Neighbor Correspondence Matching for Flow-based Video Frame Synthesis

Zhaoyang Jia∗
jzy_ustc@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Yan Lu
yanlu@microsoft.com
Microsoft Research China
Beijing, China

Houqiang Li
lihq@ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Figure 1: A difficult example of video frame interpolation in X4K1000FPS[33]. From top to bottom are the synthesised results, the zoomed details, and the residuals between synthesised images and ground truth. Previous methods fail on the car with large motion and produce blurs or flickers, while our method can generate high-quality results.

ABSTRACT

Video frame synthesis, which consists of interpolation and extrapolation, is an essential video processing technique that can be applied to various scenarios. However, most existing methods cannot handle small objects or large motion well, especially in high-resolution videos such as 4K videos. To eliminate such limitations, we introduce a neighbor correspondence matching (NCM) algorithm for flow-based frame synthesis. Since the current frame is not available in video frame synthesis, NCM is performed in a current-frame-agnostic fashion to establish multi-scale correspondences in the spatial-temporal neighborhoods of each pixel. Based on the powerful motion representation capability of NCM, we further propose to estimate intermediate flows for frame synthesis in a heterogeneous coarse-to-fine scheme. Specifically, the coarse-scale module is designed to leverage neighbor correspondences to capture large motion, while the fine-scale module is more computationally efficient to speed up the estimation process. Both modules are trained progressively to eliminate the resolution gap between training dataset and real-world videos. Experimental results show that NCM achieves state-of-the-art performance on several benchmarks. In addition, NCM can be applied to various practical scenarios such as video compression to achieve better performance.

CCS CONCEPTS

• Computing methodologies → Motion capture; Matching; Image compression.

KEYWORDS

video frame synthesis; correspondence matching

ACM Reference Format:
Zhaoyang Jia, Yan Lu, and Houqiang Li. 2022. Neighbor Correspondence Matching for Flow-based Video Frame Synthesis. In Proceedings of the 30th ACM International Conference on Multimedia (MM ’22), October 10–14, 2022, Lisboa, Portugal. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548163

1 INTRODUCTION

Video frame synthesis is a classic video processing task to generate frame in-between (interpolation) or subsequent to (extrapolation) reference frames. It can be applied to many practice applications,
with estimated optical flows. Many flow-based interpolation schemes predict the current frame by warping reference frames matched neighbor correspondence matrix can effectively model long-term correlations in multimedia tasks like video object segmentation [5, 42] and optical flow estimation [13, 37]. In these scenarios, by matching pixels of the current frame in the reference frame, a correspondence matrix can be established to guide the generation of mask or flow (Fig.2, left). However, in video frame synthesis, we only have two inference frames and the current frame is not available. As a result, the correspondence matching cannot be performed directly. So how to perform correspondence matching in video frame synthesis is still an unanswered question.

In this paper, we introduce a neighbor correspondence matching (NCM) algorithm to enhance flow estimation in video frame synthesis, which can establish correspondences in a current-frame-agnostic fashion. Observing that objects usually move continuously and locally within a small region in natural videos, we propose to perform correspondence matching between the spatial-temporal neighbors of each pixel. Specifically, for each pixel in the current frame, we only use pixels in the local windows of adjacent reference frames to calculate the correspondence matrix (Fig.2, middle and right), so the current frame is not required in this process. The matched neighbor correspondence matrix can effectively model the object correlations, from which we can infer sufficient motion cues to guide the generation of flows. In addition, multi-scale neighbor correspondence matching is further preformed to extend the receptive field and capture large motion.

Compared with previous cost-volume-based schemes [27, 36, 37] that guide the matching region by estimated flow, NCM works better on video frame synthesis. Due to the complexity of frame synthesis, the estimated flows are usually inaccurate at the beginning of estimation. In this case, if the matching region is guided by the inaccurate flow, the matching region may move away from where the current pixel is, leading to ineffective matching. On the contrary, NCM directly matches correspondences in fixed neighbor regions to avoid misleading of inaccurate flows, which is more stable and more efficient since the correspondences only need to be computed once to be applied in different stages of estimation.

Based on NCM, we further propose a unified video frame synthesis network for both interpolation and extrapolation. The proposed model can accurately estimate intermediate flows in a heterogeneous coarse-to-fine scheme. Specifically, the coarse-scale module is designed to utilize multi-scale neighbor correspondence matrix to capture accurate motion, while the fine-scale module refines the coarse flows in a computationally efficient fashion. With the proposed heterogeneous coarse-to-fine structure, our model is not only effective but also efficient, especially for high-resolution videos.

For flow-based video frame synthesis schemes, another existing problem is the resolution gap between training dataset and real-world high-resolution videos. To eliminate such gap, we propose to train coarse and fine-scale modules using a progressive training strategy. Combining all above designs, we can augment RIFE [13] framework to a novel NCM-based network, which demonstrates state-of-the-art results in several video frame synthesis benchmarks. Specifically, In challenging X4F1000FPS benchmark, our model improves PSNR by 1.47dB (from 30.16dB of ABME [28] to 31.63dB), which shows its capability in capturing large motion and handling real-scenario videos.

NCM can be extended to many practical applications such as video compression, motion effects generation, and jitters removal in real-time communication. We build a video compression algorithm based on NCM, and results show out model can save 10% bits in HEVC dataset compared with H265-HM [10].

In summary, the main contributions in this paper are:

(1) We introduce a neighbor correspondence matching algorithm for video frame synthesis, which is simple yet effective in capturing large motion or small objects.
We propose a heterogeneous coarse-to-fine structure, which can generate intermediate flows both accurately and efficiently. We further train them in a progressive fashion to eliminate the resolution gap between training and inference.

(3) Combining all designs above, we propose a unified framework for video frame synthesis. It achieves the new state-of-the-art in both interpolation and extrapolation, which improves PSNR by 1.47dB (30.16dB→31.63dB) compared with previous SOTA ABME [28] on X4K1000FPS.

2 RELATED WORKS

2.1 Video Frame Interpolation

Video frame interpolation (VFI) is a sub-task of video frame synthesis, which aims to predict the intermediate frame between input frames. Learning-based VFI methods can be categorised as kernel-based methods [16, 26] and flow-based methods [3, 11, 12, 20, 27, 28]. Kernel-based VFI learns motion implicitly using dynamic kernels [26] and deformable kernels [16], which can preserve structural stability but might generate blurry frames because of the lack of explicit motion guidance. On the contrary, flow-based VFI explicitly model the motion with dense pixel-wise flows and perform forward-warping [24, 25] or backward-warping [11, 12, 27, 28] to predict the frame, which can achieve superior performance. Since forward-warping can cause holes and overlaps in the warped image, backward-warping is more widely exploited and applied in flow-based VFI.

For flow-based VFI that perform backward-warping, the key is how to estimate the intermediate flows. The intermediate flows should be spatially aligned with the current synthesised frame, but such spatial information is agnostic during inference. That makes it difficult to estimate accurate intermediate flows. Early flow-based VFI leverage advanced optical flow methods [13, 30, 36, 37] to estimate bi-directional flows, and perform flow reversal to generate intermediate flows. Later, Park et al. [27] estimates symmetric bilateral motion with a bilateral cost volume, which is further improved by Park et al. [28] through introducing asymmetric motion to achieve superior performance. Recently, Huang et al. [11] proposed to estimate intermediate flows directly with a privileged distillation supervision, which shows a new paradigm for intermediate flow estimation. However, these schemes cannot handle large motion of small objects well, and are limited by the solution gap between training and inference. It inspires us to explore more effective motion representations for intermediate flow estimation.

2.2 Video Frame Extrapolation

Video frame extrapolation aims to predict the frame subsequent to input frames. It is much more challenging than interpolation because unseen objects may exist in the current frame. Liu et al. [20] proposed a unified framework for both interpolation and extrapolation, which models intermediate flow as a 3D voxel flow and synthesises current frame by trilinear sampling. However, due to the difficulty of synthesis frame only by two previous frames, many following works focus more on multi-frame extrapolation or video prediction [15, 17, 38]. These works can generate more accurate results, but they usually need a sequence of frames to warm up, which are computationally expensive. In this paper, our synthesis algorithm is more like Liu et al. [20], that only needs two frames as input and can adapt to both interpolation and extrapolation.

2.3 Correspondence Matching

Correspondence matching is a technique to establish correspondences between images, which has been widely used in many computer vision and graphics tasks. In many 3D vision tasks [1, 8, 31], correspondences are computed between different views to explore the 3D structure. In video object segmentation [5, 42], correspondence matching is performed to search the similar pixels in the reference frames to propagate the mask. Benefiting from the long-term correlation modeling capability of correspondence matching, these schemes achieve remarkable performance.

Recently, correspondence is leveraged in flow estimation [27, 37] and achieve superior performance. RAFT [37] builds an all-pair correspondence matrix and looks up it to refine estimated optical flow recurrently, but it cannot be effectively applied in video frame synthesis because the current frame is not available to compute correspondences. BMVC [27] establishes a bilateral cost volume in video frame interpolation, but it is limited by the symmetric linear motion assumption. In addition, these schemes introduce correspondence as a means to refine estimated flows, which can be easily misled if inaccurate flows are given. In this paper, we rethink the correspondence matching in flow estimation. Based on the assumption that objects usually move continuously and locally in natural videos, we introduce neighbor correspondence matching as a new manner for motion correlation matching.

3 METHODS

The overview of the proposed video frame synthesis network is shown in Fig.3. The network consists of three parts: 1) neighbor correspondence matching with a feature pyramid (yellow in Fig.3), 2) heterogeneous coarse-to-fine motion estimation (blue in Fig.3), and 3) frame synthesis. For completeness, we first briefly introduce RIFE [11] from which we adopt some block designs, and then demonstrate details of each module in this section.

3.1 Background

We base our network design on the RIFE [11] framework. In RIFE, given a pair of reference frames $I_0, I_1 \in \mathbb{R}^{3xHxW}$, three IBlocks (Fig.5, left) are used to estimate intermediate flows from the coarse to fine-scale. The flows $F = (F_{t\rightarrow0}, F_{t\rightarrow1})$ and fusion map $M$ are refined by residual estimation in each IFBlock, and the current image at time $t$ can be generated by:

$$I_t = M \odot warp(I_0, F_{t\rightarrow0}) + (1-M) \odot warp(I_1, F_{t\rightarrow1})$$

(1)

where $\odot$ denotes pixel-wise product and $warp$ means backward warping operation. Then $I_t, F$ and $M$ are fed to a U-Net-like refine network (i.e., the synthesis network) to generate the synthesised frame $\hat{I}_t$.

RIFE is light weight and real-time, but the synthesis quality is not satisfactory due to the limited receptive field and motion capture capability of the designed fully-convolutional network. In addition, RIFE cannot adapt to high-resolution videos where the motion is even larger. To eliminate these limitations, we propose
where the estimated flows point. The flow-centric manners have volume-based manners [11, 27] that establish a cost volume around videos, the core idea of NCM is to explore the motion information by the correspondences in \( d \) correspondence matching to generate motion feature.

3.2 Neighbor Correspondence Matching

3.2.1 Overview. Based on the observation that an object usually move continuously and locally within a small region in natural videos, the core idea of NCM is to explore the motion information by establishing spatial-temporal correlation between the neighboring regions. In detail, we compute the correspondences between the local windows of two adjacent reference frames for each pixel, as shown in Fig.2. It means we need no information about the current frame, so the matching can be performed in a current-frame-agnostic fashion to meet the need of frame synthesis.

It is worth noting that the position of local windows are determined by the position of pixel, which is different from the cost-volume-based manners [11, 27] that establish a cost volume around where the estimated flows point. The flow-centric manners have a potential problem that if the estimated flow is inaccurate, the matching may be performed in a wrong region. As a result, the cost volume cannot compute effective correlations to refine the estimated flows. On the contrary, NCM is pixel-centric and will not be misled by the inaccurate flows. Experiments also show that compared with flow-guided matching, NCM can lead to better performance and stability.

3.2.2 Mathematical Formulation. Given a pair of reference frames \( I_0, I_1 \in \mathbb{R}^{3xH\times W} \), a n-layer feature pyramid \( f^l_i \in \mathbb{R}^{C_l \times H_l \times W_l} \), \( i \in \{0, 1\} \) is first extracted with several residual blocks[9], where \( l \in \{1, \ldots, n\} \) denotes different layers and \( C_l, H_l, W_l \) are the channel number, height and width of the feature from the l-th layer. The first \( n-1 \) features only serve as the image features for subsequent modules, while feature from the deepest layer \( f^n \) is used for correspondence matching to generate motion feature.

For a pixel at spatial position \((i, j)\), we perform NCM to compute the correspondences in \( d \times d \) windows:

\[
corr^0(i, j) = \{f^0_0(i + \delta_{l,1}, j + \delta_{l,1}) \cdot f^n_1(i + \delta_{l,1}, j + \delta_{l,1})\}_{\delta_{l,1}} \tag{2}
\]

where \( \delta_{l,1} \in \{-d/2, -d/2 + 1, \ldots, d/2\} \) denote different location pairs in the window, and \( \cdot \) denotes the channel-wise dot production. The computed correspondence matrix \( corr^0 \in \mathbb{R}^{d\times W} \) contains correlations of all pairs in the neighborhoods, which can be further leveraged to extract motion information.

To enlarge the receptive field and capture large motion, we further perform multi-scale correspondence matching. As shown in Fig.4, we first downsample \( f^n \) to \( s = 1/2^k \) resolution to generate multi-scale features \( f^k, k \in \{0, 1, \ldots, K\} \). For each level \( k \), the correspondences can be computed by:

\[
corr^k(i, j) = \{f^0_0(i_k + \delta_{l,1}, j_k + \delta_{l,1}) \cdot f^n_1(i_k + \delta_{l,1}, j_k + \delta_{l,1})\}_{\delta_{l,1}} \tag{3}
\]

where \( (i_k, j_k) = (i/2^k, j/2^k) \) is the position of pixel in the down-sampled feature map. We use bilinear interpolation for non-integer position. And the final multi-scale neighbor correspondences can be generated by simply concatenating correspondences at different levels:

\[
corr = corr^0 | corr^1 | \cdots | corr^K \tag{4}
\]

where \( | \) denotes channel concatenation. In this paper, we extract \( n = 4 \) layers feature pyramid with 1, 1/2, 1/4, 1/8 resolutions of input frames, and perform NCM in 4 scales (\( K = 3 \)) with window size \( d = 3 \).
For the fine-scale module, we directly adopt original IFBlocks to be more computationally efficient. Two IFBlocks receive the fine-scale flows as input, and refine it only using the high-resolution frames. Finally, the estimated intermediate flows are fed into the synthesis network to generate the current frame \( I_t \) as output.

The estimation resolution in each IFBlock can be controlled flexibly by the size \((h, w)\) and the downsample factor \(K_{c,f}\) to adapt to the resolution of the input video. Assume \(H < W\), we use parameter \(a\) to control the size by \((h, w) = (a, W/H \times a)\). In the fine-scale module, we set \(K_f = (2, 1)\) if \(a/H < 1/2\) otherwise set \(K_f = (1, 1)\). We set \(K_c = (4, 2, 1)\) in the coarse-scale module.

### 3.4 Progressive Learning

Many existing frame synthesis schemes cannot be well extended to applications due to the resolution gap between training and inference. That is, the training data is low-resolution (e.g., 256 \( \times \) 256) but the resolution of real-world high-resolution images may be much higher (e.g., 1080p or 4K). To address this problem, we design a progressive learning scheme for the proposed network. The basic idea is to separate the end-to-end training into two stages to simulate the inference on high-resolution videos:

- In stage I, only the coarse-scale module is trained on low-resolution 256 \( \times \) 256 frames. It can be regarded as training on the low-resolution version of real-world high-resolution images.
- In stage II, the coarse-scale module is fixed, and the fine-scale module is trained to refine the coarse-scale flows to high-resolution. We randomly downsample images to 64 \( \times \) 64, 128 \( \times \) 128 or keep 256 \( \times \) 256 in each mini-batch to generate pseudo low-high resolution data pairs for training.

In inference, the coarse-module can estimate accurate low-resolution flows with stage I, and the fine-scale module can refine such flows to high-resolution with stage II. As a result, our model can be effectively adapted to high-resolution videos.

### 3.5 Loss function

Following RIFE, we adopt a self-supervised privileged distillation scheme to supervise the estimated flows directly. In detail, an additional teacher IFBlock is stacked to refine the estimated flows using the current frame \( I_t \) as input, and the generated \( F^{Tea} \) and \( M^{Tea} \) can supervise the intermediate flows with a distillation loss:

\[
L_{dis} = \sum_{i \in \{0,1\}} \|F_t - i - F^{Tea}_t\|_1
\]

which is applied over all estimated flows from each IFBlock. The gradient of the distillation loss is stopped for the teacher module.

The overall training loss consists of the reconstruction loss of the student \( L_{rec} \), the teacher \( L^{Tea}_{rec} \) and the privileged distillation loss \( L_{dis} \):

\[
L = L_{rec} + L^{Tea}_{rec} + \lambda_d L_{dis}
\]

where the reconstruction loss is defined as the \( L_1 \) loss between the Laplacian pyramid representations of the synthesized frame and the ground truth. \( \lambda_d \) is set to 0.01 by default.
We evaluate our scheme on various benchmarks to verify its performance and generalization ability. On each dataset, we measure the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) for quantitative evaluation.

** X4K1000FPS [33]: It is a high-quality dataset of 4K videos, which is challenging due to the high resolution, occlusion, large motion and the scene diversity. The provided X-TEST set supports 8× interpolation evaluation on two frames with a temporal distance of 32 frames, and we also use it to evaluate 2× extrapolation by synthesising the 32–nd frame using the 0–th and the 16-th.

** SNU-FILM [6]: It contains 1,240 triplets of 240fps videos with resolution from 640×368 to 1280×720. Four settings - Easy, Medium, Hard, and Extreme - are provided to evaluate from small motion to large motion, and the temporal distance of each setting increases from 2 (120fps → 240fps) to 16 (15fps → 30fps).

**UCF101** [34]: Liu et al. [20] selected 379 triplets from UCF101 with weight decay

**Vimeo90K** [40]: The test set of Vimeo90K contains 3,782 triplets with resolution of 448 × 256. We evaluate on it to verify the robustness of our model on low-resolution videos.

**Different downsample size (h, w) is set to adapt to the resolution of different benchmark. We set a = 384 for X4K1000FPS and a = 256 for other benchmarks (the definition of a is in Section 3.4).**

### 4 EXPERIMENTS

#### 4.1 Experimental Setup

**Training Data.** We use the Vimeo-90k [40] training split, which has 51,312 triplets with a resolution of 448 × 256. We augment the dataset by randomly flipping, temporal order revering, and cropping 256 × 256 patches.

**Training Strategy.** We use AdamW [21] to optimize our model with weight decay $10^{-3}$. The learning rate is gradually reduced from $3 \times 10^{-4}$ to $3 \times 10^{-5}$ using cosine annealing for each stage in progressive learning. The batch size is set to 64, and we use 4 Telsa V100 GPU to train the coarse-scale module for 230k iterations in stage I and train the other parts for 76k iterations in stage II. It takes about 40 hours for training in total.

#### 4.2 Benchmarks

We evaluate our scheme on various benchmarks to verify its performance and generalization ability. On each dataset, we measure the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) for quantitative evaluation.

**X4K1000FPS** [33]: It is a high-quality dataset of 4K videos, which is challenging due to the high resolution, occlusion, large motion and the scene diversity. The provided X-TEST set supports 8× interpolation evaluation on two frames with a temporal distance of 32 frames, and we also use it to evaluate 2× extrapolation by synthesising the 32–nd frame using the 0–th and the 16-th.

**SNU-FILM** [6]: It contains 1,240 triplets of 240fps videos with resolution from 640×368 to 1280×720. Four settings - Easy, Medium, Hard, and Extreme - are provided to evaluate from small motion to large motion, and the temporal distance of each setting increases from 2 (120fps → 240fps) to 16 (15fps → 30fps).

**UCF101** [34]: Liu et al. [20] selected 379 triplets from UCF101 human actions dataset for video frame synthesis evaluation. However, there are some dirty data in the test set. We will demonstrate it in detail in the supplementary material.

**Vimeo90K** [40]: The test set of Vimeo90K contains 3,782 triplets with resolution of 448 × 256. We evaluate on it to verify the robustness of our model on low-resolution videos.

Different downsample size $(h, w)$ is set to adapt to the resolution of different benchmark. We set $a = 384$ for X4K1000FPS and $a = 256$ for other benchmarks (the definition of $a$ is in Section 3.4).

### 4.3 Comparison with Previous Methods

For video frame interpolation, we compare the proposed scheme with previous methods: DAIN [2], AdaCoF [16], XVFI [33], SoftSplat [25], BMBC [27], ABME [28] and RIFE [11]. For extrapolation, we compare with DVF [20], and re-implement RIFE on the extrapolation task for comparison.

Following RIFE [11], we also introduce a Large version of our model to meet the need of different scenarios with different computation cost. Two modification are performed: 1) test-time augmentation, to inference twice with the original input frames and the flipped frames then average the results, and 2) model scaling, to double the resolution of feature map in each IFBlock and the synthesis network by removing the first stride. More information can be found in the supplementary material.
we find that RIFE-Large shows much worse improvement of 0.23dB (base model) and 2.08dB (large model) in terms of PSNR. And we find that RIFE-Large shows better performance in all benchmarks except SNU-FILM Easy and Medium setting, leading to an average performance on 256p videos. It shows the efficiency to design different motion features. The image features mainly improves performance on low-resolution videos, and the matching-based motion features can enhance performance on both high-resolution (0.65dB) and low-resolution (0.12dB).

4.4.1 Add-on Study. To understand the effectiveness of the proposed components, we perform an ablation study to add each component on a baseline in Table3. The baseline (last line in the table) consists of five IFBlocks, and between each two blocks the upsampling rate is set to 2 in both training and inference.

**Progressive learning.** It makes the model adapt to high resolution 4K videos (0.93 dB improvement on X4K1000FPS). However, the performance on 256p videos slightly drops. This is because the baseline is only trained for low-resolution 256p videos, while progressive learning is designed to refine high-resolution videos on low-resolution 256p videos. So if we resize 256p videos to 480p, such limitation is eliminated and achieves 0.14dB improvement.

**Neighbor correspondence matching** comprises of a feature pyramid to extract image features and a matching process to extract motion features. The image features mainly improves performance on low-resolution videos, and the matching-based motion features can enhance performance on both high-resolution (0.65dB) and low-resolution (0.12dB).

**4.4.2 Coarse-to-fine and the matching region.** We also perform an ablation study on the coarse-to-fine structure and the matching region in NCM in Table4.

We also report the quantitative results of extrapolation in Table2, and the visual comparison is shown in Fig.6. Compared with RIFE, our model achieves better performance in all benchmarks except for SNU-FILM Easy and Medium setting, leading to an average improvement of 0.23dB (base model) and 2.08dB (large model) in terms of PSNR. And we find that RIFE-Large shows much worse performance than RIFE in high-resolution videos (e.g., 4.49dB on X4K1000FPS and 2.02dB drop on SNU-FILM), which is also observed on the extrapolation benchmark where RIFE-Large shows 0.2dB drop in X4K1000FPS. It is because when doubling feature resolution, the receptive field of the network is halved. However, our model is not affected much by the resolution of feature map because the receptive field is guaranteed by NCM instead of convolution layers. It means our scheme can be better extended to the large version for better performance.

### Table 3: Add on study of progressive learning and NCM. PSNR/SSIM and the runtime(ms) are reported.

| Setting                | X4K1000FPS     | Vimeo90K      | Runtime @1080p |
|------------------------|----------------|---------------|-----------------|
| w/ normal coarse-to-fine | 31.35/0.9151   | 35.93/0.9797  | 244             |
| w/ flow-guided matching | 31.56/0.9180   | 35.83/0.9793  | 92              |
| Ours-base              | 31.63/0.9185   | 35.88/0.9795  | 82              |
| (Heterogeneous coarse-to-fine, neighbor matching) |                |               |                 |

**Table 4: Ablation study on coarse-to-fine manner and matching region. PSNR/SSIM and runtime(ms) are reported.**

| Setting          | X4K1000FPS       | Vimeo90K      | Runtime @1080p |
|------------------|------------------|---------------|-----------------|
| ✓✓✓              | 31.63/0.9185     | 35.88/0.9795  | 82              |
| ✓✓x              | 30.98/0.9087     | 35.76/0.9788  | 69              |
| ✓x               | 30.93/0.9088     | 35.36/0.9773  | 63              |
| x                | 30.00/0.8937     | 35.53/0.9779  | 63              |
flow-guided matching is more likely to cause collapse in training. It indicates the flow-guided matching is less stable since the matching region is influenced by estimated flows.

5 APPLICATION: VIDEO COMPRESSION

The proposed scheme can be applied to various scenarios due to the powerful capability in capture large motion in high-resolution videos. For example, it can be applied in video compression, jitters removal in real-time communication system, and motion effects generation. In this section, we use video compression as an example to show its potential. Experiments and demos about other applications are detailed in the supplementary material.

Most video compression schemes [18, 22, 32, 41] adopt a motion estimation and motion compensation (MEMC) paradigm. They predict the current frame using the optical flow, and then compress the prediction residual to reconstruct the compressed frame. Many works [18, 32, 41] focus on improving the residual compression process, but the temporal redundancy in the optical flow is not well considered. We can eliminate such redundancy partly by using the proposed module for motion prediction. We design a scalable bi-directional video compression model using NCM. It can serve as a plugin-in on any uni-direction codec (e.g., P-frame codec [10, 32]) to compress B-frame, and the whole video can be compressed in order of I-B-P-B-... . The model supports a bit-free mode for low-cost compression and a bit-need mode for high-quality compression.

The bit-free mode simply adopts our video frame synthesis model to interpolate the B-frame. It means the codec does not need to encode and decode code streams of B-frame, and this frame can be interpolated using much less cost. We evaluate it on the state-of-the-art learned P-frame codec TCM [32], and the rate-distortion curve on HEVC test videos [35] is shown in Fig.7. Our model can save 10.4% bits (0.065bpp → 0.058bpp) under PSNR=32.0dB and save 17.2% bits (0.063bpp → 0.052bpp) under MS-SSIM=0.9650. In addition, our model can inference about 9 times faster in B-frames, from 500ms encoding and 252ms decoding time of TCM to 0ms encoding and 82ms interpolation time. It demonstrates the efficiency of bit-free mode in low-cost scenarios such as video communication.

The bit-need mode adopts our video frame synthesis model as a motion prediction module, and compress the residual motion instead of the entire motion. Experiments show it can save 17.5% bits (from 0.065bpp to 0.053bpp) under PSNR=32.0dB on TCM. We also compare it with H.265-HM [10] under the same sequence order in Fig.8, and results show 10% bits saving (from 0.111bpp to 0.100bpp) on the whole sequence under PSNR=34.0dB. It shows our model can serve as a superior motion prediction module for video compression. More details of experiment settings and experiment results can be found in the supplementary material.

6 LIMITATIONS

Even the proposed neighbor correspondence matching algorithm can capture large motion for high-resolution videos, it may estimate inaccurate flows on similar details in the video. Because the matching is guided by feature similarity, neighbor regions with similar appearance cannot be distinguished by matching. As shown in Fig.9, our model fails on estimating repeated stripes on the house. It also limits NCM to be used in fine-scale module in 4K videos because in such scale the neighbor region is usually full of similar regions. We hope to solve it in the future by distinguishing regions with similar appearance using techniques like positional embedding.

7 CONCLUSION

In this paper, we propose a neighbor correspondence matching algorithm for video frame synthesis, which can capture large motion even in high-resolution videos. With the proposed heterogeneous coarse-to-fine structure design and the progressive learning, our model is both effective and efficient and can adapt to high-resolution videos. Experiments show the superiority of our model in both interpolation and extrapolation. In addition, our model can be used in many applications, and we use video compression as an example to show its potential. We hope it can be extended to more applications in the future.
REFERENCES

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. 2011. Buildingrome in a day. Commun. ACM 54, 10 (2011). 105–112.

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019. Depth-aware video frame interpolation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 3703–3712.

[3] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. IEEE transactions on pattern analysis and machine intelligence 43, 3 (2019), 933–948.

[4] Tim Brooks and Jonathan T Barron. 2019. Learning to synthesize motion blur. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 6840–6848.

[5] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. 2021. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. Advances in Neural Information Processing Systems 34 (2021).

[6] Myungsuh Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyong Mu Lee. 2020. Channel attention is all you need for video frame interpolation. In Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. 10663–10671.

[7] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world’s imagery. In Proceedings of the IEEE conference on computer vision and pattern recognition. 5515–5524.

[8] Richard Hartley and Andrew Zisserman. 2003. Multiple view geometry in computer vision. Cambridge university press.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[10] HM-16.20. 2022. https://vcgid.hhi.fraunhofer.de/jvet/HM/-/tags/HM-16.20. Accessed: 4/4/2022.

[11] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2021. HM-16.20. 2022. https://vcgid.hhi.fraunhofer.de/jvet/HM/-/tags/HM-16.24. Accessed: 4/4/2022.

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

[13] Huainan Jiang, Dengqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 9000–9008.

[14] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. 2021. Learning to estimate hidden motions with global motion aggregation. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 9000–9008.

[15] Nima Khademi Kalantari, Ting-Chan Wang, and Raviv Ramsamoorthi. 2016. Learning-based view synthesis for light field cameras. ACM Transactions on Graphics (TOG) 35, 6 (2016), 1–10.

[16] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. 2021. CCVS: Context-aware Controllable Video Synthesis. Advances in Neural Information Processing Systems 34 (2021).

[17] Hyoengjun Lee, Taeho Kim, Tae-young Chung, Daeyoun Pak, Yuseok Ban, and Sangyoun Lee. 2020. Addaf: Adaptive collaboration of flows for video frame interpolation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 5316–5325.

[18] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. 2021. Temporal Context Mining for Learned Video Compression. arXiv preprint arXiv:2111.13850 (2021).

[19] Xuefan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Learning for optical flow using pyramid, warping, and cost volume. In Proceedings of the IEEE conference on computer vision and pattern recognition. 8934–8943.

[20] Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In European conference on computer vision. Springer, 402–419.

[21] Yunbo Wang, Haixu Wu, Jianjun Zhang, Zhifeng Gao, Jianmin Wang, Philip S Yu, and Minghong Long. 2021. PredRNN: A recurrent neural network for spatiotemporal predictive learning. arXiv preprint arXiv:2103.09504 (2021).

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

[23] Zongxiao Yang, Yunchao Wei, and Yi Yang. 2021. Collaborative video object segmentation by multi-scale foreground-background integration. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).