A Unified Pyramid Recurrent Network for Video Frame Interpolation

Xin Jin[1]   Longhai Wu[1]   Jie Chen[1]   Youxin Chen[1]   Jayoon Koo[2]   Cheul-hee Hahm[2]

[1]Samsung Electronics (China) R&D Center   [2]Samsung Electronics, South Korea

{xin.jin,longhai.wu,ada.chen,yx113.chen,j.goo,chhahm}@samsung.com

Abstract

Flow-guided synthesis provides a common framework for frame interpolation, where optical flow is estimated to guide the synthesis of intermediate frames between consecutive inputs. In this paper, we present UPR-Net, a novel Unified Pyramid Recurrent Network for frame interpolation. Cast in a flexible pyramid framework, UPR-Net exploits lightweight recurrent modules for both bi-directional flow estimation and intermediate frame synthesis. At each pyramid level, it leverages estimated bi-directional flow to generate forward-warped representations for frame synthesis; across pyramid levels, it enables iterative refinement for both optical flow and intermediate frame. In particular, we show that our iterative synthesis strategy can significantly improve the robustness of frame interpolation on large motion cases. Despite being extremely lightweight (1.7M parameters), our base version of UPR-Net achieves excellent performance on a large range of benchmarks. Code and trained models of our UPR-Net series are available at: https://github.com/srcn-ivl/UPR-Net.

1. Introduction

Video frame interpolation (VFI) is a classic low-level vision task. It aims to increase the frame rate of videos, by synthesizing non-existent intermediate frames between consecutive frames. VFI technique supports many practical applications including novel view synthesis [10], video compression [21], cartoon creation [32], *etc*.

Despite great potential in applications, video frame interpolation remains an unsolved problem, due to challenges like complex and large motions, occlusions, and illumination changes in real-world videos. Depending on whether or not optical flow is incorporated to compensate for inter-frame motion, existing methods can be roughly classified into two categories: flow-agnostic methods [5,6,25,28], and flow-guided synthesis [2,14,19,26,27,29,30]. With recent advances in optical flow [12,13,34,35], flow-guided synthesis has developed into a popular framework with compelling performance for video frame interpolation.
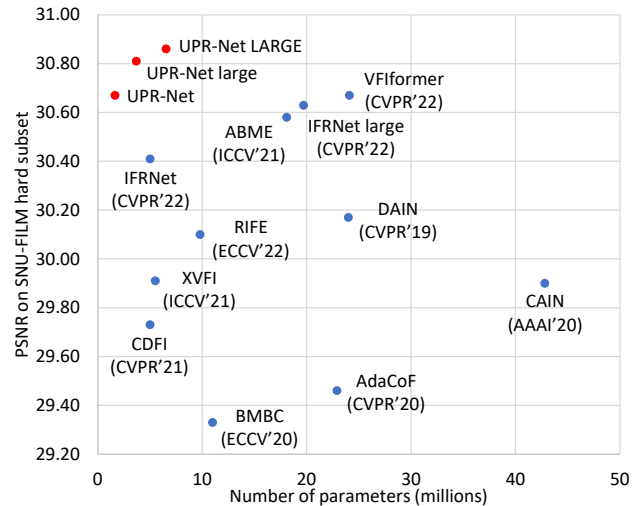

Figure 1. Comparison of performance and model size on the hard subset of SNU-FILM benchmark [6]. Our UPR-Net series achieve state-of-the-art accuracy with extremely small parameters.

Most of existing flow-guided methods follow a similar procedure: estimating optical flow for desired time step, warping input frames and their context features based on optical flow, and synthesizing intermediate frame from warped representations. Where technical choices may diverge in this procedure, is the warping operation and the optical flow it requires. Backward-warping is traditionally used for frame interpolation [2,14,19,29,30], but acquiring high-quality bilateral intermediate flow for it is often challenging. Forward-warping can directly use linearly-scaled bi-directional flow between input frames (which is easier to obtain), and thus has recently emerged as a promising direction for frame interpolation [26,27].

In common flow-guided synthesis pipeline [1,18,27,30], optical flow is typically estimated from coarse to fine by a pyramid network, but intermediate frame is synthesized just once by a synthesis network. Despite promising performance on low-resolution videos, this practice misses the opportunity of iteratively refining the interpolation for high-resolution inputs. Second, for large motion cases, an important issue has been overlooked by previous works: even

when estimated motion is visually plausible, in many cases, the obvious artifacts in warped frames (*e.g.*, large holes in forward-warped frames) may also degrade the interpolation performance. Last, existing methods typically rely on heavy model architectures to achieve good performance, blocking them from being deployed on platforms with limited resources, *e.g.*, mobile devices.

Aiming at these issues, we introduce UPR-Net, a novel Unified Pyramid Recurrent Network for frame interpolation. Within a pyramid framework, UPR-Net exploits lightweight recurrent modules for both bi-directional flow estimation and forward-warping based frame synthesis. It enables iterative refinement of both optical flow and intermediate frame across pyramid levels, producing compelling results on complex and large motion cases.

Our work draws inspirations from many existing works, but is significantly distinguished from them in three aspects. First, UPR-Net inherits the merit of recent pyramid recurrent bi-directional flow estimators [15,31], allowing to customize the number of pyramid levels in testing to estimate extremely large motions. But, it goes one step further, by exploiting pyramid recurrent network for coarse-to-fine frame synthesis, and unifying motion estimation and frame synthesis within a single pyramid recurrent network.

Second, we reveal that our coarse-to-fine iterative synthesis can significantly improve the robustness of frame interpolation on large motion cases. At high-resolution pyramid levels, forward-warped frames may suffer from obvious holes due to large motions, resulting in poor interpolation for many cases. We show that this issue can be remedied to a large extent, by feeding the frame synthesis module with the intermediate frame estimate upsampled from previous lower-resolution pyramid level.

Third, both of our optical flow and frame synthesis modules are extremely lightweight. Yet, they are still carefully integrated with the key ingredients from modern researches on optical flow [34,35] and frame synthesis [26]. Specifically, at each pyramid level, UPR-Net firstly extracts CNN features for input frames, then constructs a correlation volume for simultaneous bi-directional flow estimation. It predicts refined intermediate frame from forward-warped input frames and their CNN features, along with upsampled intermediate frame estimate.

We conduct extensive experiments to verify the effectiveness of UPR-Net for frame interpolation. Our base version of UPR-Net only has 1.7M parameters. Yet, it achieves excellent performance on both low- and high-resolution benchmarks, when trained with low-resolution data. Figure 1 gives a comparison of accuracy and model size on the hard subset of SNU-FILM [6], where our UPR-Net series achieve state-of-the-art accuracy with much fewer paprameters. In addition, we validate various design choices of UPR-Net by ablation studies.

2. Related Work

Pyramid recurrent optical flow estimator. PWC-Net [34] has been traditionally used for optical flow by frame interpolation methods [1,26,27]. However, the fixed number of pyramid levels makes it difficult to handle extremely large motions beyond the training phase. Recently, pyramid recurrent optical flow estimators [31,38] are developed to handle large motion, by sharing the structure across pyramid levels and customizing the pyramid levels in testing. A larger number of pyramid levels can better handle large motions that often appear in high-resolution videos.

Previous pyramid recurrent estimators typically employ a plain U-Net as the base estimator at each pyramid level. However, U-Net is over-simplified for optical flow due to the lack of correlation volume [34,35]. Very recently, EBME [15] incorporates correlation volume into pyramid recurrent network for simultaneous bi-directional flow estimation. We follow the basic idea in [15] for bi-directional flow, but modify it to better adapt to our unified pyramid network for frame interpolation.

Coarse-to-fine image synthesis. Coarse-to-fine processing is a mature technology for high-resolution image synthesis, where low-resolution images are firstly synthesized, and then iteratively refined until generating the desired high-resolution output. It has many successful applications, including photographic image synthesis conditioned on semantic layouts [4], adversarial image generation [7], and recent diffusion model based image synthesis [9].

However, coarse-to-fine synthesis has been largely overlooked by existing frame interpolation methods. Zhang *et al*. [38] iteratively estimate the occlusion mask within a pyramid recurrent framework, but still needs an extra refinement network to obtain the final result. XVFI [31] estimates multi-scale intermediate frames during training, but does not perform iterative refinement of intermediate frame. IFRNet [16] gradually refines the intermediate feature (rather than frame) until generating the desired output, but it is not recurrent, and has limited capacity in handling large motion. In this work, we iteratively refine the intermediate frame within a pyramid recurrent framework.

Artifacts in warped frames. Although warping can compensate for per-pixel motion, it often creates distortion and artifacts. If a pixel is moved to a new location, and no other pixels are moved to fill the old location, this pixel will appear twice in backward-warped frame [18,23], or leave a hole at original location in forward-warped frame [26].

To robustly synthesize intermediate frame from warped frames, existing frame interpolation methods typically feed the synthesis network with both warped frames and their context features [11,26,27]. The synthesis network can
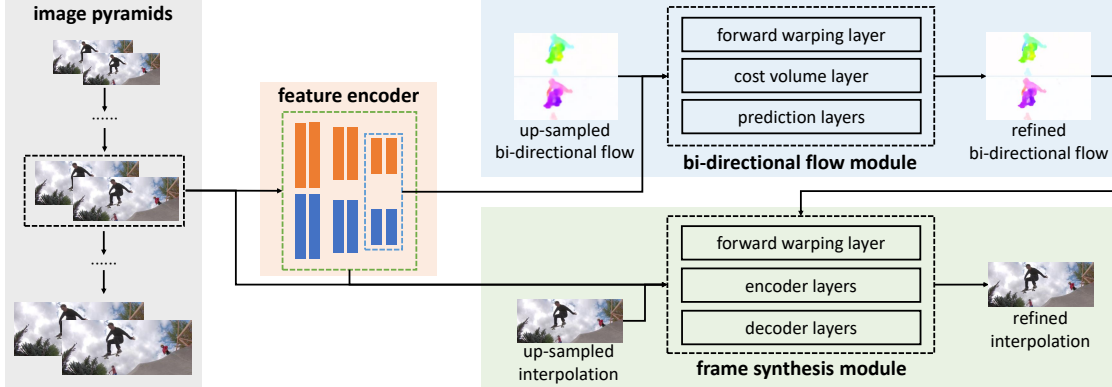
Figure 2. Overview of our UPR-Net. Given two input frames, we first construct image pyramids for them, then apply a **recurrent** structure across pyramid levels to repeatedly refine estimated bi-directional flow and intermediate frame. Our recurrent structure consists of a feature encoder that extracts multi-scale features for input frames, a bi-directional flow module that refines bi-directional flow with correlation-injected features, and a frame synthesis module that refines intermediate frame estimate with forward-warped representations.

leverage rich contextual cues to infer the intermediate frame from warped representations, even when artifacts exist in warped frames.

In this work, we observe that the synthesis network does work well for small motion cases where forward-warped frames contain slight artifacts. However, in presence of large motions, in many cases, the obvious holes in forward-warped frames may lead to artifacts in interpolation. We show that our iterative synthesis can significantly improve the robustness of frame interpolation on large motion cases.

Additionally, during forward-warping, the conflicted pixels mapped to the same target should be addressed by simple averaging or certain weighted averaging operation (*e.g.*, softmax splatting [27]). In this work, we adopt the average splatting [27] as forward-warping for simplicity.

3. Our Approach

3.1. Unified Pyramid Recurrent Network

We illustrate the overall pipeline of UPR-Net in Figure 2. It unifies bi-directional flow estimation and frame synthesis within a pyramid structure, and shares the weights across pyramid levels. This macro pyramid recurrent architecture has two advantages: (i) reducing the parameters of the full pipeline; (ii) allowing to customize the number of pyramid levels in testing to handle large motions.

Given a pair of consecutive frames I_0, I_1, and the desired time step t ($0 \leq t \leq 1$), our goal is to synthesize the non-existent intermediate frame I_t. UPR-Net tackles this task via an iterative refinement procedure across L image pyramid levels, from the top level with down-sampled frames I_0^{L-1} and I_1^{L-1}, to the bottom (zeroth) pyramid level with the original input frames I_0^0 and I_1^0.

At each pyramid level, UPR-Net employs a feature encoder to extract multi-scale CNN features for both input

frames. Then, the features from the last layer of feature encoder and the optical flow upsampled from previous level are processed by a bi-directional flow module to produce refined bi-directional flow. The refined optical flow is leveraged to forward-warp input frames and multi-scale CNN features. Combining warped representations with the interpolation upsampled from previous level, a frame synthesis module is employed to generate refined intermediate frame. This estimation process is repeated until generating the final interpolation at the bottom pyramid level.

3.2. Recurrent Frame Interpolation Modules

Our recurrent structure consists of three lightweight modules: feature encoder, bi-directional flow module, and frame synthesis module.

Feature encoder. For optical flow [34, 35], a common practice is to extract per-pixel features and construct correlation volume with features. For frame interpolation, it is also common to feed the synthesis network with warped features to provide contextual cues [11, 27]. Considering these, at each pyramid level, we firstly employ a feature encoder to extract multi-scale features for input frames.

Our feature encoder has three convolutional stages: stage-0, stage-1, and stage-2. Each stage consists of four convolutional layers, and the first layers of the stage-1 and stage-2 perform down-sampling. We use the features from each stage's last convolutional layer. Given images I_0^l and I_1^l at the l-th pyramid level, we denote the features as $\{C_0^{l,0}, C_1^{l,0}\}$, $\{C_0^{l,1}, C_1^{l,1}\}$ and $\{C_0^{l,2}, C_1^{l,2}\}$ for I_0 and I_1 extracted at stage 0, 1, 2, respectively. $\{C_0^{l,2}, C_1^{l,2}\}$ will be used for bi-directional flow estimation, and all multi-scale features will be used for context-aware frame synthesis.
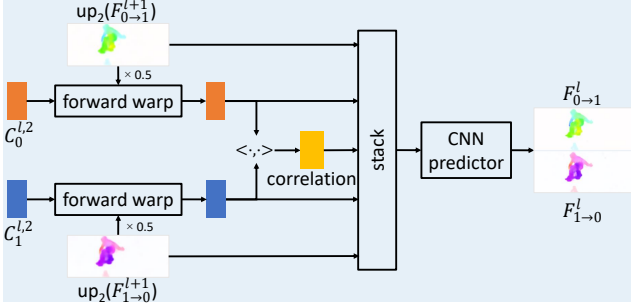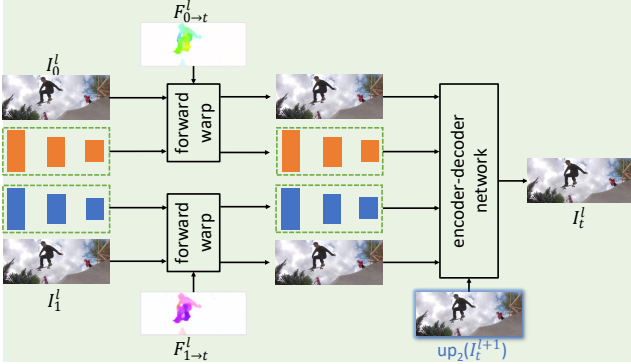
Figure 3. Our bi-directional flow module.

Figure 4. Our frame synthesis module.

Bi-directional flow module. Let $F_{0\to1}^{l+1}$ and $F_{1\to0}^{l+1}$ denote the refined bi-directional optical flow at level $l+1$. At the l-th image pyramid level, we firstly initialize the bi-directional flow by $\times2$ upsampling $F_{0\to1}^{l+1}$ and $F_{1\to0}^{l+1}$: $\hat{F}_{0\to1}^l = \text{up}_2(F_{0\to1}^{l+1})$, $\hat{F}_{1\to0}^l = \text{up}_2(F_{1\to0}^{l+1})$. In particular, the initial flow at top level is set to zero. Based on initial flow, we can obtain the optical flow from the input frames I_0^l and I_1^l to the hidden middle frame $I_{0.5}^l$ by linear scaling: $\hat{F}_{0\to0.5}^l = 0.5 \cdot \hat{F}_{0\to1}^l$, $\hat{F}_{1\to0.5}^l = 0.5 \cdot \hat{F}_{1\to0}^l$.

With $\hat{F}_{0\to0.5}^l$ and $\hat{F}_{1\to0.5}^l$, we forward-warp the CNN features $C_0^{l,2}$ and $C_1^{l,2}$ to the middle frame to align their pixels. Then, we construct a partial correlation volume [34] using warped features, and use a 6-layer CNN to predict refined bi-directional flow $F_{0\to1}^l$ and $F_{1\to0}^l$. In particular, the input to the CNN predictor is the concatenation of correlation volume, warped features, initial flow $\hat{F}_{0\to1}^l$ and $\hat{F}_{1\to0}^l$, and the upsampled feature from the 5-th layer of the CNN predictor at previous pyramid level. Since the warped features are of 1/4 resolution of input frames, the predicted optical flow is also of 1/4 resolution. We use bi-linear interpolation to up-sample optical flow to original scale.

Figure 3 illustrates our bi-directional flow module. Its high-level design is similar to EBME [15]. We adapt EBME for iterative synthesis, by using the features from feature encoder shared with synthesis module, and forward-warping CNN features rather than input frames.

Frame synthesis module. Our frame synthesis module is based on a U-Net structure. The encoder part has three convolutional stages, and each of them is composed of three convolutional layers, with the first layers of the second and third stages performing down-sampling. The decoder part also has three convolutional stages, with two transpose convolutional layers for up-sampling.

At pyramid level l, given refined bi-directional flow $F_{0\to1}^l$ and $F_{1\to0}^l$, we can obtain the optical flow from input frames I_0^l and I_1^l to target frame I_t^l by linear scaling: $F_{0\to t}^l = t \cdot F_{0\to1}^l$, $F_{1\to t}^l = (1-t) \cdot F_{1\to0}^l$. With $F_{0\to t}^l$ and $F_{1\to t}^l$, we forward-warp input frames I_0^l, I_1^l, and their multi-scale context features $\{C_0^{l,0}, C_1^{l,0}\}$, $\{C_0^{l,1}, C_1^{l,1}\}$, $\{C_0^{l,2}, C_1^{l,2}\}$. Furthermore, we generate an initial estimate $\hat{I}_t^l$ of intermediate frame by up-sampling the interpolation from previous $l+1$ level: $\hat{I}_t^l = \text{up}_2(I_t^{l+1})$. At top level, the initial estimate is set to the average of two warped frames. Based on these, we feed the first encoder stage of the synthesis module with warped frames, initial estimate of intermediate frame $\hat{I}_t^l$, original input frames, and scaled bi-directional flow $F_{0\to t}^l$ and $F_{1\to t}^l$. We feed warped context features to the second and third encoder stages, and the first decoder stage to provide multi-scale contextual cues.

The output of our frame synthesis module include two maps M_0^l and M_1^l for fusing two forward-warped frames $I_{0\to t}^l$ and $I_{1\to t}^l$, and a residual image ΔI_t^l for further refinement. Then we can obtain refined intermediate frame I_t^l by:

$$I_t^l = \frac{(1-t) \cdot M_0^l \odot I_{0\to t}^l + t \cdot M_1^l \odot I_{1\to t}^l}{(1-t) \cdot M_0^l + t \cdot M_1^l} + \Delta I_t \quad (1)$$

where $\odot$ denotes element-wise multiplication.

Figure 4 illustrates our frame synthesis module. Its design is inspired by previous context-aware synthesis network [26, 27, 30], but has two distinctive features. First, we feed the synthesis network with up-sampled estimate of intermediate frame as an explicit reference for further refinement. Second, our synthesis module is extremely lightweight, shared across pyramid levels, and much simpler than the grid-like architecture in [26, 27, 30].

Analysis of iterative synthesis for large motion. Iterative synthesis from coarse-to-fine has been proven beneficial for high-resolution images synthesis [4]. In this work, we reveal that it can also significantly improve the robustness of frame interpolation on large motion cases.

To understand this, we start with a plain synthesis strategy, which does not feed up-sampled estimate of intermediate frame into the frame synthesis module. For plain synthesis, the actual synthesis is only performed on the bottom level, because interpolations on previous levels have no path to affect the frame synthesis on the bottom level. By comprehensively comparing the interpolated frames by
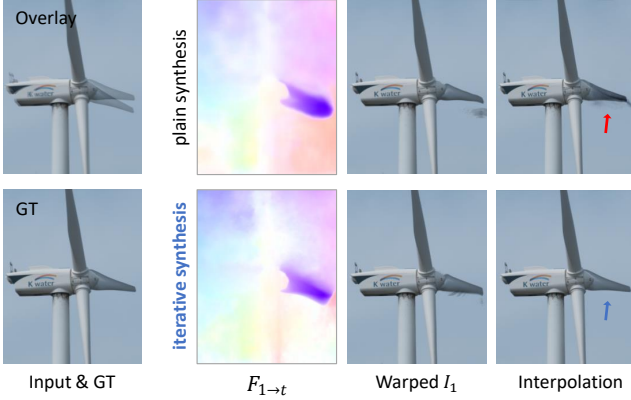
Figure 5. Visualization of warped frames and interpolations for $t = 1/8$, given a large motion case from 4K1000FPS [31]. Our iterative synthesis enables robust interpolation even when warped frame has obvious artifacts. (Best viewed by zooming in.)
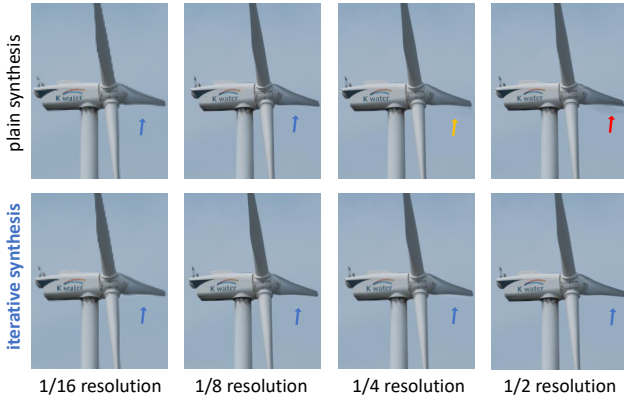

Figure 6. Interpolated frames at different input resolutions, using plain and iterative synthesis. Images are intentionally resized to same size to show the differences. (Best viewed by zooming in.)

plain synthesis and iterative synthesis on large motion cases, we draw empirical conclusions as follows.

(i) When using plain synthesis for interpolation, the obvious artifacts due to large motion in forward-warped frames may lead to obvious artifacts in interpolated frames. Figure 5 shows a typical example.

(ii) Coarse-to-fine iterative synthesis enables robust interpolation even when the warped frame also has obvious artifacts (see Figure 5). We hypothesize that the up-sampled interpolation, which is synthesized at lower-resolution pyramid level, may have less or no artifacts due to smaller motion magnitude. Thus, it can guide the synthesis module to produce robust interpolation at higher-resolution levels.

(iii) Our hypothesis is supported by the evidence shown in Figure 6, where we interpolate the same example on reduced resolutions. We find that on 1/8 resolution, plain synthesis also gives good interpolation without artifacts, since the motion magnitude is much smaller. Our iterative synthesis gives good interpolation at all scales, because it lever-

ages the interpolations from low-resolution levels.

3.3. Resolution-aware Adaptation in Testing

The flexible design of UPR-Net allows resolution-aware adaptation during inference. We describe two adaptation strategies for better estimating the extremely large motions (in high-resolution videos) beyond the training phase.

Customizing the number of pyramid levels. Benefiting from our recurrent module design, we can customize the number of pyramid levels in testing to handle large motion. We follow the method proposed in [15]. Assume that the number of pyramid levels in training is L^{train}, and the width of test image is n times of training images. Then, the number of test pyramid levels is calculated as follows.

$$L^{test} = \texttt{ceil}(L^{train} + \texttt{log}_2 n) \qquad (2)$$

where $\texttt{ceil}()$ rounds up a float number to an integer.

Skipping high-resolution levels for 4K input. By default, at pyramid level l, we up-sample optical flow and interpolation from level $l + 1$ as initializations. But, actually, the design of our UPR-Net allows to up-sample estimates from any previous levels if necessary. We empirically verify that reduced resolution is beneficial for accurate estimation of large motion for extreme 4K resolution videos. We also verify that skipping last two high-resolution levels for flow estimation and second-to-last level for frame synthesis, can improve the accuracy of frame interpolation on 4K benchmark [31].

3.4. Architecture Variants

We construct three versions of models by scaling the feature channels.

- UPR-Net: Base version of UPR-Net. The numbers of channels of feature encoder's three stages are 16, 32, 64, respectively. The channel numbers of the 6-layer CNN in our optical flow module are 160, 128, 112, 96, 64, respectively. The channel numbers of the three encoder stages of our synthesis module are 32, 64, 128.

- UPR-Net large: A large version of UPR-Net, by scaling all feature channels of the base version by 1.5.

- UPR-Net LARGE: A larger version of UPR-Net, by scaling all feature channels of the base version by 2.0.

4. Implementation Details

Loss function. Our loss is the sum of Charbonnier loss [3] and census loss [24] between ground truth I_t^{GT} and our interpolation I_t estimated at the bottom pyramid level:

$$L = \rho(I_t^{GT} - I_t) + L_{cen}(I_t^{GT}, I_t). \qquad (3)$$

| methods | UCF101 | Vimeo90K | SNU-FILM | | | | parameters | runtime |
			easy	medium	hard	extreme	(millions)	(seconds)
DAIN [1]	34.99/0.9683	34.71/0.9756	39.73/0.9902	35.46/0.9780	30.17/0.9335	25.09/0.8584	24.0	0.151
CAIN [6]	34.91/0.9690	34.65/0.9730	39.89/0.9900	35.61/0.9776	29.90/0.9292	24.78/0.8507	42.8	0.037
SoftSplat [27]	35.39/0.9520	36.10/0.9700	-	-	-	-	-	-
AdaCoF [17]	34.90/0.9680	34.47/0.9730	39.80/0.9900	35.05/0.9754	29.46/0.9244	24.31/0.8439	22.9	0.030
BMBC [29]	35.15/0.9689	35.01/0.9764	39.90/0.9902	35.31/0.9774	29.33/0.9270	23.92/0.8432	11.0	0.822
CDFI full [8]	35.21/0.9500	35.17/0.9640	40.12/0.9906	35.51/0.9778	29.73/0.9277	24.53/0.8476	5.0	0.172
ABME [30]	35.38/0.9698	36.18/0.9805	39.59/0.9901	35.77/0.9789	30.58/0.9364	25.42/0.8639	18.1	0.277
XVFI$_v$ [31]	35.18/0.9519	35.07/0.9681	39.78/0.9840	35.37/0.9641	29.91/0.8935	24.73/0.7782	5.5	0.098
RIFE [11]	35.28/0.9690	35.61/0.9780	40.06/0.9907	35.75/0.9789	30.10/0.9330	24.84/0.8534	9.8	**0.012**
EBME-H* [15]	35.41/0.9697	36.19/0.9807	40.28/0.9910	36.07/0.9797	30.64/0.9368	25.40/0.8634	3.9	0.082
VFIformer [22]	35.43/**0.9700**	**36.50/0.9816**	40.13/0.9907	36.09/0.9799	30.67/**0.9378**	25.43/**0.8643**	24.1	1.018
IFRNet [16]	35.29/0.9693	35.80/0.9794	40.03/0.9905	35.94/0.9793	30.41/0.9358	25.05/0.8587	5.0	0.022
IFRNet large [16]	35.42/0.9698	36.20/0.9808	40.10/0.9906	36.12/0.9797	30.63/0.9368	25.27/0.8609	19.7	0.038
UPR-Net	35.41/0.9698	36.03/0.9801	40.37/0.9910	36.16/0.9797	30.67/0.9365	25.49/0.8627	**1.7**	0.042
UPR-Net large	35.43/**0.9700**	36.28/0.9810	40.42/**0.9911**	36.24/0.9799	30.81/0.9370	25.58/0.8636	3.7	0.062
UPR-Net LARGE	**35.47/0.9700**	36.42/0.9815	**40.44/0.9911**	**36.29/0.9801**	**30.86**/0.9377	**25.63**/0.8641	6.6	0.081

Table 1. Qualitative (PSNR/SSIM) comparisons to state-of-the-art methods on UCF101 [33], Vimeo90K [37] and SNU-FILM [6] benchmarks. **RED**: best performance, <u>BLUE</u>: second best performance.

Training dataset.

The Vimeo90K dataset [37] contains 51,312 triplets with resolution of 448×256 for training. We augment the training images by randomly cropping 256×256 patches. We also apply random flipping, rotating, reversing the order of the triplets for augmentation.

Pyramid levels in training.

We use 3-level image pyramids for our UPR-Net during training, which is sufficient to capture the motions on Vimeo90K [37].

Optimization.

We use the AdamW [20] optimizer with weight decay 10^{-4} for 0.8 M iterations, and batch size of 32. We gradually reduce the learning rate during training from 2×10^{-4} to 2×10^{-5} using cosine annealing.

5. Experiments

5.1. Experiment Settings

Evaluation datasets. While our models are trained only on Vimeo90K [37], we evaluate them on a broad range of benchmarks with different resolutions.

- **UCF101 [33]**: The test set of UCF101 contains 379 triplets with a resolution of 256×256. UCF101 contains a large variety of human actions.

- **Vimeo90K [37]**: The test set of Vimeo90K contains 3,782 triplets with a resolution of 448×256.

- **SNU-FILM [6]**: This dataset contains 1,240 triplets, and most of them are of the resolution around 1280×720. It contains four subsets with increasing motion scales – easy, medium, hard, and extreme.

- **4K1000FPS [31]**: This is a 4K resolution benchmark (X-TEST) that enables multi-frame ($\times 8$) interpolation.

Resolution-aware adaptation. According to Equation 2, we set the test pyramid levels for UCF-101, SNU-FILM and 4K1000FPS as 3, 5 and 7, respectively. We skip the last two pyramid levels for bi-directional flow and second-to-last level for frame synthesis on 4K1000FPS. We also report the results without level skipping on 4K1000FPS.

Metrics. PSNR and SSIM [36] are used for quantitative evaluation of frame interpolation. For running time, we test all models with a RTX 2080 Ti GPU under 640×480 resolution, and average the running time by 100 iterations.

5.2. Comparisons with State-of-the-art Methods

We compare with state-of-the-art methods, including DAIN [1], CAIN [6], SoftSplat [27], AdaCoF [17], BMBC [29], CDFI [8], ABME [30], XVFI [31], RIFE [11], EBME [15], VFIformer [22], and IFRNet [16]. We report results by executing the source code and trained models, except for SoftSplat which has not released the full code. For SoftSplat, we copy the results from original paper.

Low resolution frame interpolation. Table 1 reports the comparison results on low-resolution UCF101 and Vimeo90K datasets. The transformer-based VFIformer [22] achieves the best accuracy on Vimeo90K, which however has a large number of parameters, and runs very slow. Our UPR-Net LARGE model achieves the best performance on UCF101 and second best result on Vimeo90K. Furthermore,

Figure 7. Qualitative comparisons on SNU-FILM [6]. First example from the hard subset, and second example from extreme subset.
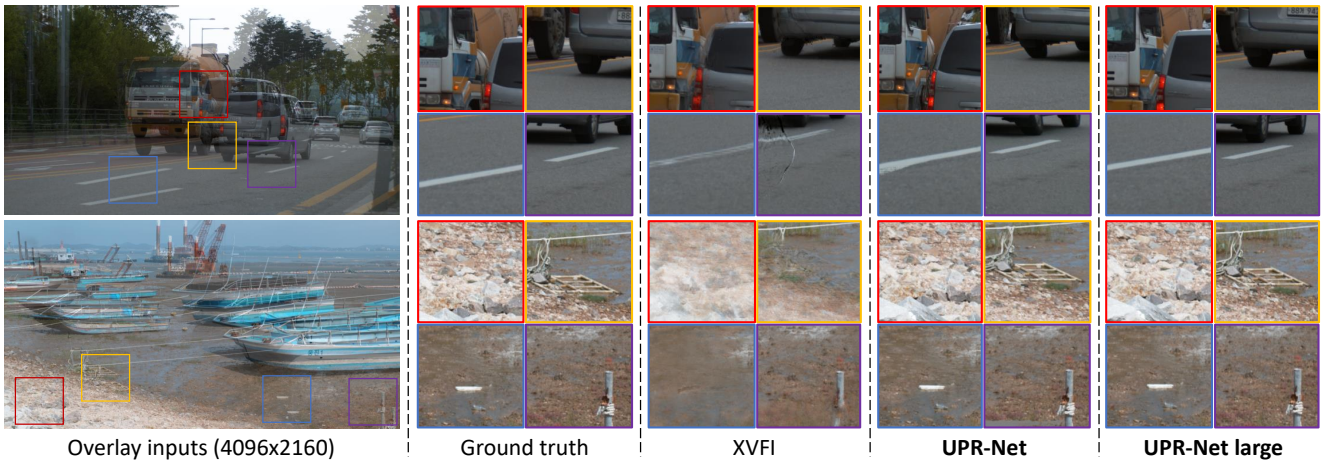

Figure 8. Qualitative comparisons on X-TEST of 4K1000FPS [31]. First example interpolated at $t = 1/8$ and second example at $t = 1/2$.

our UPR-Net large and UPR-Net models also achieve excellent accuracy on these two benchmarks. In particular, our UPR-Net model, which only has 1.7M parameters, outperforms many recent large models on UCF101, including SoftSplat [27] and ABME [30].

Moderate resolution frame interpolation. Table 1 also reports the comparison results on SNU-FILM. When measured with PSNR, our UPR-Net series outperform all previous state-of-the-art methods. In particular, our base UPR-Net model outperforms the large VFIformer model, in part due to its capability of handling challenging motion.

Figure 7 gives two examples from the hard and extreme subsets from SNU-FILM, respectively. Our methods produce better interpolation results than IFRNet large model [16] for local textures (first two rows), and give promising results for large motion cases (last two rows),

much better than CAIN [6] and VFIformer [22], and sightly better than ABME [30] and IFRNet large [16].

4K resolution multiple frame interpolation. Table 2 reports the 8x interpolation results on X-TEST. Our UPR-Net large model achieves the best performance. Skipping proper high-resolution levels enables better interpolation results, as it benefits the estimation of extremely large motions. Our UPR-Net LARGE model does not achieve the best results, as it might overfit the motion magnitude of Vimeo90K [37]. Furthermore, our method enables arbitrary-time frame interpolation, and the bi-directional flow only needs to be estimated once for multi-frame interpolation.

Figure 8 shows two interpolation examples. Our models are robust to large motion, and give better interpolation for local textures than XVFI [31].

methods	arbitrary time	reuse flow	X-TEST (4K)
DAIN [1]	✓	✓	26.78/0.8065
AdaCoF [17]	×	×	23.90/0.7271
ABME [30]	✓	×	30.16/0.8793
XVFI [31]	✓	partial	30.12/0.8704
UPR-Net	✓	✓	30.13/0.8990
UPR-Net large	✓	✓	**30.68/0.9086**
UPR-Net LARGE	✓	✓	30.50/0.9048
UPR-Net†	✓	✓	29.27/0.8877
UPR-Net large†	✓	✓	30.14/0.9046
UPR-Net LARGE†	✓	✓	29.91/0.8998

Table 2. Quantitative (PSNR/SSIM) comparisons on X-TEST (from 4K1000FPS [37]) for 8x multi-frame interpolation. †: not skip pyramid levels for optical flow and frame synthesis.

experiments	methods	Vimeo90K	Hard	X-TEST	param.
recurrent modules	only flow	35.98	30.33	24.01	3.60M
	only syn.	**36.03**	30.25	23.65	2.68M
	both	**36.03**	**30.67**	**30.13**	1.65M
iterative synthesis	plain	36.02	30.57	28.91	1.65M
	iterative	**36.03**	**30.67**	**30.13**	1.65M
unified pipeline	separate	35.96	**30.67**	29.89	1.82M
	unified	**36.03**	**30.67**	**30.13**	1.65M
correlation volume	without	35.81	30.59	29.51	1.64M
	with	**36.03**	**30.67**	**30.13**	1.65M
context feature	without	35.63	**30.68**	29.91	1.43M
	with	**36.03**	30.67	**30.13**	1.65M

Table 3. Ablation studies of our design choices on Vimeo90K [37], hard subset of SNU-FILM [6], and 4K X-TEST [31]. Default settings (independent of benchmark datasets) are marked in gray .

Parameter and inference efficiency. As shown the last two columns in Table 1, our base version of UPR-Net only has 1.7 M parameters. UPR-Net is much faster than recent ABME and VFIformer, but slower than RIFE and IFRNet.

5.3. Ablation Studies of Design Choices

In Table 3, we present ablation studies of the design choices of our UPR-Net on Vimeo90K [37], the hard subset of SNU-FILM [6], and X-TEST of 4K1000FPS [31].

Recurrent module design. We verify the effectiveness of recurrent model design by replacing the optical flow module or frame synthesis module with three cascaded non-recurrent modules. These two non-recurrent counterparts can still achieve promising performance on low-resolution Vimeo90K [37]. However, they can not customize the number of pyramid levels in testing, and thus lack of flexibility in handling large motion. As a result, they can not achieve good results on the hard subset of SUN-FILM [6], and the extreme 4K-resolution X-TEST benchmark [31].

Iterative frame synthesis. Plain synthesis without using up-sampled interpolation achieves good performance on Vimeo90K [37]. This suggests that iterative synthesis might be unnecessary for low-resolution images. Compared to many state-of-the-art methods, it also achieves good results on hard subset of SNU-FILM [6]. Even so, iterative synthesis achieves better performance on hard subset, and much better performance on X-TEST [31]. These results provide quantitative evidences about our analysis in Section 3.2.

Unified pipeline. We construct a pipeline consisting of two separate pyramid recurrent networks (with separate feature encoders) for bi-directional flow and frame synthesis. Like previous works [15, 31], optical flow network outputs single-scale bi-directional flow from the bottom pyramid level. We down-sample the bi-directional flow to generate multi-scale flow for iterative frame synthesis. This separate counterpart adds about 14% parameters (and computational cost), and achieves slight inferior performance to our unified pipeline. We prefer our unified pipeline, due to its simplicity and elegance. But, it is worth noting that our pyramid recurrent frame synthesis network can be combined with any existing optical flow model for frame synthesis.

Correlation volume for bi-directional flow. Removing the correlation volume from our optical flow module leads to performance degradation on all benchmarks. This result is consistent with the observations in [15]. Arguably, the importance of correlation volume is overlooked by many existing frame interpolation methods [11, 31, 38].

Context feature for frame synthesis. Removing the context features from our synthesis module will greatly degrade the accuracy on Vimeo90K [37]. But, surprisingly, it does not lead to obvious inferior performance on large motion benchmarks, which is worth further investigation.

6. Conclusion and Future Work

This work presented UPR-Net, a lightweight Unified Pyramid Recurrent Network for frame interpolation. UPR-Net achieved excellent performance on various frame interpolation benchmarks. In the future, we will investigate some interesting problems related to this work. Firstly, we will verify the generalization of our iterative synthesis by replacing our motion estimator with an off-the-shelf optical flow model (*e.g.*, PWC-Net [34]). Second, we will train our models with the Vimeo90K-Septuplet [37] to investigate whether multi-frame interpolation during training is beneficial for multi-frame interpolation in testing.

References

[1] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *CVPR*, 2019. 1, 2, 6, 8

[2] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *TPAMI*, 2019. 1

[3] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*, 1994. 5

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 2, 4

[5] Xianhang Cheng and Zhenzhong Chen. Video frame interpolation via deformable separable convolution. In *AAAI*, 2020. 1

[6] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *AAAI*, 2020. 1, 2, 6, 7, 8

[7] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. *NeurIPS*, 2015. 2

[8] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. CDFI: Compression-driven network design for frame interpolation. In *CVPR*, 2021. 6

[9] Patrick Esser, Robin Rombach, Andreas Blattmann, and Bjorn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. *NeurIPS*, 2021. 2

[10] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016. 1

[11] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *ECCV*, 2022. 2, 3, 6, 8

[12] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018. 1

[13] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 1

[14] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *ICCV*, 2018. 1

[15] Xin Jin, Longhai Wu, Guotao Shen, Youxin Chen, Jie Chen, Jayoon Koo, and Cheul-hee Hahm. Enhanced bi-directional motion estimation for video frame interpolation. *arXiv preprint arXiv:2206.08572*, 2022. 2, 4, 5, 6, 8

[16] Lingtong Kong, Boyuan Jiang, Donghao Luo, Wenqing Chu, Xiaoming Huang, Ying Tai, Chengjie Wang, and Jie Yang. IFRNet: Intermediate feature refine network for efficient frame interpolation. In *CVPR*, 2022. 2, 6, 7

[17] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. AdaCoF: Adaptive

[18] Sungho Lee, Narae Choi, and Woong Il Choi. Enhanced correlation matching based video frame interpolation. In *WACV*, 2022. 1, 2

[19] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 1

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[21] Guo Lu, Xiaoyun Zhang, Li Chen, and Zhiyong Gao. Novel integration of frame rate up conversion and HEVC coding based on rate-distortion optimization. *TIP*, 2017. 1

[22] Liying Lu, Ruizheng Wu, Huaijia Lin, Jiangbo Lu, and Jiaya Jia. Video frame interpolation with transformer. In *CVPR*, 2022. 6, 7

[23] Yao Lu, Jack Valmadre, Heng Wang, Juho Kannala, Mehrtash Harandi, and Philip Torr. Devon: Deformable volume network for learning optical flow. In *WACV*, 2020. 2

[24] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018. 5

[25] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *CVPR*, 2018. 1

[26] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 1, 2, 4

[27] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020. 1, 2, 3, 4, 6, 7

[28] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 1

[29] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. BMBC: Bilateral motion estimation with bilateral cost volume for video interpolation. In *ECCV*, 2020. 1, 6

[30] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *ICCV*, 2021. 1, 4, 6, 7, 8

[31] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: Extreme video frame interpolation. In *ICCV*, 2021. 2, 5, 6, 7, 8

[32] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *CVPR*, 2021. 1

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[34] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 1, 2, 3, 4, 8

[35] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 1, 2, 3

[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 6

collaboration of flows for video frame interpolation. In *CVPR*, 2020. 6, 8

[37] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 2019. 6, 7, 8

[38] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A flexible recurrent residual pyramid network for video frame interpolation. In *ECCV*, 2020. 2, 8