IAM-VFI : Interpolate Any Motion for Video Frame Interpolation with motion complexity map

Kihwan Yoon[1], Yong Han Kim[1], Sungjei Kim[2,†], and Jinwoo Jeong[2,†]

[1] University of Seoul, Seoul 02504, Republic of Korea
[2] Korea Electronics Technology Institute, Gyeonggi-do 13488, Republic of Korea
rlghksdbs@gmail.com
yhkim@uos.ac.kr
{jw.jeong, sungjei.kim}@keti.re.kr
https://github.com/rlghksdbs/IAM-VFI

Abstract. Within the video, different regions have varying motion complexity, with simple regions containing static or global motion and complex regions containing fast motion or lots of local motion. In recent years, the performance of flow-based Video Frame Interpolation (VFI) algorithms has improved significantly. However, existing training methods train on randomly cropped regions of train data without considering the complexity of the motion. As a result, they cannot handle all regions of the frame that contain varying motion complexity. To solve this problem, we propose a novel VFI approach (IAM-VFI) that can interpolate any motion by considering the motion complexity of all regions in the frame. First, we propose a training data classification method for motion optimization based on each motion complexity. Then, using the proposed data, a flow estimation network generates optimized results for each complexity. Finally, we propose a Motion Complexity Estimation Network (MCENet) to generate a Motion Complexity Map (MCM) that can estimate the motion complexity of each region. Our proposed methods can be easily applied to most flow-based VFI algorithms. Experimental results show that the proposed method can interpolate any motion and significantly improve the performance of existing VFI algorithms.

Keywords: Video Frame Interpolation, Optical Flow Estimation, Region-Aware

1 Introduction

Video Frame Interpolation (VFI) is a challenging low-level vision task that increases the frame rate by generating a non-existent intermediate frame between two consecutive frames. VFI can be applied to various tasks such as video restoration [3], novel view synthesis [7, 41], slow motion generation [14, 23], and frame up-conversion [5], etc. With the development of deep learning, the performance of VFI algorithms has significantly improved.

† Co-corresponding authors.

(a) Interpolation results for region with medium motion complexity

(b) Interpolation results for region with hard motion complexity

Fig. 1: Example of subjective results of based on motion complexity. In the case of (a), the motion complexity between the two input frames is classified as a medium, and the model trained on data classified as medium is the best. In contrast, (b) has a fast motion complexity and the model trained on the hard data shows the best results.

Existing VFI algorithms are generally divided into two main categories: kernel-based methods [28, 29] and flow-based methods [16, 17, 39]. Kernel-based methods typically use an adaptive convolution to synthesize intermediate frames by local patches. However, it suffers from a limitation when the motion is larger than the kernel size, which degrades the interpolation results. Furthermore, the memory requirement increases when a large kernel size is used. Flow-based methods, on the other hand, estimate the optical flow between two input frames and synthesize the intermediate frame by warping it with the input frames. Flow-based methods have become promising method in recent years due to advances in flow estimation networks [12, 17, 30].

Within consecutive frames, the distribution of motion complexity varies, with some regions containing simple motion patterns such as stationary motion, global motion, and linear motion, and others containing complex motion patterns such as occlusion, fast motion, and local motion. However, existing flow-based VFI algorithms typically train using patches that are cropped random regions of the training data. The motion complexity within these patches is randomly distributed from easy to hard. When the complexity is distributed over a wide range, it is not possible to train optimally for each complexity because each complexity interferes with the others. As a result, the interpolation results are degraded because it cannot optimally estimate all motions in a video with varying motion complexity. Therefore, to achieve optimal results for motion in all regions, it is necessary to interpolate by considering information about each motion complexity within the frame.

To address these issues, we propose IAM-VFI, a method that estimates the motion complexity of each region of the intermediate frames and interpolates for

all motions based on their complexity. First, we analyzed the relation between existing VFI results and motion complexity to further investigate the problem. Then, based on the analysis results, we propose a data classification method that classifies the training data into three classes (easy, medium, and hard) according to the motion complexity to optimize the motion estimation network of VFI for each motion complexity. As shown in the Fig. 1, we can see that simply training with complexity-classified data improves the results for specific complexities.

Finally, we propose a Motion Complexity Estimation Network (MCENet) that estimate the complexity of the motion in each region of the intermediate frame and adaptively interpolate all the motions according to their complexity. MCENet generates a Motion Complexity Map (MCM) that represents the motion complexity within a frame. Experimental results show that the proposed method can significantly improve performance when applied to existing VFI algorithms.

In summary, this paper has the following main contributions:

- We analyse the relation between the motion complexity and the interpolation results and propose a dataset classification method.
- We propose the IAM-VFI algorithm that can handle any motion within the frame with varying distributions of motion complexity.
- We propose a novel network called MCENet and a complexity loss to generate a MCM that to handle all region within the frame.
- The proposed methods can be simply applied to previous flow-based VFI algorithms and achieve significant performance.

2 Related Works

2.1 Flow-based Video Frame Interpolation

Deep learning based VFI algorithms typically take two consecutive frames (I_0, I_1) as input and generates the intermediate frame $\hat{I}_t$, $0 < t < 1$ through a VFI network ϕ with parameters Θ.

$$\hat{I}_t = \phi(I_0, I_1; \Theta) \tag{1}$$

In recent years, advances in optical flow estimation networks have made flow-based methods become a dominant approach for VFI algorithms. Most flow-based VFI algorithms typically involve as following steps. First estimate optical flows $F_{0\to1}, F_{1\to0}$ between two frames through a flow network. Then the flows are warped with the input frames to generate intermediate frame. For warping, some algorithms generate $F_{0\to t}, F_{1\to t}$ by scaling the flow between two frames, and then use forward warping to generate two warped frames $I_{0\to t}, I_{1\to t}$. However, the problem with forward warping is that it creates holes in the interpolated frame. For this reason, most recent VFI algorithms use backward warping by creating $F_{t\to0}, F_{t\to1}$.

$$\hat{I}_{0\to t} = \phi_w(I_0, F_{0\to t}),\ \hat{I}_{1\to t} = \phi_w(I_1, F_{1\to t}) \tag{2}$$

where ϕ_w is a backward warp operation. Then, generate a temporary intermediate frame $\tilde{I}_t$ with the two warped frames and the mask obtained by the flow network [17, 36].

$$\tilde{I}_t = M \odot \hat{I}_{0\to t} + (1 - M) \odot \hat{I}_{1\to t} \tag{3}$$

Finally, generate an intermediate frame $\hat{I}_t$ by adding the estimated residuals β using the synthesis network [8, 33, 39].

$$\hat{I}_t = \tilde{I}_t + \beta \tag{4}$$

Previous flow-based VFI algorithms suffer from the issues that the receptive field size is limited, and therefore it is challenging to handle fast motion with large pixel displacements or complex videos with occlusion and context details. To address this issues, Sim *et al.* proposed XVFI, a recursive multi-scale structure for processing 4K video with large motion, and also released the X4K1000FPS dataset with extreme motion [34]. To handle large and complex motion, Jin *et al.* proposed an EBME [17] that guides the flow through correlation volumes. And various flow networks have been proposed in recent years that utilize pyramid structure [16, 17] to enlarge receptive field, and utilize transformer [24, 40] to capture long-range correspondence. As a result, these methods have achieved impressive results for handling large and complex motions.

However, interpolating complex frames with many occlusion regions or various motion distributions within the frame still remains as a challenging problem. In addition, there is the problem of poor performance of interpolation on data other than training data.

2.2 Region-aware Image & Video Restoration

In a video (or image), each region has various characteristics of spatial complexity, such as edges and noise, and temporal complexity, such as the degree of motion between frames. Therefore, in the restoration task, it is important to understand the characteristics of each region for accurate restoration.

As an example of utilizing spatial complexity, Rad *et al.* proposed SROBB [32], which uses different weights for loss based on the OBB (Object, Background, Boundary) labels within the frame. And Kong *et al.* proposed ClassSR [20], which categorizes each patch in a frame as easy, medium, or hard based on the PSNR and restores each patch appropriately. Recently, the Segmentation Anything Model (SAM) [18] has been utilized for region aware restoration tasks [15, 37].

In VFI, it is also important to interpolate the intermediate frame with considering the spatial and temporal characteristics of the video. However, existing VFI algorithms have been interpolating without considering these characteristics. As a result, it is difficult to interpolate all regions of large videos with varying distributions of motion complexity. To address the limitations of existing VFI algorithms, we propose IAM-VFI, a method for interpolating all regions in videos with motion complexity.

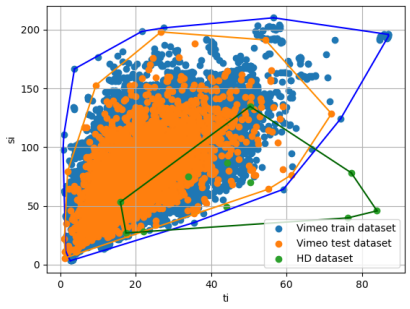


Fig. 2: SI & TI distribution of VFI benchmark

Table 1: Interpolation results of subclips in HD data

	Name	PSNR
HD Dataset	Parkrun	29.587
	Shields	36.236
	Stockholm	35.304
	Bluesky	41.438
	Kimono	35.983
	Parkscene	37.482
	Sunflower	35.994
	Sintel Alley2	30.452
	Sintel Market5	21.077
	Sintel Temple1	27.57
	Sintel Temple2	23.19

3 Proposed Methods

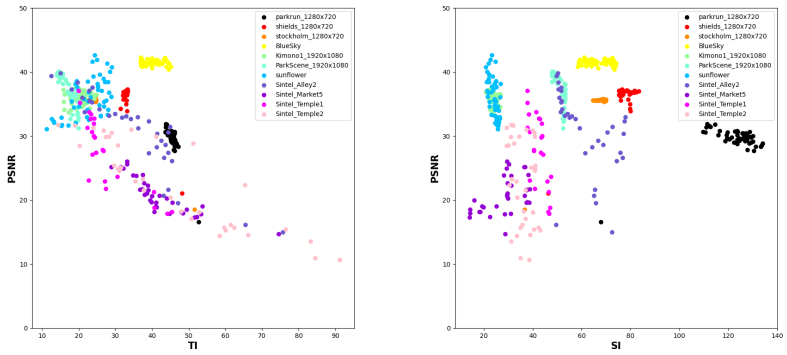
In this section, we first explore the problems with existing algorithms by analysing the relation between video complexity and VFI results in Sec. 3.1. Then, in Sec. 3.2, we describe a data classification method to address the existing problems. Finally, we describe the overall structure of our proposed IAM-VFI to interpolate any motion within the frame in Sec. 3.3, and describe the proposed MCENet to estimate the motion complexity of each region in Sec. 3.4.

3.1 Relation between Video Complexity & Interpolate results

We conducted the analysis in this section to explore the limitations of existing algorithms. The algorithm we used for the analysis is RIFE [12], a representative flow-based VFI algorithm. And we used Spatial Information (SI) and Temporal Information (TI) [13] to find the relation between the complexity of the video with the interpolation results. SI and TI are often used to ensure that videos span the appropriate range of spatio-temporal complexity and are used to classify video data according to complexity [9, 35]. It is also used when creating source videos to ensure that the distribution of video complexity is appropriate [2], and recently SI/TI has been used as a feature of deep learning-based video quality metrics [10]. SI is a measure of the amount of spatial detail by computing the sobel filter vertically and horizontally to detect edges within a frame and considering the standard deviation of the detected edges:

$$SI = std_{space}[Sobel(F_n)] \tag{5}$$

Where F_n is the n_{th} frame of the video. Higher SI value indicate that the frame contains more detail are has a higher spatial complexity. TI is a measure of the amount of temporal variation in a video sequence. TI first calculates



(a) Relation between TI and PSNR in HD **(b)** Relation between SI and PSNR in HD

Fig. 3: Relation between spatiotemporal information (SI/TI) & PSNR

the amount of variation between two adjacent frames by calculating the difference between the current frame and the previous frame, and then considers the standard deviation of the variation to calculate temporal complexity.

$$M_n(i,j) = F_n(i,j) - F_{n-1}(i,j) \tag{6}$$

$$TI = std_{space}(M_n(i,j)) \tag{7}$$

Where M_n represents the difference between the current frame F_n and the previous frame F_{n-1}. A higher value of TI indicates that more motion is included between two frames.

Video complexity is an important factor in VFI, so we analysed the relation between SI/TI and interpolation results to find out the reason why the existing VFI algorithms perform poorly on data that deviates from the characteristics of the training data. As shown in the Fig. 2, we can see that the SI/TI distribution of the Vimeo90K [38] train dataset, which is commonly used for training, contains the entire Vimeo90K test dataset. As a result, the existing training methods perform well on interpolate the Vimeo90K test data. However, some of the subclips in the HD [1] dataset have SI/TI that are out of the distribution of the training data. In this case, we can observe that the PSNR of subclips outside the distribution of the training data degrades in Tab. 1. Based on this, we further analyze the relation between SI/TI and PSNR. As shown in Fig. 3a, PSNR is more closely related to TI, which is important for accurately estimating motion in VFI. It shows that PSNR degrades as the complexity of motion between frames increases. In contrast, SI, the spatial complexity of the interpolation, is not closely related to the interpolation result, as shown in Fig. 3b.

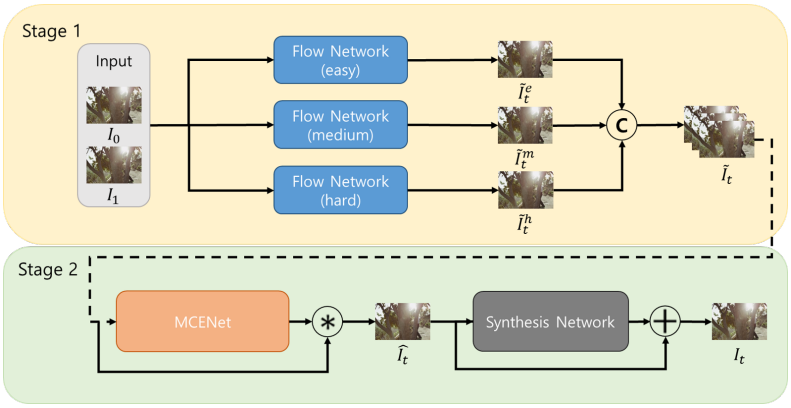


Fig. 4: Overview of our IAM-VFI. The Flow Network in stage 1 is the Flow Network of the existing VFI algorithm. It is trained with our proposed easy, medium, and hard datasets, respectively, to optimize for the motion complexity. The Synthesis Network in stage 2 is the synthesis network of the existing VFI algorithm.

3.2 Data classification Method

Previous training methods typically train with patches cropped from random regions of the input image. However, the complexity of motion within the patch is also randomized, leading to suboptimal training for each complexity. As a result, it suffers from poor results for videos with varying complexity distribution within the video. To address this problem, we propose a data classification method that classifies the existing training dataset into easy, medium, and hard datasets according to motion complexity based on the analysis of Sec. 3.1.

First we created Vimeo90K(X2), which is a $\times 2$ upsampling of Vimeo90K [38] via EDSR [22] to generate more samples for larger motions, and we use both Vimeo90K and Vimeo90K(X2) data for classification. To avoid training randomly cropped regions, we first overlapped parts of the dataset and cropped them into patches of 256x256 size. We then calculated the TI to classify the data based on the motion complexity of each cropped patch. Based on the calculated TI, we set a threshold to divide the number of data into three parts: easy, medium, and hard datasets. The effectiveness of our proposed data classification method can be seen in Sec. 5.1.

3.3 IAM-VFI Overview

To identify the motion complexity for each region in the frame and to handle all motion, we propose IAM-VFI. Our proposed IAM-VFI is trained in two stages, as shown in Fig. 4. In the first stage, we train a flow estimate network with easy, medium, and hard datasets, each classified by motion complexity, to perform motion optimization for that complexity. The flow estimation network trained on each data generates optimized frames $\tilde{I}_t^e, \tilde{I}_t^m, \tilde{I}_t^h$, for each complexity. The generated frames are then concatenated and used as input for the second stage.

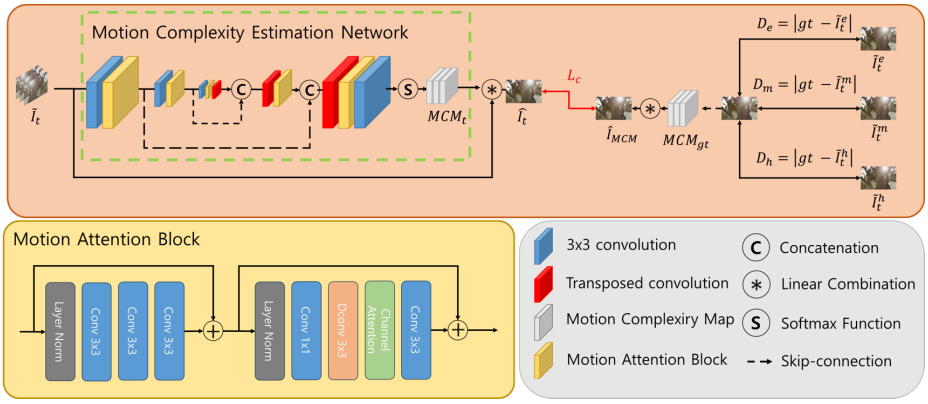


Fig. 5: The network structure of Motion Complexity Estimation Network

$$\tilde{I}_t = Concat(\tilde{I}_t^e, \tilde{I}_t^m, \tilde{I}_t^h) \tag{8}$$

The second stage receives each optimized frame as input and generates a Motion Complexity Map (MCM_t) using a Motion Complexity Estimation Network (MCENet) to estimate the complexity of each region in the intermediate frame. Then perform a linear combination of the generated MCM_t and the input image $\tilde{I}_t$ to generate a temporal intermediate frame $\hat{I}_t$. And the generated MCM_t is used as input to provide information about the motion complexity to the synthesis network of the existing VFI algorithm to estimate the residual. Finally, the residual is added to the temporal intermediate frame to generate the final interpolated frame I_t.

3.4 Motion Complexity Estimation Network

We propose MCENet, a network for estimating the motion complexity of each region of the intermediate frames to be interpolated and handling them according to the motion complexity of each region. First, MCENet takes as input $\tilde{I}_t^e, \tilde{I}_t^m$, and $\tilde{I}_t^h$, the optimized results for each motion complexity, and generates an $MCM_t \in \mathbb{R}^{3\times H\times W}$ that estimate the motion complexity of the region. The generated MCM_t consists of three channels, each representing a probability value for whether this region is close to easy, medium, or hard. We then perform a linear combination of multiplying each channel by the $\tilde{I}_t^e, \tilde{I}_t^m$, and $\tilde{I}_t^h$ frames respectively to generate $\hat{I}_t$, the frame that reflects the complexity. As a result, MCM_t also controls the region on the boundary of each class by giving more weight to the most probable values among the three classified easy, medium, and hard values.

$$\hat{I}_t = \sum_{c\in e,m,h} MCM_t^c \tilde{I}_t^c \tag{9}$$

We construct MCENet with a U-Net structure for simple application to existing VFI algorithms and propose Motion Attention Block (MAB) for estimating motion complexity more efficiently. The detailed architecture of MCENet and MAB can be seen in Fig. 5.

To effectively train MCM_t, we propose a novel loss function, complexity loss. The frame $\hat{I}_t$ generated by the MCM_t, contains information about the complexity of the motion in each region. Therefore, it is not effective to use a loss function to reduce the error with GT, which does not contain information about complexity. For this reason we generate MCM_{gt}, which contains information about the complexity of the motion. To generate the MCM_{gt}, we compute the error between the optimized frame and the ground truth for each pixel by motion complexity. We then assign a 1 to the channel of the MCM_{gt} that represents the class with the minimum error at each pixel, and a 0 to the other channels. This allows the MCM_{gt} to represent the best motion complexity class at each pixel.

$$MCM_{gt}^{c}(i,j) = \begin{cases} 1, & \text{if } c = \underset{c \in \{e,m,h\}}{\operatorname{argmin}} \left(\left| gt(i,j) - \tilde{I}_t^c(i,j) \right| \right) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Finally, we compute a linear combination of the generated MCM and each frame to generate $\hat{I}_{MCM}$

$$\hat{I}_{MCM} = \sum_{c \in \{e,m,h\}} MCM_{gt}^{c} \tilde{I}_t^c \tag{11}$$

We update the MCM_t by taking the L1 loss between two frames $\hat{I}_t$ and $\hat{I}_{MCM}$ that contain the motion complexity.

$$\mathcal{L}_c = \left\| \hat{I}_t - \hat{I}_{MCM} \right\|_1 \tag{12}$$

The effect on $\mathcal{L}_c$ was verified in Sec. 5.3.

4 Experiment

4.1 Implement details

Training Dataset. We first trained the flow network using the classified datasets introduced in Sec. 3.2. Then, we freeze the flow network and trained the MCENet and synthesis network. Since the classified datasets are the data proposed to optimize each motion, we used the original Vimeo90K to train MCENet and synthesis network. The Vimeo90K data consists of 51,312 triplets and has a resolution of 448 × 256.

Training Strategy. We applied the proposed methods to the flow-based algorithms RIFE [12] and EMA-VFI [40]. The experiments were implemented based on the official code presented in the paper. For both proposed stages, we used the same optimization techniques such as batch size, optimizer, scheduler, and

learning rate as used in RIFE and EMA-VFI, and trained for the same iterations. Our training was performed on a single NVIDIA RTX A6000.

Loss Function. In the first stage, we used the same loss function used in the original networks for training the flow estimation network. In the second step, we used $\mathcal{L}_{syn}$, the reconstruction loss used to train synthesis network in RIFE and EMA-VFI, and $\mathcal{L}_c$, the loss function proposed in Sec. 3.4 to train the MCENet.

$$\mathcal{L}_{total} = \lambda_s \mathcal{L}_{syn} + \lambda_f \mathcal{L}_c \tag{13}$$

where we set $\lambda_s = 0.5$, $\lambda_f = 0.5$ to balance the loss values.

4.2 Benchmarks and Evaluation Metrics

Our proposed methods are to handle various motion complexities. Therefore, we evaluated our proposed method on the following datasets with various motion complexities.

SNU-FILM [4] : 1280x720 resolution with a total of 1240 frames, categorized into easy, medium, hard, and extreme according to the motion magnitude.

HD [1] : It consists of 11 videos. The HD dataset consists of four 1080p, three 720p and four 544p videos, and we used the first 100 frames of each video

UVG [25] : It consists of 7 videos with 3840x2160 resolution, and we used the first 300 frames of each video.

Xiph [26] : It contains 19 video sequences, each consisting of 31 consecutive 4K resolution frames. We followed [27] method to downsample and center-crop the 4K images to create a 2K resolution and evaluate both Xiph4K and Xiph2K.

Metrics. We used Peak Signal-To-Noise Ratio (PSNR) and Structural-Similarity-Image-Metric (SSIM), commonly used objective quality metrics, where a higher PSNR and SSIM indicate better quality. To evaluate runtime, we measured all algorithms at 720p resolution to be fair, and we used a single NVIDIA RTX A6000 GPU.

5 Results

5.1 Data Classification method

We first validate our proposed data classification method. The proposed method aims to optimize for the complexity of each motion, so we exclude the influence

Table 2: Quantitative results comparison on benchmark datasets.(Red indicates best PSNR value)

Train Data \ Benchmark	Vimeo90K			UVG		
	easy	medium	hard	easy	medium	hard
Vimeo90K	43.543	36.495	32.131	48.338	36.317	30.052
Ours$_{easy}$	43.955	36.402	31.727	49.027	36.291	29.338
Ours$_{medium}$	43.951	37.012	32.588	48.509	36.851	30.289
Ours$_{hard}$	43.632	36.991	32.721	47.831	36.815	30.611

Table 3: Quantitative results comparison on benchmark datasets.(Red indicates best PSNR/SSIM values within each dataset and Blue indicates second best and Green indicates third best.) We followed the test procedures of [12] for HD, [19] for SNU-FILM, and [11] for Xiph to ensure a fair comparison. "OOM" indicates "Out of Memory" and "†" indicates the results we obtained by retraining to compare results.

Algorithm	HD	SNU-FILM				UVG	Xiph	
		Easy	Medium	Hard	Extreme		2K	4K
SepConv [29]	30.87/0.930	39.41/0.990	34.97/0.976	29.36/0.925	24.31/0.844	26.11/0.864	34.77/0.929	32.06/0.880
CAIN [4]	31.45/0.939	39.91/0.990	35.57/0.977	29.87/0.929	24.73/0.850	30.32/0.902	35.21/0.937	32.56/0.901
AdaCoF [21]	31.43/0.933	39.84/0.990	35.07/0.975	29.47/0.924	24.31/0.843	OOM	34.86/0.928	32.19/0.882
CDFI [6]	31.46/0.937	40.11/0.990	35.51/0.977	29.74/0.927	24.54/0.847	OOM	35.48/0.940	32.47/0.903
ABME [31]	32.17/0.943	39.59/0.990	35.51/0.978	29.47/0.936	25.23/0.863	OOM	35.18/0.964	32.36/0.940
IFRNet [19]	32.15/0.943	40.02/0.990	35.93/0.979	30.40/0.935	25.05/0.858	31.33/0.904	36.24/0.964	33.38/0.941
M2M-PWC [11]	31.23/0.937	39.59/0.990	35.68/0.979	30.27/0.936	25.07/0.860	30.89/0.910	36.40/0.967	33.77/0.943
VFIformer [24]	OOM	40.12/0.991	36.09/0.980	30.67/0.937	25.20/0.863	OOM	OOM	OOM
EBME [17]	32.44/0.945	39.98/0.991	35.74/0.979	30.40/0.935	25.24/0.861	31.79/0.915	36.33/0.965	33.71/0.942
RIFE† [12]	32.08/0.942	39.98/0.990	35.78/0.978	30.14/0.933	24.84/0.853	31.13/0.900	36.15/0.964	33.25/0.940
RIFE_{ours}	32.46/0.946	40.18/0.991	36.05/0.980	30.58/0.937	25.35/0.863	32.17/0.917	36.60/0.966	34.20/0.946
EMA-VFI† [40]	32.62/0.948	40.17/0.991	35.97/0.979	30.64/0.935	25.27/0.857	31.19/0.907	36.71/0.966	33.69/0.944
EMA-VFI_{ours}	32.89/0.951	40.28/0.992	36.30/0.981	30.88/0.938	25.59/0.864	32.12/0.916	37.01/0.968	34.29/0.947

of synthesis networks for accurate validation. We use the proposed easy, medium, and hard datasets to train IFNet, the flow network of RIFE. And warped the estimated flows and input frames then generate temporal intermediate frames using Eq. (3). The Benchmark used for validation was Vimeo90K test and UVG, and was divided into 256x256 patches to avoid varying complexity within the region and the TI of each patch was calculated and classified as easy, medium, or hard.

As shown in Tab. 2, we can see that the training with the complexity classified data optimizes the motion estimation performance, resulting in improved PSNR compared to the results trained with the original dataset. This result shows that classification by complexity is effective in optimizing motion.

5.2 Comparison with the State-of-the-Art Methods

To compare the proposed methods, we used the kernel-based algorithms SepConv [29], AdaCoF [21], and CDFI [6], the flow-based algorithms ABME [31], IFRNet [19], M2M-PWC [11], VFIformer [24], EBME [17], and the other method CAIN [4]. We applied our methods to the flow-based algorithms RIFE [12] and EMA-VFI [40].

Tab. 3 summarizes the quantitative comparison with existing VFI algorithms on various benchmarks to show the effectiveness of the proposed methods. The performance of the original RIFE is slightly worse compared to the state-of-the-art algorithms, but we can see that our proposed method improves performance on all benchmarks. In particular, for UVG and Xiph4K, which are high-resolution videos with various motion complexity, we can see the PSNR improvements of **1.04 dB** and **0.95 dB**, respectively, with the best results for UVG data

(a) Visual comparison of IAM-VFI applied to EMA-VFI with the SOTA algorithm

(b) Visual comparison of IAM-VFI applied to RIFE with the SOTA algorithm

Fig. 6: Visual comparison on extreme subset of SNU-FILM [4] and HD [1] benchmarks.

compared to the state-of-the-art algorithms. Other benchmarks also achieved second or third best results. And when the IAM-VFI method is applied to the recently proposed EMA-VFI, we can see that it achieves the best performance in all benchmarks except UVG, with an improvement of **0.93 dB** in UVG and **0.60 dB** in Xiph4K compared to the original algorithm.

In Fig. 6, we compare the subjective image quality of the results of applying IAM-VFI to RIFE and EMA-VFI with other algorithms. In the first column of Fig. 6a, we can see that IAM-VFI can handle large motions and the second column demonstrated that images with complex motion and detail are also improved. In the first column of Fig. 6b, the occlusion regions of the object and background are sharply interpolated and the face is interpolated clearly. And in the second column, the edge regions are also interpolated sharply.

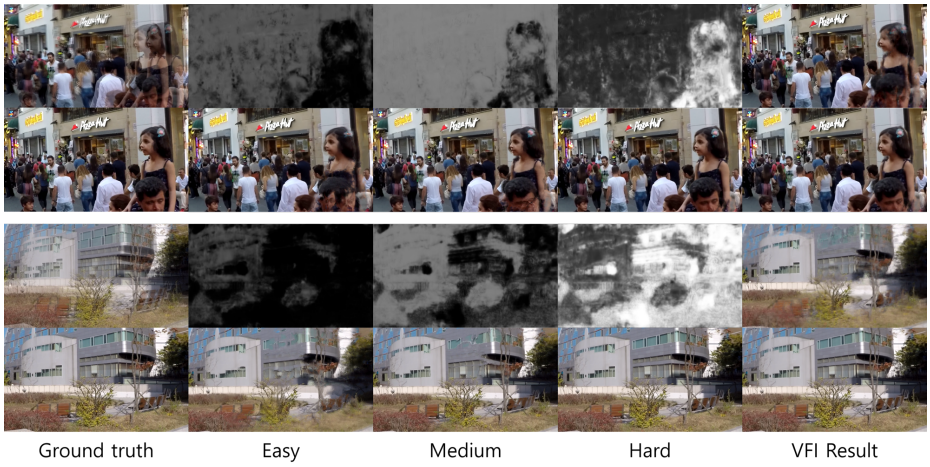


Fig. 7: Subjective image quality comparison for the effectiveness of MCM. The first column shows, from left, the overlap of the two input images, each channel of the MCM_t representing the probabilities of easy, medium, and hard, and the interpolation results of the original RIFE. The second column shows, from left, the ground truth of the intermediate frames to be interpolated, the intermediate frames optimized to easy, medium and hard using the data classification method, and the interpolation results of IAM-VFI.

5.3 Ablation Studies

We performed the ablation study to demonstrate the effectiveness of the proposed methods. For comparison, we used all benchmark datasets and compared the inference time of applying the proposed method to the existing VFI algorithm. In Tab. 4, the baseline without applying all the proposed methods is the original RIFE [12] algorithm.

Effect of MCENet. The second column is the result of simply weighting each of the frames optimized for each class by the classified data by 0.33. We can see that the PSNR degrades for most of the benchmarks because we do not take into account the complexity of each region. In contrast, when we generate an MCM_t using MCENet to estimate the complexity of each region and adaptively weight each region, we see that performance improves compared to the baseline for all benchmarks.

In Fig. 7, we can see the optimized results for each motion and a visualisation of each channel of the MCM_t generated by MCENet. We can see that the regions with large motion are classified as hard motion complexity, and the optimized result with hard data interpolates the large motion regions clearly. Also in the occlusion region, we can see that the motion complexity is classified as hard due to the different motion of the two overlapped objects, and the edges are sharply interpolated. And global motions are classified as easy or medium. Finally, when small objects such as branches move, the distribution of their motions varies,

Table 4: Ablation studies for our proposed methods. M.O stands for Motion Optimisation with classified data. (Red indicates the best PSNR).

$M.O$	$MCENet$	L_c	MAB	Syn_{MCM}	HD	SNU-FILM				UVG	Xiph		Runtime
						Easy	Medium	Hard	Extreme		2K	4K	
×	×	×	×	×	32.08	39.98	35.78	30.14	24.83	31.13	36.15	33.25	48.3ms
✓	×	×	×	×	31.97	38.39	35.15	30.22	25.23	31.60	35.80	33.60	68.3ms
✓	✓	×	×	×	32.32	40.05	36.03	30.59	25.35	32.02	36.33	33.92	78.7ms
✓	✓	✓	×	×	32.42	40.11	36.04	30.58	25.34	32.08	36.56	34.06	78.7ms
✓	✓	✓	✓	×	32.45	40.14	36.05	30.59	25.35	32.15	36.60	34.12	82.7ms
✓	✓	✓	✓	✓	32.46	40.18	36.08	30.60	25.35	32.17	36.60	34.20	87.5ms

so they are classified as complex motions, and you can see that the previously blurred parts are interpolated clearly.

Effect of Motion Complexity Loss. The $\hat{I}_t$ generated by MCM_t contains information about the complexity of each regions. Therefore, we can see that using the loss function for frames with complexity information reflected through MCM_{gt} rather than using the loss with GT is effective.

Effect of MAB. Each channel of MCM represents the probability value for whether the complexity of the corresponding area falls into easy, medium, or hard. Therefore, the PSNR is improved when MAB is used in MCENet to give attention for each channel.

Synthesis Network with MCM. Using the MCM_t as an additional input to the synthesis network to provide information about motion complexity can improve performance on most of the benchmarks.

5.4 Limitations

By applying the proposed method to the existing algorithm, we achieved a significant improvement in interpolation results, but it still has several limitations. First, the flow network needs to be trained for each classified dataset to optimize for motion, resulting in increased training time. Second, compared to the existing network, the runtime increases. In future work, we will attempt to optimize for all motion complexities using a single flownet.

6 Conclusion

This paper aims to address the problem that existing VFI algorithms do not consider motion complexity and therefore fail to perform optimally for all regions. We solve this problem by combining three intermediate frames, which are derived by different flow networks, with motion complexity map. Experimental results show that the proposed method of interpolating frame by considering motion complexity improves the interpolation performance of existing VFI algorithms and achieves enhanced quality for all regions in the video.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00087, Development of high-quality conversion technology for SD/HD low-quality media)

References

1. Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: 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), 933–948 (2019)
2. Barman, N., Zadtootaghaj, S., Schmidt, S., Martini, M.G., Möller, S.: Gamingvideoset: a dataset for gaming video streaming applications. In: 2018 16th Annual Workshop on Network and Systems Support for Games (NetGames). pp. 1–6. IEEE (2018)
3. Brooks, T., Barron, J.T.: Learning to synthesize motion blur. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6840–6848 (2019)
4. Choi, M., Kim, H., Han, B., Xu, N., Lee, K.M.: Channel attention is all you need for video frame interpolation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 10663–10671 (2020)
5. Dar, Y., Bruckstein, A.M.: Motion-compensated coding and frame rate up-conversion: Models and analysis. IEEE Transactions on Image Processing **24**(7), 2051–2066 (2015)
6. Ding, T., Liang, L., Zhu, Z., Zharkov, I.: Cdfi: Compression-driven network design for frame interpolation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8001–8011 (2021)
7. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deepstereo: Learning to predict new views from the world's imagery. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5515–5524 (2016)
8. Fourure, D., Emonet, R., Fromont, E., Muselet, D., Tremeau, A., Wolf, C.: Residual conv-deconv grid network for semantic segmentation. arXiv preprint arXiv:1707.07958 (2017)
9. Fremerey, S., Göring, S., Rao, R.R.R., Huang, R., Raake, A.: Subjective test dataset and meta-data-based models for 360° streaming video quality. In: 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). pp. 1–6. IEEE (2020)
10. Göring, S., Rao, R.R.R., Feiten, B., Raake, A.: Modular framework and instances of pixel-based video quality models for uhd-1/4k. IEEE Access **9**, 31842–31864 (2021)
11. Hu, P., Niklaus, S., Sclaroff, S., Saenko, K.: Many-to-many splatting for efficient video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3553–3562 (2022)
12. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Real-time intermediate flow estimation for video frame interpolation. In: European Conference on Computer Vision. pp. 624–642. Springer (2022)

13. Installations, T., Line, L.: Subjective video quality assessment methods for multimedia applications. Networks **910**(37), 5 (1999)
14. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9000–9008 (2018)
15. Jiang, J., Holz, C.: Restore anything pipeline: Segment anything meets image restoration. arXiv preprint arXiv:2305.13093 (2023)
16. Jin, X., Wu, L., Chen, J., Chen, Y., Koo, J., Hahm, C.h.: A unified pyramid recurrent network for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1578–1587 (2023)
17. Jin, X., Wu, L., Shen, G., Chen, Y., Chen, J., Koo, J., Hahm, C.h.: Enhanced bidirectional motion estimation for video frame interpolation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 5049–5057 (2023)
18. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)
19. Kong, L., Jiang, B., Luo, D., Chu, W., Huang, X., Tai, Y., Wang, C., Yang, J.: Ifrnet: Intermediate feature refine network for efficient frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1969–1978 (2022)
20. Kong, X., Zhao, H., Qiao, Y., Dong, C.: Classsr: A general framework to accelerate super-resolution networks by data characteristic. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12016–12025 (2021)
21. Lee, H., Kim, T., Chung, T.y., Pak, D., Ban, Y., Lee, S.: Adacof: Adaptive collaboration of flows for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5316–5325 (2020)
22. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 136–144 (2017)
23. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: Proceedings of the IEEE international conference on computer vision. pp. 4463–4471 (2017)
24. Lu, L., Wu, R., Lin, H., Lu, J., Jia, J.: Video frame interpolation with transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3532–3542 (2022)
25. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)
26. Montgomery, C., Lars, H.: Xiph. org video test media (derf's collection). Online, https://media. xiph. org/video/derf **6** (1994)
27. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5437–5446 (2020)
28. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE international conference on computer vision. pp. 261–270 (2017)
29. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE international conference on computer vision. pp. 261–270 (2017)

30. Park, J., Ko, K., Lee, C., Kim, C.S.: 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. pp. 109–125. Springer (2020)
31. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14539–14548 (2021)
32. Rad, M.S., Bozorgtabar, B., Marti, U.V., Basler, M., Ekenel, H.K., Thiran, J.P.: Srobb: Targeted perceptual loss for single image super-resolution. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 2710–2719 (2019)
33. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
34. Sim, H., Oh, J., Kim, M.: Xvfi: extreme video frame interpolation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 14489–14498 (2021)
35. Song, L., Tang, X., Zhang, W., Yang, X., Xia, P.: The sjtu 4k video sequence dataset. In: 2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX). pp. 34–35. IEEE (2013)
36. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8934–8943 (2018)
37. Xiao, Z., Bai, J., Lu, Z., Xiong, Z.: A dive into sam prior in image restoration. arXiv preprint arXiv:2305.13620 (2023)
38. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision **127**, 1106–1125 (2019)
39. Yoon, K., Huh, J., Kim, Y.H., Kim, S., Jeong, J.: Textural detail preservation network for video frame interpolation. IEEE Access (2023)
40. Zhang, G., Zhu, Y., Wang, H., Chen, Y., Wu, G., Wang, L.: Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5682–5692 (2023)
41. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 286–301. Springer (2016)