Low-Latency Neural Stereo Streaming

Qiqi Hou   Farzad Farhadzadeh   Amir Said   Guillaume Sautiere   Hoang Le*
Qualcomm AI Research†
{qhou, ffarhadz, asaid, gsautie, hoanle}@qti.qualcomm.com

Abstract

The rise of new video modalities like virtual reality or autonomous driving has increased the demand for efficient multi-view video compression methods, both in terms of rate-distortion (R-D) performance and in terms of delay and runtime. While most recent stereo video compression approaches have shown promising performance, they compress left and right views sequentially, leading to poor parallelization and runtime performance. This work presents Low-Latency neural codec for Stereo video Streaming (LLSS), a novel parallel stereo video coding method designed for fast and efficient low-latency stereo video streaming. Instead of using a sequential cross-view motion compensation like existing methods, LLSS introduces a bidirectional feature shifting module to directly exploit mutual information among views and encode them effectively with a joint cross-view prior model for entropy coding. Thanks to this design, LLSS processes left and right views in parallel, minimizing latency; all while substantially improving R-D performance compared to both existing neural and conventional codecs.

1. Introduction

The rise in popularity of autonomous vehicles (AVs) equipped with stereo cameras, along with the widespread use of virtual reality (VR) headsets, has led to a significant increase in stereo video data. For AVs, stereo cameras serve as a cost-effective alternative to sensors like LIDAR or RADAR. The data they capture is crucial for time-sensitive safety analyses during vehicle operation, necessitating low-latency data transmission. In VR, to achieve an immersive user experience, the demands for both resolution and latency are even higher. For both AV and VR applications, it's crucial that the codec encodes stereo video efficiently while maintaining low latency.

A basic approach to stereo video coding would apply a low-delay, single-view codec like AVC [62] or HEVC [23]

* Corresponding author
† Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

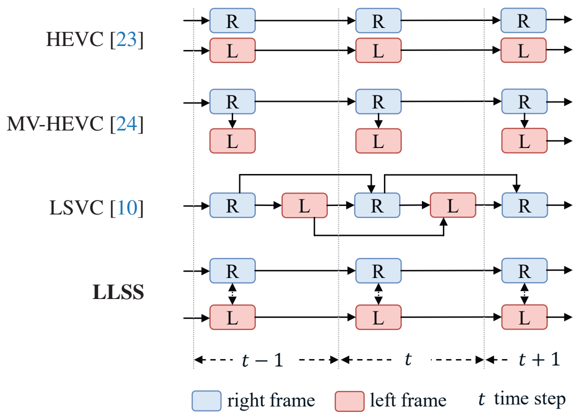

Figure 1. Comparison of multi-view compression strategies. In contrast to LSVC [10], our approach processes the left and right frames simultaneously. This parallel processing not only facilitates more rate-efficient coding, it also reduces the latency between the left and right views.

to each view independently. While these traditional codecs yield promising results, and are used in commercial products, such as Meta Quest [16], they double the rate and ignore the similarities between two views. Consequently, several standard codecs have been proposed to reduce the redundancy between two views through *disparity compensation* [54, 59]. They typically first encode the right view frame using a single-view codec. Then left view frame is predicted from the encoded right view frame. However, these sequential processing limits the ability to process multiple views simultaneously.

Recent years have seen rapid progress in single-view neural video codecs [2, 25, 26, 31, 32, 36, 38, 47, 49], particularly in low-delay settings. For instance, recent DCVC-DC Li et al. [36] has shown better Rate-Distortion (R-D) performance than the H.266 standard codec [7]. Our method is inspired by the most recent work LSVC [10], which is the first neural codec for stereo video. Although LSVC achieves great results and shows significant superiority over MV-HEVC [24] codec, it sequentially processes the right and left view frames, which constrains its suitability for low-latency applications like VR and AVs.

In this work, we present a Low-Latency Stereo video

Streaming (LLSS) codec designed for parallel stereo video coding. This codec's development is grounded in two key insights. First, inspired from the recent progresses in the stereo matching methods [9, 20, 50], the *disparity compensation* module between left and right views can be greatly simplified, compared to complex motion compensation schemes in LSVC [10]. It can be efficiently represented with horizontal shifts. Second, we observed that these disparity compensations can be executed concurrently for both views. A careful encoder design, sharing horizontally-shifted features across views, can implicitly estimate disparity, while facilitating parallel processing of both views, thereby achieving low-latency inference. Figure 1 shows a schematic comparison of these approaches. We introduce a novel component, *BiShiftMod* (Bidirectional Shifting Module), which facilitates the connections and information exchange between views in our network. This module is integrated into both the codec and hypercodec [5, 42], which enables data-dependent optimization of the cross-view mutual information. By following this approach, we replace the sequential disparity compensation with a parallel coding network that can exploit cross-view mutual information in a "disparity-agnostic" fashion.

Finally, we show that our solution substantially improves R-D performance compared to the state-of-the-art method on three common stereo video benchmarks, with 50.6% BD-rate savings on the CityScapes dataset [11], 18.2% on the KITTI 2012 dataset [17] and 15.8% on the KITTI 2015 dataset [41]. Besides, we also provide a neural network complexity and inference time study, and show that our model has only 35% of the complexity of LSVC [10] in terms of FLOPS. We further ablate each design choice to showcase the contribution of the proposed modules toward the final R-D performance.

The contributions of this paper include:

- A novel low-latency neural stereo video codec architecture that replaces sequential inter-view compensation with an efficient and parallelizable learned module to connect parallel autoencoders
- A bidirectional shift module that effectively captures and exhibits redundancy between inter-view features
- A set of thorough experiments demonstrating that our method is fast, efficient, and obtaining comparable and often better than state-of-the-art methods

2. Related Work

2.1. Neural video codecs

Neural networks have been successfully applied to data compression in many domains, including the image [1, 3, 5, 18, 22, 40, 42, 43, 55, 56] and video [2, 21, 25, 26, 31, 32, 34–36, 38, 48, 49, 51] settings. Most of these lossy compression systems are composed of one or more variational autoencoders referred to as *compressive autoencoders*[21, 29, 55]. Lossy compression is achieved through quantization of the latent variables in the bottleneck. These latents are further compressed in a lossless manner via entropy coding, typically using a learned, data-dependent prior model.

The main advantage of neural codecs is that they learn to compress from example data, whereas handcrafted codecs require expert design. This allows for easy customization to new domains [21], or even to specific videos or datapoints [52, 57, 58, 67]. Additionally, they may provide advantages from a deployment perspective. In practice, standard codecs often use hardware-based implementations to enable efficient operation, especially on mobile devices. However, these implementations tend to require a longer deployment process. In contrast, software-based neural codecs only need generic and ubiquitous AI accelerators for operation, making them more flexible and with the potential to enhance various application domains, especially where hardware-based codecs are not available. Lastly, neural codecs can be optimized end-to-end to improve *perceptual quality* through the use of perceptual loss functions [1, 18, 40, 43, 66], or take the semantics of the video into account via region-of-interest coding [8, 15]. Despite these dissimilarities, neural video codecs have taken inspiration from handcrafted codecs. Early works used temporal architectures [19, 21, 64], but follow-up work quickly adopted subnetworks for motion compensation and residual coding in the low latency [2, 38, 46, 48, 49] and streaming setting [30, 46]. Recently, neural video codecs have adopted advanced motion compensation techniques [25] and conditional coding, allowing them to become competitive with standard codecs in the low latency setting [35, 36].

2.2. Standard stereo video codecs

Although single-view codecs achieve strong compression performance, applying them to the stereo (and more broadly multi-view) domain by independently coding each view would lead to a suboptimal linear increase in rate. For this reason, early works in image coding extended support to stereo images by using *disparity compensation* [39, 45]. The idea is to encode one view independently, then predict the other view, for instance, with motion compensation. Then, the difference between this prediction and the ground truth is quantized and transmitted.

Subsequent standard works, like the Multiview Video Coding (MVC) [59], extended standard video codecs [53, 61] using variations of disparity compensation. In particular, the most recent standard MV-HEVC [54] adopted new techniques like the coding tree unit [27] to compress the disparity information. Such techniques have a few major drawbacks. First, they use many handcrafted components which cannot be optimized end-to-end. This makes it chal-

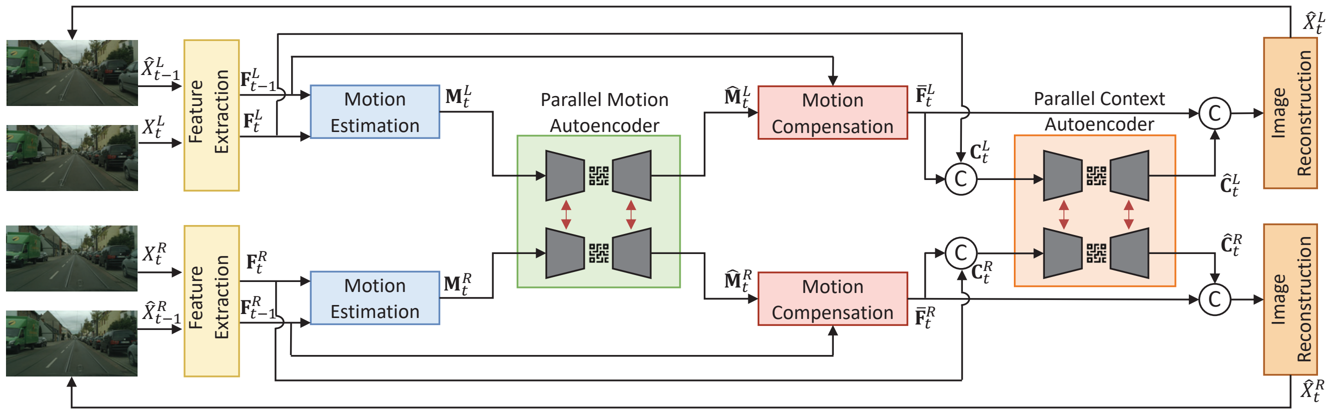

Figure 2. Overall architecture of our network. It contains two branches dedicated to processing the left and right view. It incorporates a parallel motion autoencoder and a parallel context autoencoder to reduce the redundant motion and context information across views, respectively. The weights are shared across views, including the feature extraction module, the motion estimation module, the motion compensation module, and the image reconstruction module.

lenging to optimize reconstructions for perceptual quality or for downstream vision tasks, such as in the automotive use case. Second, relying on explicit disparity compensation requires sequential processing of each view and therefore leads to poor parallelization across views.

2.3. Neural stereo video codecs

There are multiple works on neural stereo *image* coding [14, 33, 37, 63]. To the best of our knowledge, there is only one prior work on neural compression of stereo video, called LSVC [10]. All of these works apply some form of explicit disparity compensation. As an example of a recent stereo image coding work, SASIC by Wödlinger et al. [63] uses a shared codec between left and right views but only encodes the differences in latent space between the horizontally shifted right latent and the left latent when compressing the left view. By operating in feature space rather than pixel space, this codec allows capturing big disparities with few parameters due to the heavy spatial subsampling in the encoder. Recently, LSVC [10] was the first method to propose an end-to-end neural method for stereo video coding. The main idea is to first encode the right view, then use this to conditionally encode the left view. LSVC uses a reference buffer that keeps track of the last encoded inter and intra view frames. Using these frames, explicit feature-based motion (as originally introduced in FVC [25]) and disparity compensation are used. LSVC vastly outperforms the MV-HEVC standard on three common benchmarks.

3. Method

3.1. Redundancy and mutual information reduction

Consider a stereo video denoted by $\{\mathbf{X}_t^L, \mathbf{X}_t^R\}_{t\in\{1\cdots T\}}$ consisting of T frames captured concurrently by the left (L) and right (R) cameras. This video contains two primary types of redundancy: (1) temporal redundancy between consecutive frames ($\mathbf{X}_t^L$, $\mathbf{X}_{t+1}^L$) and ($\mathbf{X}_t^R$, $\mathbf{X}_{t+1}^R$); and (2) cross-view redundancy between $\mathbf{X}_t^L$ and $\mathbf{X}_t^R$.

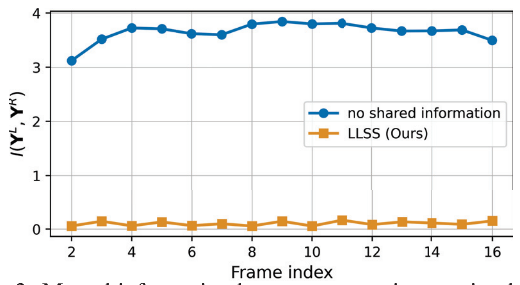

Figure 3. Mutual information between cross-view motion latents. $I(\mathbf{Y}^R;\mathbf{Y}^L) = -1/2\log_2(1-\rho^2)$ for a joint Gaussian distribution with a normalized cross-correlation ρ.

The performance of a video codec is significantly influenced by its ability to eliminate redundant information. For instance, temporal redundancy is commonly addressed using motion compensation techniques, where one image is aligned with another to "reuse" decoded information through a set of highly compressible motion vectors such as optical flows [2, 38, 51] or deformable kernels [25, 26, 36]. For stereo video compression, as illustrated in Figure 1, each frame at time t typically needs to perform two motion compensation steps: an *intra-view* step, where a prediction of the the current frame is based on the same camera view from the previous frame, and an *inter-view* step, which relies on the other camera view in the current frame for prediction. Conventionally, these processes are executed in a sequential pattern [10, 54] which hinders the opportunities for exploiting parallel processing and leveraging specific mutual information characteristics of the stereo videos.

Stereo videos are typically rectified and highly-correlated. For instance, the disparity between two views are always in the horizontal direction [9, 20, 50]. By reducing the redundant information between two views, we could compress the stereo videos more effectively. From

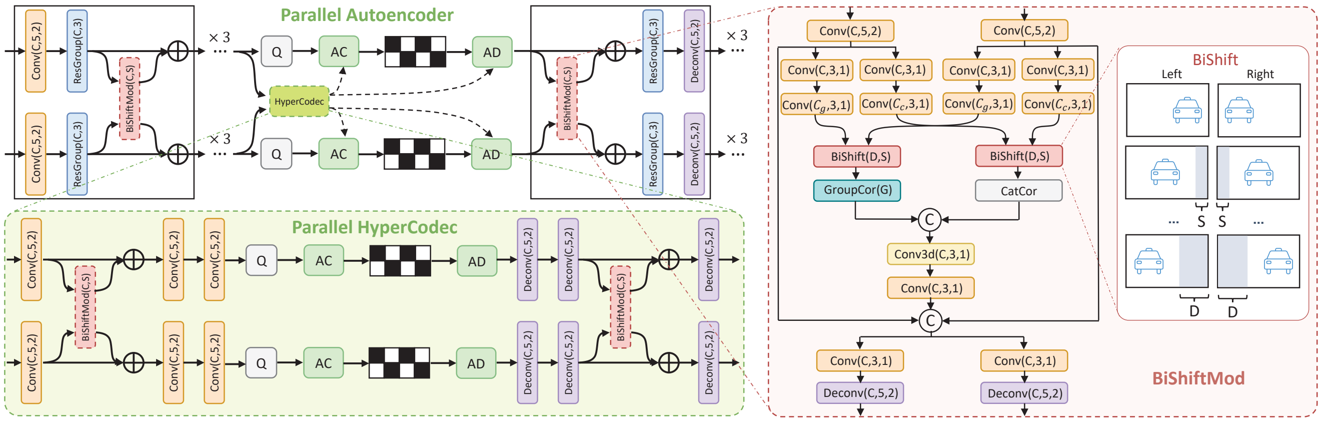

Figure 4. The architecture of a parallel autoencoder. It contains two parallel branches to compress the left and right features at the same time. The format reads "BlockType(channel, kernel_size, stride)". The Bidirectional Shift Module (BiShiftMod) is designed to learn the correlation between the left and right branches. It shifts the left and right features bidirectionally, estimating the Groupwise Correlation (GroupCor) features and Concatenation-based Correlation (CatCor) features between them. We omit activation layers for conciseness.

the rate-distortion theory [12], the total bit rate $\mathcal{R}_L + \mathcal{R}_R$ to encode the two separate latents $\mathbf{Y}^L$ and $\mathbf{Y}^R$ generated by encoders/decoders of left and right views (with shared information)

$$\mathcal{R}_L + \mathcal{R}_R \geq I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L, \mathbf{Y}_t^R) \tag{1}$$

where $\mathcal{R}_L, \mathcal{R}_R$ indicates the bit rate for the left and right view, respectively, and $I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L, \mathbf{Y}_t^R)$ indicates the mutual information between the pair of the random variables $(\mathbf{X}_t^R, \mathbf{X}_t^L)$ and $(\mathbf{Y}_t^L, \mathbf{Y}_t^R)$. It is upper bounded by

$$I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L, \mathbf{Y}_t^R) \leq I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L) + I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^R), \tag{2}$$

where $I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L)$ is indicating the mutual information [12] between joint random variable $(\mathbf{X}_t^R, \mathbf{X}_t^L)$ and random variable $\mathbf{Y}_t^L$. Similarly, $I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^R)$ is the mutual information [12] between joint random variable $(\mathbf{X}_t^R, \mathbf{X}_t^L)$ and random variable $\mathbf{Y}_t^R$.

This causes the bit rate overhead $\mathcal{W} \geq 0$

$$\begin{aligned}\mathcal{W} &= \Big[I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L) + I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^R)\Big] - I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L, \mathbf{Y}_t^R) \\ &= I(\mathbf{Y}_t^L; \mathbf{Y}_t^R) - I(\mathbf{Y}_t^L; \mathbf{Y}_t^R | \mathbf{X}_t^R, \mathbf{X}_t^L),\end{aligned} \tag{3}$$

comparing to the case we had a single joint encoder/decoder with the single joint latent $\mathbf{U}_t = (\mathbf{Y}_t^L, \mathbf{Y}_t^R)$ and $\mathcal{R} \geq I(\mathbf{X}_t^R, \mathbf{X}_t^L; \mathbf{Y}_t^L, \mathbf{Y}_t^R)$. $I(\mathbf{Y}_t^L; \mathbf{Y}_t^R | \mathbf{X}_t^R, \mathbf{X}_t^L)$ is a Conditional mutual information [12] of random variables $\mathbf{Y}_t^L$ and $\mathbf{Y}_t^R$ conditioned on joint random variables $(\mathbf{X}_t^R, \mathbf{X}_t^L)$.

If $I(\mathbf{Y}_t^L; \mathbf{Y}_t^R) = 0$, since $I(\mathbf{Y}_t^L; \mathbf{Y}_t^R | \mathbf{X}_t^R, \mathbf{X}_t^L) \geq 0$, the total bit rate overhead $\mathcal{W}$ has to be equal to 0. Consequently, to reduce the total bit rate overhead $\mathcal{W}$, the network should be designed to minimize $I(\mathbf{Y}_t^L; \mathbf{Y}_t^R)$. In our architecture, we enabled this via information-sharing between the two parallel autoencoders. Specifically, we make use of two autoencoders for left and right views, and let them share information through a (learned) shifted attention module after each convolutional block. These modules enable the information flow between the two left and right branches, helping the network to learn to reduce their redundancy, or equivalently the mutual information $I(\mathbf{Y}^L; \mathbf{Y}^R)$. Besides, the parallel design is beneficial for the parallel processing.

As shown in Figure 3, we observe a substantial reduction in the mutual information $I(\mathbf{Y}_t^L; \mathbf{Y}_t^R)$ between the left and right view latents when the two autoencoders are configured to share information compared to operating independently.

Given this approach of enabling more information flow, the remaining question is how to design a module that can efficiently capture and exhibit the mutual information between two branches of a codec. As inspired by the recent success of the recent state-of-the-art approach for stereo matching [50], this paper introduces a Bidirectional Shift module to effective capture and transfer the mutual information between views in stereo video compression. As a learnt component, this module naturally adapts to both flow and context latent between two views. The following sections present in details the network design of our method.

3.2. Low-Latency Neural Stereo Streaming

Figure 2 shows an overview of the architecture of LLSS. At each time step t, our method compresses both left and right view in a parallel manner. LLSS uses two branches, left and right, to accordingly compress the left and right image $\mathbf{X}_t^L$ and $\mathbf{X}_t^R$ into two separated latents $\mathbf{Y}_t^L$ and $\mathbf{Y}_t^L$. The network of each branch is partially adopted from the recent feature-based video compression methods [25, 34, 36], including the feature extractor, motion estimation, motion compensation module, and the image reconstruction module. Please find the implementation details in the supplementary materials.

To enable inter-view information flow, LLSS introduces a novel Bidirectional Shift Module, dubbed *BiShiftMod* that is inspired from the recent work from Wödlinger et al. [63]. This module is used to bridge the intermediate features of the network in both codec and hyper codec of the two branches.

3.2.1 Parallel AutoEncoders

As illustrated in Figure 2, LLSS has two pairs of parallel autoencoders: *parallel motion autoencoder* and *parallel context autoencoder*. Inside each of them, there are two autoencoders running in parallel corresponding to the left and right views. The architecture of each single autoencoder is adopted from recent state-of-the-art feature-based video codec [25]. Due to the limited space, we would like to refer the reader to the original paper [25] for more details of its architecture. Briefly, each autoencoder contains a residual-based encoder to transform its input into highly compressible latent, which is then coded with the help of a hyper prior network before being decoded back to the expected output via another residual-based decoder.

To enhance the intra-view information flow, the residual autoencoder compressing the residual feature $\mathbf{R}_t = \mathbf{F}_t - \bar{\mathbf{F}}_t$, originally in FVC [25] is replaced by a conditional autoencoder inspired by [35]. In this conditional autoencoder, $\mathbf{F}_t$ is fed directly into the encoder, and both encoder and decoder are conditioned on the warped feature $\bar{\mathbf{F}}_t$. Additionally, $\bar{\mathbf{F}}_t$ is fed into hyper codec to enhance the estimation of the parameters of the prior model.

To boost the inter-view information flow, we propose the "*Bidirectional Shift Module*". In summary, this block connects the modules of encoders and decoders of the left and right branches together to enable the flow of information across views, as illustrated in Figure 4. The next section details the implementation of this block.

3.2.2 Bidirectional Shift Module

Figure 4 shows the architecture of a Bidirectional Shift Module. It takes as input a pair of inter-view features, one from each branch of the codec, and outputs a pair of enhanced features. Inside, the inter-view features will be first transformed into a more representative form of intermediate features via a set of group-based convolutions. These intermediate features are then shifted via a module `BiShift(D,S)`. The shifted features are then passed through a set of Groupwise and Concatenation-based blocks to estimate their correlation.

The estimated correlations are then further transformed together with the input inter-view features. These sets of transformations and correlations help capturing the redundancy between the inter-view features while feeding them into each of the left and right encoders/decoders. The encoders/decoders can then share their information efficiently and reduce the mutual information between the inter-view latents before compression.

We describe some key components of Bidirectional Shift Module in detail:

- **Bidirectional shift (`BiShift(D,S)`)**: horizontally shifts left feature $\mathbf{F}^L$ to the left and the right feature $\mathbf{F}^R$ to the right with a max disparity D and a stride S.
- **Groupwise correlation (`GroupCor(G)`)**: inspired by the stereo matching networks [9, 20, 50], measures similarity between the shifted features. It splits the features into groups and calculates the cosine distance for each group. Following [20, 50], the shifted features are evenly divided into G groups along the channel dimension. The groupwise correlation is calculated by

$$\mathbf{V}_{gwc}(d,x,y,g) = \frac{1}{C_g/G}\langle \mathbf{F}_g^L(x,y), \mathbf{F}_g^R(x-d,y)\rangle, \tag{4}$$

  where $\langle\cdot,\cdot\rangle$ and (x,y) respectively indicates the inner product and the pixel coordinates. g and d designate the index of the groups and the disparity levels. $\mathbf{V}_{gwc}$ is defined in $[D,H,W,G]$, where H,W indicate the height and width of the feature map, respectively.
- **Concatenation-based correlation (`CatCor`)**: adopting from [9, 20, 50], it captures the similarity between the shifted features by just concatenating them. Compared to the Groupwise correlation, it would provide more context information, thus help guide the network to learn the redundancy between the left and right branches. To get the Concatenation-based feature maps [50], the shifted features are concatenated as follows

$$\mathbf{V}_{concat}(d,x,y) = \mathbf{F}^L(x,y) \| \mathbf{F}^R(x-d,y), \tag{5}$$

  where $\|$ indicates the concatenation operator along the shifted channels.

3.3. Loss Function

We optimize the entire network for the left and right views in an end-to-end manner. We adopt the typical rate-distortion loss [21, 55] as follows

$$\mathcal{L} = \sum_{v\in\{L,R\}}\sum_t D(\mathbf{X}_t^v, \hat{\mathbf{X}}_t^v) + \beta\left(\mathbb{H}(\mathbf{Y}_{M,t}^v) + \mathbb{H}(\mathbf{Y}_{C,t}^v)\right), \tag{6}$$

where $D(\cdot)$ indicates the distortion metric for the reconstructed frames. Depending on the training phase, it can be either the MSE or MS-SSIM loss. The superscript v indicates which view is considered between Left and Right. For each view v and time step t, $\mathbf{X}_t^v$ indicates the ground truth frame, $\hat{\mathbf{X}}_t^v$ the reconstructed frame, $\mathbf{Y}_{M,t}^v$ the quantized motion latent and $\mathbf{Y}_{C,t}^v$ the quantized context latent. $\mathbb{H}(\cdot)$ indicates the entropy function, which is proportional to the bitrate. β indicates the hyper-parameter used to control the trade-off between the frame distortion and the rate. Note that we omit the hyper latents entropy for conciseness.

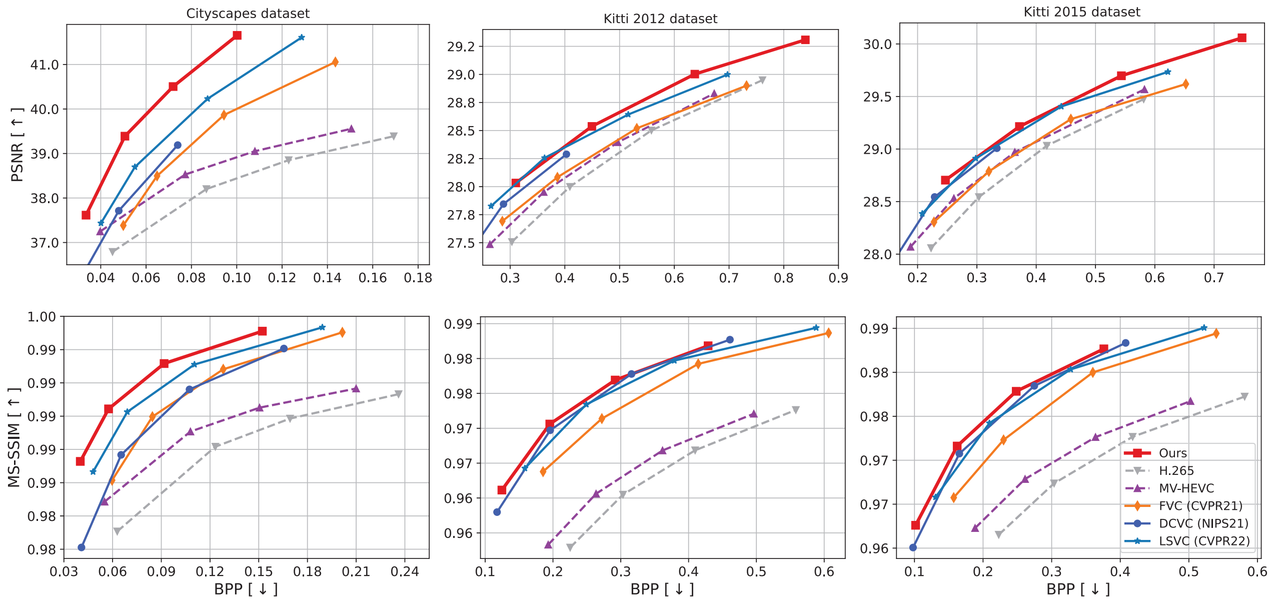

Figure 5. Rate-distortion curves in terms of PSNR and MS-SSIM on the CityScapes [11] , KITTI 2012 [17] and KITTI 2015 [41] datasets.

Table 1. BD-rate (%) on the CityScapes [11], KITTI 2012 [17], and KITTI 2015 [41] datasets. MV-HEVC [54] is set as the baseline. Lower is better, a negative number indicating bitrate *savings*.

Method	CityScapes [11]	KITTI 2012 [17]	KITTI 2015 [41]
HEVC [23]	33.3	7.9	12.7
FVC [25]	-15.6	-2.3	1.0
DCVC [34]	-15.2	-13.7	-12.3
LSVC [10]	-32.7	-17.1	-13.4
Ours	**-50.6**	**-18.2**	**-15.8**

4. Experiments

Datasets. We make use of 4 different datasets. For training, we use the single-view Vimeo90K dataset [65] for pre-training and then the stereo-camera CityScape dataset [11] train set for finetuning. For evaluation we use Cityscapes test set and the stereo-video KITTI 2012 [17] and 2015 [41] datasets.

The CityScapes [11] testing dataset comprises 1,525 30-frame stereo sequence pairs, with each containing two streams of size 2048×1024. The KITTI 2012 and 2015 testing datasets include 195 and 200 stereo sequence pairs, respectively, each containing 21 frames. We follow LSVC [10] data pre-processing for the CityScape and KITTI datasets, all frames are cropped into size 1920×704 and 1216×320, respectively.

Evaluation metrics. We measure rate in bits-per-pixel (BPP), and assess reconstruction fidelity with the commonly used Peak Signal-to-Noise Rate (PSNR) and Multi-Scale Structural SIMilarity (MS-SSIM) [60] metrics. To summarize the rate-distortion curve in a single number, we also report the Bjøntegaard-Delta rate (BD-rate) [6], which can be interpreted as an average bitrate saving for a fixed quality compared to a reference codec. All scores are reported in the RGB color space. We evaluate our methods with a Group-of-Picture (GoP) size equal to the total sequence length i.e., 30 and 21 frames for the CityScape and KITTI datasets, respectively. For model efficiency, we report the number of parameters along with FLOP and MAC per pixel. We measure the inference GPU time using the function `torch.cuda.Event()` as well as the function `torch.cuda.synchronize()` from the official PyTorch library [44], while FLOPs and MACs are calculated using `get_model_profile` from the `DeepSpeed` library [4, 13].

Training details. We implemented our neural stereo video codec using PyTorch [44]. Following a similar strategy to [10], we train our models in 3 stages:

First, a single view version of our model (hence without the "BiShiftMod" modules) is randomly initialized. We train the single view model on the Vimeo90k dataset and make use of its size and diversity. We train for 2M iterations with a learning rate of $5 \cdot 10^{-5}$, using MSE as distortion loss. Second, the resulting pre-trained weights from the first stage are used to initialize both branches of the full stereo network. We then train the BiShiftMod modules while freezing all other modules. During this step, we train the network for 10k iterations on the CityScape dataset with a learning rate of $1 \cdot 10^{-5}$. We found this step to greatly stabilize the training process. Finally, we finetune the entire network on the CityScape dataset for 200k iterations with a learning rate of $1 \cdot 10^{-5}$. When reporting MS-SSIM performance, we use a version of our network which is further finetuned using MS-SSIM as distortion loss for an additional 100k iterations.

Across all stages, we use the Adam optimizer [28] and train with various β values, specifically $[0.0002, 0.0004, 0.0008, 0.0016, 0.0032]$, to obtain rate

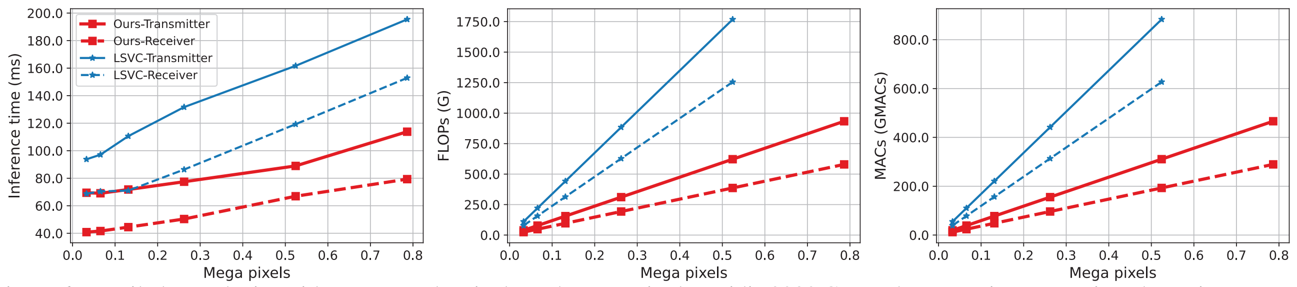

Figure 6. Detailed complexity with respect to the pixel number on a single Nvidia 3080 GPU. The transmitter comprises the entire network, while the receiver is tasked solely with a subset of functions to reconstruct the frame from bits.

curves. We use a batch size of 8 for the Vimeo dataset and 4 for the CityScape dataset. Additionally, we applied standard data augmentation during training. Specifically, we generated training samples by randomly cropping with size 256×256 for Vimeo-90k, and size 384×256 for CityScape. Our network was trained on two NVIDIA V100 GPUs for the first stage and only one for the other stages.

Standard Baselines. We compare our work to two standard baselines: H.265 [53] and its multi-view extension MV-HEVC [54]. We obtain the results of the standard codecs from the LSVC [10] paper. For H.265, it uses the HM-16.20 [23] implementation in the "lowdelay_P_main" preset on each view independently. MV-HEVC is from the HTM-16.3 implementation [24] with "baseCfg_2view" preset.

Learned Baselines. The only learned stereo video codec to date is LSVC by Chen et al. [10]. Like them, we include a comparison to a single-view codec FVC by Hu et al. [25], in which feature-based warping and residual compensation were introduced and inspired LSVC architecture. We report the scores of H.265, MV-HEVC, FVC, and LSVC as recorded in Chen et al. [10]. Besides, we also compare with DCVC [34]. Since DCVC didn't release the training codes, we evaluated it without finetuning. Therefore, the comparison between our method with DCVC should be interpreted with a grain of salt.

4.1. Comparison with state-of-the-art methods

Figure 5 and Table 1 respectively show the rate-distortion curves and BD-rate (with MV-HEVC as anchor) of all methods considered on the CityScapes, KITTI 2012 and 2015 test sets. Our LLSS method outperforms both learned and standard state-of-the-art methods. On the CityScape dataset, our method achieves **50.6%** BD-rate savings compared to MV-HEVC, while LSVC only saved 32.7%. On the KITTI 2012 and 2015 datasets, our method attains **18.2%** and **15.8%** BD-rate savings, respectively.

Note that for KITTI datasets, the gap in R-D performance to LSVC has tightened. We tested our method on the KITTI datasets without finetuning, following LSVC. However, KITTI and CityScapes datasets have different data distributions due to camera settings and baselines, and image processing settings. These differences lead to significant variations in BPP and PSNR ranges when applying conventional and data-driven methods codecs. Especially all neural codecs relying on training data so tend to be less performant in this setting. Despite these challenges, our LLSS method still achieves comparable and often better results than all existing conventional and neural codecs. Compared to LSVC, our method obtains an improvement of 1.1% and 2.4% BD-rate gain on KITTI 2012 and KITTI 2015, respectively, while being much faster. This demonstrates the effectiveness and generalization capability of our method and its potential to further improve its performance in the future with more generalized datasets.

4.2. Computational complexity study

In this study, we evaluate the complexity of both the transmitter and receiver components. The transmitter encompasses the entire network, as both encoding and decoding operations are carried out in order to create the bit streams. The receiver however only encompasses the feature extractor for previous frames, the parallel motion decoding module, the motion compensation, the parallel context decoding module, and the image reconstructor. As stated in Section 4, note that LSVC does not report complexity numbers, hence we re-implemented their architecture in order to get the complexity numbers.

We reduced the complexity of our method by observing that the pixel displacement in cross-view disparity of a pair of stereo frames is simpler and more predictable than the one caused by temporal motion. When compressing temporal motion and disparity cross views, LSVC requires large and complex networks (MRC and DRC). Our method greatly simplifies this by designing an efficient BiShift-Mod to align features cross-view. In the supplementary, we showed that our BiShiftMod accounts for only a small fraction of the overall computational complexity. Due to BiShiftMod, our parallel autoencoders have been designed to be more streamlined and efficient.

We examine the complexity of the transmitter and receiver in terms of inference time, FLoating-point OPerations (FLOPs), and Multiply-Add Cumulation (MACs). To investigate how these metrics perform for various video sizes, we crop the videos to the sizes including 128×128,

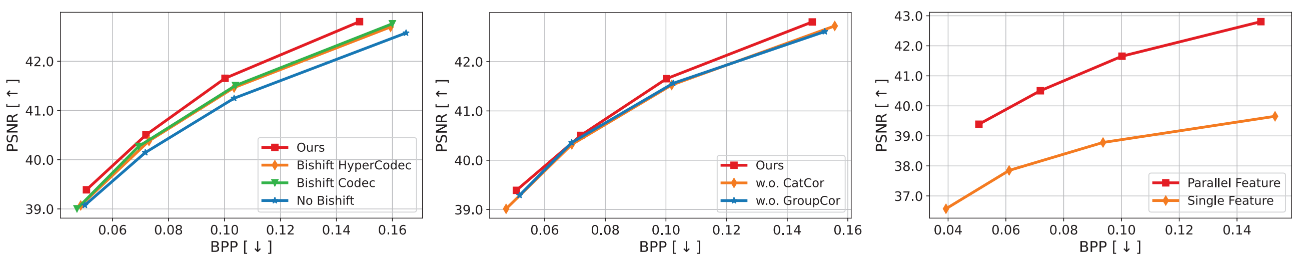

Figure 7. The effectiveness of the BiShiftMod, the components in BishiftMod, and the parallel feature on the CityScape dataset [11].

256×128, 256×256, 512×256, 512×512, and 768×512. Our experiments are conducted on a single Nvidia 3080 GPU, with a batch size of 1. We report the complexity for one pair of stereo P-frames and compare our method to the state-of-the-art stereo video compression approach, LSVC [10]. As illustrated in Figure 6, our method successfully reduces computational complexity across all examined metrics. For instance, considering a pair of stereo frames of size 512×512, our transmitter achieves an inference time $1.7\times$ times faster than LSVC, while our receiver is $1.9\times$ times quicker. In terms of FLOPs, LSVC exhibits $2.8\times$ and $3.2\times$ times higher complexity for the transmitter and receiver, respectively. Similarly, for MACs, LSVC demonstrates $2.8\times$ and $3.3\times$ times higher complexity for the transmitter and receiver, respectively.

4.3. Ablation study

Effectiveness of the BiShiftMod. We evaluate the effectiveness of the BiShiftMod by conducting tests it on the codec and hypercodec. When training the network without BiShiftMod, we skip the second training step. Figure 7 demonstrates that BiShiftMod significantly enhances the rate-distortion (RD) performance on the CityScape dataset. Specifically, employing BishiftMod on the codec and hypercodec lead to BD-rate savings of 7.3% and 6.1% compared to the configuration without BishftMod, respectively. When applied to the entire network, BishiftMod achieves 13.0% BD-rate reduction. These improvements can be attributed to BiShiftMod's robust ability to reduce rates, highlighting the efficacy of our BiShiftMod architecture.

Effectiveness of the BishiftMod components. We remove the component from BishftMod, including the groupwise correlation (CroupCor) and concatenation-based correlation (CatCor). Figure 7 shows the RD curve on the cityscape dataset. Both GroupCor and CatCor improve the results. Removing GroupCor and CatCor results in 3.7% and 4.3% BD-rate increasing, respectively, which demonstrates the effectiveness of our network architecture.

Effectiveness of parallel feature. We concatenate the left and right features as a single feature. Figure 7 shows that our paralleled feature streams approach achieves 62% BD-rate saving compared to using single shared feature. Our approach to split features into views and explicitly model their cross-view redundancy is more effective than simply asking a network to perform all of these operations implicitly, which is consistent with the many other works on monocular video compression tasks that compressing frames without modeling the motion tends to spend more bits.

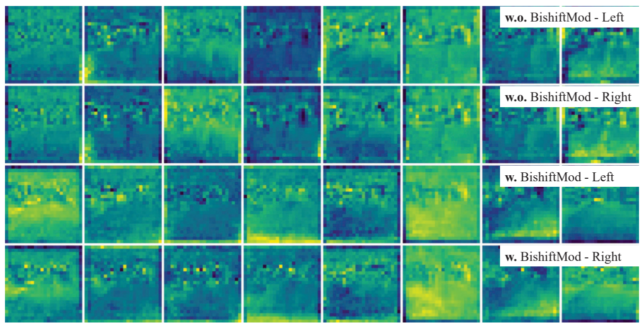

Figure 8. Visualization of sorted channels in the motion latents with the top-8 largest average energy. With BishiftMod, the latent features between left and right views become less alike.

Visualization of latent features. Figure 8 displays the latent features from the left and right branches with the top-8 largest average energy. We compare the differences between the left and right branches. The first two rows are from the model without BiShiftMod, while the bottom two rows are from the model with BiShiftMod. In the absence of BiShiftMod, the latent features appear very similar. However, when BiShiftMod is present, the latent features become less alike, indicating that BiShiftMod successfully reduces redundancy between the left and right branches.

5. Conclusion

We present a low-latency neural stereo video compression method designed to simultaneously compress left and right views. We develop a bidirectional-shift compression network for this purpose. The bidirectional-shift module effectively and efficiently captures the redundancy between the left and right frames. Our experiments demonstrate that our method significantly outperforms other state-of-the-art approaches. Furthermore, the experiments show that our bidirectional-shift module and parallel autoencoders contribute to the reduced bit rates and improved frame quality.

Acknowledgements We thank Auke Wiggers, Reza Pourreza for their insightful feedback. Additionally, we thank Wenyu Xia for providing the voiceover in our video demo.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 221–231, 2019. 2

[2] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020. 1, 2, 3

[3] Eirikur Agustsson, David Minnen, George Toderici, and Fabian Mentzer. Multi-realism image compression with a conditional generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22324–22333, 2023. 2

[4] Reza Yazdani Aminabadi, Samyam Rajbhandari, Minjia Zhang, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Olatunji Ruwase, et al. Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale. *arXiv preprint arXiv:2207.00032*, 2022. 6

[5] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. 2

[6] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001. 6

[7] 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. 1

[8] Chunlei Cai, Li Chen, Xiaoyun Zhang, and Zhiyong Gao. End-to-end optimized roi image compression. *IEEE Transactions on Image Processing*, 29:3442–3457, 2019. 2

[9] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5418, 2018. 2, 3, 5

[10] Zhenghao Chen, Guo Lu, Zhihao Hu, Shan Liu, Wei Jiang, and Dong Xu. LSVC: A Learning-Based stereo video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6073–6082, 2022. 1, 2, 3, 6, 7, 8

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2, 6, 8

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006. 4

[13] DeepSpeed. Deepspeed flops profiler. https://github.com/microsoft/DeepSpeed/tree/master/deepspeed/profiling/flops_profiler, 2023. [Online; accessed 1-May-2023]. 6

[14] Xin Deng, Wenzhe Yang, Ren Yang, Mai Xu, Enpeng Liu, Qianhan Feng, and Radu Timofte. Deep homography for efficient stereo image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1492–1501, 2021. 3

[15] Noor Fathima, Jens Petersen, Guillaume Sautière, Auke Wiggers, and Reza Pourreza. A neural video codec with spatial rate-distortion control. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5365–5374, 2023. 2

[16] Meta Quest for Creators. Encoding immersive videos for meta quest 2. https://creator.oculus.com/getting-started/media-production-specifications-for-delivery-to-meta-quest-2-headsets/, 2022. Accessed: 2023-11-16. 1

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 2, 6

[18] Noor Fathima Ghouse, Jens Petersen, Auke Wiggers, Tianlin Xu, and Guillaume Sautiere. A residual diffusion model for high perceptual quality codec augmentation. *arXiv preprint arXiv:2301.05489*, 2023. 2

[19] Adam Goliński, Reza Pourreza, Yang Yang, Guillaume Sautière, and Taco S. Cohen. Feedback recurrent autoencoder for video compression. *ACCV*, 2020. 2

[20] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019. 2, 3, 5

[21] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042, 2019. 2, 5

[22] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[23] HEVC. Hevc test model (hm). https://hevc.hhi.fraunhofer.de/HM-doc/, 2023. [Online; accessed 19-Apr-2023]. 1, 6, 7

[24] MV HEVC. Multiview high efficiency video coding (mv-hevc). https://hevc.hhi.fraunhofer.de/mvhevc, 2023. [Online; accessed 19-Apr-2023]. 1, 7

[25] Zhihao Hu, Guo Lu, and Dong Xu. FVC: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1502–1511, 2021. 1, 2, 3, 4, 5, 6, 7

[26] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5921–5930, 2022. 1, 2, 3

[27] Il-Koo Kim, Junghye Min, Tammy Lee, Woo-Jin Han, and JeongHoon Park. Block partitioning structure in the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1697–1706, 2012. 2

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[30] Théo Ladune, Pierrick Philippe, Wassim Hamidouche, Lu Zhang, and Olivier Déforges. Conditional coding for flexible learned video compression. *ICLR neural compression workshop*, 2021. 2

[31] Hoang Le, Reza Pourreza, Amir Said, Guillaume Sautiere, and Auke Wiggers. Gamecodec: Neural cloud gaming video codec. In *BMVC*, page 204, 2022. 1, 2

[32] Hoang Le, Liang Zhang, Amir Said, Guillaume Sautiere, Yang Yang, Pranav Shrestha, Fei Yin, Reza Pourreza, and Auke Wiggers. Mobilecodec: neural inter-frame video compression on mobile devices. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 324–330, 2022. 1, 2

[33] Jianjun Lei, Xiangrui Liu, Bo Peng, Dengchao Jin, Wanqing Li, and Jingxiao Gu. Deep stereo image compression via Bi-Directional coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19669–19678, 2022. 3

[34] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 4, 6, 7

[35] Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In *Proceedings of the 30th ACM International Conference on Multimedia*, 2022. 2, 5

[36] Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, Canada, June 18-22, 2023*, 2023. 1, 2, 3, 4

[37] Jerry Liu, Shenlong Wang, and Raquel Urtasun. DSIC: Deep stereo image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3136–3145, 2019. 3

[38] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 1, 2, 3

[39] M. Lukacs. Predictive coding of multi-viewpoint image sets. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 521–524, 1986. 2

[40] Fabian Mentzer, George Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33:11913–11924, 2020. 2

[41] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070. IEEE, 2015. 2, 6

[42] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018. 2

[43] Matthew J Muckley, Alaaeldin El-Nouby, Karen Ullrich, Hervé Jégou, and Jakob Verbeek. Improving statistical fidelity for neural image compression with implicit local likelihood models. *arXiv preprint arXiv:2301.11189*, 2023. 2

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[45] M.G. Perkins. Data compression of stereopairs. *IEEE Transactions on Communications*, 40(4):684–696, 1992. 2

[46] Reza Pourreza and Taco S Cohen. Extending neural p-frame codecs for b-frame coding. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6680–6689, 2021. 2

[47] Reza Pourreza, Hoang Le, Amir Said, Guillaume Sautière, and Auke Wiggers. Boosting neural video codecs by exploiting hierarchical redundancy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5355–5364, 2023. 1

[48] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[49] Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14479–14488, 2021. 1, 2

[50] Zhelun Shen, Yuchao Dai, Xibin Song, Zhibo Rao, Dingfu Zhou, and Liangjun Zhang. Pcw-net: Pyramid combination and warping cost volume for stereo matching. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 280–297. Springer, 2022. 2, 3, 4, 5

[51] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. *IEEE Transactions on Multimedia*, 2022. 2, 3

[52] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. *arXiv preprint arXiv:2112.04267*, 2021. 2

[53] 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. 2, 7

[54] Gerhard Tech, Ying Chen, Karsten Müller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang. Overview of the

multiview and 3d extensions of high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):35–49, 2016. 1, 2, 3, 6, 7

[55] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *ICLR*, 2017. 2, 5

[56] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017. 2

[57] Ties van Rozendaal, Johann Brehmer, Yunfan Zhang, Reza Pourreza, and Taco S Cohen. Instance-adaptive video compression: Improving neural codecs by training on the test set. *arXiv preprint arXiv:2111.10302*, 2021. 2

[58] Ties van Rozendaal, Iris AM Huijben, and Taco S Cohen. Overfitting for fun and profit: Instance-adaptive data compression. *arXiv preprint arXiv:2101.08687*, 2021. 2

[59] Anthony Vetro, Thomas Wiegand, and Gary J. Sullivan. Overview of the stereo and multiview video coding extensions of the h.264/mpeg-4 avc standard. *Proceedings of the IEEE*, 99(4):626–642, 2011. 1, 2

[60] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003*, pages 1398–1402 Vol.2, 2003. 6

[61] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003. 2

[62] 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. 1

[63] Matthias Wödlinger, Jan Kotera, Jan Xu, and Robert Sablatnig. SASIC: Stereo image compression with latent shifts and stereo attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 661–670, 2022. 3, 4

[64] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video compression through image interpolation. *Proceedings of the European conference on computer vision (ECCV)*, pages 416–431, 2018. 2

[65] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019. 6

[66] Ren Yang, Luc Van Gool, and Radu Timofte. Perceptual learned video compression with recurrent conditional GAN. *arXiv preprint arXiv:2109.03082*, 2021. 2

[67] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit neural video compression. *arXiv preprint arXiv:2112.11312*, 2021. 2