RESGVC: RESIDUAL CODING FOR HIGH-FIDELITY GENERATIVE VIDEO COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative video compression (GVC) methods usually employ decoder-side diffusion models to denoise reconstructed pictures based on generative priors. Besides, most of these methods are applied in latent space rather than directly in pixel domain. In this paper, we propose **ResGVC**, a residual coding framework for GVC that guides denoising toward original pictures by transmitting residual signals. ResGVC first encodes the source pictures within a learned video codec and then decomposes the distortion between the codec reconstruction and the source pictures into a sequence of residual segments, where each residual segment is acquired at each denoising step. Residual segments are then compressed using a novel residual codec with a diffusion transformer as entropy model. Unlike prior methods, ResGVC introduces explicit source constraints throughout denoising rather than relying on decoder-side generation alone. The residual codec also enables more efficient context modeling capability based on the powerful generative priors of diffusion transformer. Moreover, operating directly in pixel space avoids the irreversible information loss caused by latent representation bottlenecks. Experimental results show that, compared with the previous state-of-the-art method, ResGVC achieves 63.7% and 83.8% BD-rate reductions in terms of LPIPS and PSNR, respectively, while also delivering superior visual quality.

1 INTRODUCTION

Traditional video coding standards (Wiegand et al., 2003; Sullivan et al., 2012; Bross et al., 2021) and recent neural video compression (NVC) methods (Li et al., 2021; 2024; Jia et al., 2025; Li et al., 2026c) have shown non-trivial improvements in rate-distortion performance. As the optimization prioritizes signal fidelity, most of these codecs may lose fine textures or produce blurry results at low bitrates, which deviates from human perception. This limitation has motivated generative video compression (GVC), where generative models are integrated into codecs to synthesize plausible details that are not explicitly transmitted. Recent GVC methods commonly employ decoder-side diffusion models (Rombach et al., 2022; Yang et al., 2025; Wan et al., 2025) to denoise reconstructed pictures and improve perceptual quality (Ma & Chen, 2025a; Xue et al., 2026; Mao et al., 2026; Li et al., 2026d). Moreover, most work operates in latent space rather than directly in pixel domain.

However, in existing GVC methods, the decoder-side denoising process is largely driven by generative priors without sufficient source constraints along the denoising path. As a result, the denoised signal may drift from the original signal during iterative denoising. Besides, because these methods are typically built on latent autoencoders, the bottleneck discards fine details that cannot be reliably recovered at the decoder (Kingma & Welling, 2013). These limitations suggest that faithful generative video compression requires explicit source constraints throughout denoising, and performs denoising directly in the pixel space without relying on autoencoders.

In this paper, we propose **ResGVC**, a **Res**idual coding framework for high-fidelity **G**enerative **V**ideo **C**ompression, which guides diffusion-based denoising toward the source video by transmitting residual signals. ResGVC first encodes the source video with a learned video codec and then reformulates the distortion between the codec reconstruction and the source video into a sequence of residual segments, where each residual segment is acquired at each denoising step. ResGVC selectively encodes and transmits fidelity-critical residual segments, so that source constraints are injected into the de-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: Visual comparisons on the *BQTerrace* sequence from the HEVC B dataset.

noising path. Besides, denoising is directly operated in pixel space further avoiding the irreversible information loss introduced by latent autoencoders.

For efficient residual coding, we propose a residual codec by extending a pixel-space video diffusion transformer from a generator to a conditional probability estimator. This model estimates the distribution of each residual segment, allowing fidelity-critical residual segments to be entropy-coded while predicting the remaining residual segments from the previously transmitted ones. Therefore, residual coding is reserved for information that must be faithfully transmitted to preserve reconstruction fidelity, wheres residual generation is used for content that can be well predicted by the generative model, thereby reducing bitrate. By jointly applying residual coding and residual generation along the denoising path, ResGVC achieves an improved perception-fidelity-rate trade-off.

To the best of our knowledge, this work is the first to explore residual coding in GVC and to repurpose a diffusion model for distribution modeling in both residual coding and residual generation. Experiments demonstrate that ResGVC achieves state-of-the-art compression performance across multiple metrics. Compared with the latest GVC method, ResGVC achieves 63.7% and 83.8% BD-rate reductions in terms of LPIPS and PSNR, respectively, while producing superior visual results.

Our contributions are summarized as follows:

- We propose ResGVC, a residual coding framework for GVC that guides denoising toward original pictures by transmitting residual signals. It decomposes the distortion between the codec reconstruction and the original pictures into residual segments over intermediate denoising states, thereby introducing explicit source constraints throughout denoising.
- We design a novel residual codec with a pixel-space video diffusion transformer as the entropy model. This codec is capable of effective context modeling for residual compression, enabling entropy coding of transmitted residual segments while preserving generative modeling capability for non-transmitted ones.
- Compared with existing methods, ResGVC achieves superior compression performance in terms of both perception quality and signal fidelity, while delivering better visual quality.

2 Related Work

2.1 Generative Video Compression

Generative video compression (GVC) incorporates generative models into codecs to improve perceptual quality beyond the signal-fidelity objective of traditional video coding and fidelity-oriented NVC. Early GVC methods (Yang et al., 2022; Mentzer et al., 2022) mainly introduced generative adversarial networks (GANs) and perceptual losses into neural codecs. Although these methods improved perceptual realism, they often suffered from training instability and visual artifacts. Other work explored generative reconstruction in latent or tokenized representation spaces. For example,

GLC-Video (Qi et al., 2025) performed compression in the latent space of a vector-quantized variational autoencoder, while ProGVC (Li et al., 2026b) proposed a progressive generative codec based on hierarchical multi-scale residual token maps. However, these methods remained constrained by the representational capacity of their latent or tokenized bottlenecks.

Recently, the emerging diffusion models provides strong generative priors for GVC due to their ability to synthesize realistic spatial and temporal details. Existing diffusion-based methods (Ma & Chen, 2025a; Mao et al., 2026; Li et al., 2026d;a; Ma & Chen, 2025b; 2026) typically adopt a latent compression pipeline, where the source video is encoded into compact latent representations and a diffusion model is employed in the decoder to remove compression distortions from decoded latents. DiffVC (Ma & Chen, 2025a) follows this paradigm by compressing videos in the latent domain and using an image diffusion model to enhance decoded latent frames. GNVC-VD (Mao et al., 2026) further introduces a video diffusion transformer to jointly enhance decoded latent representations. YODA (Li et al., 2026d) explores one-step diffusion-based video compression by embedding multiscale temporal reference features for both latent coding and latent denoising.

Despite their promising perceptual quality, these methods still primarily perform decoder-side denoising in latent space. Their denoising process is largely guided by generative priors, while explicit source constraints along the denoising path remain weak. Moreover, when built upon latent autoencoders, they inherit an irreversible reconstruction bottleneck that discards fine details. In contrast, our method introduces residual coding into pixel-space denoising, enabling explicit source constraints on the denoising process.

2.2 Diffusion Model with Flow Matching

Diffusion models have become a popular paradigm for visual generation because they progressively transform noisy samples into clean data using learned denoising dynamics (Ho et al., 2020; Song et al., 2021a;b). Recent work has developed diffusion architectures based on U-Net (Ronneberger et al., 2015) and Transformer backbones (Vaswani et al., 2017; Peebles & Xie, 2023; Wan et al., 2025) for high-quality image and video generation. To improve efficiency, many methods (Rombach et al., 2022; Ma et al., 2024) perform denoising in a compressed latent space which is produced by a variational autoencoder (VAE) (Kingma & Welling, 2013). While this design substantially reduces computation, it inherits the reconstruction bottleneck of the autoencoder. In contrast, pixel-space diffusion models directly denoise visual signals in the original signal space and can better preserve fine-grained details (Li & He, 2026; Wang et al., 2026; Yu et al., 2026; Ma et al., 2026).

Flow matching (Lipman et al., 2023; Liu et al., 2023) provides an efficient formulation for diffusion-based generative modeling by learning a continuous denoising path from a random noise sample to clean data. Specifically, the intermediate noisy state at denoising step t is commonly defined as

$$\mathbf{x}_t = (1 - \sigma_t)\mathbf{x}_0 + \sigma_t \mathbf{x}_T, \quad 0 \le t \le T, \tag{1}$$

where $\mathbf{x}_0$ denotes the clean data. $\mathbf{x}_T$ denotes the denoising start, which is usually a random noise sample. σ_t controls the noise level (Karras et al., 2022). The denoising network is trained to predict the corresponding velocity field by minimizing

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_T}\left[\|\hat{\boldsymbol{\epsilon}}(\mathbf{x}_t, \mathbf{c}, t) - (\mathbf{x}_T - \mathbf{x}_0)\|_2^2\right], \tag{2}$$

where $\mathbf{c}$ denotes the condition and $\hat{\boldsymbol{\epsilon}}(\mathbf{x}_t, \mathbf{c}, t)$ is the predicted velocity. During inference, the denoising network iteratively updates the intermediate noisy state along the predicted denoising path toward the clean data:

$$\hat{\mathbf{x}}_{t-1} = \hat{\mathbf{x}}_t + (\sigma_{t-1} - \sigma_t)\hat{\boldsymbol{\epsilon}}(\hat{\mathbf{x}}_t, \mathbf{c}, t), \tag{3}$$

where $\hat{\mathbf{x}}_t$ represents the predicted noisy state at denoising step t.

In GVC, flow-matching based diffusion models are mainly used as decoder-side denoiser to refine reconstructed latent representations with generative priors (Ma & Chen, 2025a; Mao et al., 2026; Li et al., 2026d;a; Ma & Chen, 2025b). Under this formulation, restoration can be viewed as estimating a denoising path from the codec reconstruction toward the source pictures. Existing methods primarily predict this path from generative priors, with limited source constraints along intermediate noisy states. This motivates our design: instead of treating the denoising path as purely generative, we introduce residual coding into it so that explicit source information can constrain the denoising process. Moreover, unlike latent-space diffusion used in most prior methods, our approach performs denoising in pixel space, avoiding the irreversible reconstruction bottleneck of latent autoencoders.

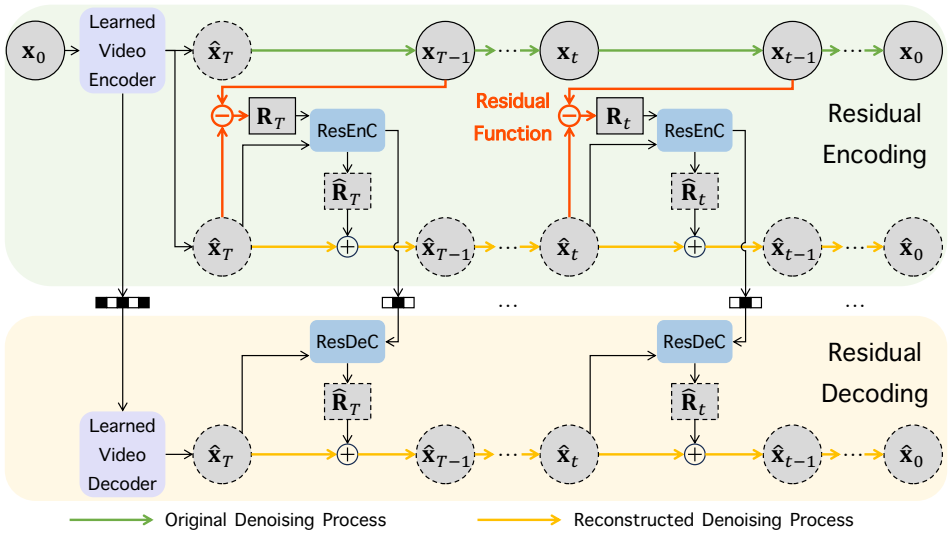


Figure 2: Overview of the proposed residual coding framework. $\mathbf{x}_0$ denotes the original pictures, and $\hat{\mathbf{x}}_T$ represents the reconstruction decoded by the learned video codec. $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$ denote the predefined and reconstructed noisy states at denoising step t, respectively. $\mathbf{R}_t$ denotes the to-be-encoded residual segment, and $\hat{\mathbf{R}}_t$ denotes its reconstruction produced by the residual codec. $\hat{\mathbf{x}}_0$ denotes the final reconstructed pictures after residual decoding.

3 METHODOLOGY

ResGVC is a residual coding framework for high-fidelity reconstruction refinement. It first compresses source pictures with a learned video codec and then refines the decoded pictures using residual information through iterative residual coding. Furthermore, the diffusion process is performed directly in pixel domain, avoiding the irreversible bottleneck of latent compression. Concretely, the distortion between the decoded pictures and the original signal is decomposed into a sequence of residual segments, each obtained at a denoising step. To encode these segments efficiently, we design a residual codec in which a video diffusion model serves as a conditional probability estimator for entropy coding. Section 3.1 formulates the residual segments from the codec reconstruction to the source signal. Section 3.2 presents the proposed residual codec and its diffusion-based entropy model. Section 3.3 describes the skip mode for residual coding, which skips residual coding at selected denoising steps and replaces it with decoder-side residual generation for reducing bitrate.

3.1 FROM DENOISING PATH TO RESIDUAL CODING

Fig. 2 presents the proposed framework, which introduces residual coding along a diffusion-based denoising path. Given the original pictures $\mathbf{x}_0$, $\hat{\mathbf{x}}_T$ denote the reconstruction decoded by the learned video codec. In diffusion-based GVC, denoising usually starts from $\hat{\mathbf{x}}_T$ rather than pure Gaussian noise (Mao et al., 2026). Following flow matching, the predefined noisy state at step t is written as

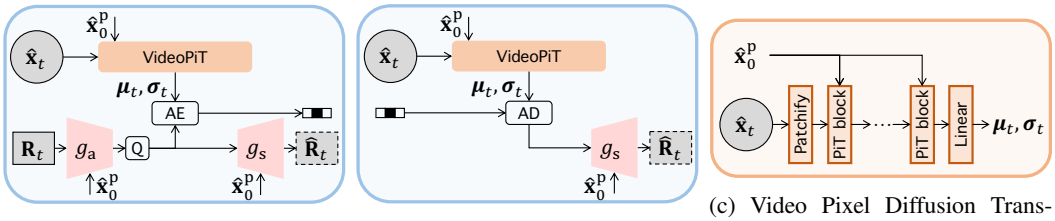
$$\mathbf{x}_t = (1 - \sigma_t)\mathbf{x}_0 + \sigma_t \hat{\mathbf{x}}_T, \tag{4}$$

where σ_t denotes the noise level of the intermediate noisy state at step t. The sequence $\{\mathbf{x}_t\}_{t=0}^{T-1}$ defines the *original denoising process* from the codec reconstruction $\hat{\mathbf{x}}_T$ to the original data $\mathbf{x}_0$. Accordingly, the *original denoising path* is formulated as $\hat{\mathbf{x}}_T \to \mathbf{x}_{T-1} \to \cdots \to \mathbf{x}_t \to \cdots \to \mathbf{x}_0$.

Prior diffusion-based methods recover this denoising path at the decoder by predicting the denoising update at each step. From step t to step $t-1$, the update can be written as

$$\hat{\mathbf{x}}_{t-1} = \hat{\mathbf{x}}_t + (\sigma_{t-1} - \sigma_t)\hat{\mathbf{R}}_t, \tag{5}$$

where $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{R}}_t$ denote the reconstructed noisy state and the predicted residual information at step t, respectively. $(\sigma_{t-1} - \sigma_t)\hat{\mathbf{R}}_t$ represents the correction applied to the current reconstructed noisy state to approach the next cleaner state. Since this correction is predicted solely from generative priors, the reconstructed denoising process may deviate from the original denoising process.



(a) Residual Encoder (ResEnC). (b) Residual Decoder (ResDeC). (c) Video Pixel Diffusion Transformer (VideoPiT).

Figure 3: Proposed residual encoder, residual decoder, and video pixel diffusion transformer.

In contrast, as illustrated in Fig. 2, we reformulate the same stepwise denoising update into residual coding which aims to faithfully approximate the original denoising path at the decoder based on the transmitted residual information. At step t, the to-be-encoded residual segment is defined as the scaled difference between the predefined noisy state $\mathbf{x}_{t-1}$ and the reconstructed noisy state $\hat{\mathbf{x}}_t$:

$$\mathbf{R}_t = (\mathbf{x}_{t-1} - \hat{\mathbf{x}}_t)/(\sigma_{t-1} - \sigma_t), \tag{6}$$

where the scaling factor $(\sigma_{t-1} - \sigma_t)$ standardizes the variance of $\mathbf{R}_t^i$, thereby aligning its distributions across steps to facilitate robust probability estimation in an entropy model. $(\sigma_{t-1} - \sigma_t)\mathbf{R}_t$ represents the correction that moves $\hat{\mathbf{x}}_t$ toward $\mathbf{x}_{t-1}$, *i.e.*, the next predefined state on the original denoising path. The sequence $\{\hat{\mathbf{x}}_t\}_{t=0}^{T-1}$ defines the *reconstructed denoising process*. Accordingly, the *reconstructed denoising path* is formulated as $\hat{\mathbf{x}}_T \to \hat{\mathbf{x}}_{T-1} \to \cdots \to \hat{\mathbf{x}}_t \to \cdots \to \hat{\mathbf{x}}_0$.

At the encoder, each $\mathbf{R}_t$ is compressed into a bitstream, and then reconstructed as $\hat{\mathbf{R}}_t$ that is used to simulate the decoder-side reconstructed denoising process. Starting from $\hat{\mathbf{x}}_T$, the encoder iteratively updates $\hat{\mathbf{x}}_t$ by applying Eq. 5, which yields the reconstructed denoising process $\{\hat{\mathbf{x}}_t\}_{t=0}^{T-1}$.

At the decoder, the reconstructed denoising process also starts from the same codec reconstruction $\hat{\mathbf{x}}_T$. The reconstructed residual segments $\{\hat{\mathbf{R}}_t\}_{t=1}^{T}$ are sequentially recovered from the transmitted bitstream, for which the reconstructed denoising process can be reproduced step by step, according to Eq. 5. In this way, our method turns denoising from a purely decoder-side generation process into a high-fidelity reconstruction process constrained by transmitted residual information.

3.2 Residual Coding with Diffusion-based Entropy Model

Fig. 3 shows the proposed residual codec which consists of a residual encoder (ResEnC) and a residual decoder (ResDeC), with a shared diffusion-based entropy model implemented by a video pixel diffusion transformer (VideoPiT). Generally, $\mathbf{x}_0$ can denote either a group of pictures (GOP) (Li et al., 2026c) or a single frame (Jia et al., 2025). In ResGVC, we define $\mathbf{x}_0$ as a GOP. The codec starts from the learned-codec reconstruction $\hat{\mathbf{x}}_T$ and progressively transmits the residual segments $\{\mathbf{R}_t\}_{t=1}^{T}$, for which the reconstructed denoising process is recovered step by step, ultimately yielding the final reconstruction $\hat{\mathbf{x}}_0$.

At denoising step t, the residual segment $\mathbf{R}_t$ is first transformed as below:

$$\mathbf{y}_t = g_{\mathrm{a}}(\mathbf{R}_t \mid \hat{\mathbf{x}}_0^{\mathrm{p}}), \tag{7}$$

where g_{a} denotes the analysis transform. $\hat{\mathbf{x}}_0^{\mathrm{p}}$ is the final reconstruction of the previous GOP which is used as temporal context. The transformed representation $\mathbf{y}_t$ is then quantized as

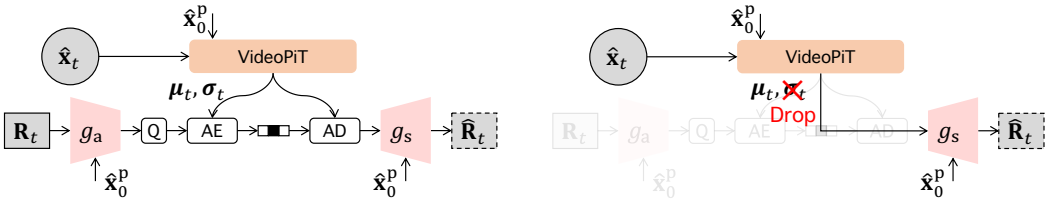
$$\hat{\mathbf{y}}_t = \lfloor \mathbf{y}_t \rceil. \tag{8}$$

To entropy-code $\hat{\mathbf{y}}_t$, we estimate its conditional probability distribution from the reconstructed noisy state $\hat{\mathbf{x}}_t$ in the reconstructed denoising process. Specifically, the diffusion-based entropy model takes the reconstructed noisy state $\hat{\mathbf{x}}_t$, and the final reconstruction of the previous GOP $\hat{\mathbf{x}}_0^{\mathrm{p}}$ as inputs, and predicts the mean and scale parameters (*i.e.*, $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t$) of a Gaussian probability distribution:

$$\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t = \mathrm{VideoPiT}(\hat{\mathbf{x}}_t \mid \hat{\mathbf{x}}_0^{\mathrm{p}}). \tag{9}$$

where $\hat{\mathbf{x}}_t$ is first patchified into tokens, and subsequently processed by a cascade of PiT blocks (Yu et al., 2026) with $\hat{\mathbf{x}}_0^{\mathrm{p}}$ as the condition. The resulting features are projected by a linear layer to generate $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ for entropy coding. More implementation details of the VideoPiT are provided in the appendix. $\hat{\mathbf{y}}_t$ is then arithmetic-encoded as

$$B_t = \mathrm{AE}(\hat{\mathbf{y}}_t; \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t), \tag{10}$$



(a) Residual Coding. (b) Residual Generation.

Figure 4: Residual coding and residual generation under the skip mode.

where B_t denotes the bitstream of the residual segment at step t. AE denotes the arithmetic encoding. During training, the quantization and arithmetic coding operations are replaced by differentiable approximations for end-to-end optimization, following the GVC practice (Mao et al., 2026).

At the decoder, each residual segment is reconstructed from the transmitted bitstream using the same conditional probability model. Since the decoder has access to the same conditioning information, namely $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{x}}_0^{\mathrm{p}}$, the Gaussian probability parameters in Eq. 9 can be reproduced exactly at the decoder side. The received bitstream is first arithmetic-decoded as

$$\hat{\mathbf{y}}_t = \mathrm{AD}(B_t; \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t), \tag{11}$$

where AD denotes the arithmetic decoding. $\hat{\mathbf{y}}_t$ is then mapped back to $\hat{\mathbf{R}}_t$:

$$\hat{\mathbf{R}}_t = g_{\mathrm{s}}(\hat{\mathbf{y}}_t \mid \hat{\mathbf{x}}_0^{\mathrm{p}}), \tag{12}$$

where g_{s} denotes the synthesis transform. Once $\hat{\mathbf{R}}_t$ is recovered, it is used to update the current reconstructed noisy state toward the cleaner one using Eq. 5. The updated state $\hat{\mathbf{x}}_{t-1}$ then serves as the context for the next denoising step. Therefore, starting from $\hat{\mathbf{x}}_T$, the decoder sequentially recovers all residual segments and reproduces the reconstructed denoising process to obtain $\hat{\mathbf{x}}_0$.

3.3 Skip Mode for Residual Coding

ResGVC supports a skip mode, in which residual coding is applied only to selected denoising steps. For the skipped steps, the corresponding residual segments are not transmitted but instead generated at the decoder by the diffusion model from the available reconstruction context, as illustrated in Fig. 4. Accordingly, the denoising process operates in two modes: *residual coding*, where the residual segment is encoded and transmitted by the residual codec, and *residual generation*, where the residual segment is inferred at the decoder without additional bitrate.

In our implementation, residual coding is used in the early denoising steps, while the later steps are skipped and handled by residual generation. For steps assigned to residual coding, $\hat{\mathbf{R}}_t$ is reconstructed from the transmitted bitstream and provides explicit source constraints for the denoising trajectory. For skipped steps, $\hat{\mathbf{R}}_t$ is predicted by the residual generator from previously decoded residual context, without incurring extra bits. Therefore, given a total of T denoising steps, the number of coded steps determines the trade-off between bitrate and reconstruction quality, and its effect is further analyzed in Section 4.3.

Although obtained differently, transmitted and generated residual segments are incorporated through the same state update rule in Eq. 5. Thus, transmitted source information and generative priors are jointly integrated throughout denoising to guide reconstruction toward the source video.

4 Experiment

4.1 Experimental Protocol

Training details. ResGVC is trained on the OpenVid-1M dataset (Nan et al., 2025) with a rate-distortion loss function, *i.e.*, $L = R + \lambda D$. R denotes the bitrate from the learned video codec and the proposed residual codec. D denotes the reconstruction distortion, computed as the average of mean squared error (MSE), LPIPS (Zhang et al., 2018), and DISTS (Ding et al., 2020). To evaluate compression performance at multiple bitrates, we train four independent models with trade-off parameter λ sampled at four points by linear interpolation in log space over the range [0.025,

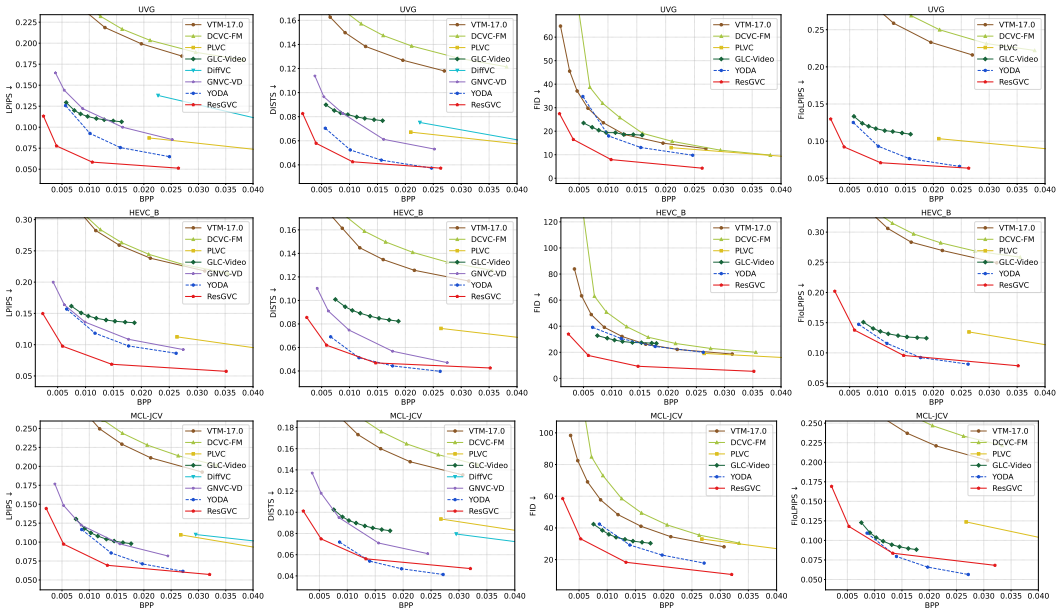


Figure 5: Rate-perception curves on UVG, HEVC B, and MCL-JCV.

Table 1: BD-rate (%) ↓ / BD-metric ↑ on UVG, HEVC B, and MCL-JCV. The best and second-best results are highlighted in red and blue, respectively. "N/A" indicates that BD-rate cannot be computed due to insufficient overlap between the rate–metric curves. "-" denotes that the baseline does not report the result.

	UVG				HEVC B				MCL-JCV			
Method	LPIPS	DISTS	FID	FloLPIPS	LPIPS	DISTS	FID	FloLPIPS	LPIPS	DISTS	FID	FloLPIPS
VTM-17.0	N/A / -0.136	N/A / -0.092	21.5 / -3.0	N/A / -0.177	N/A / -0.163	N/A / -0.093	10.1 / -2.1	N/A / -0.192	N/A / -0.150	N/A / -0.109	89.6 / -14.3	N/A / -0.163
DCVC-FM	N/A / -0.145	N/A / -0.108	66.9 / -8.8	N/A / -0.203	N/A / -0.168	N/A / -0.108	52.1 / -11.1	N/A / -0.203	N/A / -0.165	N/A / -0.126	143.5 / -24.3	N/A / -0.188
PLVC	116.7 / -0.019	369.8 / -0.027	33.2 / -2.2	255.5 / -0.034	91.3 / N/A	594.2 / N/A	N/A / N/A	200.5 / N/A	208.1 / -0.048	658.7 / -0.052	129.7 / -14.9	319.4 / -0.067
DiffVC	425.5 / -0.070	395.8 / -0.036	- / -	- / -	- / -	- / -	- / -	- / -	275.1 / N/A	382.3 / N/A	- / -	- / -
GLC-Video	35.4 / -0.018	N/A / -0.027	-4.2 / 0.4	56.2 / -0.022	44.2 / -0.025	N/A / -0.038	-15.2 / 1.6	33.1 / -0.019	24.5 / -0.013	N/A / -0.030	0.1 / -1.3	19.8 / -0.011
GNVC-VD	75.4 / -0.023	126.4 / -0.022	- / -	- / -	18.4 / -0.008	89.0 / -0.015	- / -	- / -	36.1 / -0.015	97.0 / -0.020	- / -	- / -
YODA	0.0 / 0.000	0.0 / 0.000	0.0 / 0.0	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000	0.0 / 0.0	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000	0.0 / 0.0	0.0 / 0.000
ResGVC	-74.9 / 0.031	-46.7 / 0.008	-68.4 / 10.2	-61.0 / 0.019	-68.1 / 0.042	-14.9 / 0.002	-78.9 / 18.6	-18.4 / 0.008	-48.3 / 0.016	-1.8 / 0.001	-54.4 / 11.2	0.9 / -0.003

2.5]. Each model is trained with a GOP size of 8. T is set to 5, indicating that five denoising steps are performed under the noise scheduler adopted in (Wan et al., 2025) to determine σ_t. The detailed training strategies are provided in the appendix.

Test details. We evaluate ResGVC on the HEVC B (Sullivan et al., 2012), UVG (Mercat et al., 2020), and MCL-JCV (Wang et al., 2016) datasets. Following (Li et al., 2023; Mao et al., 2026), raw YUV420 videos are converted to RGB using BT.709 (Li et al., 2023), and the first 96 frames of each sequence are used for evaluation. Compression performance is measured by BD-rate and BD-metric (Bjontegaard, 2001). Signal fidelity is measured by PSNR and MS-SSIM (Wang et al., 2003). Perceptual quality is assessed by LPIPS (Zhang et al., 2018) and DISTS (Ding et al., 2020). To assess temporal perceptual consistency, we report FloLPIPS (Danier et al., 2022), a flow-guided LPIPS variant. Generative realism is measured by FID (Heusel et al., 2017), following (Mentzer et al., 2020). Bitrate is reported in bits per pixel (BPP). ResGVC is evaluated with intra-period = -1 and GOP size = 8. Since ResGVC is trained with five denoising steps, we use two steps for residual coding and three steps for residual generation during testing, which gives the best overall compression performance. Analysis of this setting is provided in Section 4.3.

Baselines. We compare ResGVC with three categories of baselines: 1) the traditional video codec VTM-17.0 (Bross et al., 2021); 2) the fidelity-oriented NVC method DCVC-FM (Li et al., 2024); and 3) GVC methods, including PLVC (Yang et al., 2022), GLC-Video (Qi et al., 2025), DiffVC (Ma & Chen, 2025a), GNVC-VD (Mao et al., 2026), and YODA (Li et al., 2026d). For VTM-17.0, we follow the evaluation protocol as in (Li et al., 2023). Since the code for DiffVC and GNVC-VD is publicly unavailable, we report the results from their original papers. Other baselines are tested with their default settings. We use the latest open-source method YODA as the anchor for BD-rate computation, rather than VTM-17.0 as usual, because the RD curves of VTM-17.0 are often far from those of GVC methods and overlap only marginally in quality range.

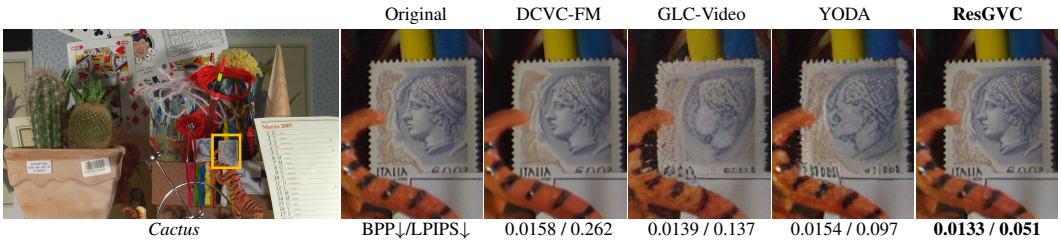


Figure 7: Visual comparisons with open-source baselines on the *Cactus* sequence from HEVC B.

4.2 Comparison Results

Quantitative comparisons. Fig. 5 shows the rate-perception curves on the three test sets. The proposed ResGVC outperforms all baselines on most perceptual metrics, achieving notably lower LPIPS and FID at comparable bitrates. Table 1 summarizes the BD-rate and BD-metric results. Averaged over the three test sets, ResGVC achieves bitrate savings of 63.7% in LPIPS, 21.2% in DISTS, 67.2% in FID, and 26.1% in FloLPIPS relative to the second-best method YODA. Additionally, as shown in Fig. 6, ResGVC delivers the best PSNR and MS-SSIM performance among all GVC methods, and is even comparable to the fidelity-oriented NVC method DCVC-FM at low bitrates. Notably, on the HEVC B test set, ResGVC surpasses DCVC-FM in MS-SSIM when the bitrate is below 0.005 BPP. Detailed BD-rate and BD-metric results for PSNR and MS-SSIM are provided in the appendix. Overall, these results show that ResGVC advances both perceptual quality and signal fidelity over previous GVC methods.

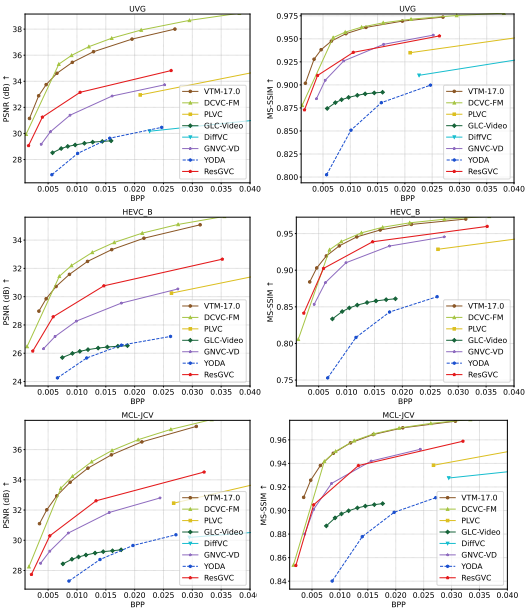


Figure 6: Rate-fidelity curves on UVG, HEVC B, and MCL-JCV.

Qualitative comparisons. Fig. 7 presents a visual comparison of different methods. Owing to its optimization objective that prioritizes signal fidelity, DCVC-FM tends to produce blurred regions and fails to preserve fine-grained textures and structural details. Although GVC methods such as GLC-Video and YODA achieve higher perceptual quality than DCVC-FM, they often introduce spatial inconsistencies and hallucinated details, reflecting an overemphasis on perceptual quality at the expense of signal fidelity. By contrast, ResGVC more faithfully preserves object structures while maintaining rich textural details. These observations further suggest that, for GVC methods, signal fidelity should not be neglected when optimizing perceptual quality; otherwise, noticeable visual degradation and structural collapse may occur.

Complexity comparisons. We compare the per-frame encoding and decoding times of ResGVC with those of two recent methods, *i.e.*, GNVC-VD and YODA. Among the GVC baselines, GNVC-VD delivers superior compression performance on fidelity-oriented metrics, whereas YODA excels on perceptual metrics. Since the official implementation of GNVC-VD is unavailable, we reproduce its model architecture solely for complexity evaluation, without training. As presented in Table 2, ResGVC is faster than GNVC-VD in both encoding and decoding, mainly because GNVC-VD has a substantially larger model size. Compared with YODA, which uses single-step denoising, ResGVC has a slightly slower decoding speed due to its five-step denoising design. However, the compact model size of ResGVC helps offset the overhead introduced by the additional denoising steps, allowing it to maintain competitive decoding speed. Despite using fewer parameters, ResGVC still outperforms YODA in both perceptual quality and signal fidelity, further demonstrating the effectiveness of our pixel-space residual coding framework.

Table 2: Complexity comparison. "Enc. T" / "Dec. T" denotes the per-frame encoding / decoding time, tested on an NVIDIA H20 GPU.

Method	Model Params (M)	Enc. T (ms)	Dec. T (ms)
GNVC-VD	2334.5	439	4225
YODA	1063.2	1139	729
Ours	593.9	346	772

Table 3: BD-rate (%)↓ comparisons across various residual coding (RC) settings. The model with five-step RC in pixel space serves as the anchor.

Method	LPIPS	DISTS	PSNR	FloLPIPS
w/ RC in pixel space	0.0	0.0	0.0	0.0
w/o RC in pixel space	44.6	59.1	12.1	34.2
w/ RC in latent space	17.4	21.3	28.6	15.1
w/ CNN-based entropy model	13.4	18.9	7.4	16.1

4.3 Ablations

We conduct a series of ablation studies to verify the effect of key designs in the proposed ResGVC. All BD-rate results are averaged on three test sets.

Effect of pixel-space residual coding. ResGVC performs residual coding directly in pixel space. To validate this design, we conduct the ablations in Table 3, using the variant with five-step pixel-space residual coding as the anchor. Removing residual coding causes a substantial performance degradation, which confirms the necessity of residual coding. We then shift residual coding from pixel space to latent space by inserting ResGVC between the VAE encoder and decoder of Wan2.1 (Wan et al., 2025), whose implementation details are provided in the appendix. This variant incurs 17.4% and 28.6% BD-rate losses in terms of LPIPS and PSNR, respectively, demonstrating that residual coding is more effective in pixel space than in latent space.

Effect of diffusion-based entropy model. Existing fidelity-oriented NVC and GVC methods typically rely on CNN-based entropy models to estimate the distributions used for entropy coding. In contrast, ResGVC adopts a diffusion-based entropy model, which leverages stronger generative priors for more accurate conditional probability estimation. As shown in Table 3, replacing our entropy model with the latest CNN-based design (Li et al., 2026c) degrades performance across all metrics, confirming the advantage of diffusion-based entropy modeling for residual compression.

Effect of the skip mode for residual coding. In ResGVC, residual coding can be skipped at selected denoising steps, where the skipped residual segments are instead generated at the decoder by the video pixel diffusion model. This skip mode enables a flexible allocation of denoising steps between transmitted residuals and decoder-side generation to optimize rate-distortion performance. As shown in Table 4, we fix the total number of denoising steps to five and vary the number of residual-coding steps from zero to five, with the remaining steps handled by residual generation. The best performance is achieved with two coding steps and three skipped steps, indicating that the most fidelity-critical residual segments are transmitted in the first two steps. Using more coding steps degrades performance by spending additional bitrate on residuals that can be reliably predicted from generative priors at the decoder, while using fewer coding steps weakens source constraints and thus reduces reconstruction fidelity.

Table 4: BD-rate (%)↓ comparison under different skip settings for residual coding. RC and RG denote residual coding and generation, respectively.

Method	LPIPS	DISTS	PSNR	FloLPIPS
0-step RC + 5-step RG	11.4	11.6	9.2	8.8
1-step RC + 4-step RG	7.7	8.2	7.1	6.5
2-step RC + 3-step RG (Ours)	0.0	0.0	0.0	0.0
3-step RC + 2-step RG	2.4	4.5	1.8	2.7
4-step RC + 1-step RG	5.4	5.8	3.5	3.6
5-step RC + 0-step RG	6.7	7.3	3.4	4.2

5 Conclusion

In this paper, we present ResGVC, a residual coding framework for generative video compression that guides denoising toward the source pictures by transmitting residual signals. ResGVC first compresses the source pictures with a learned video codec and then decomposes the distortion between the codec reconstruction and the source pictures into a sequence of residual segments over denoising steps. These residual segments are compressed by a novel residual codec with a diffusion transformer as the entropy model, enabling explicit source constraints to be introduced throughout denoising rather than relying on decoder-side generation alone. By operating directly in pixel space, ResGVC also avoids the irreversible information loss caused by latent representation bottlenecks. Experimental results demonstrate that ResGVC achieves state-of-the-art compression performance, reducing BD-rate by 63.7% in LPIPS and 83.8% in PSNR compared with the previous state-of-the-art method, while also delivering superior visual quality.

AI USE STATEMENT

In this work, we used GPT-5.4 to improve the clarity, grammar, and formal tone of the writing in all sections, and to retrieve and translate related work in support of a more comprehensive understanding of this research field. We did not use generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design or provide feedback on research methodology or experiments, implement methods, clean or reformat datasets, support qualitative and thematic data analysis, or interpret results. The generation of synthetic datasets is not applicable to this work. All AI-generated suggestions were reviewed and edited by the authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

Duolikun Danier, Fan Zhang, and David Bull. Flolpips: A bespoke video quality metric for frame interpolation. In *Picture Coding Symposium*, pp. 283–287. IEEE, 2022.

Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, 44(5):2567–2581, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12543–12552, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Daowen Li, Ding Ding, Zifu Zhang, Kai Li, and Ying Chen. Generative video compression based on hierarchical referencing. *arXiv preprint arXiv:2608.11618*, 2026a.

Daowen Li, Ruixiao Dong, Ying Chen, Kai Li, Ding Ding, and Li Li. Progvc: Progressive-based generative video compression via auto-regressive context modeling. *arXiv preprint arXiv:2603.17546*, 2026b.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22616–22626, 2023.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with feature modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26099–26108, 2024.

Jiahao Li, Wenxuan Xie, Zhaoyang Jia, Bin Li, Zongyu Guo, Xiaoyi Zhang, and Yan Lu. Ultra-fast neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41311–41321, 2026c.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Xingchen Li, Junzhe Zhang, Junqi Shi, Ming Lu, and Zhan Ma. Yoda: Yet another one-step diffusion-based video compressor. *IEEE Transactions on Circuits and Systems for Video Technology*, 2026d.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Wenzhuo Ma and Zhenzhong Chen. Diffusion-based perceptual neural video compression with temporal diffusion information reuse. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(12):1–22, 2025a.

Wenzhuo Ma and Zhenzhong Chen. Diffvc-osd: One-step diffusion-based perceptual neural video compression framework. In *International Conference on Visual Communications and Image Processing*, pp. 1–5. IEEE, 2025b.

Wenzhuo Ma and Zhenzhong Chen. Diffvc-rt: Towards practical real-time diffusion-based perceptual neural video compression. *arXiv preprint arXiv:2601.20564*, 2026.

Zehong Ma, Ruihan Xu, and Shiliang Zhang. PixelGen: Improving pixel diffusion with perceptual supervision. *arXiv preprint arXiv:2602.02493*, 2026.

Qi Mao, Hao Cheng, Tinghan Yang, Libiao Jin, and Siwei Ma. Generative neural video compression via video diffusion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43239–43248, 2026.

Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in neural information processing systems*, 33:11913–11924, 2020.

Fabian Mentzer, Eirikur Agustsson, Johannes Ballé, David Minnen, Nick Johnston, and George Toderici. Neural video compression using gans for detail synthesis and propagation. In *European Conference on Computer Vision*, pp. 562–578. Springer, 2022.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM multimedia systems conference*, pp. 297–302, 2020.

Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *International conference on learning representations*, volume 2025, pp. 1045–1064, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Linfeng Qi, Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Generative latent coding for ultra-low bitrate image and video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(10):10500–10515, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10674–10685, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *IEEE international conference on image processing*, pp. 1509–1513. IEEE, 2016.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 43559–43580, 2026.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar conference on signals, systems & computers*, volume 2, pp. 1398–1402, 2003.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

Naifu Xue, Zhaoyang Jia, Jiahao Li, Bin Li, Zihan Zheng, Yuan Zhang, and Yan Lu. Single-step diffusion-based video coding with semantic-temporal guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9752–9761, 2026.

Ren Yang, Radu Timofte, and Luc Van Gool. Perceptual learned video compression with recurrent conditional gan. In *IJCAI*, pp. 1537–1544, 2022.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14273–14282, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

A APPENDIX

A.1 MODEL IMPLEMENTATIONS

Analysis Transform and Synthesis Transform. The proposed residual codec employs analysis transform g_{a} and synthesis transform g_{s} to reduce the redundancy within each residual segment. g_{a} and g_{s} are built upon the encoder and decoder architecture of the inter model in DCVC-UF HT-L (Li et al., 2026c), with two principal distinctions. First, our g_{a} and g_{s} are shared by both intra-coding and inter-coding, with an adapter introduced for the intra-frame to align the input channels of g_{a}. By contrast, DCVC-UF employs two separate models for intra- and inter-coding. Second, we employ the previously reconstructed GOP $\hat{\mathbf{x}}_0^{\mathrm{p}}$ as contextual information, whereas DCVC-UF relies on memory updated using features from decoded frames.

Diffusion-based Entropy Model. In our framework, the video pixel diffusion transformer (VideoPiT) serves as an entropy model, in which each PiT block is built upon the T2I variant of PixelDiT (Yu et al., 2026). The distinction is that we use the previously reconstructed GOP, $\hat{\mathbf{x}}_0^{\mathrm{p}}$, as the conditioning input, whereas PixelDiT conditions on features derived from the text embedding. In VideoPiT, the input $\hat{\mathbf{x}}_t$ first undergoes 16×16 patchification, followed by six PiT blocks. The output feature of the final PiT block is then linearly projected to produce the probability parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ for entropy coding. Notably, both $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ have spatial dimensions downsampled by a factor of 16 relative to $\hat{\mathbf{x}}_t$.

A.2 MORE TRAINING DETAILS

We initialize the learned video codec (Li et al., 2026c) with its the pretrained weight, and the proposed residual codec is trained from scratch. AdamW (Loshchilov & Hutter, 2019) is used as the optimizer. For the training set OpenVid-1M (Nan et al., 2025), we resize and crop each train sample to 448×448. Adopting the noise schedule from Wan2.1 (Wan et al., 2025), the noise levels σ_t from $t = 5$ down to $t = 0$ are configured as [0.83, 0.77, 0.68, 0.56, 0.36, 0.0], where the initial magnitude of 0.83 reflects our assumption that the decoded video from the learned video codec exhibits a noise (distortion) ratio of 0.83. The loss function is formulated as:

$$L = R_{\mathrm{LVC}} + R_{\mathrm{ResC}} + \lambda(\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_2^2 + L_{\mathrm{LPIPS}}(\mathbf{x}_0, \hat{\mathbf{x}}_0) + L_{\mathrm{DISTS}}(\mathbf{x}_0, \hat{\mathbf{x}}_0)), \tag{13}$$

where R_{LVC} denotes the bitrate cost from the learned video codec and R_{ResC} denote the bitrate cost from the proposed residual codec. L_{LPIPS} and L_{DISTS} represent the perceptual losses measured by LPIPS (Zhang et al., 2018) and DISTS (Ding et al., 2020). $\mathbf{x}_0$ and $\hat{\mathbf{x}}_0$ denote the original video and its reconstruction.

To stabilize model optimization, we adopt a multi-stage training strategy as follows. All training is conducted on eight NVIDIA H20 GPUs.

Stage 1. We train only the residual codec with single-step denoising, where $\sigma_1 = 0.83$ and $\sigma_0 = 0$. The trade-off parameter λ is set to 8. The number of training frames is progressively increased from 2 to 33, while the batch size is reduced from 256 to 16 accordingly. The learning rate is set to 5×10^{-5}.

Stage 2. We jointly train the learned video codec and the proposed residual codec. λ is sampled at four points using linear interpolation in logarithmic space over the range [0.025, 2.5], resulting in four independent models for evaluating compression performance at different bitrates. The number of training frames is set to 33, the batch size to 8, and the learning rate to 5×10^{-5}.

Stage 3. We progressively scale the number of denoising steps, *i.e.*, T, from 1 to 5. Standard flow matching typically samples a random step t during each training iteration and optimizes the objective in Eq. 2 under a single-step denoising paradigm, akin to the setting adopted in Stage 1. However, given our multi-step residual coding design, compression distortions from the decoded residuals propagate across successive steps. Consequently, we employ multi-step denoising during training to adapt the model to such error propagation. To ensure stable optimization, we only backpropagate the gradient at step 0. The number of training frames is set to 33, the batch size to 8, and the learning rate to 2×10^{-5}.

Stage 4. We increase the number of training frames to 97. All other training configurations remain the same as in Stage 3.

Table 5: BD-rate (%) ↓ / BD-metric ↑ for PSNR and MS-SSIM on UVG, HEVC B, and MCL-JCV, using YODA as the anchor. The best and second-best results are highlighted in red and blue, respectively. "N/A" indicates that BD-rate cannot be computed due to insufficient overlap between the rate–metric curves. "-" denotes that the baseline does not report the result.

	UVG		HEVC B		MCL-JCV	
Method	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
VTM-17.0	N/A / 7.221	N/A / 0.101	N/A / 6.886	N/A / 0.132	N/A / 6.582	N/A / 0.082
DCVC-FM	-94.0 / 7.766	-93.1 / 0.103	-92.7 / 7.334	-89.5 / 0.135	-87.3 / 6.834	-84.9 / 0.083
PLVC	N/A / 2.848	N/A / 0.040	N/A / N/A	N/A / N/A	N/A / 2.134	N/A / 0.028
DiffVC	9.5 / -0.158	N/A / 0.014	- / -	- / -	20.9 / N/A	N/A / N/A
GLC-Video	-29.1 / 0.760	-48.8 / 0.040	-24.7 / 0.530	-44.9 / 0.042	-28.2 / 0.710	-45.4 / 0.032
GNVC-VD	-72.3 / 3.265	-79.0 / 0.075	-74.2 / 3.020	-81.0 / 0.105	-69.5 / 2.876	-74.7 / 0.061
YODA	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000	0.0 / 0.000
ResGVC	-87.6 / 4.535	-88.1 / 0.079	-85.1 / 4.625	-87.4 / 0.118	-78.9 / 3.977	-77.8 / 0.060

Table 6: BD-rate (%)↓ comparison of performing residual coding (RC) and residual generation (RG) in the latent space *vs.* the pixel space.

	Method	LPIPS	DISTS	PSNR	FloLPIPS
(a)	5-step RC in latent space	15.4	17.5	30.8	10.3
(b)	5-step RC in pixel space	6.7	7.3	3.4	4.2
(c)	5-step RG in latent space	7.4	8.1	22.6	4.6
(d)	5-step RG in pixel space	11.4	11.6	9.2	8.8
(e)	2-step RC and 3-step RG in latent space	13.2	13.1	24.7	5.6
(f)	**2-step RC and 3-step RG in pixel space (Ours)**	**0.0**	**0.0**	**0.0**	**0.0**

Stage 5. During multi-step residual coding, we allow gradients to backpropagate across steps. The number of training frames is set to 97, the batch size to 4, and the learning rate to 1×10^{-5}.

A.3 Further Comparison Results

We report the BD-rate and BD-metric results for fidelity-oriented measures, *i.e.*, PSNR and MS-SSIM, in Table 5. Among all methods, the fidelity-oriented NVC approach DCVC-FM achieves the best performance, while the traditional video codec VTM-17.0 delivers the second-best results. This is expected, as both are optimized to prioritize signal fidelity rather than perceptual quality. Among the GVC methods, the proposed ResGVC demonstrates clear superiority over the other baselines, achieving approximately 10% bitrate savings on each test set compared with the second-best method GNVC-VD. Notably, the performance gap between DCVC-FM and ResGVC is relatively small, with the BD-rate difference remaining below 10% across all three test sets.

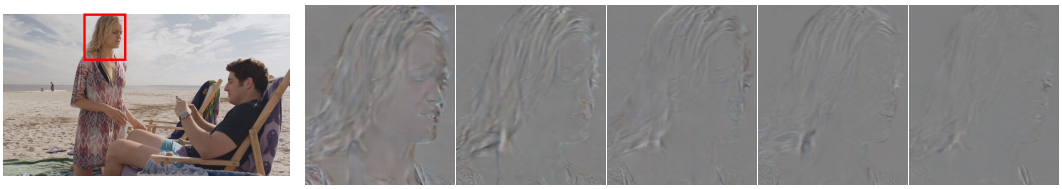
A.4 Additional Analysis

A.4.1 Employing our framework in pixel space *vs.* latent space

The proposed ResGVC is employed in pixel space rather than in latent space produced by autoencoders commonly adopted in previous GVC methods. In Section 4.3, we conduct an ablation study in which residual coding is instead performed in latent space by inserting ResGVC between the VAE encoder and decoder. Here, we provide the implementation details.

Following (Mao et al., 2026), we adopt the VAE from Wan2.1 (Wan et al., 2025) as the autoencoder, which performs $8\times$ spatial downsampling for all frames and $4\times$ temporal downsampling for inter frames only. As described in Section 3.2, g_{a} transforms the to-be-encoded $\hat{\mathbf{R}}_t$ into $\hat{\mathbf{y}}_t$ for subsequent entropy coding. The spatial resolution of $\hat{\mathbf{y}}_t$ is reduced by a factor of 16 relative to $\hat{\mathbf{R}}_t$ through an 8×8 patchification and a convolutional layer with $2\times$ downsampling within g_{a}. To shift ResGVC to latent space, we encode two latent frames into $\hat{\mathbf{y}}_t$, corresponding to eight frames as a GOP in pixel space, and remove the patchification to ensure that $\hat{\mathbf{y}}_t$ retains the same dimensions as in pixel-space residual coding. Besides, 16×16 patchification in the video PiT is replaced with 2×2 patchification.

We provide additional experimental results in Table 6 to evaluate the effectiveness of performing residual coding and residual generation in latent space *vs.* pixel space. A comparison between line



Original	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5
BPP↓	0.0188	0.0216 (+0.0028)	0.0227 (+0.0011)	0.0230 (+0.0003)	0.0231 (+0.0001)	0.0232 (+0.0001)
LPIPS↓	0.0725	0.0453 (-0.0272)	0.0394 (-0.0059)	0.0369 (-0.0025)	0.0354 (-0.0015)	0.0340 (-0.0014)

Figure 8: Visual comparisons of the reconstructed noisy states during five-step residual coding for the *videoSRC23* sequence from the MCL-JCV dataset. Wherein, residual maps depicted above are derived by subtracting the reconstruction of the preceding step from that of the current step.

Table 7: Distribution of bitrate cost across different residual coding steps. Note that the percentage in "Step 0" denotes the proportion of the total bitrate consumed by the learned video codec.

Step 0	Step 1	Step 2	Step 3	Step 4	Step 5
82.9%	10.1%	4.7%	1.8%	0.3%	0.2%

(a) and line (b) shows that when five-step residual coding is performed in latent space, compression performance declines, particularly in terms of BD-rate *w.r.t.* PSNR. By contrast, comparing line (c) with line (d), the latent-space variant outperforms the pixel-space counterpart on perceptual metrics, *i.e.*, LPIPS, DISTS, and FloLPIPS. This may indicate that under a fixed model capacity, compact residual latent representations are more amenable to generation than pixel-space signals. Comparing line (a) with line (c), both of which operate in latent space, residual generation proves considerably more tractable than residual coding. This might be because compact latent representations are more susceptible to quantization, while quantization errors in latent space may be magnified in pixel space by the VAE decoder. Overall, as evidenced by the comparison between line (e) and line (f), the pixel-space model surpasses the latent-space model under our default setting of two-step coding and three-step generation.

A.4.2 Distribution of Bitrate Cost across Different Residual Coding Steps

Table 7 presents the distribution of bitrate cost across different residual coding steps. The learned video codec accounts for the largest portion of the total bitrate. During residual coding, the combined bitrate contribution of Step 3, Step 4, and Step 5 is 2.3% of the total bitrate. Notably, as shown in Table 4, taking our default setting (two residual-coding steps and three residual-generation steps) as the anchor, using five residual-coding steps leads to more than a 3% BD-rate increase under all metrics. This discrepancy may stem from the fact that residual coding in the later steps introduces quantization errors that degrade reconstruction quality, while the residual generated by VideoPiT are already sufficiently accurate at these denoising steps.

Furthermore, we provide a visual example produced with five-step residual coding to illustrate the distribution of bitrate cost across different steps and the corresponding variation in perceptual quality. As shown in Fig. 8, the first two residual-coding steps consume the most bitrate and yield significant perceptual quality improvements. At step 3, step 4, and step 5, perceptual quality continues to improve through residual generation using predicted residual segments.

A.5 More Visual Examples

We present additional qualitative comparisons on three datasets: HEVC Class B, MCL-JCV, and UVG. As illustrated in Fig. 9 - Fig. 11, our method yields visually sharper and more realistic reconstructions than the competing baselines. Fig. 12 presents a comparison of temporal consistency

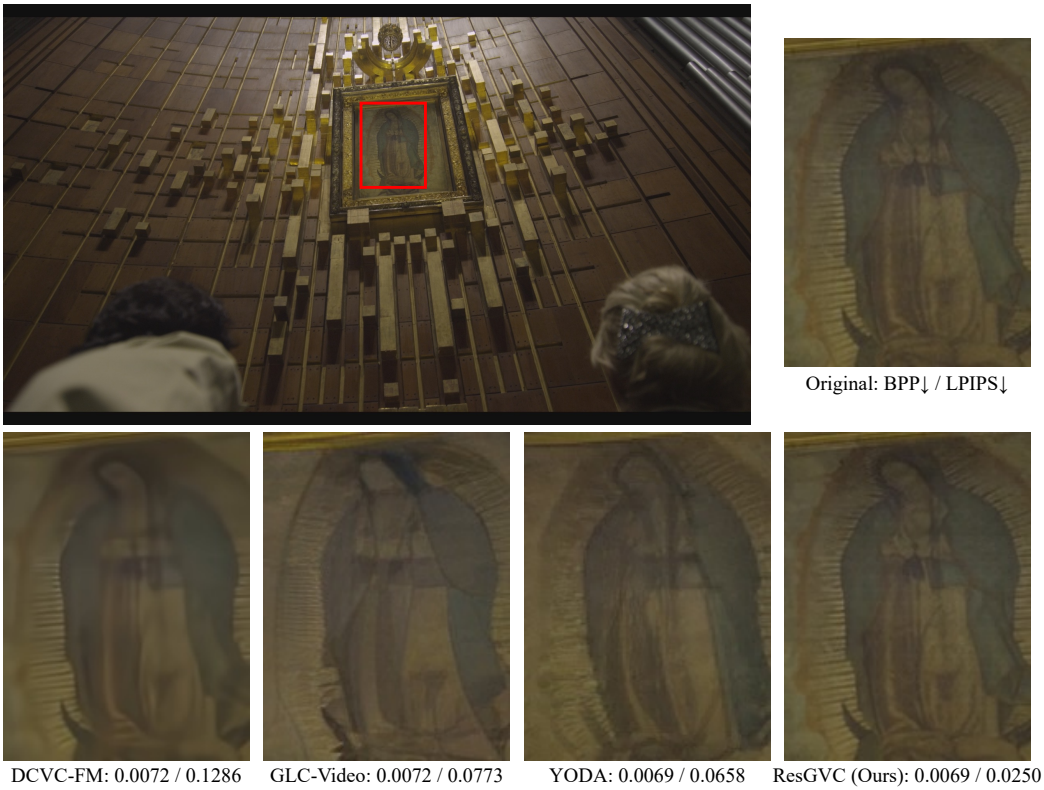


Figure 9: Visual comparisons with open-source baselines on the *videoSRC03* sequence from the MCL-JCV dataset.

across different methods. DCVC-FM generally produces blurry reconstructions, while GLC-Video and YODA tend to introduce jitter and flickering artifacts, thereby compromising temporal coherence. By contrast, our method delivers temporally stable reconstructions, especially in regions containing dense textures. These examples further underscore that our approach excels not only on quantitative metrics but also in preserving fine textures, realistic details, and perceptual naturalness, thereby delivering superior visual quality.

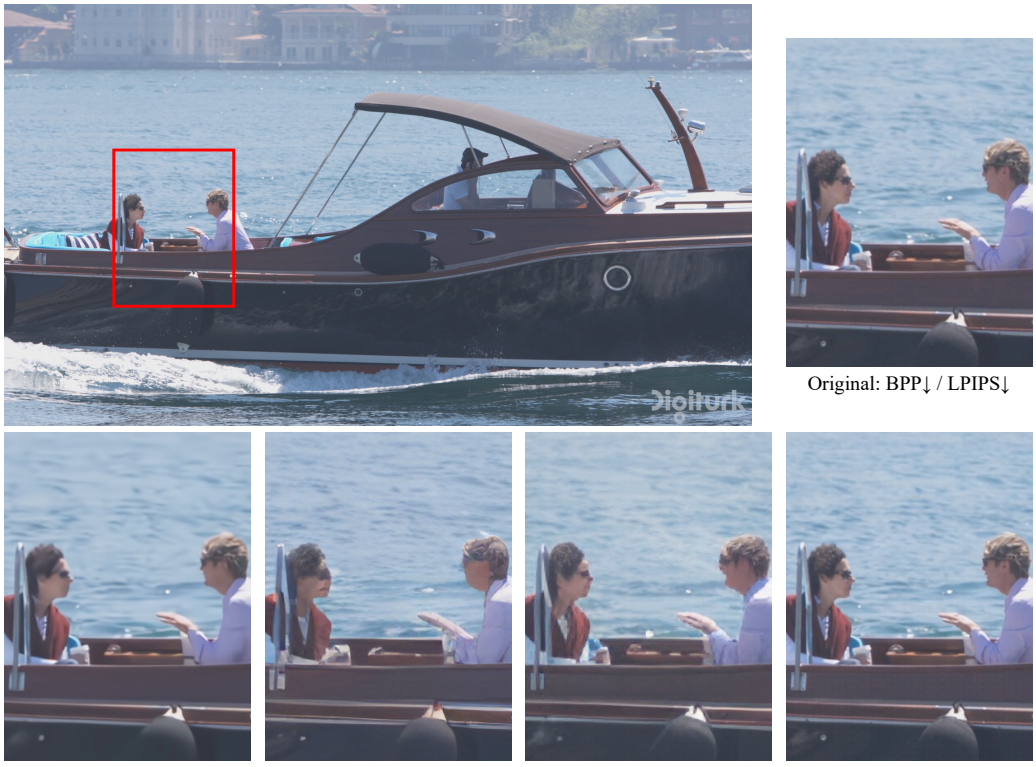


Figure 10: Visual comparisons with open-source baselines on the *YachtRide* sequence from the UVG dataset.



Figure 11: Visual comparisons with open-source baselines on the *Kimono* sequence from the HEVC B dataset.

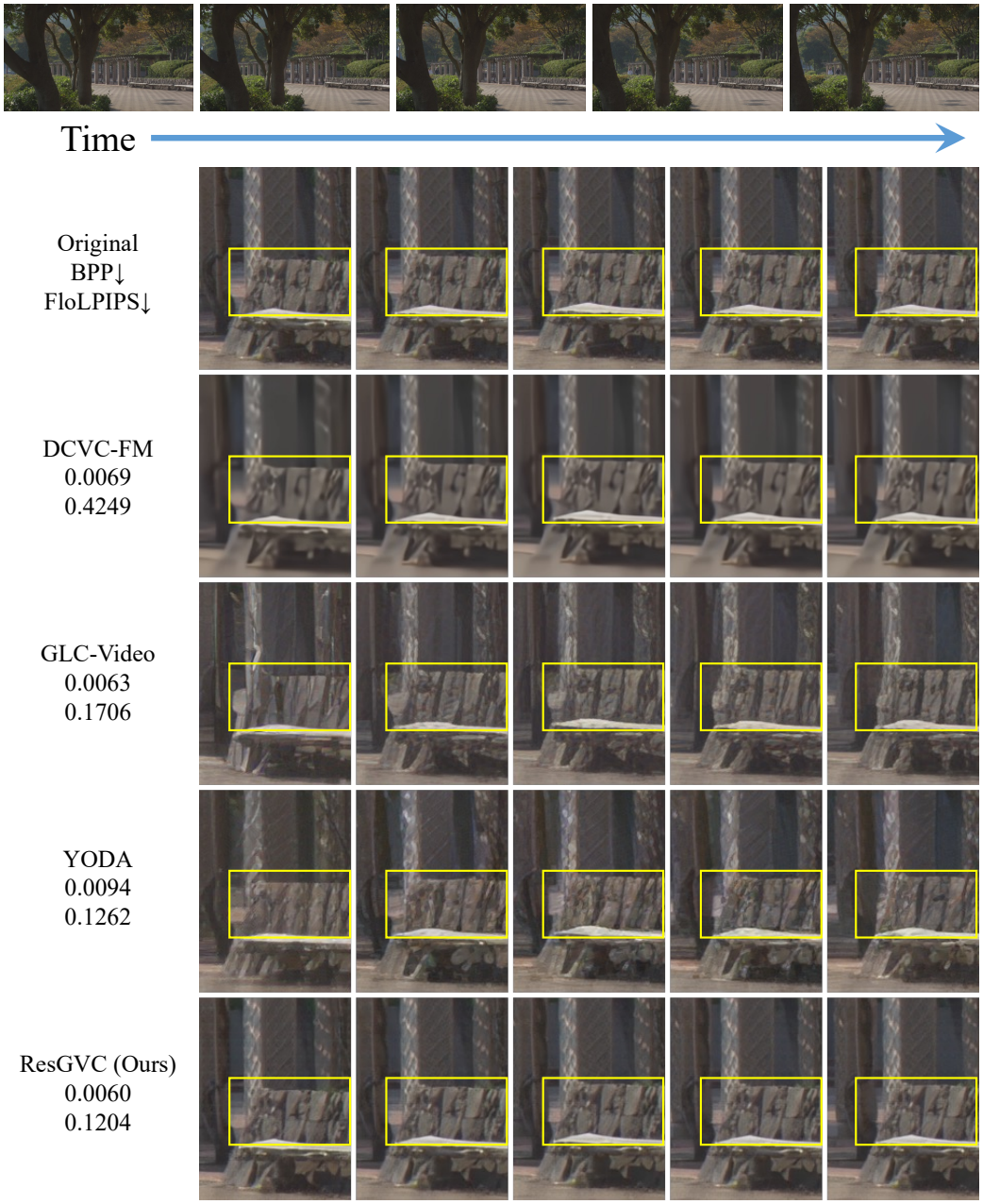


Figure 12: Visual comparisons across frames on the *ParkScene* sequence from the HEVC B dataset.