Supplementary Material: Video Frame Interpolation with Transformer

Liying Lu[1] Ruizheng Wu[2] Huaijia Lin[2] Jiangbo Lu[2] Jiaya Jia[1]
[1] The Chinese University of Hong Kong [2] SmartMore
{lylu,rzwu,hjlin,leojia}@cse.cuhk.edu.hk, jiangbo@smartmore.com

1. More Implementation Details

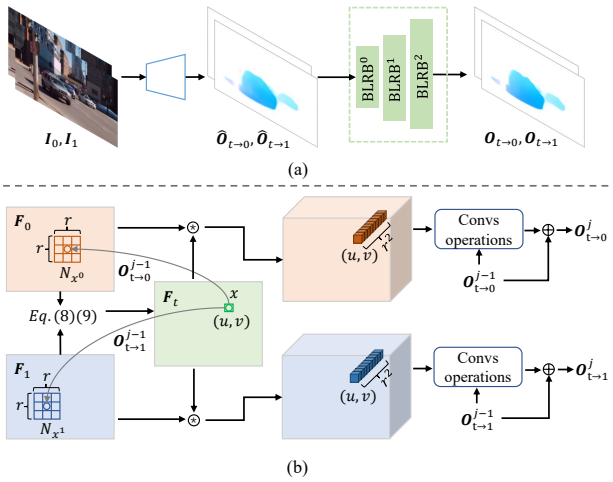

Figure 1. (a). The pipeline of our flow estimator, where the Bilateral Local Refinement Block (BLRB) is used to refine the flows. (b). Details of the Bilateral Local Refinement Block.

Flow Estimator Achitecture. The pipeline of our flow estimator is shown in Fig. 1a, a flow prediction network [2] is first used to predict the coarse flows $\hat{\boldsymbol{O}}_{t\to0}$, $\hat{\boldsymbol{O}}_{t\to1}$. Then Bilateral Local Refinement Blocks (BLRBs) are used to refine the flows in a coarse-to-fine manner, whose details are shown in Fig. 1b.

For the j-th BLRB, given the flows $\boldsymbol{O}^{j-1}_{t\to0}$ and $\boldsymbol{O}^{j-1}_{t\to1}$ produced by the last BLRB and feature maps $\boldsymbol{F}^j_0$, $\boldsymbol{F}^j_1$ of two input frames extracted by the encoder Enc, we first rescale the flows to the current scale (note that we do not change their notations after rescaling for brevity) and use them to backward warp the features, obtaining $\widetilde{\boldsymbol{F}}^j_0$ and $\widetilde{\boldsymbol{F}}^j_1$. Then the warped features are fed into convolutional layers to produce the intermediate feature $\boldsymbol{F}^j_t$. The specific process is given by

$$\boldsymbol{D} = sigmoid(Convs([\widetilde{\boldsymbol{F}}^j_0, \widetilde{\boldsymbol{F}}^j_1])) \ , \tag{1}$$

$$\boldsymbol{F}^j_t = \boldsymbol{D} \odot \widetilde{\boldsymbol{F}}^j_0 + (1 - \boldsymbol{D}) \odot \widetilde{\boldsymbol{F}}^j_1 \ , \tag{2}$$

where $Convs$ denotes convolutional layers, $[\,]$ denotes concatenation in the channel dimension, $\boldsymbol{D}$ is the generated mask for blending $\widetilde{\boldsymbol{F}}^j_0$ and $\widetilde{\boldsymbol{F}}^j_1$, and $sigmoid$ is used to ensure the mask in the range of [0, 1].

Afterwards, we compute the local correlation volumes to model the relationships among pixels in $\boldsymbol{F}^j_t$ and $\boldsymbol{F}^j_0$, F^j_1. Specifically, for each pixel $x = (u, v)$ in $\boldsymbol{F}^j_t$, we map it to its estimated correspondence in $\boldsymbol{F}^j_0$ (here we take refining $\hat{\boldsymbol{O}}^j_{t\to0}$ for example): $x^{'} = (u + \hat{\boldsymbol{O}}^{j-1}_{t\to0}(u), v + \hat{\boldsymbol{O}}^{j-1}_{t\to0}(v))$. Then a local window around $x^{'}$ is defined as

$$\mathcal{N}_{x'} = \{(u + d_u, v + dv) \mid d_u, d_v \in \{-r, ..., r\}\} \ , \tag{3}$$

where r is the radius and is set to 1 in our experiments. Then we calculate the cosine similarity between x and pixels in $\mathcal{N}_{x'}$, producing a correlation vector. The correlation map is then processed by two convolutional layers. Meanwhile, the flow $\boldsymbol{O}^{j-1}_{t\to0}$ is also processed with other two convolutional layers. At last, the correlation feature, the flow feature, $\boldsymbol{F}_0$ and $\boldsymbol{F}_t$ are concatenated and fed into four convolutional layers to produce the flow residual $\Delta\boldsymbol{O}^j_{t\to0}$, the refined flow is obtained as

$$\boldsymbol{O}^j_{t\to0} = \Delta\boldsymbol{O}^j_{t\to0} + \boldsymbol{O}^{j-1}_{t\to0} \ . \tag{4}$$

Note that all the operations are similar for $\boldsymbol{O}^j_{t\to1}$.

2. More Ablation Studies

Influence of the TFL number. We also investigate the influence of the TFL number, the ablation results on the Vimeo90K [9] testing set are shown in Table 1, we set the numbers of the TFLs in each TFB to 2, 4, 6, and 8, respectively. It is observed that the more TFLs, the higher PSNR/SSIM. To balance the performance and the computational cost, we set the number to 6 in our experiments.

TFL number	PSNR/SSIM
2	36.35/0.9810
4	36.43/0.9814
6	36.49/0.9815
8	36.52/0.9817

Table 1. Ablation study on the TFL number.

Effect of CSWA. To have a thorough investigation on the Vimeo90K testing set, we split it into 4 motion levels according to the average flow values estimated by PWC-Net [8]. The PSNR gain of CSWA becomes larger as motions become larger as shown in Table 2, which further validates the effectiveness of CSWA.

flow range	avg. flow	WA	CSWA	gain
$[0, 3)$	1.9	36.51	36.52	0.01
$[3, 6)$	4.0	36.52	36.55	0.03
$[6, 10)$	7.4	36.91	36.94	0.03
$[10, +\infty)$	14.02	34.00	34.04	0.04

Table 2. PSNR of WA and CSWA under 4 motion levels.

3. More Quantitative Comparisons

Methods	Runtime (ms)	Param. (M)	Vimeo90K
SoftSplat-$\mathcal{L}$ [4]	-	-	36.10/0.9700
CDFI [1]	60	5.0	35.17/0.9640
XVFI [7]	96	5.5	35.07/0.9760
BMBC [5]	478	11.0	35.01/0.9764
RIFE-Large [2]	13	21.7	36.10/0.9801
ABME [6]	158	18.1	36.18/0.9805
Ours	365	24.1	36.50/0.9816
Ours-Small	213	17.0	36.38/0.9811

Table 3. More quantitative comparisons. The running time is tested on images with 448×256 resolution on an NVIDIA TITAN V GPU.

We compare the running time and network parameters with some recent SOTA methods in Table 3. As mentioned in the *Limitation* section of the main paper, the computational cost of our model is still heavier than CNN-based methods. To increase the practicability of our model, we further train a light-weight version, which is denoted as 'Ours-Small'. It adopts a simpler flow estimator architecture [2] and its window size and channel number are 4×4 and 136. As shown in Table 3, the light-weight version has moderate running time and parameters yet still yields the SOTA result.

4. More Visual Results

More visual results are shown in Fig. 2 and Fig. 3. We compare our proposed method and other recent state-of-the-art methods, including AdaCoF [3], RIFE-Large [2] and ABME [6]. It can be observed that our method restores more appealing results with sharper structures.

References

[1] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. Cdfi: Compression-driven network design for frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8001–8011, 2021. 2

[2] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2020. 1, 2

[3] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5316–5325, 2020. 2

[4] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020. 2

[5] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 109–125. Springer, 2020. 2

[6] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14539–14548, 2021. 2

[7] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: Extreme video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14489–14498, 2021. 2

[8] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 2

[9] 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. 1, 3, 4

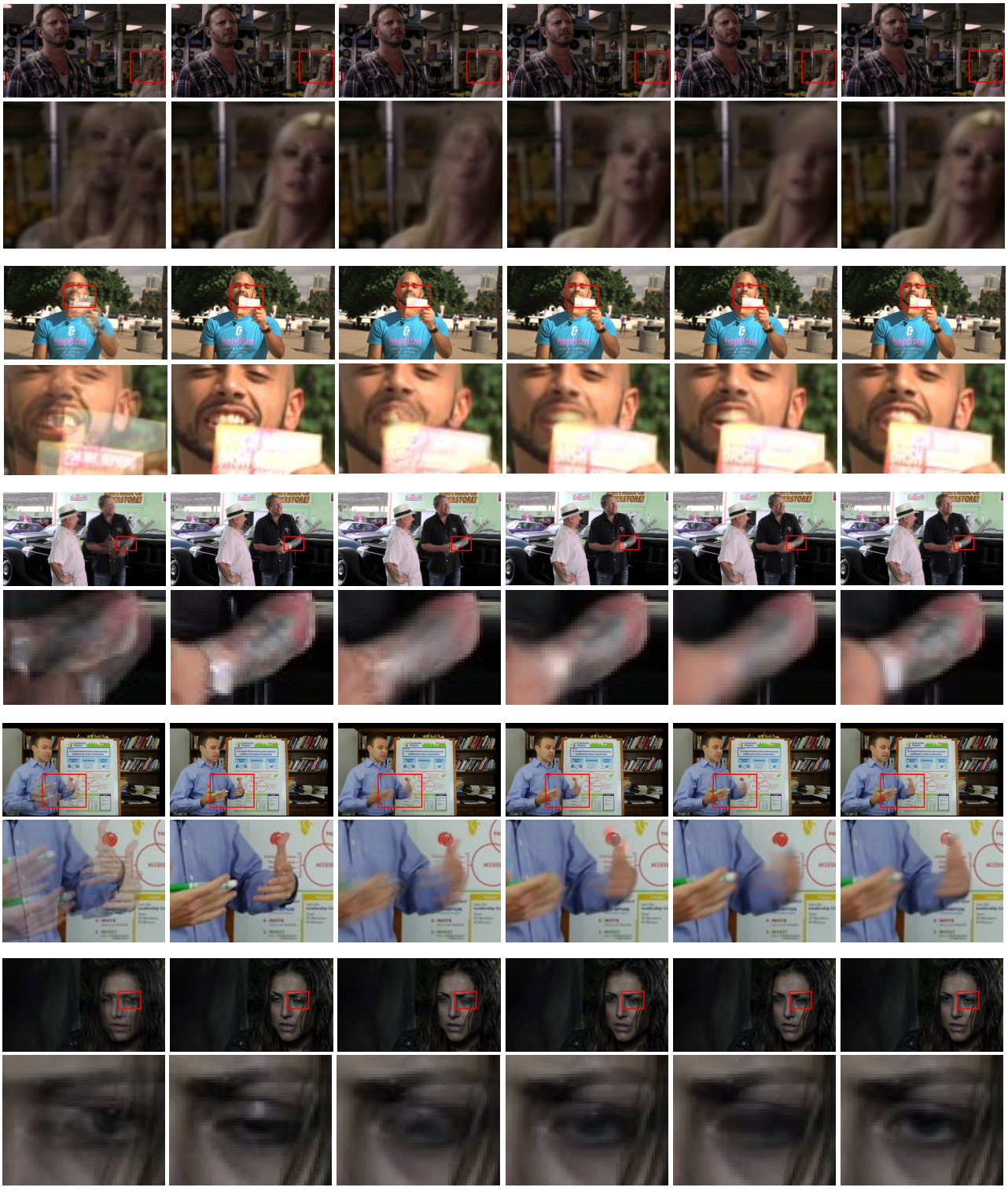

Figure 2. Visual comparison among different VFI methods on the Vimeo90K [9] testing set.

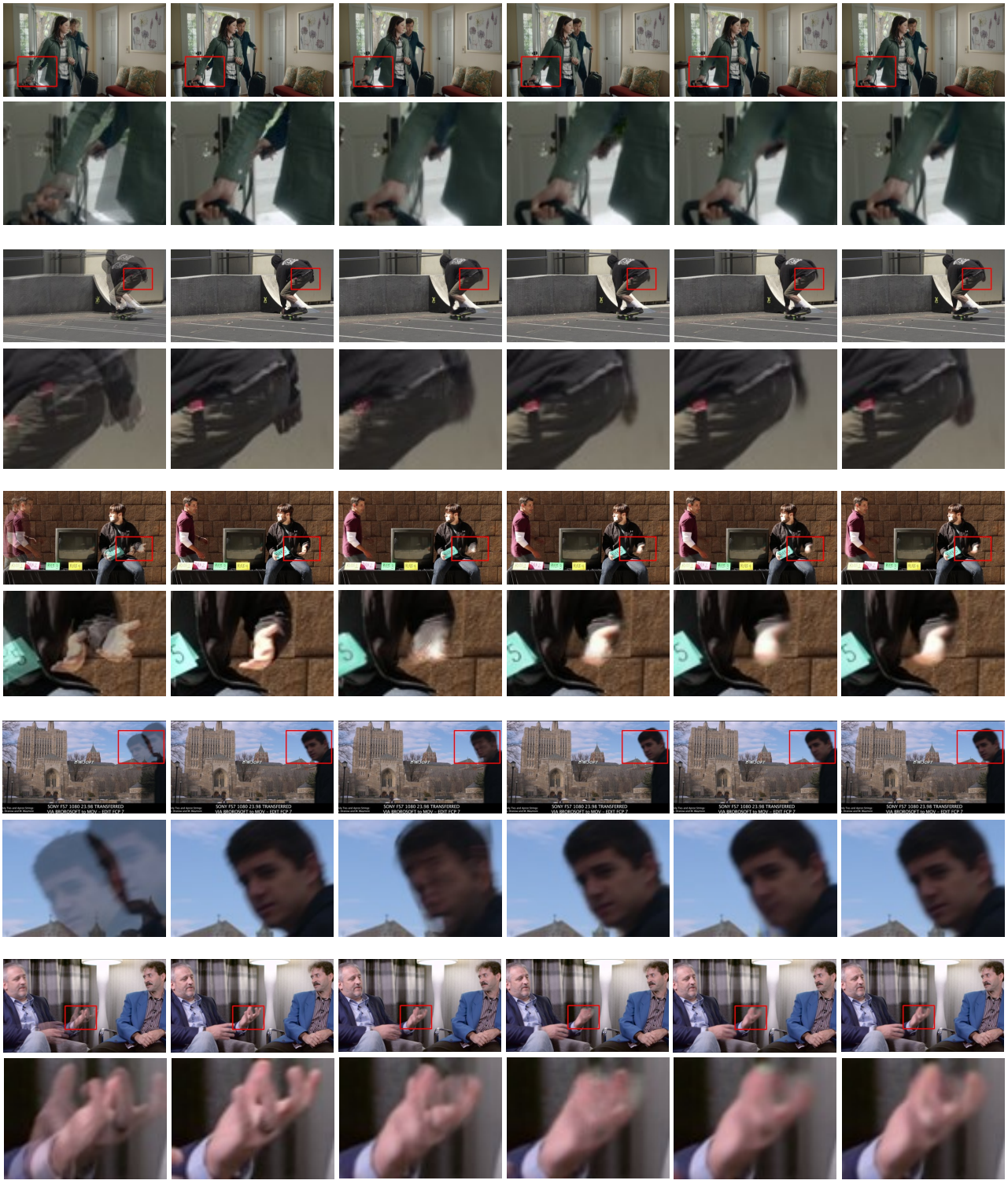

Figure 3. Visual comparison among different VFI methods on the Vimeo90K [9] testing set.