LSVC: A Learning-based Stereo Video Compression Framework

Zhenghao Chen[1], Guo Lu[2*], Zhihao Hu[3], Shan Liu[4], Wei Jiang[4], Dong Xu[1]
[1]University of Sydney, [2]Beijing Institute of Technology, [3]Beihang University, [4]Tencent America
{zhenghao.chen, dong.xu}@sydney.edu.au, guo.lu@bit.edu.cn, huzhihao@buaa.edu.cn

Abstract

In this work, we propose the first end-to-end optimized framework for compressing automotive stereo videos (i.e., stereo videos from autonomous driving applications) from both left and right views. Specifically, when compressing the current frame from each view, our framework reduces temporal redundancy by performing motion compensation using the reconstructed intra-view adjacent frame and at the same time exploits binocular redundancy by conducting disparity compensation using the latest reconstructed cross-view frame. Moreover, to effectively compress the introduced motion and disparity offsets for better compensation, we further propose two novel schemes called motion residual compression and disparity residual compression to respectively generate the predicted motion offset and disparity offset from the previously compressed motion offset and disparity offset, such that we can more effectively compress residual offset information for better bit-rate saving. Overall, the entire framework is implemented by the fully-differentiable modules and can be optimized in an end-to-end manner. Our comprehensive experiments on three automotive stereo video benchmarks Cityscapes, KITTI 2012 and KITTI 2015 demonstrate that our proposed framework outperforms the learning-based single-view video codec and the traditional hand-crafted multi-view video codec.

1. Introduction

Stereo cameras are often used as crucial sensors for autonomous driving applications as they can acquire more precise depth information than traditional mono cameras, while being less expensive than other sensors (*e.g.,* LIDAR) [7]. Due to the increasing demand for storing and transmitting the massive automotive stereo videos, how to effectively compress such data has become an emerging task.

Automotive stereo video compression can be regarded as a special case of multi-view video compression (MVC), for which several traditional MVC standards [35, 38, 41] have been proposed in the past decades. Nevertheless, such standards all rely on hand-crafted modules, which cannot be jointly end-to-end optimized. Moreover, they only aim at improving the compression performance for one view (*e.g.*, the left view) by exploiting both inter-view and intra-view redundancy, while compressing the base view (*e.g.*, the right view) by directly employing the single-view video codecs. Therefore, the compression performance could not be significantly improved without exploiting the cross-view redundancy when compressing the right-view videos.

While deep neural networks (DNNs) have achieved tremendous success in image and video compression [5,29], how to develop an effective learning-based stereo video codec remains an under-explored research problem. In this work, we propose the first learning-based stereo video compression (LSVC) framework for autonomous driving applications. Different from the traditional stereo video codecs, which simply compress the right-view videos by using the single-view video codecs, our framework effectively reduces both temporal and binocular redundancy for the stereo videos from both views, in which we use the information from one view to assist the compression of another view in an iterative fashion. Specifically, when compressing the current left/right frame, the reconstructed within-view frame (*i.e.,* the left/right frame) from the previous time-step and the latest reconstructed cross-view frame (*i.e.,* the right/left frame) are used as two reference frames. To better exploit both temporal redundancy for adjacent within-view frames and binocular redundancy for cross-view frames, in our approach, we estimate the motion offset between the current frame and the reconstructed previous frame from the same view, while estimating the disparity offset between the current frame and the latest reconstructed cross-view frame. After motion and disparity compression, we respectively perform motion compensation and disparity compensation to generate the intra-view and inter-view compensated features, which are eventually fused as the final compensated feature for the subsequent residual compression process.

Furthermore, we also propose two new approaches called motion residual compression (MRC) and disparity residual compression (DRC) to effectively compress the introduced motion and disparity offsets, which is based on

* Guo Lu is the corresponding author.

the observation that most automotive stereo sensors are particularly calibrated and their epipolar line is horizontal [12, 25]. Consequently, the produced stereo frames are rectified, which leads to a strong relationship among temporally adjacent stereoscopic frames [23, 26]. To exploit such a relationship, rather than directly compressing the raw offsets, we generate the predicted motion offset (*resp.,* disparity offset) based on the latest reconstructed motion offset (*resp.,* disparity offset) by leveraging the geometric correlation of stereoscopic frames (*resp.,* temporal correlation of adjacent within-view frames), such that we only need to compress the residual offset information between the raw offset and the predicted offset for better bit-rate saving.

Overall, we implement all our modules by using fully-differentiable DNNs and perform all coding operations in the feature space for better performance as in [19]. We conduct comprehensive experiments on three automotive stereo video benchmarks Cityscapes [10], KITTI 2012 [15] and KITTI 2015 [32]. The results demonstrate that our proposed framework significantly outperforms other codecs.

Our contributions are summarized as follows:

- Our proposed framework reduces both intra-view and inter-view redundancy by iteratively performing deep motion and disparity compensation in the feature space for both left-view and right-view videos. To the best of our knowledge, it is the first end-to-end optimized stereo video compression framework.
- We propose two new compression schemes MRC and DRC to compress the residual motion information and the residual disparity information, respectively, which enables effective compression of complex motion and disparity offsets from the automotive stereo videos.
- The extensive experiments on three automotive stereo video benchmarks demonstrate that our newly proposed framework achieves the state-of-the-art results for compressing the automotive stereo videos.

2. Related Work

2.1. Image and Video Compression

The traditional image and video codecs [6, 36, 37, 42, 44] mostly adopted the hand-crafted techniques to reduce the spatial and spatial-temporal redundancies. However, they cannot be end-to-end optimized based on large training datasets. Recently, several learning-based image compression methods were proposed, among which the most recent works [4, 5, 8, 9, 33, 39, 45] adopted the auto-encoder style networks to compress images. For example, Ballé *et al.* [5] proposed to use the variational auto-encoder (VAE) with hierarchical priors, which achieves better compression performance than BPG [6]. Inspired by the success of image compression, learning-based video compression also attracts increasing attention [3, 14, 16–18, 24, 27–30, 47]. Most works followed the hybrid coding framework and produced the compensated frames based on the estimated motion information. For example, DVC [29] is the first end-to-end optimized video codec by implementing all modules with DNNs, while the latest method FVC [19] performs all major coding operations in the feature space. Although such learning-based video codecs can achieve promising performance for single-view video compression, they cannot be directly applied for effective automotive stereo video compression as it is unclear how these methods can effectively handle the inter-view redundancy, which is one of the core challenges in the stereo video compression task.

2.2. Multi-view Image and Video Compression

Based on the hand-crafted modules, the traditional multi-view image compression methods usually used disparity compensation for stereo image compression [13, 31, 34]. For stereo video compression, the standard MVC methods were extended from the standard video codecs [36, 44]. For example, the MVC extension [41] of H.264 employed disparity compensation to exploit the inter-view redundancy. Extended from H.265, the most recent standard MV-HEVC [38] adopted the new techniques (*e.g.*, the coding tree unit [21]) to compress the disparity information. However, such MVC standards might not be suitable for automotive stereo video compression as they are all based on the hand-crafted modules and thus prevent the joint optimization of the compression task together with other machine vision tasks (*e.g.,* object detection) for the standard autonomous driving applications. Recently, Liu *et al.* [25] proposed the first deep stereo image coding method DSIC by adopting the parametric skip function, while Deng *et al.* [12] further employed homography estimation. However, these two approaches were only proposed for stereo image compression without considering the temporal redundancy of stereo videos. As far as we know, there are no learning-based stereo video compression approaches.

3. Methodology

3.1. Overview

Let us denote a rectified automotive stereo video pair as $(\mathcal{V}^l, \mathcal{V}^r)$, where the superscripts l and r respectively denote the left and right views from the camera. Our method compresses each frame pair (X_t^l, X_t^r) in an iterative fashion, where the subscript t denotes the time-step t. Specifically, as shown in the Fig. 1 (a), at time-step t, we use the LSVC right-view compression module to compress the right-view frame X_t^r and produce the reconstructed frame $\hat{X}_t^r$, in which we adopt the reconstructed within-view frame $\hat{X}_{t-1}^r$ from time-step $t-1$ and the most recent cross-view reconstructed frame $\hat{X}_{t-1}^l$ as two reference frames. Then, we compress the left-view frame X_t^l by using the LSVC left-view compression module, in which we employ the previ-

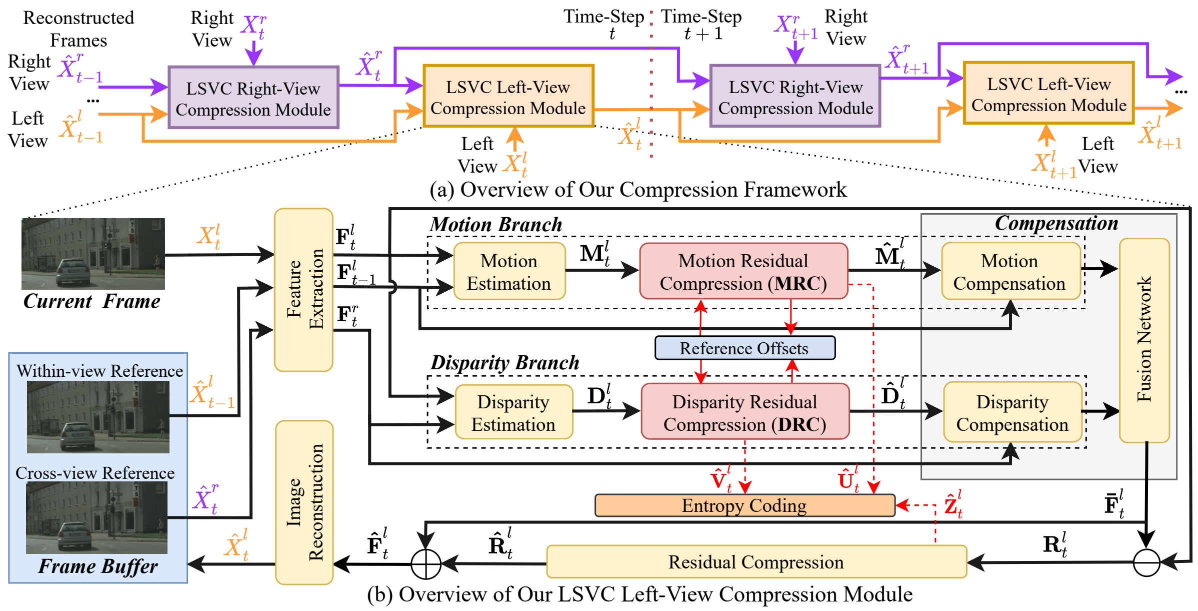

Figure 1. (a) At time-step t, we first use the reconstructed frames $\hat{X}^l_{t-1}$ and $\hat{X}^r_{t-1}$ respectively from the left-view and right-view videos at time-step $t-1$ as the reference frames to compress the right-view current frame X^r_t and then produce the reconstructed right-view current frame $\hat{X}^r_t$ by using the LSVC right-view compression module. By adopting $\hat{X}^r_t$ as the cross-view reference frame and $\hat{X}^l_{t-1}$ as the within-view reference frame, we then use the LSVC left-view compression module to compress the left-view current frame X^l_t and produce the reconstructed left-view current frame $\hat{X}^l_t$, which will be again used as the cross-view reference frame together with $\hat{X}^r_t$ for compressing the right-view frame X^r_{t+1} at next time-step $t+1$. (b) Taking the LSVC left-view compression module for compressing X^l_t as an example, we first extract the features $\mathbf{F}^l_t$, $\mathbf{F}^l_{t-1}$ and $\mathbf{F}^r_t$ respectively from the frames X^l_t, $\hat{X}^l_{t-1}$ and $\hat{X}^l_r$ by using the feature extraction module. Then, we produce the reconstructed motion offset (*resp.,* reconstructed disparity offset) by using motion estimation (*resp.,* disparity estimation) and motion residual compression (MRC) (*resp.,* disparity residual compression (DRC)) in the motion branch (*resp.,* disparity branch). After compensation including both motion and disparity compensation and the simple fusion operation, we produce a final compensated feature $\bar{\mathbf{F}}^l_t$. Following that, the residual feature $\mathbf{R}^l_t$ between $\mathbf{F}^l_t$ and $\bar{\mathbf{F}}^l_t$ is then compressed by using the residual compression module, such that we can then produce the final reconstructed feature $\hat{\mathbf{F}}^l_t$ by adding the reconstructed residual feature $\hat{\mathbf{R}}^l_t$ and $\bar{\mathbf{F}}^l_t$. Eventually, we decode the feature $\hat{\mathbf{F}}^l_t$ back to the reconstructed frame $\hat{X}^l_t$ by using the image reconstruction module, which will be then stored in the frame buffer.

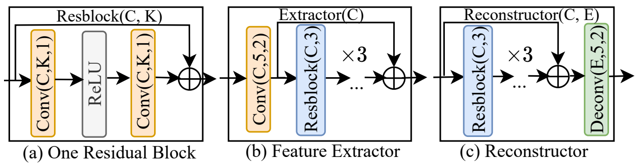

Figure 2. The structure of (a) one residual block, (b) the extractor for feature extraction and (c) the reconstructor for image/feature reconstruction, where "Conv(C, K, S)" and "Deconv(C, K, S)" represent the convolution and deconvolution operations with the number of output channels as C, the kernel size as $K \times K$ and the stride as S. In our main framework (*i.e.,* Fig. 1 (b)), the feature extraction module employs one "Extractor(64)", while the image reconstruction module employs one "Reconstructor(64, 3)".

ously reconstructed frames $\hat{X}^r_t$ as the cross-view reference frame and $\hat{X}^l_{t-1}$ as the within-view reference frame. After we obtain the reconstructed frame $\hat{X}^l_t$, we will use it as the cross-view reference frame together with $\hat{X}^r_t$ to help compress the right-view frame X^r_{t+1} at next time-step $t+1$ and then the iterative process will continue for compressing the subsequent left and right frames. The left-view and right-view compression modules are almost identical. So we will only take the left-view compression module as an example to illustrate our main framework (*i.e.,* Fig. 1 (b)) and summarise the difference between the left-view and the right-view modules in *Section 3.4*. Motivated by [19], all our operations are performed in the feature space. It is worth mentioning that our proposed framework is a general one and other advanced learning-based motion compensation and residual compression technologies beyond the current design can be readily incorporated into this framework.

Feature Extraction. We first use the feature extraction module (*i.e.,* our extractor shown in Fig. 2 (b)) to generate the features $\mathbf{F}^l_t$, $\mathbf{F}^l_{t-1}$ and $\mathbf{F}^r_t$ respectively from the input video frame X^l_t, and two reference frames including the reconstructed within-view frame $\hat{X}^l_{t-1}$ at time-step $t-1$ and the latest reconstructed frame $\hat{X}^r_t$ from the right view.

Motion Estimation and Disparity Estimation. Inspired by the existing works in video restoration [19,20,40], we also use deformable convolution [11] as our warping strategy (see details in *Section 3.2*). Therefore, we estimate the motion and disparity offsets $\mathbf{M}^l_t$ and $\mathbf{D}^l_t$ for the

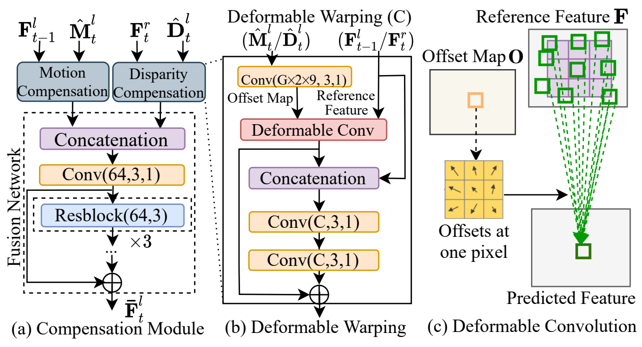

Figure 3. The structure of (a) our compensation module, (b) our deformable warping operation and (c) the deformable convolution operation. For motion (*resp.,* disparity) compensation, we adopt one "Deformable Warping (64)". Our deformable warping operation first uses a "Conv($G \times 2 \times 9, 3, 1$)" to produce the offset map. Note "G", "2", "9" denote the channel groups (G=8), two directions (the horizontal and vertical direction) of the offset map and the size of each kernel (3×3), respectively. Last, the deformable convolution module will use the offset map to warp the reference feature $\mathbf{F}$ from another view or the previous time-step to generate the predicted feature, which is followed by the standard post-processing operations such as concatenation and convolution.

deformable warping operation based on ($\mathbf{F}_t^l$, $\mathbf{F}_{t-1}^l$) and ($\mathbf{F}_t^l$, $\mathbf{F}_t^r$). For both motion and disparity estimation, we simply concatenate the two input features (*e.g.,* ($\mathbf{F}_t^l$, $\mathbf{F}_{t-1}^l$)) and then employ one "Conv(64, 3, 1)" and three "Resblock(64, 3)" operations to generate the estimated offsets.

Motion Residual Compression (MRC) and Disparity Residual Compression (DRC). Here, the MRC and DRC modules are respectively used to compress the left-view current motion offset $\mathbf{M}_t^l$ and the left-view current disparity offset $\mathbf{D}_t^l$. We generate the predicted motion (*resp.,* disparity) offset feature by warping the features from the reconstructed right-view current motion offset $\hat{\mathbf{M}}_t^r$ (*resp.,* the latest reconstructed left-view disparity offset $\hat{\mathbf{D}}_{t-1}^l$) using the deformable warping operation and then introduce the residual offset compression module for better compressing $\mathbf{M}_t^l$ and $\mathbf{D}_t^l$. Please see *Section 3.3* for more details.

Compensation. As shown in Fig. 3 (a), based on the previous temporal reference feature $\mathbf{F}_{t-1}^l$ and the reconstructed motion offset $\hat{\mathbf{M}}_t^l$, the motion compensation module employs our deformable warping operation (*i.e.,* Fig. 3 (b)) to warp the feature $\mathbf{F}_{t-1}^l$ from the previous time-step $t-1$ to the current time-step t and generate the motion compensated feature. Meanwhile, based on the reconstructed disparity offset $\hat{\mathbf{D}}_t^l$ and the right-view feature $\mathbf{F}_t^r$, we can produce the disparity compensated feature by using the same deformable warping operation again for disparity compensation. Then, a fusion network with a set of standard operations such as concatenation and convolution is proposed to combine these two compensated features and generate a more accurate left-view compensated feature $\bar{\mathbf{F}}_t^l$.

Residual Compression. The residual feature $\mathbf{R}_t^l$ between $\bar{\mathbf{F}}_t^l$ and $\mathbf{F}_t^l$ is then compressed by using the same auto-encoder style network as in [19] and the output is the reconstructed residual feature $\hat{\mathbf{R}}_t^l$.

Image Reconstruction. After adding the reconstructed residual feature $\hat{\mathbf{R}}_t^l$ and the compensated feature $\bar{\mathbf{F}}_t^l$, we produce the final reconstructed feature $\hat{\mathbf{F}}_t^l$, from which the final reconstructed image is generated by using the image reconstruction module, which directly employs the reconstructor operation shown in Fig. 2 (c).

Entropy Coding. We use the quantization approach and the learning-based entropy coding method as in [5] to losslessly compress the quantized motion feature $\hat{\mathbf{U}}_t^l$, the quantized disparity feature $\hat{\mathbf{V}}_t^l$ and the quantized residual feature $\hat{\mathbf{Z}}_t^l$, respectively produced from the motion branch, the disparity branch and the residual compression module.

3.2. Deformable Warping

We adopt deformable convolution [11] as our primary compensation and prediction technique in the feature space, which is intuitively illustrated in Fig. 3 (c). Based on the learned offset map $\mathbf{O}$ and the reference feature $\mathbf{F}$, for each kernel, we use the offsets to control the sampling locations in the reference feature. Then, all the values from the sampled locations in the reference feature are fused through the standard deformable convolution layer [11] to generate the predicted feature, which is further post-processed by using standard operations such as concatenation and two convolution operations. The whole process is called "deformable warping" in our framework. As suggested in [19], we divide the reference feature channels into 8 different groups (*i.e.,* G=8) and use a shared offset map for each group.

3.3. Motion and Disparity Residual Compression

For the automotive stereoscopic frames, the motion and disparity maps have strong geometric and temporal correlation [23, 26]. For example, both $\mathbf{M}_t^l$ and $\mathbf{M}_t^r$ represent the motion offsets from $t-1$ to t. Though they are calculated from different views, their correspondence can still be intuitively captured by the disparity offset between the left and right views. Therefore, we can approximate the raw left-view motion offset $\mathbf{M}_t^l$ based on the reconstructed right-view motion offset $\hat{\mathbf{M}}_t^r$. Hence, we only need to compress the residual offset information between the predicted and raw offset features for better bit-rate saving.

Our MRC module is shown in Fig. 4 (a), we first extract the reference feature $f(\hat{\mathbf{M}}_t^r)$ from the reconstructed right-view current motion offset $\hat{\mathbf{M}}_t^r$. Based on the feature $f(\hat{\mathbf{D}}_{t-1}^l)$ (extracted from the reconstructed left-view disparity offset $\hat{\mathbf{D}}_{t-1}^l$ at previous time-step $t-1$), we perform the deformable warping operation (*i.e.,* Fig. 3 (b)) to warp this reference feature $f(\hat{\mathbf{M}}_t^r)$ and generate a predicted offset feature. Then we only need to compress the residual feature between the raw offset feature extracted from $\mathbf{M}_t^l$ and the

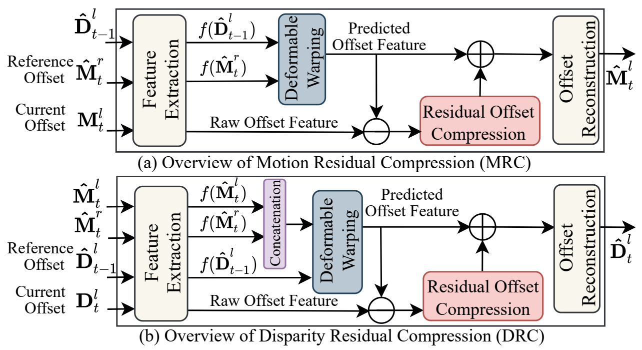

Figure 4. The network structure of (a) our motion residual compression (MRC) and (b) our disparity residual compression (DRC) modules. For MRC (*resp.,* DRC), we extract $f(\hat{\mathbf{M}}_t^r)$ (*resp.,* $f(\hat{\mathbf{D}}_{t-1}^l)$) as the reference feature from the reconstructed right-view current motion offset $\hat{\mathbf{M}}_t^r$ (*resp.,* the reconstructed left-view disparity offset $\hat{\mathbf{D}}_{t-1}^l$ at time-step $t-1$) to produce the predicted offset feature. Therefore, we only need to compress the residual offset feature between the raw offset feature extracted from the current raw motion offset $\mathbf{M}_t^l$ (*resp.,* the current raw disparity offset $\mathbf{D}_t^l$) and the predicted offset feature by using the residual offset compression module, in which we adopt 2 "Conv(128, 3, 1)" and 1 "Conv(128, 5, 2)" operations to compress the residual offset feature, then produce the quantized residual motion feature (*resp.,* quantized residual disparity feature), and eventually reconstruct it by using 2 "Deconv(128, 3, 1)" and 1 "Deconv(128, 5, 2)" operations. For both MRC and DRC, we employ 3 "Extractor(128)" operations for the feature extraction module, 3 "Reconstructor(128, 128)" operations for the offset reconstruction module and 1 "Deformable Warping (128)" operation (*i.e.,* Fig. 3 (b)) to generate the predicted offset feature. For MRC, we use the feature $f(\hat{\mathbf{D}}_{t-1}^l)$ extracted from the reconstructed left-view disparity offset $\hat{\mathbf{D}}_{t-1}^l$ at time-step $t-1$ to help warp the reference feature. While for DRC, we first concatenate two features $f(\hat{\mathbf{M}}_t^l)$ and $f(\hat{\mathbf{M}}_t^r)$ respectively extracted from the reconstructed left and right motion offsets $\hat{\mathbf{M}}_t^l$ and $\hat{\mathbf{M}}_t^r$ at time-step t to help warp the reference feature.

predicted offset feature by using an auto-encoder style network called residual offset compression. The corresponding quantized residual motion feature is denoted as $\hat{\mathbf{U}}_t^l$.

As shown in Fig. 4 (b), the scheme our DRC module is almost identical to our MRC module, in which we extract the reference feature $f(\hat{\mathbf{D}}_{t-1}^l)$ from the latest reconstructed left-view disparity offset $\hat{\mathbf{D}}_{t-1}^l$ at time-step $t-1$ to predict the raw offset feature extracted from the left-view current disparity offset $\mathbf{D}_t^l$. To produce better prediction, we employ the features from the latest left-view and right-view reconstructed motion offsets $\hat{\mathbf{M}}_t^l$ and $\hat{\mathbf{M}}_t^r$ for warping the reference feature $f(\hat{\mathbf{D}}_{t-1}^l)$, in which we simply concatenate the two features $f(\hat{\mathbf{M}}_t^l)$ and $f(\hat{\mathbf{M}}_t^r)$ from both views. Again, we only need to compress the residual offset feature between the raw disparity feature extracted from $\mathbf{D}_t^l$ and the predicted offset feature. The corresponding quantized residual disparity feature is denoted as $\hat{\mathbf{V}}_t^l$.

3.4. LSVC Right-view Compression Module

We adopt the most recent reconstructed left-view frame $\hat{X}_{t-1}^l$ as the cross-view reference frame and the reconstructed right-view adjacent frame $\hat{X}_{t-1}^r$ from time-step $t-1$ as the within-view reference frame, and then use the LSVC right-view compression module to compress the right-view current frame X_t^r. The right-view compression module is almost identical to the left-view compression module (*i.e.,* Fig. 1 (b)), which also consists of three main blocks: motion branch, disparity branch and residual compression. There are only two minor modifications.

First, for the motion branch, we directly compress the right-view motion offset $\mathbf{M}_t^r$ without using MRC as the corresponding reference motion offset from the left-view at time-step t is not available. Specifically, we use the feature extraction module (see Fig. 4 (a)) to generate the raw offset feature from $\mathbf{M}_t^r$ and then directly quantize it as $\hat{\mathbf{U}}_t^r$ for lossless entropy coding, and eventually we use the offset reconstruction module (see Fig. 4 (a)) to generate the reconstructed right-view motion offset $\hat{\mathbf{M}}_t^r$.

Second, as the reconstructed cross-view reference frame $\hat{X}_{t-1}^l$ at time-step $t-1$ is different from the current right-view frame X_t^r in terms of both timp-steps and views, it will be difficult to directly perform the disparity compensation based on its extracted feature $\mathbf{F}_{t-1}^l$. To improve the performance, we generate an intermediate feature $\tilde{\mathbf{F}}_t^l$ at time-step t by compensating $\mathbf{F}_{t-1}^l$ based on the reconstructed left-view current motion offset $\hat{\mathbf{M}}_t^l$. Note, we can perform the operations from the motion branch of the LSVC left-view compression module before the disparity branch of the LSVC right-view compression module to generate $\hat{\mathbf{M}}_t^l$, as this step does not require the reconstructed right-view frame $\hat{X}_t^r$ to be ready. Therefore, we eventually adopt the intermediate feature $\tilde{\mathbf{F}}_t^l$ to perform all operations in the disparity branch of the LSVC right-view compression module.

3.5. Training Strategy

We optimize our framework for compressing the videos from both left and right views. Specifically, we jointly compress the stereoscopic frame pair (X_t^l, X_t^r) and the corresponding objective function is formulated as follows,

$$\begin{aligned}\mathcal{L} =& \lambda D(X_t^l, \hat{X}_t^l) + R(\hat{\mathbf{U}}_t^l) + R(\hat{\mathbf{V}}_t^l) + R(\hat{\mathbf{Z}}_t^l) \\ &+ \lambda D(X_t^r, \hat{X}_t^r) + R(\hat{\mathbf{U}}_t^r) + R(\hat{\mathbf{V}}_t^r) + R(\hat{\mathbf{Z}}_t^r)\end{aligned} \tag{1}$$

where $D(\cdot)$ represents the distortion between the reconstructed and original frames. $R(\cdot)$ represents the bit-rate cost in the compression procedure. Here, $\hat{\mathbf{U}}_t^l$ and $\hat{\mathbf{U}}_t^r$ represent the quantized residual motion feature produced by MRC from the left-view video and the quantized motion feature produced by motion compression from the right-view video. Then, $\hat{\mathbf{V}}_t^l$ and $\hat{\mathbf{V}}_t^r$ represent the quantized residual disparity features produced by DRC from the left-view

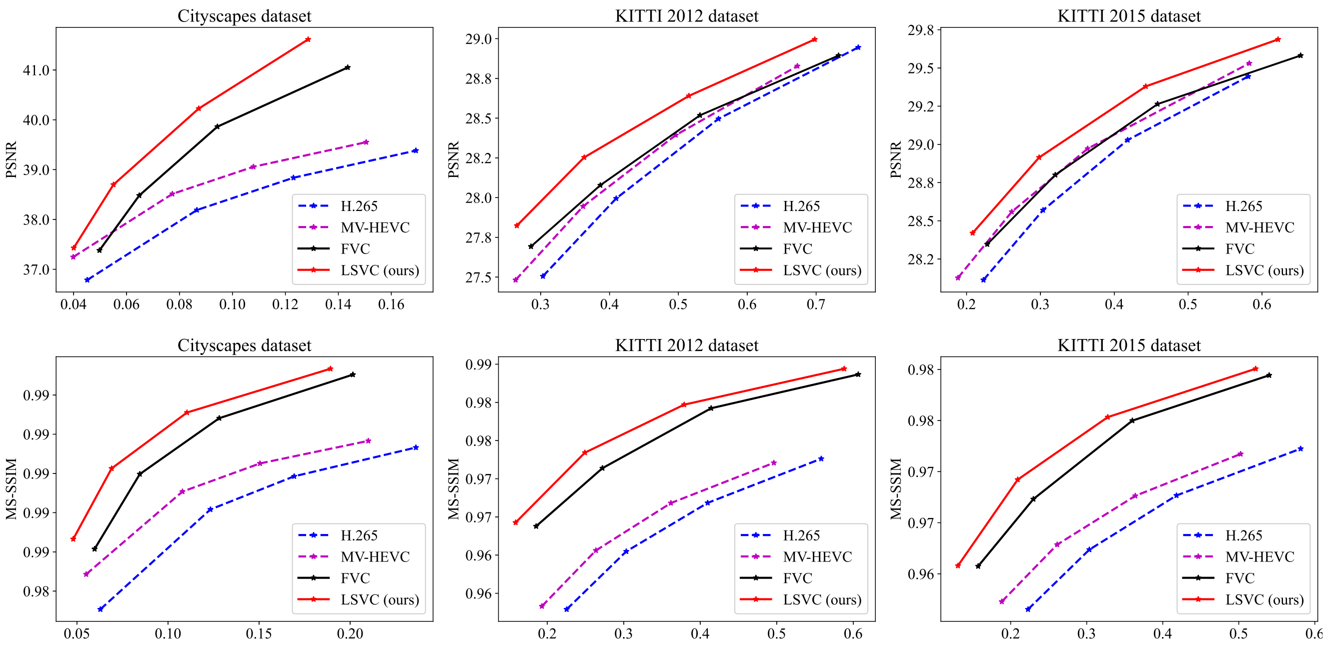

Figure 5. Experimental results on three automotive stereo video benchmark datasets Cityscapes, KITTI 2012 and KITTI 2015. We report both PSNR and MS-SSIM compression results. We use the models trained based on the Cityscapes dataset when evaluating the PSNR results on all three benchmark datasets. When using MS-SSIM for compression performance evaluation, we additionally fine-tune both FVC [19] and our LSVC models based on the Cityscapes dataset by using MS-SSIM as the distortion loss.

and right-view videos, respectively. Last, $\hat{\mathbf{Z}}_t^l$ and $\hat{\mathbf{Z}}_t^r$ represent the quantized residual features produced by residual compression from the left-view and right-view videos, respectively. We learn the whole network by solving the above optimization problem in an end-to-end fashion.

4. Experiments

4.1. Evaluation Datasets

Cityscapes. The Cityscapes raw sequence [10] consists of 2975, 500 and 1525 sequence pairs in the training, validation and testing sets, respectively. Each sequence pair contains two 30-frames videos with the resolution of 2048×1024. In our experiments, we follow [10] to use the default training/testing split. For the preprocessing procedure, we follow the evaluation setting as in [25], where we crop out the top and left parts with rectification artifacts as well as the bottom part with the ego vehicle.

KITTI. We further evaluate our method on two stereo video benchmark datasets KITTI 2012 [15] and KITTI 2015 [32]. KITTI 2012 consists of 195 stereo sequence pairs, while KITTI 2015 consists of 200 stereo sequence pairs. Each sequence has 21 frames and we crop all sequences to the resolution of 1216×320 for fair evaluation.

4.2. Experiment Protocols

Implementation Details. In our implementation, we train our models with four different λ values ($\lambda = 512, 1024, 2048, 4096$). We first pre-train all modules in the motion branch and residual compression based on the Vimeo-90K dataset [46], in which the parameters of those modules are shared for both left-view and right-view modules. A two-stage training scheme is then used for training our model based on the Cityscapes dataset. At the first stage, we train our framework without using the MRC and DRC modules, in which we first use the learning rate of 5e-5 in the first 6 epochs and then reduce the learning rate as 5e-6 in the subsequent 3 epochs. In the second stage, we first fix all other parameters, but only train both MRC and DRC modules based on the learning rate of 5e-5 for the first 3 epochs and then we train the network in an end-to-end fashion with all parameters based on the learning rate of 5e-6 for the subsequent 3 epochs. To reduce memory consumption and computational cost, we directly reuse the encoded motion and disparity offset features, which are produced when previously compressing the latest two frames. So we can bypass the feature extraction procedures in both MRC and DRC modules. To achieve better MS-SSIM [43] results, we further fine-tune the learned model for 3 more epochs based on the learning-rate of 5e-6 by using the MS-SSIM-based loss function. Note, we use the same model trained based on the Cityscapes dataset to evaluate our framework on all three benchmark datasets Cityscapes, KITTI 2012 and KITTI 2015. The framework is implemented by PyTorch with CUDA support and optimized by using Adam Optimizer [22] on a machine with a single NVIDIA 2080Ti GPU, in which we set the batch size as 4. We use the ran-

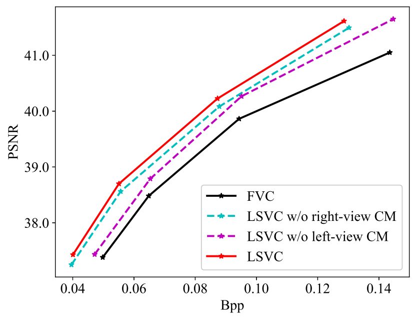

Figure 6. Ablation study of the joint video compression strategy for both left-view and right-view videos on the Cityscapes dataset. (1) **FVC**: the baseline method FVC for compressing both view videos, (2) **LSVC w/o right-view CM** (*resp.,* (3) **LSVC w/o left-view CM**): we remove the LSVC right-view compression module (*resp.,* the LSVC left-view compression module) and directly use FVC to compress the right-view videos (*resp.,* the left-view videos). (4) **LSVC**: our proposed complete framework.

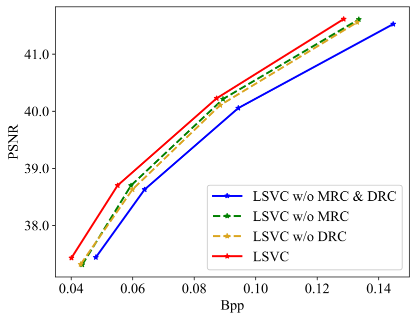

Figure 7. Ablation study of our newly proposed MRC and DRC modules on the Cityscapes dataset. (1) **LSVC w/o MRC & DRC**: LSVC without adopting both MRC and DRC modules to compress the motion offset and the disparity offset. (2) **LSVC w/o MRC**: LSVC without adopting the MRC module to compress the motion offset. (3) **LSVC w/o DRC**: LSVC without adopting the DRC module to compress the disparity offset. (4) **LSVC**: our proposed complete framework.

dom flip technique and randomly crop each training image to the size of 320×256 for data augmentation.

Metrics. We use bits per pixel (Bpp) to measure the bit-rate. In addition, we use PSNR and MS-SSIM to measure the distortion between the compressed and original videos.

Baseline Codecs. In our experiments, we include the traditional video codec H.265 [36] and its MVC extension MV-HEVC [38]. We adopt the most recent state-of-the-art learning-based video codec FVC [19], which also employs deformable convolution and performs all coding operations in the feature space as our baseline method. To pre-train FVC, we follow its original training scheme in [19] based on the Vimeo-90K dataset. Then, we fine-tune the models based on the Cityscapes dataset by using the same training scheme as in our LSVC (See *Implementation Details*). For H.265, we use HM-16.20 [1] with the *lowdelay_P_main* configuration setting, while for MV-HEVC, we use the HTM-16.3 [2] software with the *baseCfg_2view* configuration setting. Note, we set *Intra-Period=-1* for all codecs (*i.e.,* only the first frame of each video will be compressed as I-frame). For two learning-based video codecs FVC and our LSVC, we use the learning-based image compression method in [9] to compress I-frame, which is pre-trained as in [9] and then fine-tuned based on the Cityscapes dataset. Inspired by MV-HEVC, our LSVC only compresses the right-view first frame as the I-frame, while compressing the left-view first frame by using our left-view compression module. To minimize the cumulative error, our LSVC (*resp.,* FVC) also follows the HTM (*resp.,* HM) to use a better compression model for compressing the 4^{th} frame pair (*resp.,* frame) of every 4 frame pairs (*resp.,* frames).

4.3. Experiment Results

We provide the compression results on three datasets in Fig. 5. It is noted that our method outperforms all other video codecs including the traditional MVC method MV-HEVC [38] and the baseline method FVC [19]. We would like to highlight that our LSVC significantly outperforms FVC on all datasets. These results indicate that it is beneficial to reduce both intra-view redundancy along the temporal dimension and the inter-view binocular redundancy when compressing the stereo video pairs. Whereas, the standard learning-based video codecs such as FVC only focus on reducing the temporal redundancy. Moreover, in Table 1, we provide the BDBR results by using MV-HEVC [38] as the anchor method. It shows that our method respectively saves around 32.66%, 17.14% and 13.35% bit-rates on the Cityscapes, KITTI 2012 and KITTI 2015 datasets, which clearly indicates the superiority of our proposed framework over the traditional MVC methods.

Table 1. BDBR (in %) results of H.265 [36], FVC [19] and our LSVC, in which MV-HEVC [38] is used as the anchor method. Negative values indicate bit-rate savings, while positive values indicate additional bit-rate costs.

	H.265	FVC	LSVC (Ours)
Cityscapes	33.27	-15.55	**-32.66**
KITTI 2012	7.86	-2.29	**-17.14**
KITTI 2015	12.73	1.02	**-13.35**

4.4. Ablation Study

Effectiveness of LSVC left-view and right-view compression modules. We take the Cityscapes dataset as an example to evaluate two variants of our framework. Specifically, the alternative method LSVC w/o left-view CM (*resp,* LSVC w/o right-view CM) only keeps the LSVC right-view compression module (*resp.,* the LSVC left-view compression module) and directly employs FVC [19] to compress the left-view videos (*resp.,* the right-view videos), in which the reconstructed left-view current frame $\hat{X}_t^l$ (*resp.,* the reconstructed right-view current frame $\hat{X}_t^r$) comes from the output of FVC instead of the LSVC left-view compression module (*resp.,* the LSVC right-view compression module). The design of our alternative methods is similar to the traditional stereo video codecs that usually compress one base view by the single-view video codecs. As shown in Fig. 6, when compared with the baseline method FVC, the first alternative method LSVC w/o left-view CM can save around 8.95% bit-rates, while the second alternative method LSVC w/o right-view CM can save up to 15.38% bit-rates. When compressing the right-view current frame X_t^r, the LSVC right-view compression module directly adopts the reconstructed left-view frame $\hat{X}_{t-1}^l$ at previous time-step $t-1$ as the cross-view reference frame, therefore it might be harder to perform the disparity compensation due to the additional information mismatch between different time-steps, which degrades the performance of the first alternative method LSVC w/o left-view CM. Last, our complete framework outperforms both alternative methods and saves 19.23% bit-rates when compared with FVC, which indicates that it is beneficial to simultaneously use our proposed left-view and right-view compression modules in an iterative fashion.

Effectiveness of MRC and DRC. To evaluate the effectiveness of our newly proposed modules MRC and DRC, we take the Cityscapes dataset as an example to perform several new experiments and the results are reported in Fig. 7. Specifically, for LSVC w/o MRC (*resp.,* LSVC w/o DRC), we replace the MRC module (*resp.,* the DRC module) by directly compressing the motion offset (*resp.,* the disparity offset) with the feature extraction and offset reconstruction modules without going through the deformable warping and residual offset compression processes. Namely, we directly quantize the raw motion offset feature (*resp.,* the raw disparity offset feature) after using the feature extraction module (See Fig. 4) for lossless entropy coding. When compared with our LSVC complete model, we observe that the first alternative method LSVC w/o MRC will additionally consume 5.71% bit-rates, while the second alternative method LSVC w/o DRC will bring 7.61% extra bit-rates. Moreover, the third alternative method without using both MRC and DRC (*i.e.,* LSVC w/o MRC & DRC) achieves the worst results and introduces 16.07% extra bit-rates.

Visualization. We further provide the visualization results to demonstrate the effectiveness of our newly proposed MRC and DRC modules, in which we take the 2^{nd} frame of Bielefeld-028 from the Cityscapes dataset as an example. Fig. 8 (d) shows the raw motion offset $\mathbf{M}_t^l$ estimated based on the current frame (See Fig. 8 (a)) and the within-view reference frame at time-step $t-1$ (See Fig. 8 (b)), while Fig. 8 (e) and Fig. 8 (f) represent the predicted motion offset and the residual motion offset reconstructed from the corresponding features (note both offsets are reconstructed by using the offset reconstruction module (See Fig. 4)). Here we select the representative offset map (from the total number of 72 offset maps) for better visualization. It is observed that

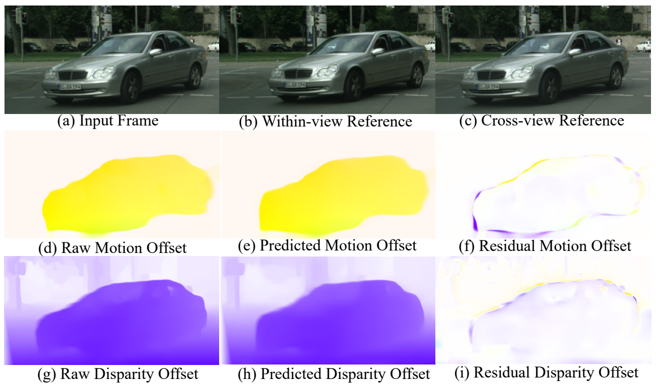
(a) Input Frame
(b) Within-view Reference
(c) Cross-view Reference
(d) Raw Motion Offset
(e) Predicted Motion Offset
(f) Residual Motion Offset
(g) Raw Disparity Offset
(h) Predicted Disparity Offset
(i) Residual Disparity Offset

Figure 8. The images of (a) the input left-view current frame X_t^l, (b) the within-view reference frame $\hat{X}_{t-1}^l$ and (c) the cross-view reference frame $\hat{X}_t^r$. The visualization results of (d) the raw motion offset $\mathbf{M}_t^l$ produced based on X_t^l and $\hat{X}_{t-1}^l$, (e) the corresponding predicted motion offset and (f) the residual motion offset produced from our MRC, (g) the raw disparity offset $\mathbf{D}_t^l$ produced based on X_t^l and $\hat{X}_t^r$, (h) the corresponding predicted disparity offset and (i) the residual disparity offset produced from our DRC.

our predicted motion offset from the MRC module is very close to the raw motion offset. Therefore, it takes much less bit-rates for compressing the corresponding residual motion offset (See Fig. 8 (f)). We have similar observations for the disparity offset (See Fig. 8 (g), (h) and (i)).

Complexity. Last, we take the Cityscapes dataset as an example to report the encoding time (in seconds per frame). All experiments are performed on the machine with one Intel i9-10900X CPU and one NVIDIA 2080Ti GPU. The encoding times of our LSVC are respectively 10.34s on the CPU platform and 0.35s on the GPU platform, which is much faster than the traditional hand-crafted MVC standard MV-HEVC (*i.e.,* 18.27s on the CPU platform).

5. Conclusion

In this work, we have proposed the first end-to-end optimized learning framework for compressing automotive stereo videos, in which we introduce a new iterative deep compression paradigm to effectively compress both left-view and right-view videos by reducing both temporal and binocular redundancy. Two new compression schemes MRC and DRC are also proposed for better compressing the introduced motion and disparity offsets. Extensive experiments show that our framework outperforms the traditional MVC standard and the most recent single-view learning-based video codec, which provides a strong baseline for compressing automotive stereo videos and facilitates the future research works for the learning-based MVC task.

Acknowledgement This work was supported by Key Research and Development Program of Guangdong Province of China (No. 2019B010154003) and in part by the National Natural Science Foundation of China (No. 62102024).

References

[1] Hevc test model (hm). https://hevc.hhi.fraunhofer.de/HM-doc/. Accessed: 2021-11-06. 7

[2] Mv-hevc test model (htm). https://hevc.hhi.fraunhofer.de/\svn/svn_3DVCSoftware/. Accessed: 2021-11-06. 7

[3] 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. 2

[4] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations (ICLR)*, 2017. 2

[5] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 4

[6] Fabrice Bellard. Bpg image format. *URL https://bellard.org/bpg*, 2015. 2

[7] Richard Bishop. Emerging from stealth, nodar introduces "hammerhead 3d vision" platform for automated driving. *Forbes*. 1

[8] Zhenghao Chen, Shuhang Gu, Guo Lu, and Dong Xu. Exploiting intra-slice and inter-slice redundancy for learning-based lossless volumetric image compression. *IEEE Transactions on Image Processing*, 2022. 2

[9] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020. 2, 7

[10] 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

[11] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 3, 4

[12] 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. 2

[13] Its' hak Dinstein, Gideon Guy, Joseph Rabany, Joseph Tzelgov, and Avishai Henik. On the compression of stereo images: Preliminary results. *Signal Processing*, 17(4):373–382, 1989. 2

[14] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6421–6429, 2019. 2

[15] 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. IEEE, 2012. 2, 6

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

[17] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *European Conference on Computer Vision*, pages 193–209. Springer, 2020. 2

[18] 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*, 2022. 2

[19] 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. 2, 3, 4, 6, 7

[20] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3224–3232, 2018. 3

[21] 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

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015. 6

[23] Hsueh-Ying Lai, Yi-Hsuan Tsai, and Wei-Chen Chiu. Bridging stereo matching and optical flow via spatiotemporal correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1890–1899, 2019. 2, 4

[24] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3546–3554, 2020. 2

[25] Jerry Liu, Shenlong Wang, and Raquel Urtasun. Dsic: Deep stereo image compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3136–3145, 2019. 2, 6

[26] Pengpeng Liu, Irwin King, Michael R Lyu, and Jia Xu. Flow2stereo: Effective self-supervised learning of optical flow and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6648–6657, 2020. 2, 4

[27] Salvator Lombardo, Jun Han, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pages 9287–9298, 2019. 2

[28] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*, pages 456–472. Springer, 2020. 2

[29] 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 Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 1, 2

[30] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3292–3308, 2020. 2

[31] M Lukacs. Predictive coding of multi-viewpoint image sets. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 521–524. IEEE, 1986. 2

[32] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 2, 6

[33] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 2

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

[35] Gary J Sullivan, Jill M Boyce, Ying Chen, Jens-Rainer Ohm, C Andrew Segall, and Anthony Vetro. Standardized extensions of high efficiency video coding (hevc). *IEEE Journal of selected topics in Signal Processing*, 7(6):1001–1016, 2013. 1

[36] 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

[37] David S Taubman and Michael W Marcellin. Jpeg2000: Standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, 2002. 2

[38] 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, 2015. 1, 2, 7

[39] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *International Conference for Learning Representations*, 2017. 2

[40] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. TDAN: Temporally-deformable alignment network for video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3360–3369, 2020. 3

[41] 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

[42] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 2

[43] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 6

[44] 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. 2

[45] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018. 2

[46] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 6

[47] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020. 2