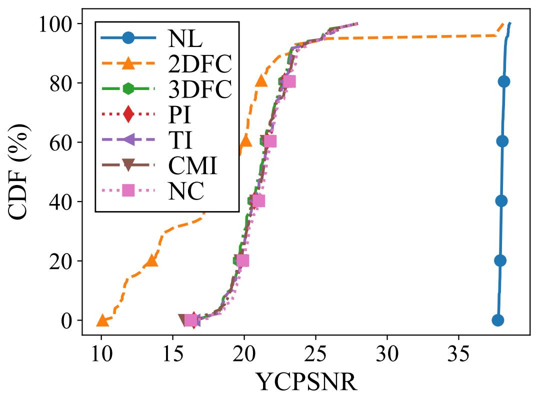


(f)

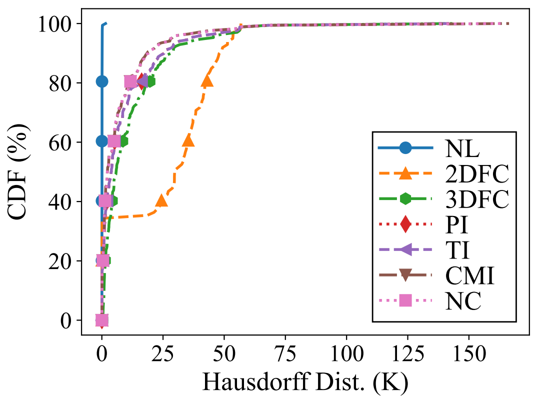
Figure 6.37: Per-frame visual quality distribution of distorted frames of 5% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



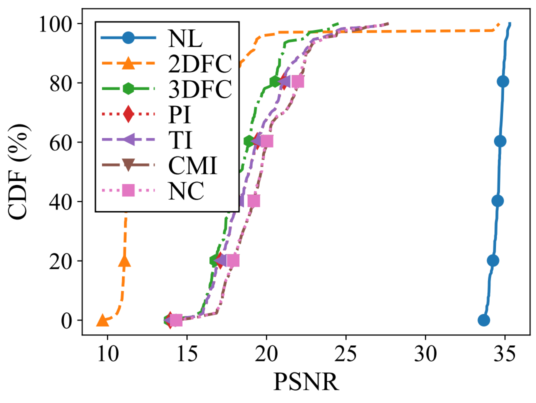
(a)



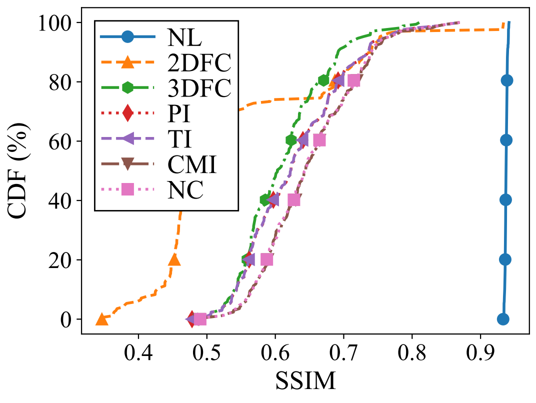
(b)



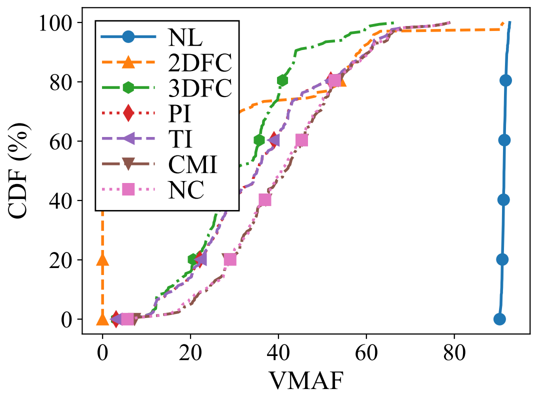
(c)



(d)

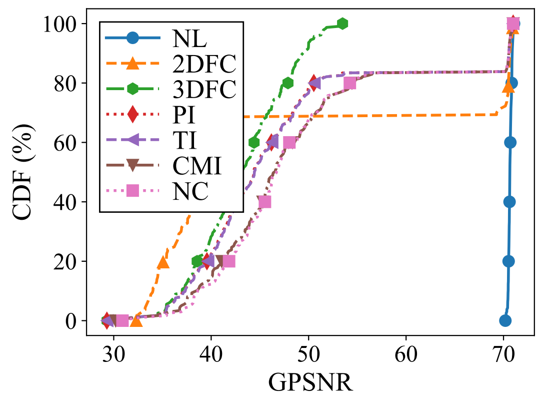


(e)

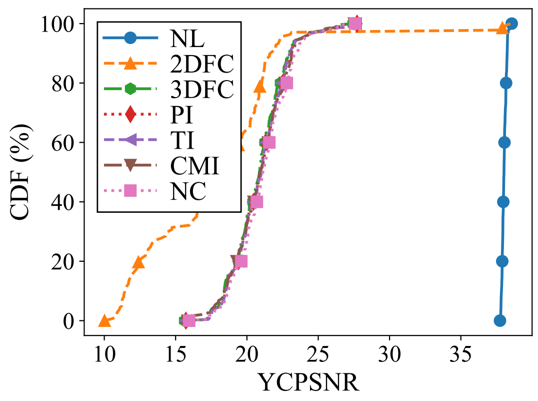


(f)

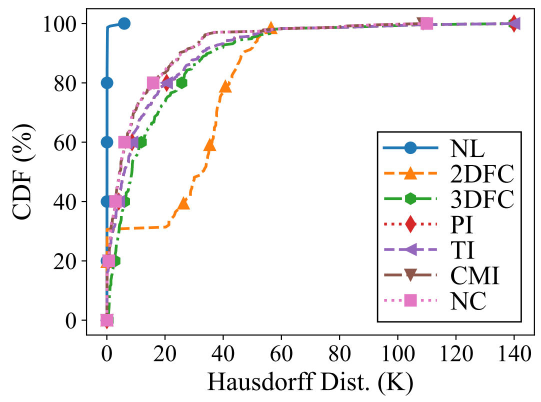
Figure 6.38: Per-frame visual quality distribution of distorted frames of 10% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



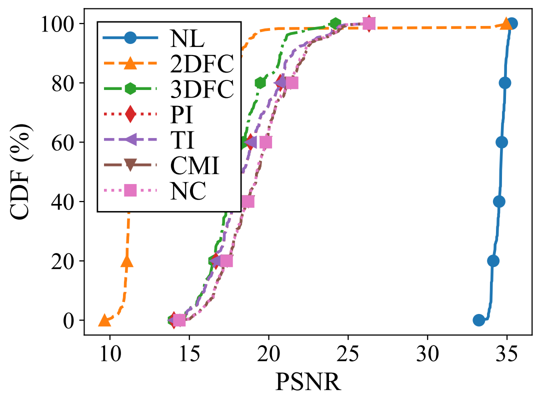
(a)



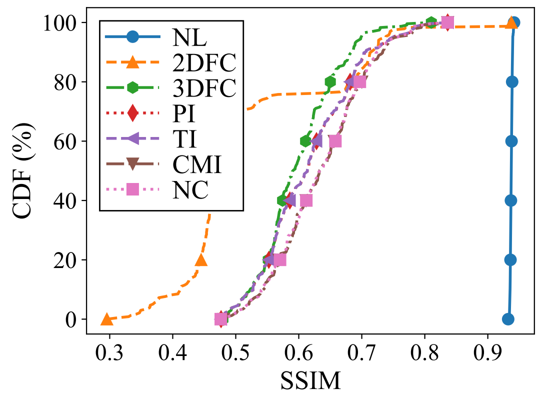
(b)



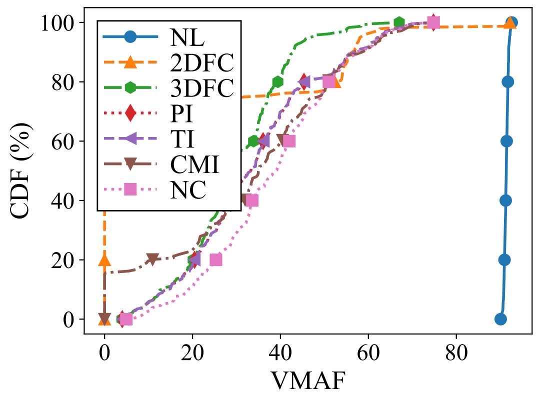
(c)



(d)

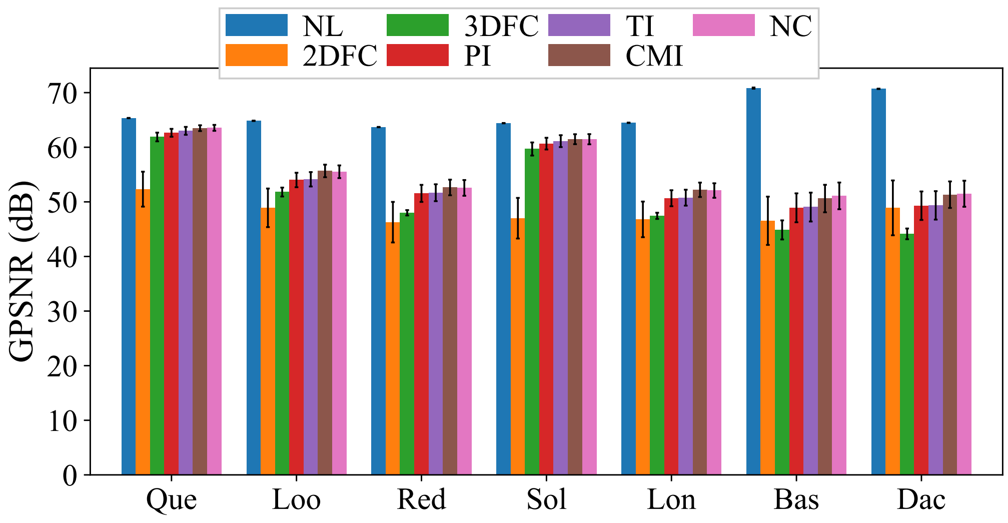


(e)

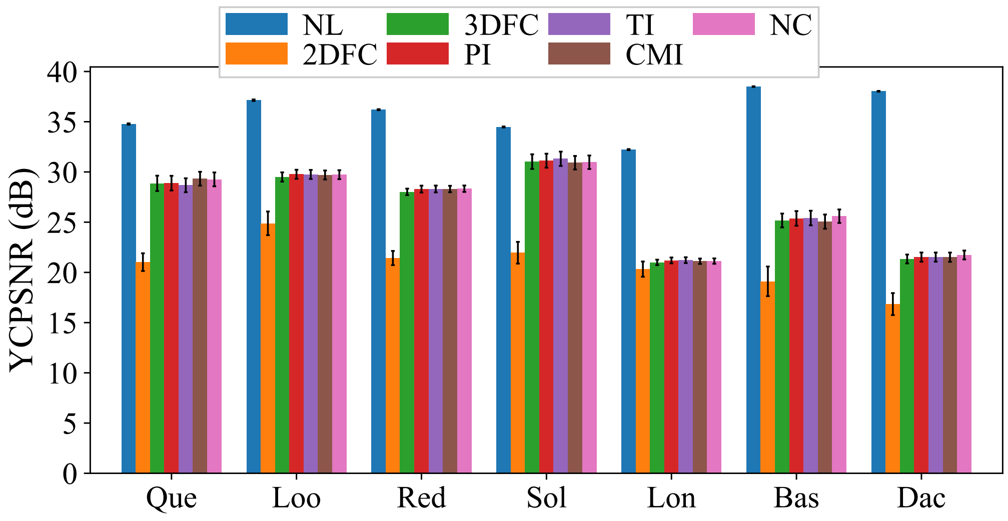


(f)

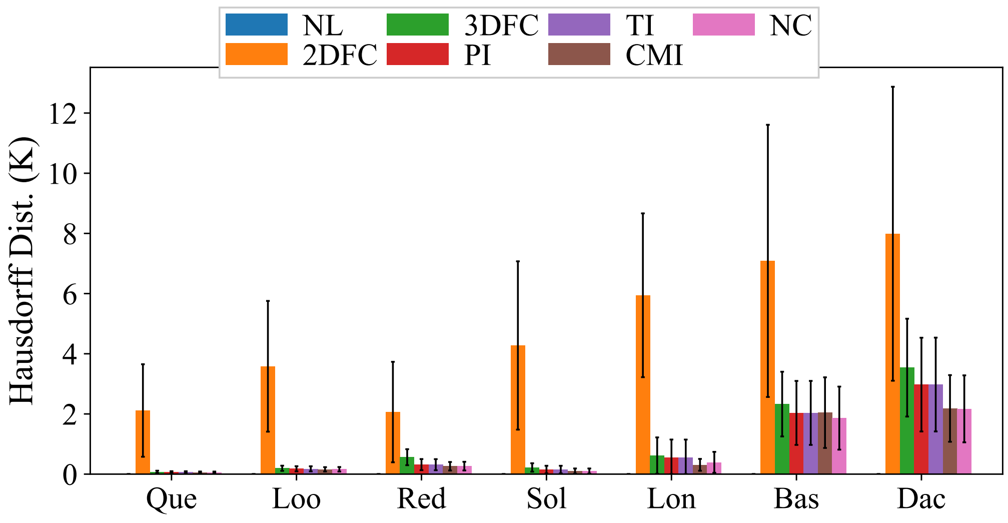
Figure 6.39: Per-frame visual quality distribution of distorted frames of 15% packet loss from Dancer: (a) GPSNR, (b) YCPSNR, (c) Hausdorff distance, (d) PSNR, (e) SSIM, and (f) VMAF.



(a)

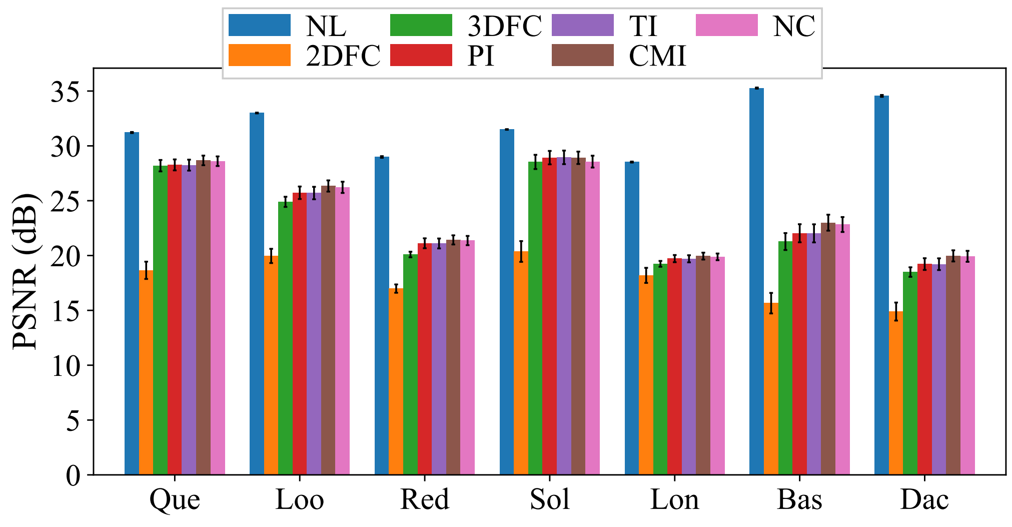


(b)

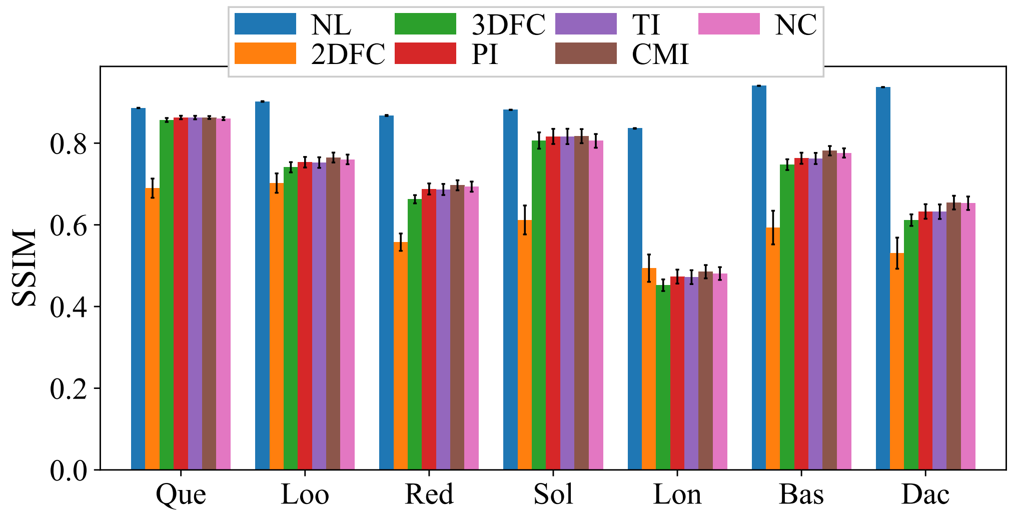


(c)

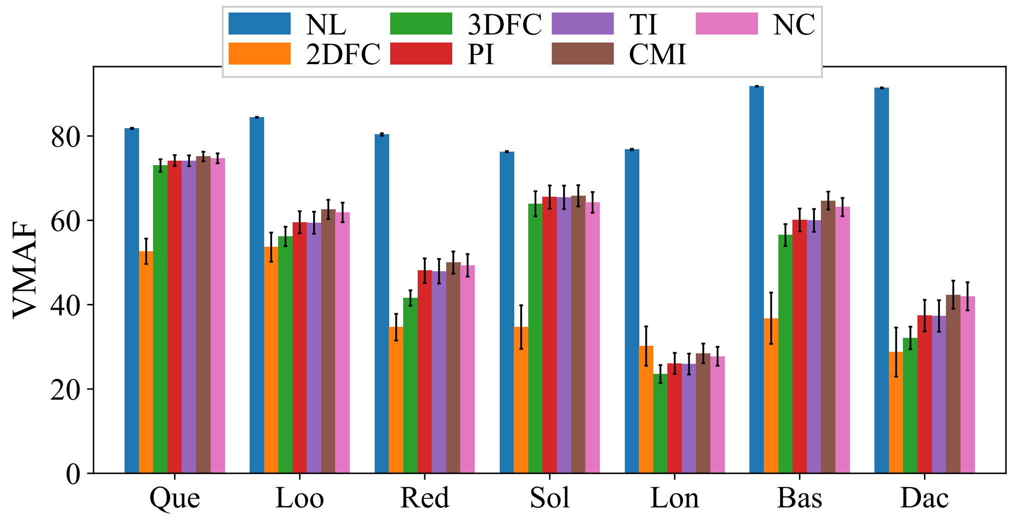
Figure 6.40: Overall 3D visual quality of distorted frames from individual sequences of 5% packet loss: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.



(a)

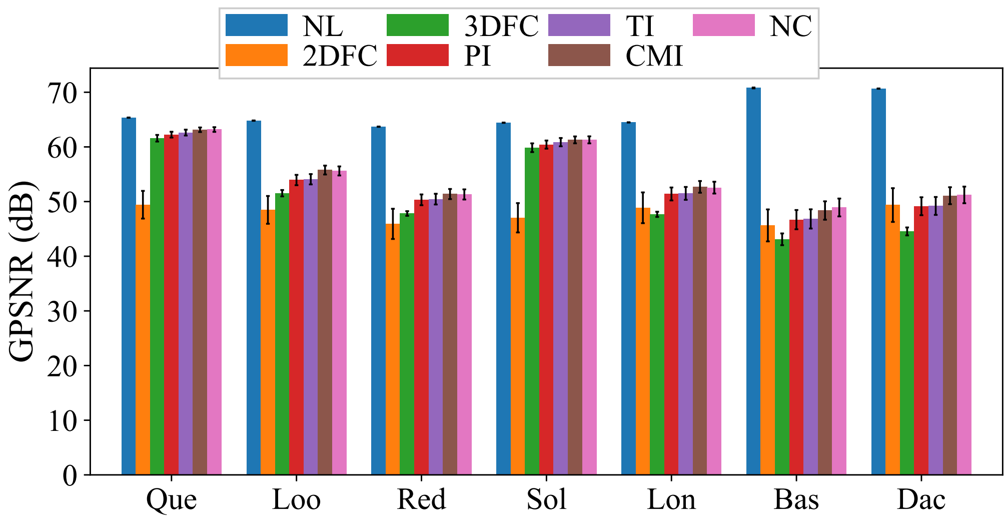


(b)

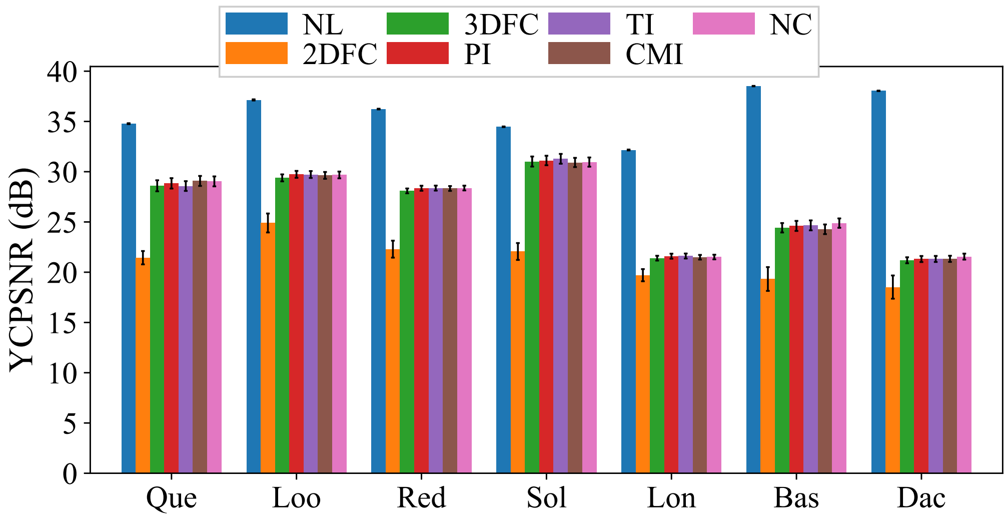


(c)

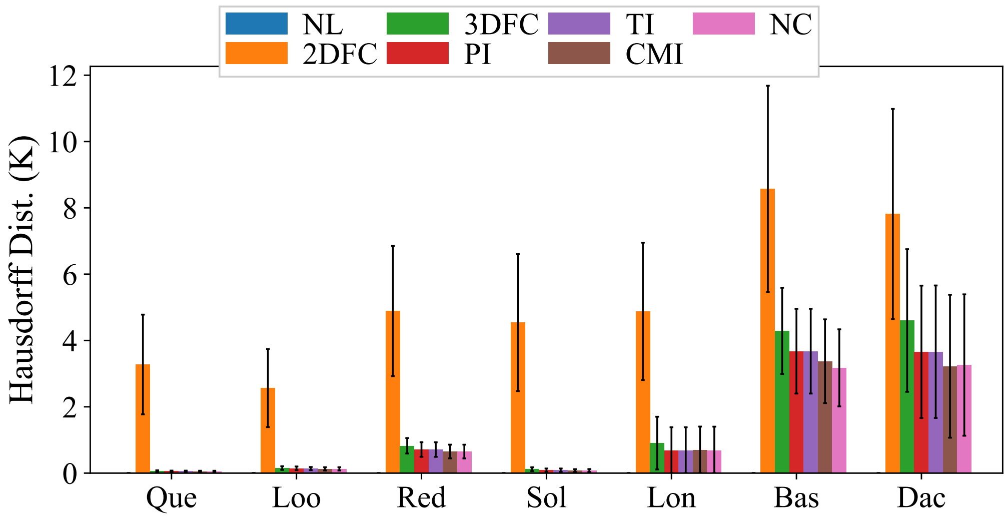
Figure 6.41: Overall 2D visual quality of distorted frames from individual sequences of 5% packet loss: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)

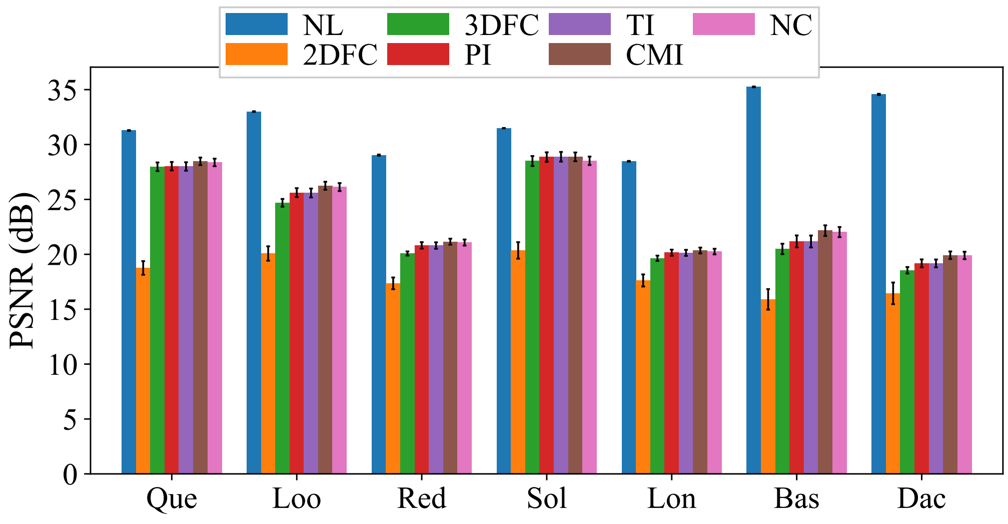


(b)

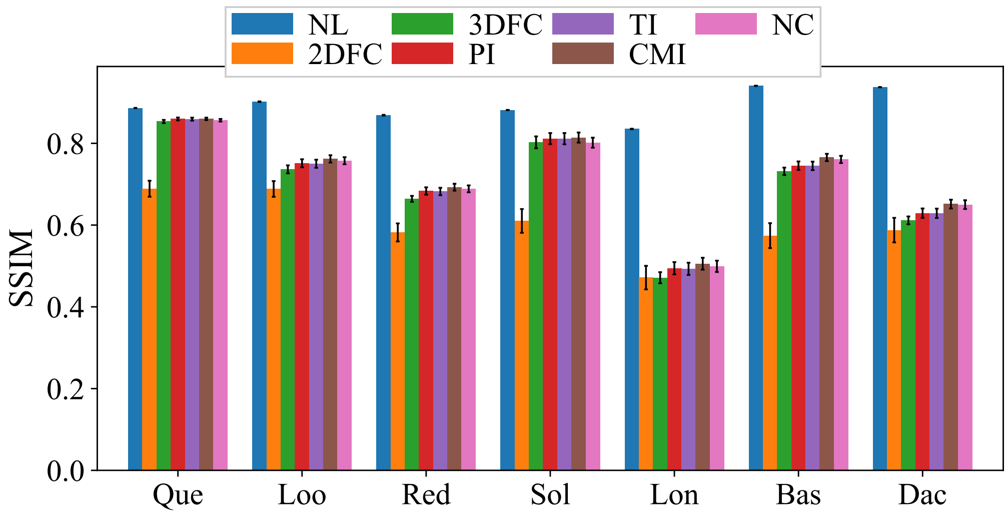


(c)

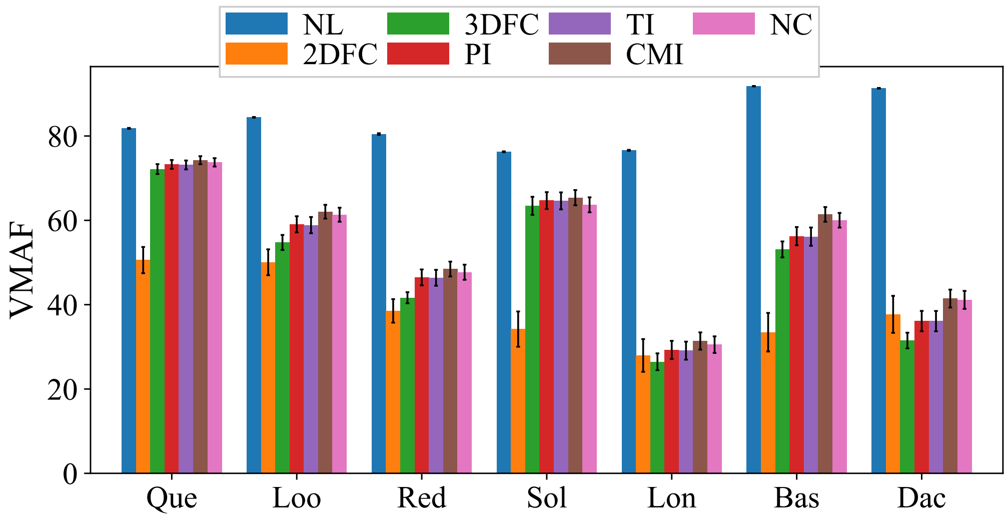
Figure 6.42: Overall 3D visual quality of distorted frames from individual sequences of 10% packet loss: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.



(a)

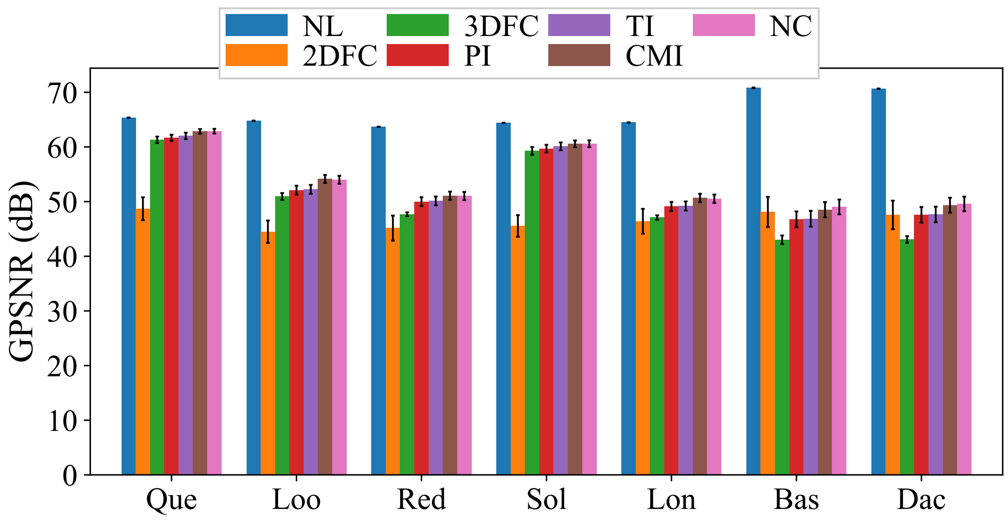


(b)

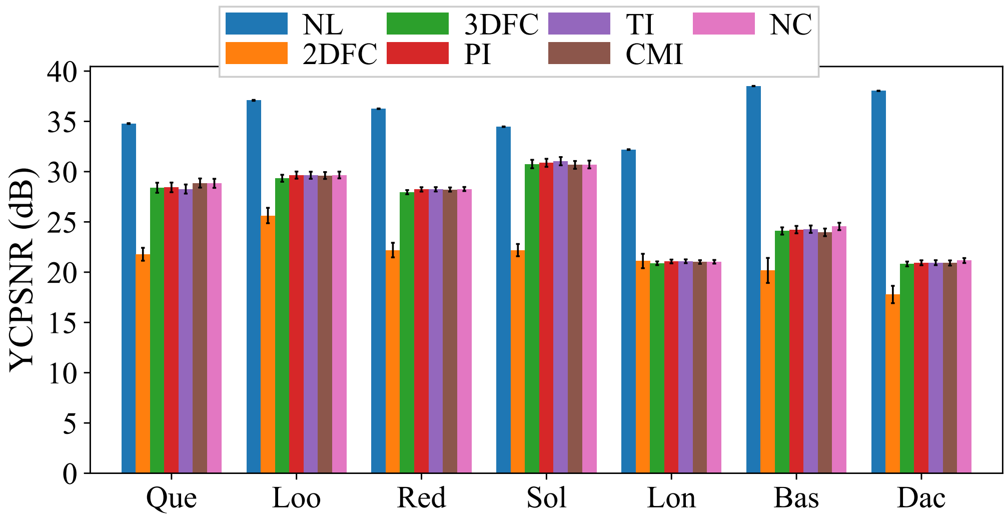


(c)

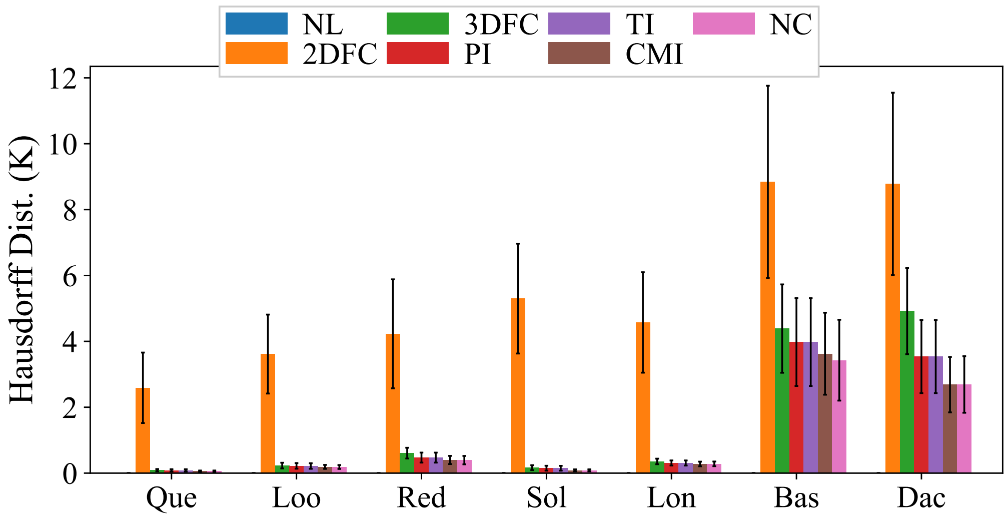
Figure 6.43: Overall 2D visual quality of distorted frames from individual sequences of 10% packet loss: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)

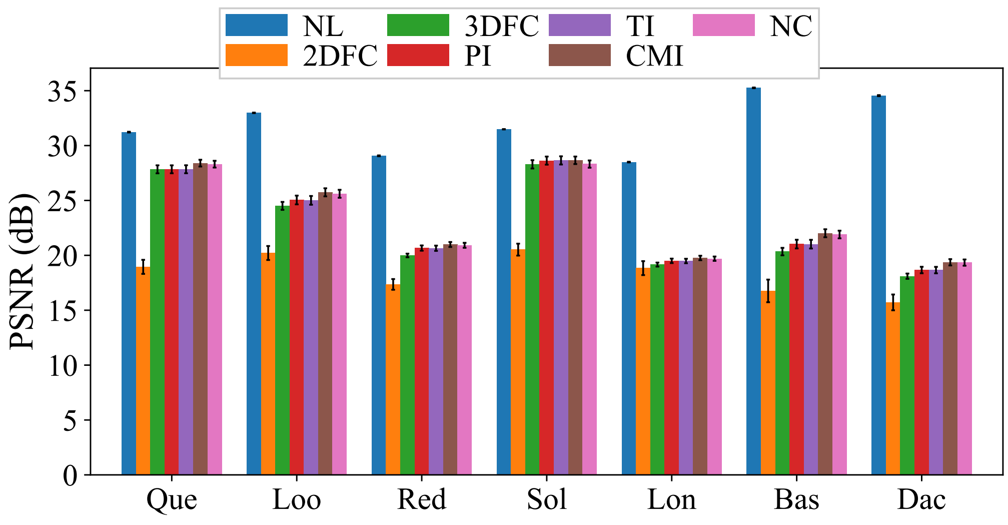


(b)

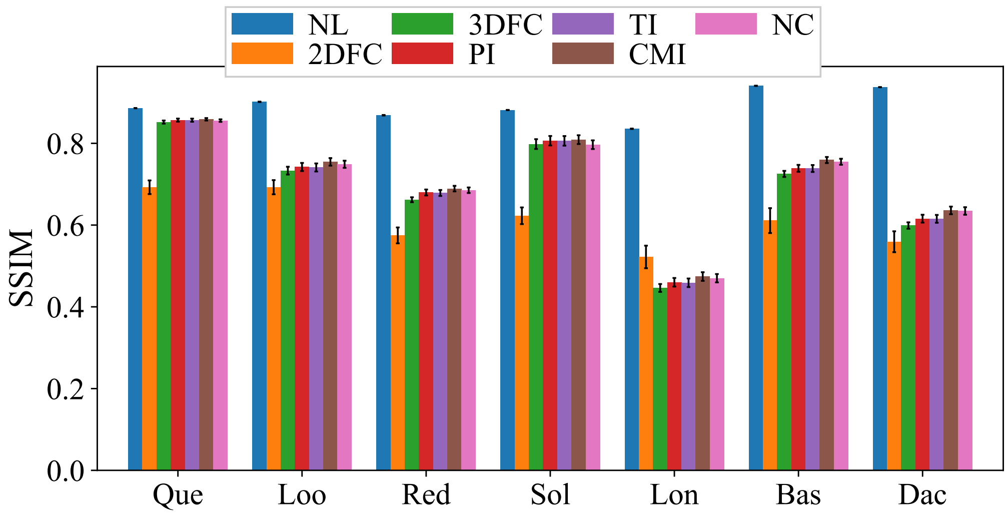


(c)

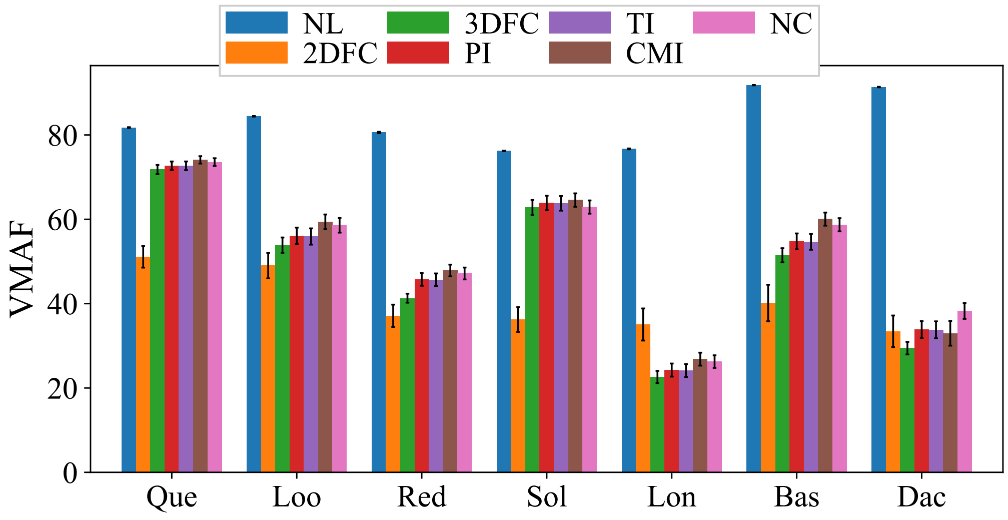
Figure 6.44: Overall 3D visual quality of distorted frames from individual sequences of 15% packet loss: (a) GPSNR, (b) YCPSNR, and (c) Hausdorff distance.



(a)

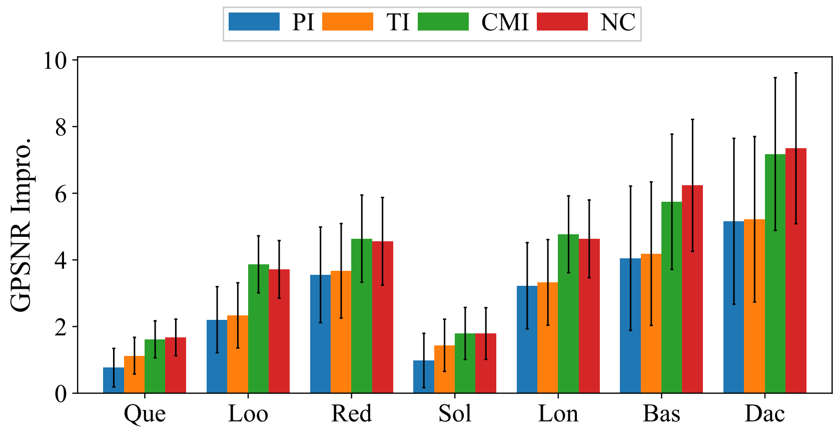


(b)

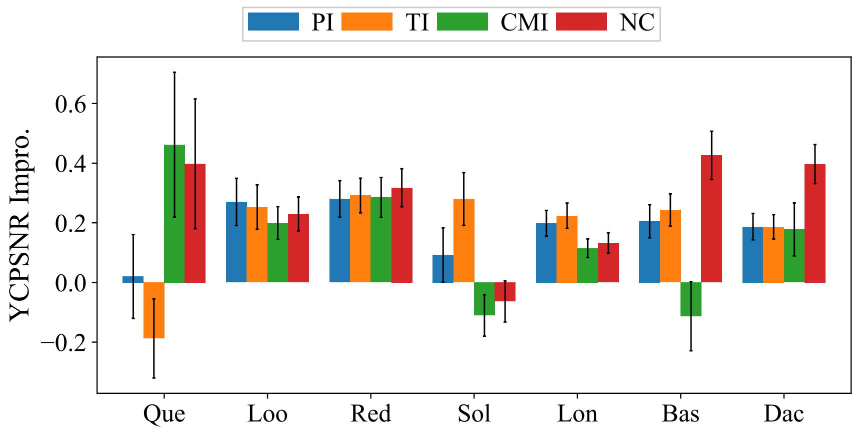


(c)

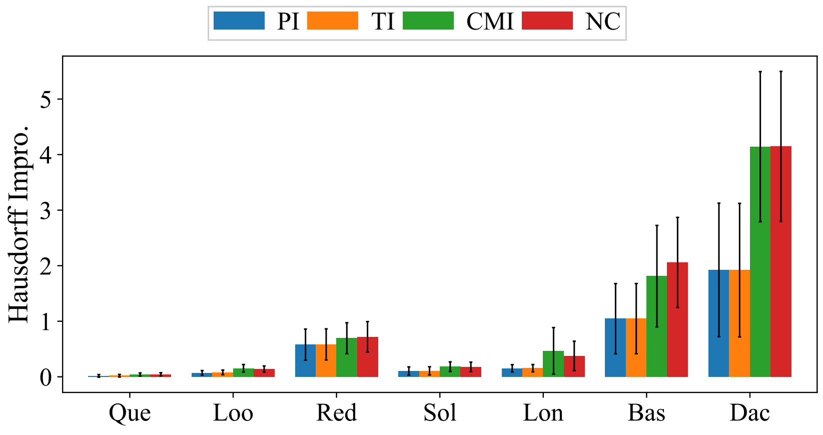
Figure 6.45: Overall 2D visual quality of distorted frames from individual sequences of 15% packet loss: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)

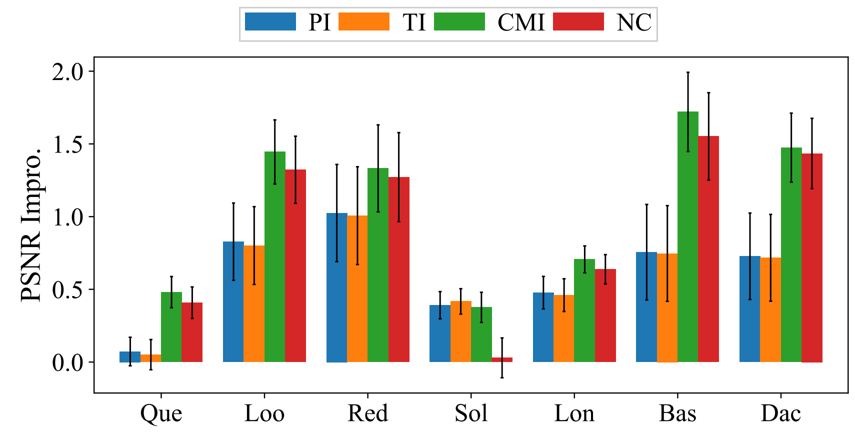


(b)

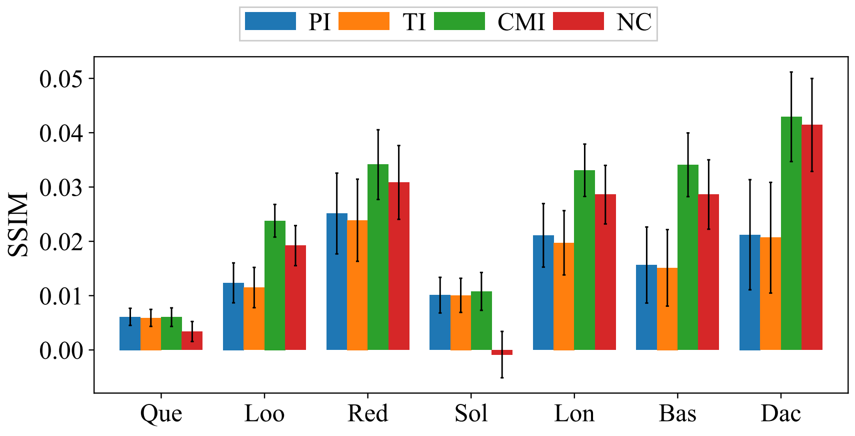


(c)

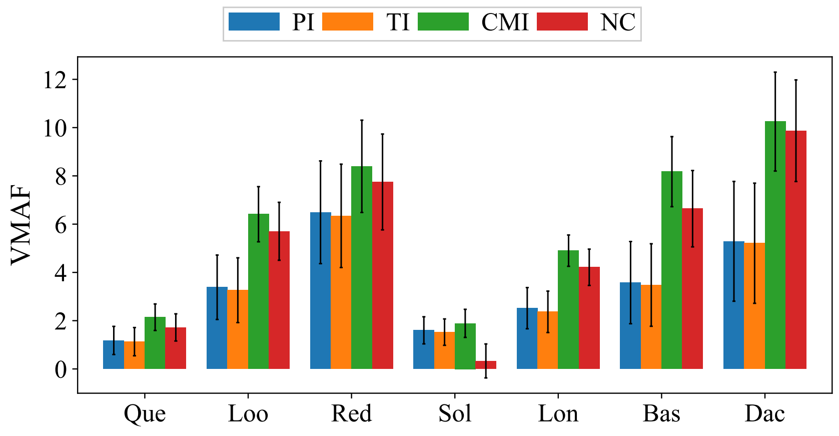
Figure 6.46: Per-sequence average 3D quality improvement of our error concealment algorithms over 3DFC of 5% packet loss: (a) GPSNR ,(b) YCPSNR, and (c) Hausdorff distance.



(a)

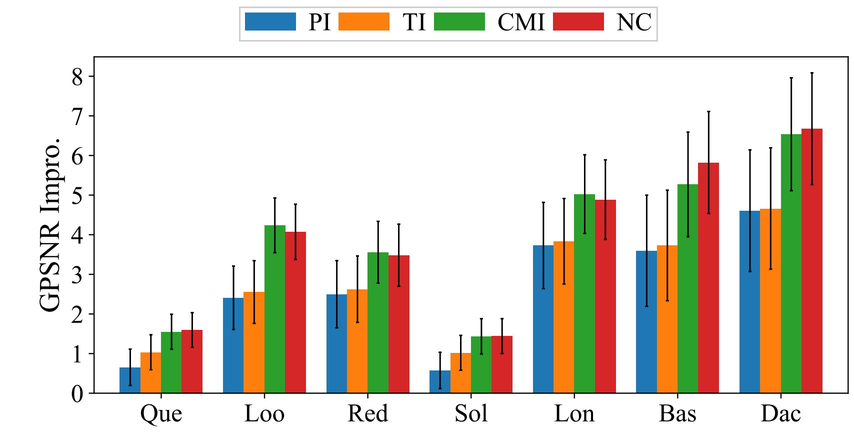


(b)

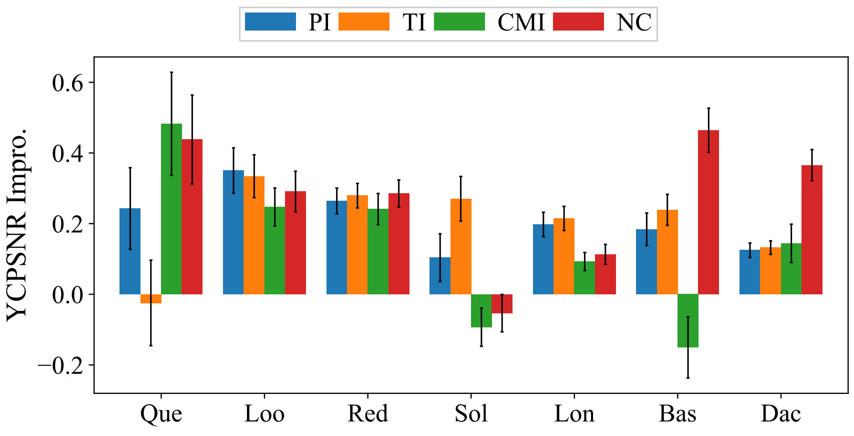


(c)

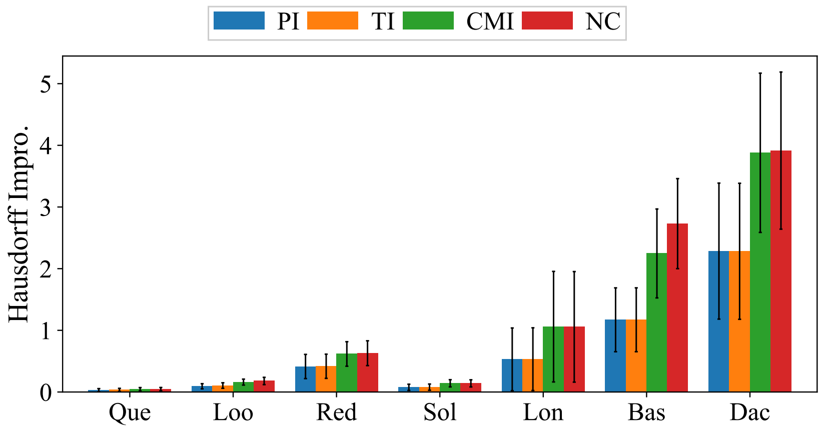
Figure 6.47: Per-sequence average 2D quality improvement of our error concealment algorithms over 3DFC of 5% packet loss: (a) PSNR ,(b) SSIM, and (c) VMAF.



(a)

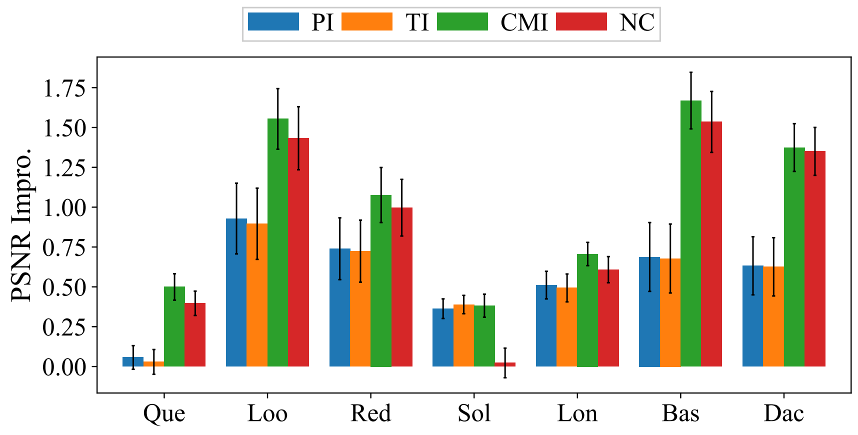


(b)

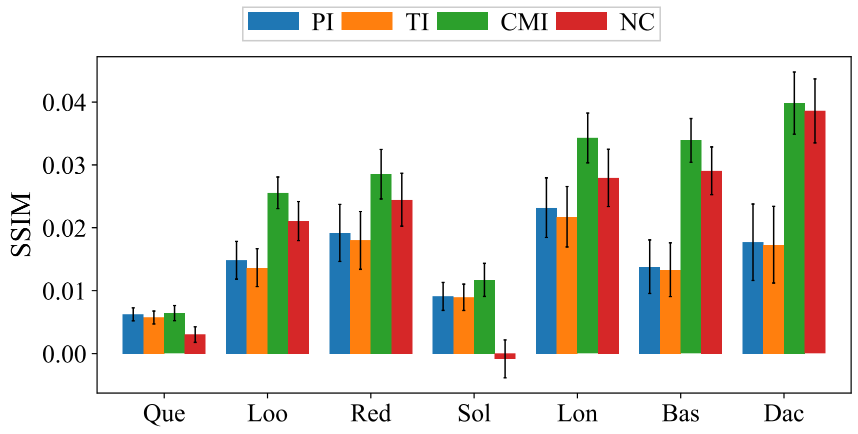


(c)

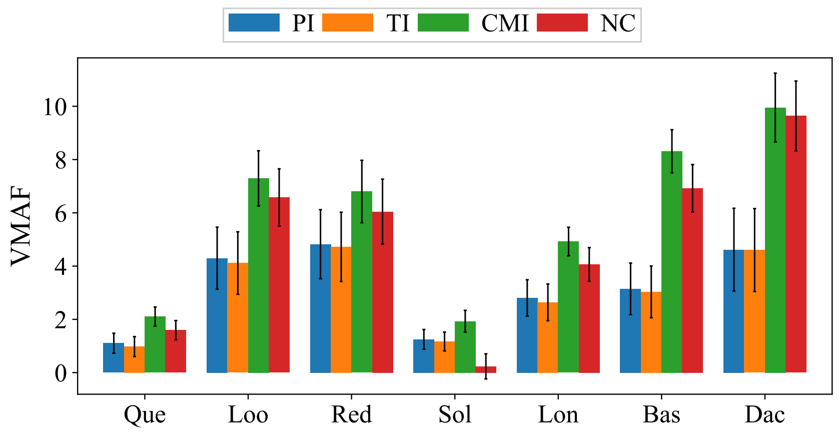
Figure 6.48: Per-sequence average 3D quality improvement of our error concealment algorithms over 3DFC of 10% packet loss: (a) GPSNR ,(b) YCPSNR, and (c) Hausdorff distance.



(a)

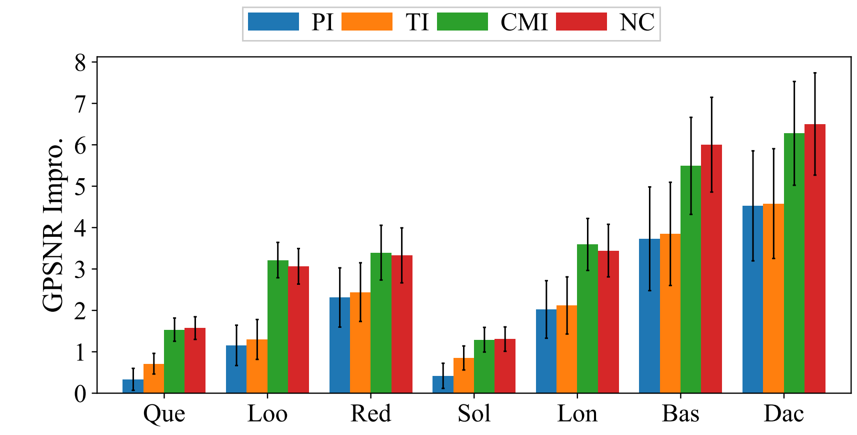


(b)

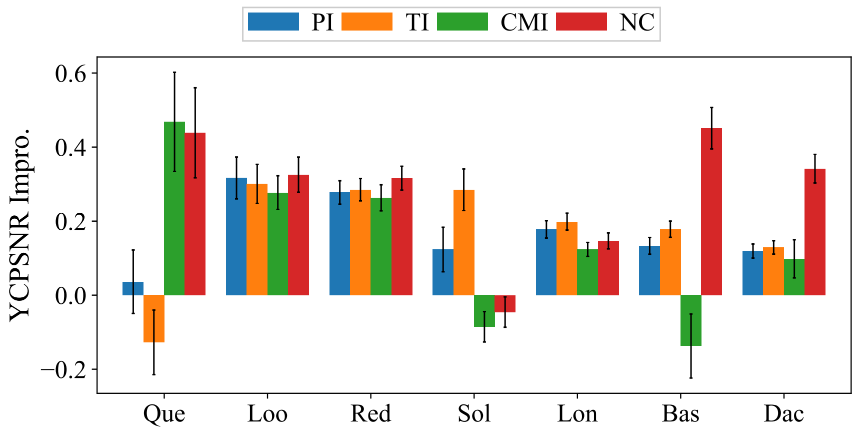


(c)

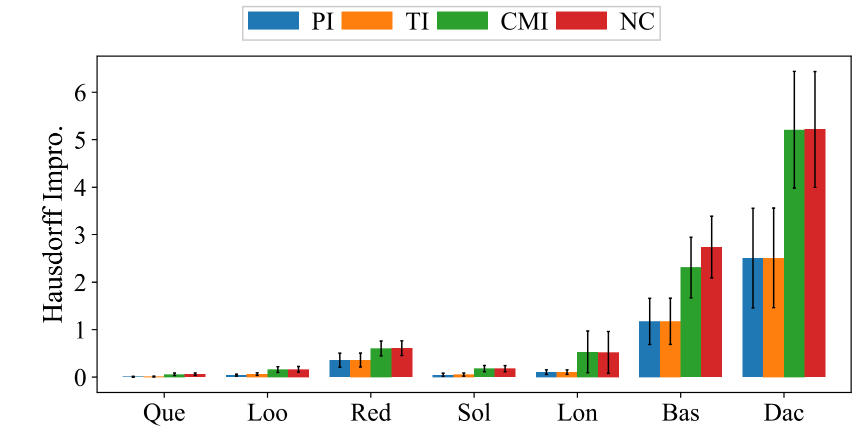
Figure 6.49: Per-sequence average 2D quality improvement of our error concealment algorithms over 3DFC of 10% packet loss: (a) PSNR ,(b) SSIM, and (c) VMAF.



(a)

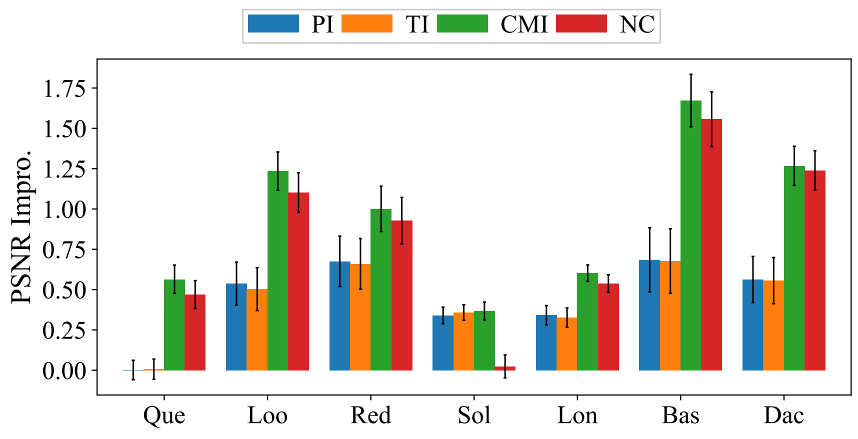


(b)

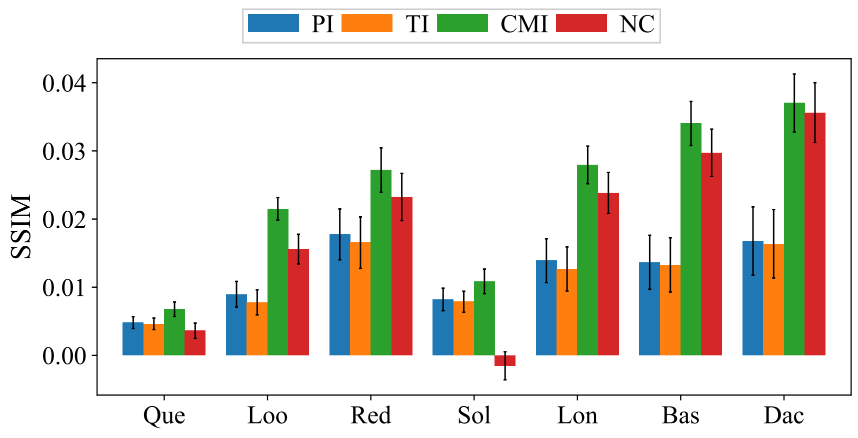


(c)

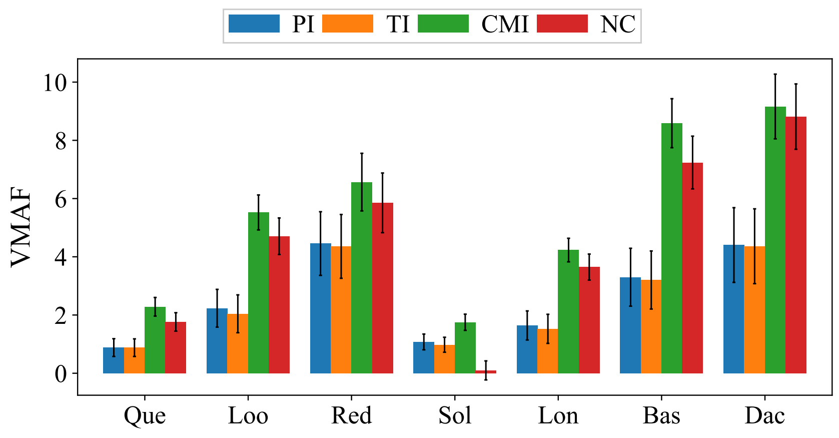
Figure 6.50: Per-sequence average 3D quality improvement of our error concealment algorithms over 3DFC of 15% packet loss: (a) GPSNR ,(b) YCPSNR, and (c) Hausdorff distance.



(a)



(b)



(c)

Figure 6.51: Per-sequence average 2D quality improvement of our error concealment algorithms over 3DFC of 15% packet loss: (a) PSNR ,(b) SSIM, and (c) VMAF.

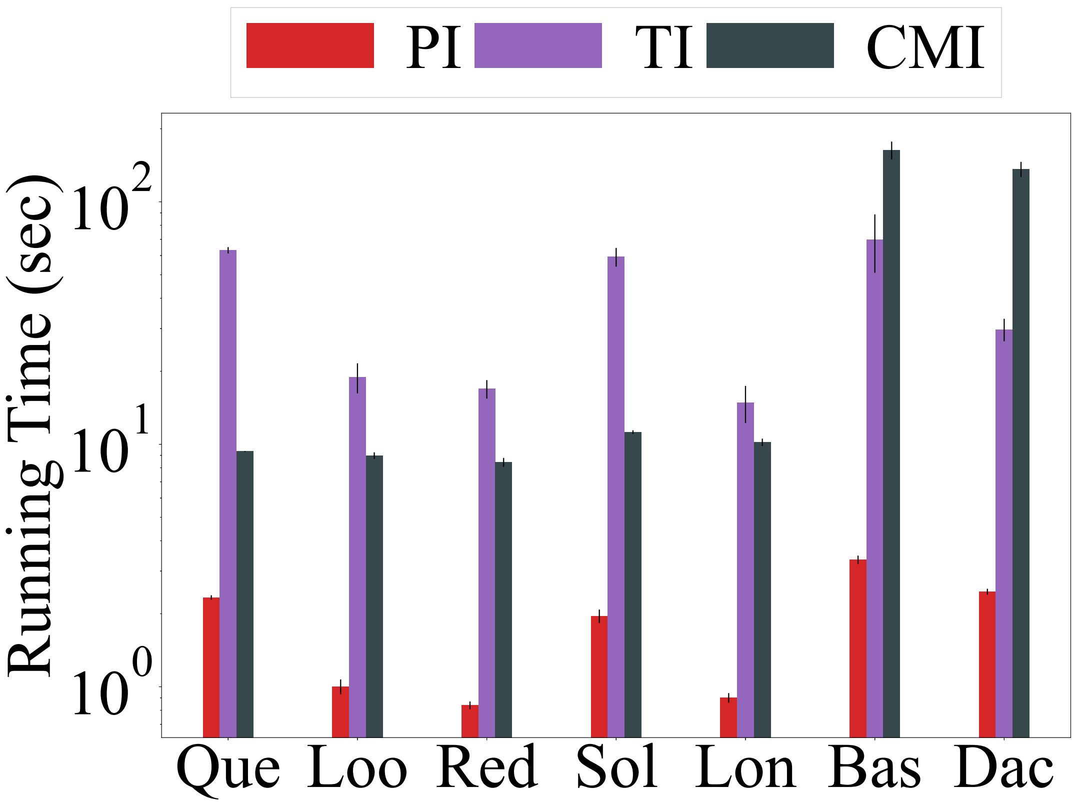


Figure 6.52: Average per-frame running time of our error concealment algorithms.

Chapter 7

User Study

To collect human evaluations of the techniques, we carried out a pair-wise comparison user study ($n = 21$, mean age X, Y female). Participants had no prior experience in picture quality evaluation, and were recruited via a call to participation through a university advertisement page.

7.1 Design

The user studies followed ITU recommendations [34], to take place in a dimly lit room, with a fixed viewing distance (i.e., $4H = 120$cm distance between the viewer and the screen). Video clips of the same model, but from different source techniques, were used as pairs for comparison. Video order was chosen randomly for each participant with a pairs of videos being played one at a time and separated by a two-second mid-grey page. Participants saw 35 pairs of videos (each combination of videos from 7 models and 5 techniques). After seeing each pair of videos, participants were posed three questions on a single voting page: (i) "*Which video was smoother?*", (ii) "*Which video had better image quality?*", and (iii) "*Which video did you prefer?*", and for each of these questions participants could answer either "*First video*" or "*Second video*". On average participants took 40 minutes to complete training and main task.

Upon beginning the user study, the workflow and goal of the study was explained to participants and they were asked to provide consent. Next, (as per ITU recommendations) participants completed two vision tests to test for visual acuity and colour blindness. After this, participants received detailed task instructions, and completed training to familiarise themselves with the interface and experiment (training used models not included in the main user study).

Chapter 8

Conclusion and Outlook

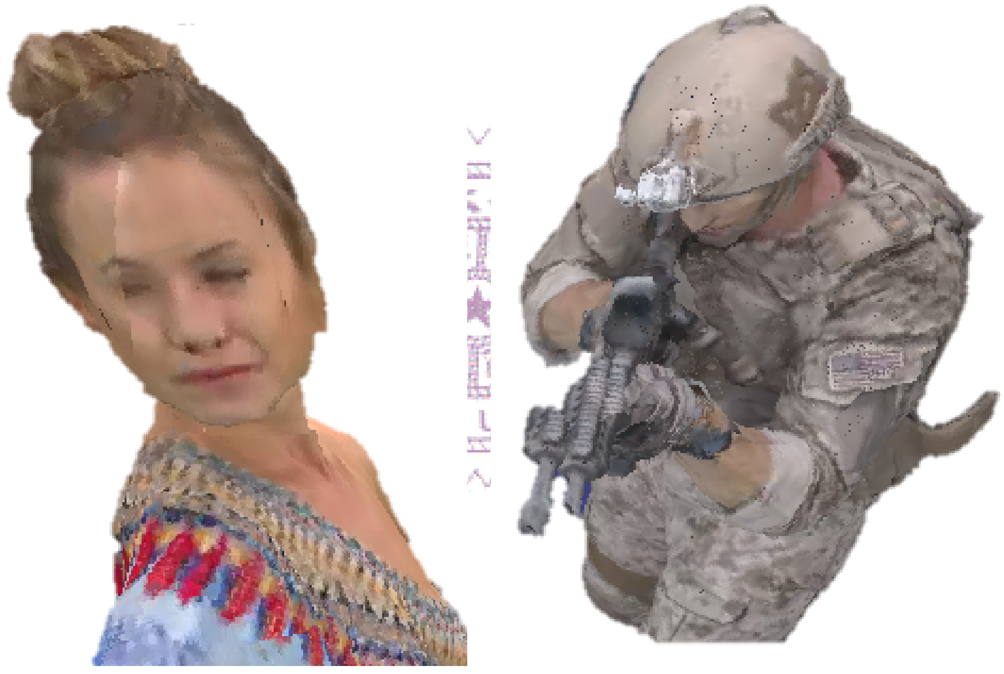

(a) (b)

Figure 8.1: Less than perfect concealed point clouds due to: (a) extrusion of facial expression and (b) rotation of the rifle.

We conclude our work and list the future work in this chapter.

8.1 Conclusion

In this paper, we studied the uninvestigated problem of error concealment techniques of the emerging dynamic 3D point cloud streaming. We developed a general framework to explore the design space of these error concealment algorithms as a sequence of steps. By introducing different techniques in individual steps and skipping some steps, we present a suite of error concealment algorithms, which perform point cloud interpolation (PI, TI, CMI, and NCMI) or extrapolation (PE, TE, CME, and NCME). Through extensive experiments, we show that our proposed error concealment algorithms: (i) significantly outperform the baseline 2DFC and 3DFC in both 2D and 3D visual quality and (ii) achieve different tradeoff points between the computational complexity and visual quality. Our experiments shed some light on the unique challenges of concealing dynamic 3D point clouds, and our proposed framework will stimulate more future development of novel error concealment algorithms.

8.2 Future Work

This work can be extended in multiple dimensions. First, different usage scenarios, in addition to (online) tele-conferencing and (offline) frame-rate upsampling, can be studied, which dictates different requirements under diverse constraints. New design decisions, such as the random-access and low-delay modes of V-PCC, could be leveraged for these usage scenarios. We conclude the future work as the following:

- **Graphic Processing Unit (GPU) Acceleration.** We have reported the running time with single-cored executions. We believe that general-purpose GPU (GPGPU) and parallel computations can bring our implementation one step closer to real-time applications.

- **Cube Motion Improvement.** We want to look deeply into motion vectors of each divided cube for avatars through mathematically formulate problems of vectors. We also want to store residual value in advance during the encoding process for better matching of the cubes and lower processing time. If we could relax cubes into cuboids, the artifacts due to rotation and extrusion, shown in Fig. 8.1(a), may be eliminated.

- **Rotations.** Some of the cubes moving across frames are in the manner of rotations rather than rigid transformations, as depicted in Fig. 8.1(b). Although there won't be obvious artifacts for point-based algorithms like PI and NCMI, the effectiveness will still degrade. The concern for implementing rotation is that high complexity may gain significant running time for dense point cloud applications.

- **Real System.** We want to establish the end-to-end system starting from dynamic 3D point clouds, encoding, transmitting through real or simulated networks, decoding, error concealment, and rendering to users with adaptive bitrate mechanism. Then the concealed streaming of dynamic point clouds can be consumed in 6DoF via HMDs, providing immersive experiences under non-perfect network conditions.

- **Spatial Concealment.** We assume all point cloud frames are lost, which may not be always the case. All of our concealment algorithms are in the temporal domain because from our reference software V-PCC, the catastrophic distortion happens in damaged frames. There could be still some useful information residing in damaged frames or metadata that can be used for intra-frame concealment. When only partial frames are lost, intra-frame techniques can be employed for better visual quality.

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 *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1732–1736, Quebec City, QC, Canada, 2015.

[2] ASKA3D. ASKA3D, 2022. https://aska3d.com/en/.

[3] R. Bar-Yehuda and C. Gotsman. Time/space tradeoffs for polygon mesh rendering. *ACM Transactions on Graphics*, 15(2):141–152, April 1996.

[4] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[5] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus. In *Proc of 11th Pacific Conference on Computer Graphics and Applications, 2003.*, pages 335–343, Canmore, Canada, October 2003.

[6] J. M. Boyce, R. Doré, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. M. Vadakital, and L. Yu. Mpeg immersive video coding standard. *Proceedings of the IEEE*, 109(9):1521–1536, 2021.

[7] C. Cao, M. Preda, and T. Zaharia. 3D point cloud compression: A survey. In *Proc. of ACM International Conference on 3D Web Technology (Web3D'19)*, pages 1–9, Los Angeles, CA, July 2019.

[8] B.-Y. Chen and T. Nishita. Multiresolution streaming mesh with shape preserving and QoS-like controlling. In *Proc. of ACM International Conference on 3D Web Technology (Web3D'02)*, pages 35–42, Tempe, Arizona, February 2002.

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

[10] W. Cheng and W. T. Ooi. Receiver-driven view-dependent streaming of progressive mesh. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'08)*, pages 9–14, Braunschweig, Germany, May 2008.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, pages 129–136, Salerno, Italy, 2008.

[12] A. Clemm, M. T. Vega, H. K. Ravuri, T. Wauters, and F. D. Turck. Toward truly immersive holographic-type communication: Challenges and solutions. *IEEE Communications Magazine*, 58(1):93–99, 2020.

[13] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics*, 34(4):1–13, 2015.

[14] C. Conti, L. D. Soares, and P. Nunes. Dense light field coding: A survey. *IEEE Access*, 8:49244–49284, 2020.

[15] K. De Miguel, A. Brunete, M. Hernando, and E. Gambao. Home camera-based fall detection system for the elderly. *Sensors*, 17(12), December 2017.

[16] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou. 8i voxelized full bodies - a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, 2017. Geneva, CH.

[17] M. Domański, O. Stankiewicz, K. Wegner, and T. Grajek. Immersive visual media — mpeg-i: 360 video, virtual navigation and beyond. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–9, Poznan, Poland, July 2017.

[18] Filament, 2022. https://github.com/google/filament.

[19] T. Forgione, A. Carlier, G. Morin, W. T. Ooi, V. Charvillat, and P. K. Yadav. DASH for 3D networked virtual environment. In *Proc. of ACM International Conference on Multimedia (MM'18)*, pages 1910–1918, Seoul, Republic of Korea, October 2018.

[20] Z. Fu, W. Hu, and Z. Guo. 3d dynamic point cloud inpainting via temporal consistency on graphs. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[21] T. Georgiev, Z. Yu, A. Lumsdaine, and S. Goma. Lytro camera technology: Theory, algorithms, performance analysis. *Multimedia Content and Mobile Devices*, 8667:458 – 467, 2013.

[22] 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, 2020.

[23] A. Guéziec, G. Taubin, B. Horn, and F. Lazarus. A framework for streaming geometry in VRML. *IEEE Computer Graphics and Applications*, 19(2):68–78, 1999.

[24] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proc. of ACM Annual International Conference on Mobile Computing and Networking (MobiCom'18)*, pages 1–13, New Delhi, India, October 2020.

[25] G. Haßlinger and O. Hohlfeld. The gilbert-elliott model for packet loss in real time services on the internet. In *14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems*, pages 1–15. VDE, 2008.

[26] J. He, Z. Fu, W. Hu, and Z. Guo. Point cloud attribute inpainting in graph spectral domain. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4385–4389. IEEE, 2019.

[27] HEVC Test Model (HM) documentation, September 2014. http://hevc.hhi.fraunhofer.de/HM-doc/.

[28] S. Hojati, M. Kazemi, and P. Moallem. Error concealment with parallelogram partitioning of the lost area. *Multimedia Tools and Applications*, pages 1–21, 2019.

[29] M. Hosseini and C. Timmerer. Dynamic adaptive point cloud streaming. In *Proc. of ACM Packet Video Workshop (PV'18)*, pages 25–30, Amsterdam, The Netherlands, June 2018.

[30] S.-Y. Hu, T.-H. Huang, S.-C. Chang, W.-L. Sung, J.-R. Jiang, and B.-Y. Chen. Flod: A framework for peer-to-peer 3D streaming. In *Proc. of IEEE Conference on Computer Communications (INFOCOM'08)*, pages 1373–1381, Phoenix, AZ, April 2008.

[31] 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, 2019.

[32] B. Hwang, J. Jo, and C. Ri. An improved multi-directional interpolation for spatial error concealment. *Springer Multimedia Tools and Applications*, 78(2):2587–2598, 2019.

[33] Intel. Intel RealSense technology, 2022.

[34] ITU-R. Parameter values for the HDTV standards for production and international programme exchange. Recommendation ITU-R BT.709-6, 2015.

[35] 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 [standards in a nutshell]. *IEEE Signal Processing Magazine*, 36(3):118–123, 2019.

[36] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel(r) realsense(tm) stereoscopic depth cameras. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1267–1276, Honolulu, USA, July 2017.

[37] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Transactions on Graphics*, 32:1–12, 2013.

[38] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. GROOT: a real-time streaming system of high-fidelity volumetric videos. In *Proc. of ACM Annual International Conference on Mobile Computing and Networking (MobiCom'20)*, pages 1–14, London, United Kingdom, September 2020.

[39] 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, 2020.

[40] Looking Glass Factory. Looking Glass Factory, 2022. https://lookingglassfactory.com/.

[41] K. Mamou, T. Zaharia, and F. Preteux. Famc: The mpeg-4 standard for animated mesh compression. In *2008 15th IEEE International Conference on Image Processing*, pages 2676–2679, San Diego, USA, October 2008.

[42] D. Miyazaki, N. Hirano, Y. Maeda, S. Yamamoto, T. Mukai, and S. Maekawa. Floating volumetric image formation using a dihedral corner reflector array device. *Applied Optics*, 52(1):A281–A289, Jan 2013.

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

[44] U. Muhammad, H. Xiangjian, L. Kin-Man, X. Min, B. S. M. Matloob, and C. Jinjun. Frame interpolation for cloud-based mobile video streaming. *IEEE Transactions on Multimedia*, 18(5):831–839, 2016.

[45] OpenCV. Depth Map from Stereo Images, 2022. https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html/.

[46] P. Paudyal, J. Gutiérrez, P. Le Callet, M. Carli, and F. Battisti. Characterization and selection of light field content for perceptual assessment. In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, Erfurt, Germany, May 2017.

[47] X. Qin, G. Wu, J. Lei, F. Fan, X. Ye, and Q. Mei. A novel method of autonomous inspection for transmission line based on cable inspection robot lidar data. *Sensors*, 18(2):596, 2018.

[48] S. Rusinkiewicz and M. Levoy. Streaming QSplat: A viewer for networked visualization of large, dense models. In *Proc. of ACM Symposium on Interactive 3D Graphics (I3D'01)*, pages 63–68, Chapel Hill, NC, March 2001.

[49] R. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proc. of IEEE International Conference on Robotics and Automation (ICRA'11)*, pages 1–4, Shanghai, China, May 2011.

[50] N. Sabater, G. Boisson, B. Vandame, P. Kerbiriou, F. Babon, M. Hog, R. Gendrot, T. Langlois, O. Bureller, A. Schubert, and V. Allie. Dataset and pipeline for multiview light-field video. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 30–40, Hawaii, HI, USA, July 2017.

[51] A. Sankisa, A. Punjabi, and A. K. Katsaggelos. Video error concealment using deep neural networks. In *IEEE International Conference on Image Processing (ICIP'18)*, pages 380–384, Athens, Greece, October 2018.

[52] A. Sankisa, A. Punjabi, and A. K. Katsaggelos. Temporal capsule networks for video motion estimation and error concealment. *Signal, Image and Video Processing*, pages 1–9, 2020.

[53] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, et al. Emerging MPEG standards for point

cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018.

[54] A. Sengupta, F. Jin, R. Zhang, and S. Cao. mm-Pose: Real-time human skeletal posture estimation using mmWave radars and CNNs. *IEEE Sensors Journal*, 20(17):10032–10044, May 2020.

[55] Y. Sun, R. Hang, Z. Li, M. Jin, and K. Xu. Privacy-preserving fall detection with deep learning on mmwave radar signal. In *Proc. of 2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, Sydney, Australia, December 2019.

[56] A. Sánchez-Rodríguez, M. Soilán, M. Cabaleiro, and P. Arias. Automated inspection of railway tunnels' power line using lidar point clouds. *Remote Sensing*, 11(21):2567, November 2019.

[57] T. I. D. Team. ImageMagick, 2022. https://imagemagick.org.

[58] Texas Instruments Inc. TI mmWave radar specification document, 2021.

[59] S. Vagharshakyan, R. Bregovic, O. Suominen, and A. Gotchev. Densely sampled light fields (Version 2), 2022. http://urn.fi/urn:nbn:fi:att:ed60be6d-9d15-4857-aa0d-a30acd16001e.

[60] S. Varakliotis, J. Ostermann, and V. Hardman. Coding of animated 3-d wireframe models for internet streaming applications. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'01)*, pages 61–61, Tokyo, Japan, August 2001.

[61] Velodyne LiDAR. Puck: compact, powerful, intelligent, 2021.

[62] R. Wang, J. Peethambaran, and D. Chen. Lidar point clouds to 3-d urban models: a review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(2):606–627, January 2018.

[63] X. Wen, T. Li, Z. Han, and Y.-S. Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1939–1948, 2020.

[64] C. Westphal. Challenges in networking to support augmented reality and virtual reality. In *Proc. of International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5, Silicon Valley, CA, January 2017.

[65] 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 Transactions on Multimedia*, pages 1–16, February 2022. Accepted to Appear.

[66] 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 ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'21)*, Istanbul, Turkey, September 2021.

[67] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. GRNet: Gridding residual network for dense point cloud completion. *arXiv preprint arXiv:2006.03761*, 2020.

[68] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, December 2021.

[69] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S.-h. Jung, and R. Bajscy. Teeve: The next generation architecture for tele-immersive environments. In *Proc. of IEEE International Symposium on Multimedia (ISM'05)*, pages 8–pp, Irvine, CA, December 2005.

[70] R. Zhang and S. Cao. Real-time human motion behavior detection via cnn using mmwave radar. *IEEE Sensors Letters*, 3(2):1–4, 2018.

[71] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[72] M. Ziegler, R. op het Veld, J. Keinert, and F. Zilly. Acquisition system for dense lightfield of large scenes. In *2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, Copenhagen, Denmark, June 2017.