FreqSIC: Frequency-aware Stereo Image Compression with Bi-directional Checkerboard Context Model

Supplementary Material

Network Structure

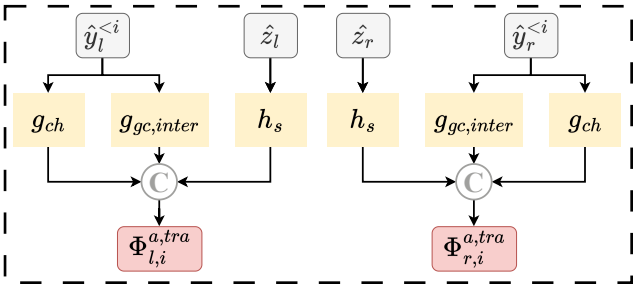
The network architectures of the anchor intra-view prior estimation, anchor entropy parameter prediction, non-anchor intra-view prior estimation, and nonanchor entropy parameter prediction modules are detailed in Figure 9. We follow the single checkboard context model in [19] to generate rich intra-view priors. Since g_{ch}, $g_{gc,inter}$, $g_{lc,attn}$, and $g_{gc,intra}$ are not the focus of this work, we refer readers to Jiang and Wang [19] for detailed definitions.

Experimental Details

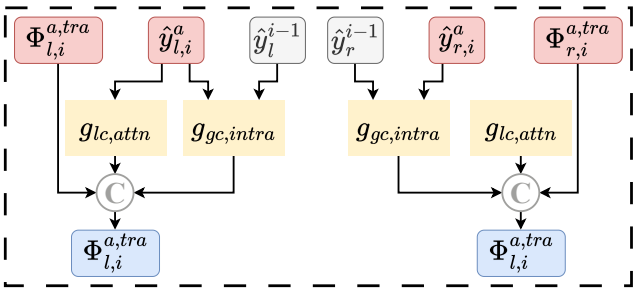
To ensure fair and consistent comparisons, all training and testing settings strictly adhere to prior works [25, 41, 42, 46]. Specifically, each image in the InStereo2K dataset is pre-processed so that its dimensions are divisible by 64. For the Cityscapes dataset, we remove rectification artifacts and the self-vehicle by cropping 64 pixels from the top, 256 pixels from the bottom, and 128 pixels from both sides of each image. During evaluation, we use image resolutions of 1,024×832 for InStereo2K and 1,792×704 for Cityscapes.

Regarding traditional codec baselines, BPG [5] is evaluated in the YUV 4:4:4 format to preserve visual fidelity. HEVC and VVC are implemented based on the JVET standard. To accommodate stereo image pairs, we convert them into YUV 4:4:4 video streams using ffmpeg, where the left view is encoded as an I-frame and the right as a P-frame. It is important to note that MV-HEVC only supports YUV 4:2:0, which causes noticeable degradation in PSNR, particularly at higher bitrates.

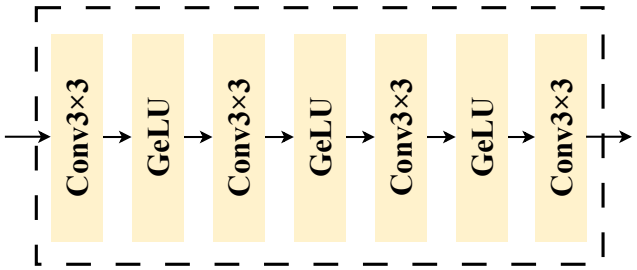
In addition, we reproduce the BCSIC [21] baseline and evaluate it under the same image resolutions used in [25, 41, 42, 46], instead of the original 512×512 setting used in [21]. This modification is necessary, as the original setting produces significantly lower rate-distortion performance. All results are reported under this unified evaluation protocol to ensure a fair and meaningful comparison across methods.



(a) Anchor intra-view prior estimation network.



(b) Nonanchor intra-view prior estimation network.



(c) Anchor/Nonanchor entropy parameter prediction network.

Figure 9. The architectural details of each component within the context model. h_s denotes the hyper-decoder. g_{ch}, $g_{gc,inter}$, $g_{lc,attn}$ and $g_{gc,intra}$ represent the channel-wise context module, inter-slice global spatial context module, shifted window-based checkerboard attention, and intra-slice global spatial context module, respectively.

BD-PSNR and BD-Rate for Image Codec

```
def BD_PSNR(R1, PSNR1, R2, PSNR2, piecewise=0):
    lR1 = np.log(R1)
    lR2 = np.log(R2)

    PSNR1 = np.array(PSNR1)
    PSNR2 = np.array(PSNR2)

    p1 = np.polyfit(lR1, PSNR1, 3)
    p2 = np.polyfit(lR2, PSNR2, 3)

    # integration interval
    min_int = max(min(lR1), min(lR2))
    max_int = min(max(lR1), max(lR2))

    # find integral
    if piecewise == 0:
        p_int1 = np.polyint(p1)
        p_int2 = np.polyint(p2)
```

```
        int1 = np.polyval(p_int1, max_int) - np.
    polyval(p_int1, min_int)
        int2 = np.polyval(p_int2, max_int) - np.
    polyval(p_int2, min_int)
    else:
        # See https://chromium.googlesource.com/
    webm/contributor-guide/+/master/scripts/
    visual_metrics.py
        lin = np.linspace(min_int, max_int, num
    =100, retstep=True)
        interval = lin[1]
        samples = lin[0]
        v1 = scipy.interpolate.pchip_interpolate
    (np.sort(lR1), PSNR1[np.argsort(lR1)],
    samples)
        v2 = scipy.interpolate.pchip_interpolate
    (np.sort(lR2), PSNR2[np.argsort(lR2)],
    samples)
        # Calculate the integral using the
    trapezoid method on the samples.
        int1 = np.trapz(v1, dx=interval)
        int2 = np.trapz(v2, dx=interval)

    # find avg diff
    avg_diff = (int2-int1)/(max_int-min_int)

    return avg_diff
```

Code 1. BD-PSNR

```
def BD_RATE(R1, PSNR1, R2, PSNR2, piecewise=0):
    lR1 = np.log(R1)
    lR2 = np.log(R2)

    # rate method
    p1 = np.polyfit(PSNR1, lR1, 3)
    p2 = np.polyfit(PSNR2, lR2, 3)

    # integration interval
    min_int = max(min(PSNR1), min(PSNR2))
    max_int = min(max(PSNR1), max(PSNR2))

    # find integral
    if piecewise == 0:
        p_int1 = np.polyint(p1)
        p_int2 = np.polyint(p2)

        int1 = np.polyval(p_int1, max_int) - np.
    polyval(p_int1, min_int)
        int2 = np.polyval(p_int2, max_int) - np.
    polyval(p_int2, min_int)
    else:
        lin = np.linspace(min_int, max_int, num
    =100, retstep=True)
        interval = lin[1]
        samples = lin[0]
        v1 = scipy.interpolate.pchip_interpolate
    (np.sort(PSNR1), lR1[np.argsort(PSNR1)],
    samples)
        v2 = scipy.interpolate.pchip_interpolate
    (np.sort(PSNR2), lR2[np.argsort(PSNR2)],
    samples)
        # Calculate the integral using the
    trapezoid method on the samples.
        int1 = np.trapz(v1, dx=interval)
        int2 = np.trapz(v2, dx=interval)

    # find avg diff
    avg_exp_diff = (int2-int1)/(max_int-min_int)
    avg_diff = (np.exp(avg_exp_diff)-1)*100
    return avg_diff
```

Code 2. BD-Rate

Visualizations

We conduct the visual comparison of our proposed FreqSIC against the current SoTA methods. Several examples from InStereo2K datasets are shown in Figure 10. From this example, we can observe that FreqSIC achieves the best rate-distortion trade-off and delivers the best visual quality. FreqSIC achieves better reconstruction of fine textures because more high-frequency information is preserved during stereo alignment.

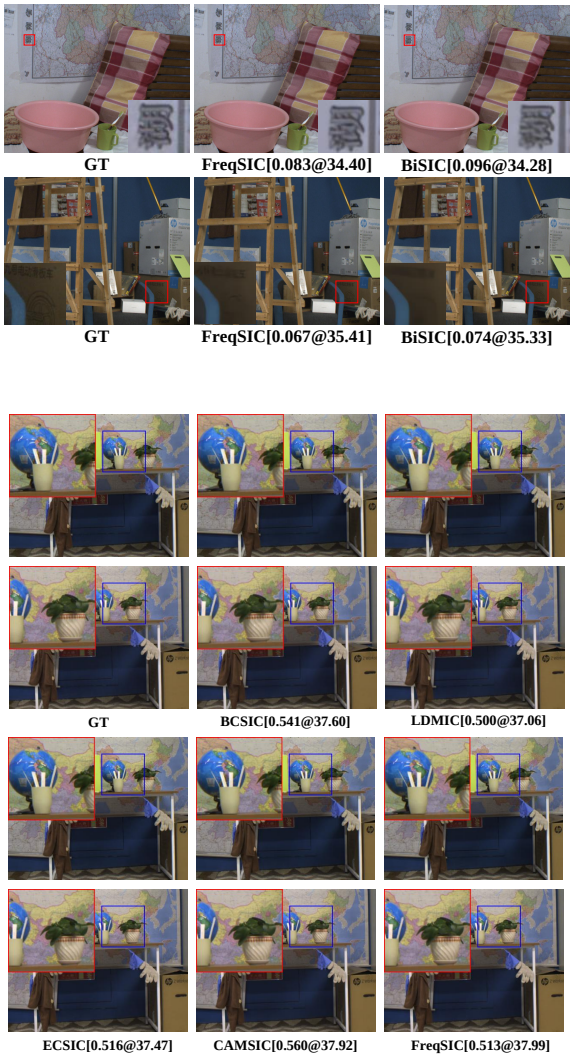


Figure 10. Visualization comparison on Instereo2K. The titles under the figures are represented as [BPP @ PSNR(dB)].