APPLICATIONS OF DEEP LEARNING TO VIDEO ENHANCEMENT

APPLICATIONS OF DEEP LEARNING TO VIDEO ENHANCEMENT

BY

ZHIHAO SHI, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2022)	McMaster University
(Electrical & Computer Engineering)	Hamilton, Ontario, Canada

TITLE:	Applications of Deep Learning to Video Enhancement
AUTHOR:	Zhihao Shi B.Eng. (Telecommunication Engineering), Zhengzhou University, Zhengzhou, China
SUPERVISOR:	Dr. Jun Chen
NUMBER OF PAGES:	xvi, 159

Abstract

Deep learning, usually built upon artificial neural networks, was proposed in 1943, but poor computational capability restricted its development at that time. With the advancement of computer architecture and chip design, deep learning gains sufficient computational power and has revolutionized many areas in computer vision. As a fundamental research area of computer vision, video enhancement often serves as the first step of many modern vision systems and facilitates numerous downstream vision tasks. This thesis provides a comprehensive study of video enhancement, especially in the sense of video frame interpolation and space-time video super-resolution.

For video frame interpolation, two novel methods, named GDConvNet and VFIT, are proposed. In GDConvNet, a novel mechanism named generalized deformable convolution is introduced in order to overcome the inaccuracy flow estimation issue in the flow-based methods and the rigidity issue of kernel shape in the kernel-based methods. This mechanism can effectively learn motion information in a data-driven manner and freely select sampling points in space-time. Our GDConvNet, built upon this mechanism, is shown to achieve the state-of-the-art performance. As for VFIT, the concept of local attention is firstly introduced to video interpolation, and a novel space-time separation window-based self-attention scheme is further devised, which not only saves costs but acts as a regularization term to improve the performance.

Based on the new scheme, VFIT is presented as the first Transformer-based video frame interpolation framework. In addition, a multi-scale frame synthesis scheme is developed to fully realize the potential of Transformers. Extensive experiments on a variety of benchmark datasets demonstrate the superiority and liability of VFIT.

For space-time video super-resolution, a novel unconstrained space-time video super-resolution network is proposed to solve the common issues of the existing methods that either fail to explore the intrinsic relationship between temporal and spatial information or lack flexibility in the choice of final temporal/spatial resolution. To this end, several new ideas are introduced, such as integration of multi-level representations and generalized pixshuffle. Various experiments validate the proposed method in terms of its complete freedom in choosing output resolution, as well as superior performance over the state-of-the-art methods.

To my family, for their love and support.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Jun Chen, for his professional leading, insightful advice, patient guidance, and comprehensive support throughout my Ph.D. study. I'm also deeply impressed by his academic ability and personal character, especially for his talents in math and kindness to everyone. How fortunate I am to be his Ph.D. student.

Sincere appreciation is also expressed to my committee members Prof. Xiaolin Wu and Prof. Shahram Shirani, for their valuable suggestions and comments to my research works. Meanwhile, I am also grateful to Prof. Ling Guan for his willingness being my external examiner.

Moreover, many thanks also go to my lab colleagues, especially to Xiaohong Liu, Chenqi Li, Kangdi Shi, Yongrui Ma, and Linhui Dai for their assistance and friendship.

Last but not least, I would like to thank my parents for their continuous love and endless support in my life. I could not make it without them. I am also thankful to my girlfriend, Xinyi Li, for her love and companionship during my Ph.D. study.

Contents

Abstract **iii**

Acknowledgements **vi**

1 Introduction **1**

1.1 Video Enhancement 1

1.2 Deep Learning 3

1.3 Contributions and Thesis Organization 5

Bibliography 8

2 Video Frame Interpolation via Generalized Deformable Convolution **12**

2.1 Abstract 12

2.2 Introduction 13

2.3 Generalized Deformable Convolution Network 16

2.4 Understanding Generalized Deformable Convolution in VFI 23

2.5 Four-Frame VFI Experiments 28

2.6 Two-Frames VFI Experiments 45

2.7 Conclusion 47

Bibliography 49

3 Video Frame Interpolation Transformer 58

3.1 Abstract 58

3.2 Introduction 59

3.3 Related Work 62

3.4 Proposed Method 64

3.5 Experimental Results 72

3.6 Conclusion 79

Bibliography 81

4 Learning for Unconstrained Space-Time Video Super-Resolution 89

4.1 Abstract 89

4.2 Introduction 90

4.3 Related Work 93

4.4 Unconstrained Space-Time Video Super-Resolution Network 97

4.5 Experiments For Unconstrained Space-Time Video Super-Resolution . 104

4.6 Experiments For Fixed Space-Time Video Super-Resolution 117

4.7 Conclusion 121

Bibliography 122

5 Conclusion 131

5.1 Conclusion 131

List of Figures

2.1 Illustration of (a) conventional convolution with $3\times3\times4 = 36$ sampling points, (b) GDConv with the same number of sampling points, and (c) visualization of interpolating one frame with GDConv. 15

2.2 Illustration of the architecture of GDConvNet with $T = 3$. Here $I_0, \cdots, I_3$ are input frames and $C_0, \cdots, C_3$ are their respective context maps; $\triangle x_n^i$ and $\triangle y_n^i$ are spatial offsets (horizontal and vertical) for support point; $\triangle x_n$, $\triangle y_n$ and $\triangle z_n$ are spatial offsets and temporal parameters for sampling points; $\triangle m_n$ is the modulation terms; I'_t is a tentative prediction of the target frame I_t while C'_t denotes the predicted context map of I_t; $\hat{I}_t$ is the final output. 17

2.3 Illustration of (a) conventional convolution, (b) AdaCoF, (c) basic GDConv, (d) advanced GDConv with $T = 1$. Here target pixels, sampling points, support points, and neighboring grid points are denoted by yellow, red, green, and blue dots, respectively. For AdaCoF, the value of each sampling point is specified via bilinear interpolation of its four neighboring grid points. For basic GDConv, the value of each sampling point is determined by its two support points via linear interpolation, or equivalently, by its eight associated grid points via trilinear interpolation. Advanced GDConv further removes the constraint that the support points need to be spatially aligned with the corresponding sampling point and allows more general numerical interpolation methods. 18

2.4 Construction of function I for the special case $T = 3$ with a sampling point $(x+\triangle x_n, y+\triangle y_n, z_n)$, its associated support points $(x+\triangle x_n^i, y+\triangle y_n^i, i), i \in \{0, 1, 2, 3\}$, and their neighboring grid points highlighted in red, green, and blue, respectively. 20

2.5 Illustration of the architecture of SEM. 22

2.6 Illustration of (a) flow-based VFI pipeline and (b) kernel-based VFI pipeline. 25

2.7 Illustration of (a) 1D version of inverse distance weighted interpolation and (b) polynomial interpolation with support points highlighted in red. Here $s_n^0 = 0.6$, $s_n^1 = 0.8$, $s_n^2 = 0.05$, and $s_n^3 = 0.4$, respectively. . . 34

2.8 Visualization of (a) ground truth, (b) error residual (ER) map generated by GDConvNet with standard polynomial (c) error residual map generated by GDConvNet with clamped polynomial, (d) distribution map of sampling points beyond limits. The error residual map is calculated by $ER = MSE_{1\_inv} - MSE_{poly}$, where MSE_{1_inv} denotes the mean squared error map between the ground truth and the result generated by GDConvNet with 1D Inv, and MSE_{poly} is similarly defined for polynomial interpolation. 36

2.9 Qualitative comparisons of different VFI algorithms. 37

2.10 Visualization of sampling points in GDCM when $t = 1.5$. Here ((a), (b)) and ((c), (d)) illustrate two different pixels in a same target intermediate frame and their associated sampling points respectively. It can be seen that sampling points are not exclusively located between I_1 and I_2. Indeed, there are some between I_1 and I_2, and some between I_2 and I_3. This indicates that the information from I_1 and I_2 is more significant for synthesizing I_t, but I_0 and I_3 also contribute to the synthesized result. 40

2.11 Visualization of failure cases. 44

3.1 Comparison of performance and model size using the Vimeo-90K dataset [14]. VFIT outperforms the state-of-the-art methods with fewer parameters. VFIT-S and VFIT-B denote the proposed small and base models. . . 60

3.2 Overview of the proposed VFIT. We first use an embedding layer to transform input frames into shallow features, followed by a Transformer-based encoder-decoder network to extract deep hierarchical features. These features together with the input frames are fed to a multi-scale frame synthesis network that is composed of three SynBlocks to obtain the final output. "$\downarrow n$" and "$\uparrow n$" denote downsampling and upsampling by a factor of n, respectively. "DConv" represents deformable convolution [29]. Please find more detailed explanations in the text of Section 3.4. 65

3.3 Illustration of different local partition strategies. (a) Regular and shifted partitions for spatial-temporal cubes of STS. (b) Regular and shifted partitions for spatial windows of Sep-STS. (c) Temporal vector partition for Sep-STS. 66

3.4 Illustration of the Sep-STS block. The Spatial MSA and Temporal MSA represent the multi-head self-attentions in spatial and temporal local windows, respectively (Section 3.4.1). 68

3.5 Qualitative comparisons against the state-of-the-art video interpolation algorithms. The VFIT generates higher-quality results with clearer structures and fewer distortions. 75

3.6 Overlap of interpolated frames and the corresponding ground truth, where a clearer overlapped image indicates a more accurate prediction. Note that for the second example, as the predictions of the baseline methods and the ground truth are not well aligned, the overlap of the red and white regions presents a blurry pink color. 75

3.7 Interpolated frames from a video with fast motion. VFIT-CNN produces severe ghosting artifacts due to it incapability of handling large motion, while the result of VFIT-STS appears blurry. In contrast, the VFIT-B generates a higher-quality intermediate frame closer to the ground truth. 78
4.1 Illustration of the architecture of USTVSRNet. 97
4.2 Illustration of the architecture of EnhanceNet. 99
4.3 Examples of the standard and generalized pixelshuffle layers, where (a) shows the standard layer while $C_{\text{in}} = 2, C_{\text{int}} = 4, C_{\text{out}} = 1$, and $s = 2$; (b) shows the generalized pixshuffle layer while $C_{\text{in}} = 2, C_{\text{int}} = 4, C_{\text{out}} = 2$, $s = 1.5$, and $p_c = 0$. 101
4.4 Illustration of the SARDB architecture. 103
4.5 Quantitative comparisons of unconstrained STVSR methods on Adobe240 dataset. 107
4.6 Quantitative comparisons of unconstrained STVSR methods on Gopro dataset. 108
4.7 Qualitative comparisons of different Unconstrained STVSR algorithms. 109
4.8 Qualitative results of ablation study regarding FINet and EnhanceNet. 111
4.9 Qualitative results of ablation study regarding channel dimension freedom. 115

List of Tables

2.1 The statistics of pixel displacement within different datasets. This table shows the average pixel displacement, the percentage of pixels with displacement larger than 15, 20 and 25 respectively for three datasets. 30

2.2 Quantitative comparisons of GDConvNet with different numerical interpolation methods on Viemo-90k test dataset, Gopro dataset, and Adobe240 dataset. 32

2.3 Mean of the squared distance. 33

2.4 The statistical distribution of sampling points beyond limits. 35

2.5 Quantitative comparisons of different VFI methods on Vimeo90K Septulet test set, Gopro dataset and Adobe240 dataset, where the first place and second place are highlighted in red and blue, respectively. . 38

2.6 Comparisons of DConv, GDConv with different numbers of sampling points, and some variants of GDConv. 39

2.7 Comparisons for different numbers of reference frames (with the number of generation frames set to be the same as that of reference frame). 42

2.8 Comparisons for different numbers of generation frames (with the reference frames fixed to be I_1 and I_2). 43

2.9 Quantitative comparisons on Vimeo90K interpolation test set, UCF101 dataset and Middlebury-Other dataset, where the first place and second place are highlighted in red and blue, respectively. 47

3.1 Quantitative comparisons on the Vimeo-90K, UCF101, and DAVIS datasets. Numbers in bold indicate the best performance and underscored numbers indicate the second best. 73

3.2 Run-time of evaluated methods in seconds per frame. The models are tested on a desktop with an Intel Core i7-8700K CPU and an NVIDIA GTX 2080 Ti GPU. The results are averaged on the Vimeo-90K dataset. 74

3.3 Effectiveness of the proposed Sep-STS block. 76

3.4 Comparison with the base models under different motion conditions. . 76

3.5 Comparison with Transformer-based resizing modules. 79

4.1 Model size and running time comparisons with $s = 4$, where the model size is reported in millions (M) and the running time is reported in second (s) per frame. 110

4.2 Quantitative results of ablation study regarding FINet and EnhanceNet with $s = 1, 2, 3, 4$, where PSNR and SSIM scores are averaged over t. 111

4.3 Comparisons between SPL and GPL in terms of the fixed scale upsampling on the Vimeo90k dataset. 113

4.4 Comparisons for different unconstrained upscale methods on Vimeo90k dataset. 115

4.5 Comparisons of FG-RDN and G-RDN with different C_{mid}. 116

4.6 Quantitative comparisons on Adobe240 dataset, Gopro dataset, and Vimeo90K dataset with $s = 4$, where 'Center' denotes the PSNR and SSIM value of the center frame $\hat{I}^H_{0.5}$ while 'Whole' is for average values of $\hat{I}^H_0$, $\hat{I}^H_{0.5}$, and $\hat{I}^H_1$. We highlight the first place and the second place in red and blue, respectively. 120

Chapter 1

Introduction

1.1 Video Enhancement

The spatial and temporal resolutions of a video is usually limited by the camera, specifically, the spatial resolution is determined by the total number of sensors, and the temporal resolution is dependent on the frame-rate and exposure time of the camera. Therefore, high-definition video quality and affordability of the camera form a trade-off for many end-users. The need for visually-pleasing video quality and low camera cost naturally calls for video enhancement, which aims at increasing spatial and temporal resolutions of a given video without resorting to any hardware upgrades. Video enhancement is a fundamental research area as it can be used in a wide range of applications including medical imaging [1], surveillance [2] and even some high-level vision applications [3] as a pre-processing step to improve the quality of the corresponding data. In general, video enhancement can be divided into three major categories: Video Super-Resolution (VSR), Video Frame Interpolation (VFI), and Space-Time Video Super-Resolution (STVSR).

VSR aims at recovering high-resolution (HR) frames from corresponding low-resolution (LR) given frames by increasing the number of pixels in the spatial domain. As an advantage over single-image super-resolution (SISR), VSR has access to multiple highly-related but misaligned frames, leading to a better visual performance than SISR. However, how to effectively aggregate information from nearby frames becomes an extra challenge for VSR. Typical VSR methods first extract features from the input frames, and then send them to the reconstruction network to generate super-resolved frames after proper alignment and fusion. By elaborately designing each module and carefully finetuning the overall network, typical VSR methods have been shown to generate satisfactory results in terms of efficiency and effectiveness on various datasets.

The goal of VFI is to synthesize intermediate frames from the given input frames, which can temporally upsample low-frame rate videos to higher-frame rate ones. There are two major categories of video interpolation methods: kernel-based and flow-based methods. Kernel-based methods generate intermediate frames by employing a set of spatial-adaptive convolution kernels to convolve with the input frames while flow-based methods achieve this goal by warping pixels from the given frames according to associated optical flows. These two categories of VFI methods are proved to be effective and reliable even though they reply on different approaches.

Distinct from VFI and VSR that only focus on one aspect of video, STVSR seeks to simultaneously increase the temporal frame rate and the spatial resolution of a given video, which is regarded as unifying VFI and VSR into one framework. A straightforward way is to directly combine state-of-the-art VFI methods and VSR methods, termed two-stage methods. However, the performance of two-stage methods is limited since the intrinsic relationship between temporal and spatial information is barely exploited. Moreover, the computational complexity of the two-stage methods is extremely high since state-of-the-art VFI and VSR methods are already computationally intensive and likely involve many redundant operations (*e.g.*, extracting features). To solve this problem, many one-stage methods have been proposed, which behave better than two-stage methods in terms of both performance and costs.

1.2 Deep Learning

Deep learning is usually based on Artificial Neural Networks (ANN), which was firstly proposed in 1943 [4] and designed to imitate biological neural networks. Through stacking multiple layers where each layer acts as a simple non-linear transformation, the ANN is able to approximate relative complex transformation. Moreover, 'deep' here emphasizes the number of stacked layers can be extremely huge (*e.g.*, 1000). As such, the DNN is believed to have the capability of achieving arbitrary non-linear transformation.

Although DNN has the potential of approximating arbitrary transformation, a careful tuning process from the initial state is needed and this process is often termed as 'training'. The training process usually takes an enormous amount of calculations, which is unaffordable for the hardware of the last century. That is the main reason why ANN was proposed early but no significant breakthrough was made. Recently, with the development of modern Cpu (*e.g.*, Intel Core series) and Gpu (*e.g.*, Nvidia series), sufficient computational power becomes available. In 2012, AlexNet [5], a deeper version of LeNet [6] which had been proposed in the last century, broke multiple AI records with a significant margin, which attracted many researchers' attention.

Later, as more researchers join the area, more large-scale datasets are constructed and more advanced algorithms are presented, which make significant contributions to the development of the modern AI area.

In general, there are four typical categories of neural network in deep learning:

1. Fully Connected Neural Network (FCN);
2. Convolutional Neural Network (CNN);
3. Recurrent Neural network (RNN).
4. Transformer.

FCN, also known as multi-layer perception machine (MLP), is composed of multiple layers of neurons, where the neurons from consecutive layers are fully-connected, leading to a better capability of learning complex transformations but with a large model size and computational complexity. FCN has been utilized in many tasks including, among others, fraud detection [7], e-mail span filtering [8], and sales forecasting [9].

Different from FCN, CNN replaces the fully-connected neurons with sparse convolutional kernels, thus dramatically reducing the total number of parameters and computational complexity. Since the kernel-based scheme is particularly suited to process array-like data (*e.g.*, images), CNN has been successfully applied in the area of computer vision (CV). The typical applications of CNN in CV include image classification [10], image segmentation [11], object detection [12], facial recognition [13], etc.

As to RNN, it can be regarded as a recursive variant of FCN, where the output layer of the previous time-step is recursively connected to the input layer of the current time-step and the hidden layers are recurrently connected along the time-steps. This characteristic allows RNN to memorize previous information, thus is widely used in the areas of natural language processing (NLP), handwriting recognition [14], and time series analysis. However, the transmitted hidden information may attenuate or even vanish when the time-steps are too long, causing the short-term memory problem. To address this problem, Gated Recurrent Units (GRU) [15] and Long Short-Term Memory (LSTM) [16] are proposed, where the various gates are introduced to keep hidden information from the far time-step still effective.

Recently, [17] proposes a novel network architecture, named Transformer, which provides an additional choice besides the above three. The new architecture purely relies on the self-attention mechanism and has shown promising results in the NLP area. Motivated by the success in the NLP area, researchers seek to adapt transformer into the CV area and have gained noticeable success, such as in the area of image classification [18] and image processing [19].

Since the focus of this thesis is on the applications of deep learning to video enhancement, CNN and Transformer are chosen as the basic backbone.

1.3 Contributions and Thesis Organization

This thesis contributes to video enhancement based on deep learning, which is composed of three articles in a *sandwich thesis* format following the terms and regulations of McMaster University. The reference information of these three articles is listed below:

1. **Zhihao Shi**, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video

frame interpolation via generalized deformable convolution. In IEEE Transactions on Multimedia, 2021.

2. **Zhihao Shi**, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

3. **Zhihao Shi**, Xiaohong Liu, Chengqi Li, Linhui Dai, Jun Chen, Timothy N. Davidson, and Jiying Zhao. Learning for Unconstrained Space-Time Video Super-Resolution. In IEEE Transactions on Broadcasting, 2021.

The first two articles belong to the topic of video frame interpolation, and the last article focuses on space-time video super-resolution.

More specifically, the first article is to tackle the VFI task, named Generalized Deformable Convolution Network. A novel mechanism named generalized deformable convolution is proposed, which can effectively learn motion information in a data-driven manner and freely select sampling points in space-time. Furthermore, a new video frame interpolation method is developed based on this mechanism. The extensive experiments demonstrate that the new method performs favorably against the state-of-the-art, especially when dealing with complex motions.

The second article is also about VFI, but pays attention to designing network backbone. In contrast to existing methods for video interpolation that heavily rely on deep convolution neural networks, a Transformer-based video interpolation framework is proposed, which allows content-aware aggregation weights and considers long-range dependencies with the self-attention operations. Moreover, a multi-scale frame synthesis scheme is developed to further improve the performance. Extensive experiments demonstrate the proposed model performs favorably against the state-of-the-art methods both quantitatively and qualitatively on a variety of benchmark datasets.

As for the last article, it focuses on realizing VFI and VSR simultaneously, *i.e.*, STVSR. An unconstrained one-stage space-time video super-resolution network is proposed, which can effectively exploit space-time correlation to boost performance. Moreover, it has complete freedom in adjusting the temporal frame rate and spatial resolution. Extensive experiments demonstrate that the proposed method not only outperforms the state-of-the-art but also requires far fewer parameters and less running time.

Bibliography

[1] H. Greenspan, “Super-resolution in medical imaging,” *The computer journal*, vol. 52, no. 1, pp. 43–63, 2009.

[2] B. K. Gunturk, A. U. Batur, Y. Altunbasak, M. H. Hayes, and R. M. Mersereau, “Eigenface-domain super-resolution for face recognition,” *IEEE transactions on image processing*, vol. 12, no. 5, pp. 597–606, 2003.

[3] S. Hao, W. Wang, Y. Ye, E. Li, and L. Bruzzone, “A deep network architecture for super-resolution-aided hyperspectral image classification with classwise loss,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4650–4663, 2018.

[4] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, pp. 1097–1105, 2012.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] R. Patidar, L. Sharma *et al.*, “Credit card fraud detection using neural network,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 1, no. 32-38, 2011.

[8] A. Alghoul, S. Al Ajrami, G. Al Jarousha, G. Harb, and S. S. Abu-Naser, “Email classification using artificial neural network,” 2018.

[9] D. J. Livingstone, *Artificial Neural Networks: Methods and Applications.* Springer, 2008.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[11] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[13] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (ICCV)*, 2019, pp. 4690–4699.

[14] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[15] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2014, p. 17241734.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the International Conference on Learning Representations*, 2020.

[19] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

The following chapter is reproduced from an IEEE published paper:

Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. "Video Frame Interpolation via Generalized Deformable Convolution." In IEEE Transactions on Multimedia, 2021.

Contribution Declaration: Zhihao Shi (the author of this thesis) is the first author and the main contributor to this paper (more than 80%). He proposed the algorithm, wrote the corresponding program, conduct all experiments, and composed the manuscript. Xiaohong Liu helped revise the manuscript; Kangdi Shi and Linhui Dai helped organize the experimental data; Jun Chen helped revise the manuscript.

Chapter 2

Video Frame Interpolation via Generalized Deformable Convolution

2.1 Abstract

Video frame interpolation aims at synthesizing intermediate frames from nearby source frames while maintaining spatial and temporal consistencies. The existing deep-learning-based video frame interpolation methods can be roughly divided into two categories: flow-based methods and kernel-based methods. The performance of flow-based methods is often jeopardized by the inaccuracy of flow map estimation due to oversimplified motion models, while that of kernel-based methods tends to be constrained by the rigidity of kernel shape. To address these performance-limiting issues, a novel mechanism named generalized deformable convolution is proposed, which can effectively learn motion information in a data-driven manner and freely select sampling points in space-time. We further develop a new video frame interpolation method based on this mechanism. Our extensive experiments demonstrate that the new method performs favorably against the state-of-the-art, especially when dealing with complex motions. Code is available at https://github.com/zhshi0816/GDConvNet.

2.2 Introduction

In recent years, owing to the hardware development and the availability of large-scale datasets, deep learning has achieved promising results in many computer vision and multimedia tasks [1] including, among others, super-resolution [2–4], optical flow estimation [5, 6], image dehazing [7, 8], action recognition [9], and VFI [10–13]. VFI is a classic problem in the multimedia area and has received significant attention with the rapid growth of streaming videos. It aims at synthesizing intermediate frames from nearby sources while maintaining spatial and temporal consistencies. VFI has two main use cases; one is to perform error concealment at the decoder side [10, 11], and the other one is to increase the frame rate of a given video for better visual performance [12, 13]. In general, VFI methods can be roughly divided into two categories: flow-based methods and kernel-based methods.

Flow-based methods generate the value of each pixel in the target intermediate frame by finding an associated optical flow. Accurate estimation of the flow map is essential for producing desirable VFI results. However, in some cases with complex motions, it is hard to obtain an accurate flow map regardless whether traditional methods [14–16] or deep-learning-based methods [6, 17–19] are employed. Flow-based methods [1, 11, 20–22] typically adopt a linear model with the oversimplified assumption of uniform motion between neighboring frames. Recently, a more sophisticated approach was proposed in [23] for estimating motion trajectories, where the naive linear model is replaced by a more accurate quadratic model that can take advantage of latent motion information by simultaneously exploiting four consecutive frames. Nevertheless, it is conceivable that the complexities and irregularities of real-world motions cannot be completely captured by a simple mathematical model. Moreover, the pixel-level displacement performed in flow-based methods is inherently inadequate for handling diffusion and dispersion effects, especially when such effects are not negligible over the time interval between two consecutive frames.

Kernel-based methods directly generate the target intermediate frame by applying spatially-adaptive convolution kernels to the given frames. They circumvent the need for flow map estimation and consequently are not susceptible to the associated issues. On the other hand, the rigidity of the kernel shape [24,25] severely limits the types of motions that such methods can handle. Indeed, one may need to choose a very large kernel size to ensure enough coverage, which is highly inefficient. As a partial remedy, reference [21] proposes adaptive deployment of convolution kernels guided by flow maps, but nevertheless, the receptive field is still constrained by the predetermined kernel shape. More recently, reference [12] introduces a new approach known as AdaCoF, which utilizes spatially-adaptive deformable convolution (DConv) to select suitable sampling points needed for synthesizing each target pixel. Although this approach eliminates the constraint on the kernel shape in the spatial domain, it does not fully exploit the degrees of freedom available in whole space-time.

In summary, flow-based methods and kernel-based methods have their respective limitations. For flow-based methods, even with the aid of sophisticated mathematical

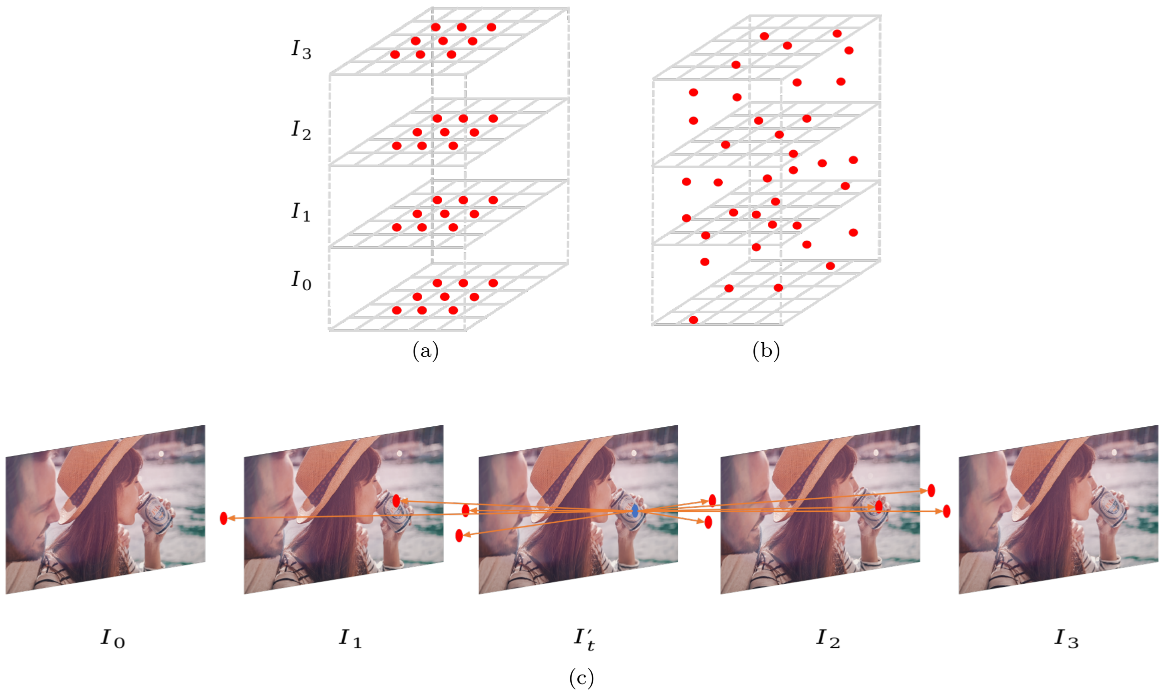


Figure 2.1: Illustration of (a) conventional convolution with $3 \times 3 \times 4 = 36$ sampling points, (b) GDConv with the same number of sampling points, and (c) visualization of interpolating one frame with GDConv.

models, flow map estimation is still a challenging task due to the intricacies of inter-frame motion trajectories. For kernel-based methods, the predetermined kernel shape lacks the flexibility to cope with a great variety of motions in terms of range and pattern. While recent innovations have alleviated the rigidity issue to a certain extent, much remains to be done.

The main contribution of this paper is a new approach to VFI that overcomes the hurdles of the aforementioned methods and retains their desirable properties. The key mechanism underlying the proposed approach is generalized deformable convolution (GDConv). An illustration of the difference between conventional convolution and our GDConv in terms of the freedom to select sampling points can be found in Fig. 2.1(a) and (b). Fig. 2.1(c) provides a rough idea of how GDConv can be leveraged for VFI:

each pixel (e.g., the blue one) in the target intermediate frame is synthesized based on the corresponding sampling points (the red ones). It is worth noting that as the sampling points are allowed to move freely in the continuous space-time, the receptive field of GDConv is basically unconstrained, making it possible to handle all kinds of motions (say, large motions). Moreover, GDConv does not directly adopt a predetermined mathematical model (e.g., linear or quadratic model) for motion estimation. Instead, it is trained to learn real-world motion trajectories and patterns via a data-driven approach. In our design, GDConv is encapsulated in a generalized deformable convolution module (GDCM). We integrate two GDCMs with several other modules, including the source extraction module (SEM), the context extraction module (CEM) and the post-processing module (PM), to form a generalized deformable convolution network (GDConvNet) for VFI. Our extensive experimental results demonstrate that owing to the effective design, the proposed GDConvNet performs favorably against the current state-of-the-art.

2.3 Generalized Deformable Convolution Network

The overall architecture of GDConvNet is shown in Fig. 2.2. Given a video clip that consists of $T+1$ source frames[1] $I_0,\ I_1,\ \cdots,\ I_T$, the task of GDConvNet is to synthesize an intermediate frame $I_t,\ t \in [0,T]$. To this end, it first generates source features through SEM and extracts context maps $C_0,\ C_1,\ \cdots,\ C_T$ through CEM from $I_0,\ I_1,\ \cdots,\ I_T$. The input frames and context maps are then warped by two separate GDCMs according to the same source features. Finally, the warped frame

[1] For notional simplicity, we assume that the source frames are equally spaced in time. However, the proposed framework can in fact handle the unequal spacing case as well.

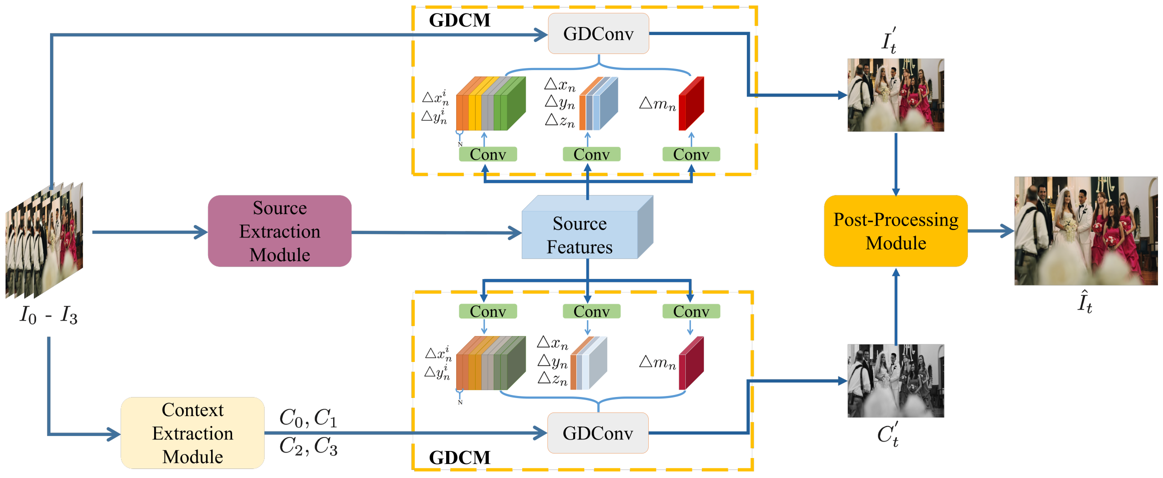


Figure 2.2: Illustration of the architecture of GDConvNet with $T = 3$. Here $I_0, \cdots, I_3$ are input frames and $C_0, \cdots, C_3$ are their respective context maps; $\triangle x_n^i$ and $\triangle y_n^i$ are spatial offsets (horizontal and vertical) for support point; $\triangle x_n$, $\triangle y_n$ and $\triangle z_n$ are spatial offsets and temporal parameters for sampling points; $\triangle m_n$ is the modulation terms; I_t' is a tentative prediction of the target frame I_t while C_t' denotes the predicted context map of I_t; $\hat{I}_t$ is the final output.

I_t' and the warped context map C_t' are fed into the PM to produce the VFI result $\hat{I}_t$, which is an approximation[2] of I_t. The proposed network accomplishes the VFI task by employing a novel GDConv mechanism. Now we proceed to give a detailed description of each module in Fig. 2.2, with a special emphasis on the GDCM where the GDConv mechanism is realized.

2.3.1 Generalized Deformable Convolution Module

The input to the GDCM consists of the $T + 1$ source frames I_0, I_1, $\cdots$, I_T (or the context maps C_0, C_1, $\cdots$, C_T) and the source features. As shown in Fig. 2.2, three different kinds of feature maps, which represent three different types of adaptive parameters, are generated through three different convolution layers, respectively.

[2]The accuracy of this approximation can be evaluated by using objective image quality metrics (to be detailed later) or subjective criteria.

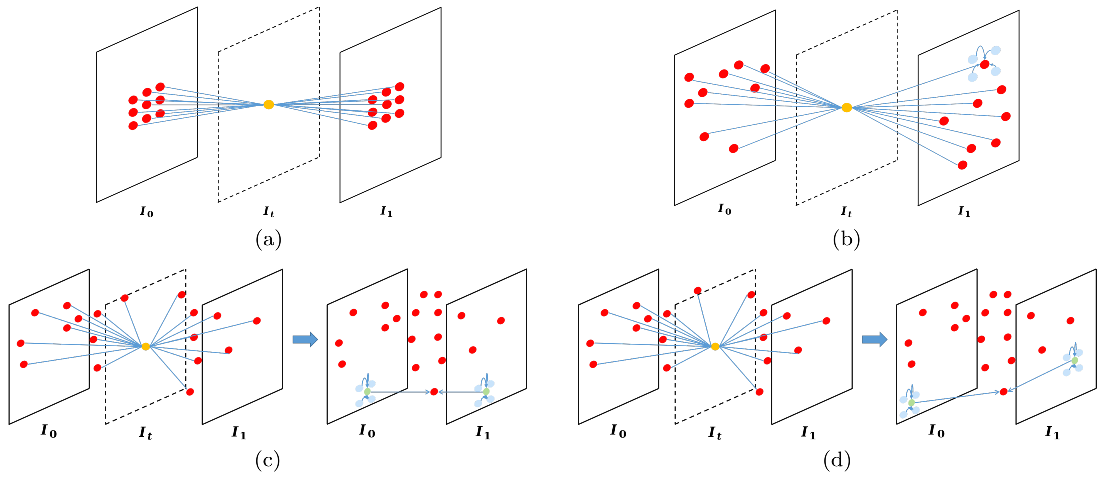


Figure 2.3: Illustration of (a) conventional convolution, (b) AdaCoF, (c) basic GDConv, (d) advanced GDConv with $T = 1$. Here target pixels, sampling points, support points, and neighboring grid points are denoted by yellow, red, green, and blue dots, respectively. For AdaCoF, the value of each sampling point is specified via bilinear interpolation of its four neighboring grid points. For basic GDConv, the value of each sampling point is determined by its two support points via linear interpolation, or equivalently, by its eight associated grid points via trilinear interpolation. Advanced GDConv further removes the constraint that the support points need to be spatially aligned with the corresponding sampling point and allows more general numerical interpolation methods.

They are then fed to GDConv along with the source frames I_0, I_1, $\cdots$, I_T (or the context maps C_0, C_1, $\cdots$, C_T) to synthesize I'_t (or C'_t). Since the two GDCMs are almost identical, here we only describe the upper one in detail. Moreover, as the operations on the three color channels are the same, we simply regard I_i as a single-channel image. For ease of exposition, we first give a brief review of VFI techniques based on conventional convolution [24] and AdaCoF [12], and then outline the improvements offered by the proposed GDConv.

Conventional convolution is employed in [24] for VFI. This can be formulated as:

$$I'_t(x, y) = \sum_{i=0}^{T} \sum_{m=1}^{M} W_m^i(x, y) \cdot I_i(x + x_m, y + y_m), \tag{2.3.1}$$

where $W^i_m(x,y)$ is a spatially-adaptive convolution weight, and $\{(x_m, y_m)\}_{m=1}^N$ is a collection of pre-defined convolution sampling offsets. Fig. 2.3(a) provides an illustration for the special case with $T = 1$, $M = 9$ and $\{(x_m, y_m)\}_{m=1}^M = \{(-1,-1),(-1,0),\cdots,(1,1)\}$. Ideally, the object (pixel) movement should be confined within the coverage of the convolution kernel. As such, in the presence of large motions, this approach is memory-inefficient due to the need for a large number of sampling points to ensure sufficient coverage.

The inefficiency of conventional convolution is largely a consequence of the pre-defined kernel shape (typically, a rectangular grid). AdaCoF [12] addresses this issue by adopting spatially-adaptive deformable convolution, resulting in the following formulation:

$$I'_t(x,y) = \sum_{i=0}^{T}\sum_{m=1}^{M} W^i_m(x,y) \cdot I_i(x + \triangle\alpha^i_m, y + \triangle\beta^i_m), \tag{2.3.2}$$

where $\{(\triangle\alpha^i_m, \triangle\beta^i_m)\}_{m=1}^M$ is a collection of adaptive sampling offsets. In the case where $\triangle\alpha^i_m$ and $\triangle\beta^i_m$ are not integers, $I_i(x + \triangle\alpha^i_m, y + \triangle\beta^i_m)$ is specified through bilinear interpolation. As a result of the introduction of adaptive sampling offsets, the kernel shape becomes adjustable, as shown in Fig. 2.3(b). For this reason, AdaCoF is able to cope with large motions using a relatively small number of sampling points. On the other hand, AdaCoF only exploits the degrees of freedom in the spatial domain. As a result, the sampling points are evenly split among the input frames. However, this is clearly suboptimal since the frames that are closer to the target intermediate frame in the temporal domain are more relevant and consequently should be allocated with more sampling points.

We shall develop a mechanism that enables flexible allocation of the sampling

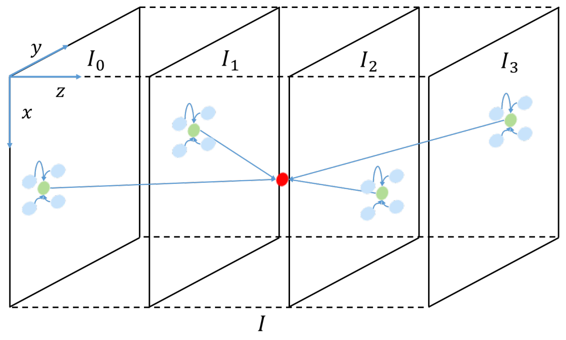


Figure 2.4: Construction of function I for the special case $T = 3$ with a sampling point $(x + \triangle x_n, y + \triangle y_n, z_n)$, its associated support points $(x + \triangle x_n^i, y + \triangle y_n^i, i), i \in \{0, 1, 2, 3\}$, and their neighboring grid points highlighted in red, green, and blue, respectively.

points across the input frames. In fact, we go one step further by allowing the sampling points to be freely distributed in whole space-time. The key idea is to associate each sampling point with an adaptive temporal parameter $z_n \in [0, T]$, leading to the following formulation:

$$I'_t(x, y) = \sum_{n=1}^{N} W_n(x, y) \cdot I(x + \triangle x_n, y + \triangle y_n, z_n). \tag{2.3.3}$$

Here, I is a function (defined on a 3D space) obtained via a judicious extension of I_0, I_1, $\cdots$, I_T to be detailed below (see Fig. 2.4 for an illustration of the special case in which $T = 3$). Note that z_n is allowed to be any real number in $[0, T]$ to facilitate end-to-end training. If z_n is an integer, we set $I(x + \triangle x_n, y + \triangle y_n, z_n) = I_{z_n}(x + \triangle x_n, y + \triangle y_n)$. (Following [12,26,27], in the case where $\triangle x_n$ and $\triangle y_n$ are not integers, $I_{z_n}(x + \triangle x_n, y + \triangle y_n)$ is specified via bilinear interpolation of four neighboring grid points.) It can be seen that Eq. (2.3.3) reduces to Eq. (2.3.2) when $N = (T+1)M$ and each value in $\{0, 1, \cdots, T\}$ is taken by the same number of z_n. Now it remains to deal with non-integer valued z_n, which occurs when the associated sampling point is not exactly located on an input frame. One simple solution is to set $I(x+\triangle x_n, y+\triangle y_n, z_n)$ as $(\lceil z_n \rceil - z_n) \cdot (I(x+\triangle x_n, y+\triangle y_n, \lfloor z_n \rfloor) + (z_n - \lfloor z_n \rfloor) \cdot I(x+\triangle x_n, y+\triangle y_n, \lceil z_n \rceil)$. (See Fig. 2.3(c) for an illustration of the special case in which $T = 1$.) More generally, we attach a set of support points $(x+\triangle x_n^i, y+\triangle y_n^i, i)$, $i \in \{0, 1, \cdots, T\}$, to each sampling point $(x+\triangle x_n, y+\triangle y_n, z_n)$, and use their values $I(x+\triangle x_n^i, y+\triangle y_n^i, i)$ (denoted as s_n^i for short), $i \in \{0, 1, \cdots, T\}$, and their relative positions, to specify $I(x+\triangle x_n, y+\triangle y_n, z_n)$ (denoted as s_n for short) via a numerical interpolation function G:

$$s_n = G(\triangle x_n, \triangle y_n, z_n, \{s_n^i, \triangle x_n^i, \triangle y_n^i\}_{i=0}^{T}). \tag{2.3.4}$$

Illustrations of special cases with $T = 1$ and $T = 3$ can be found in Fig. 2.3(d) and Fig. 2.4, respectively. Note that each support point has its own adaptive spatial offset $(\triangle x_n^i, \triangle y_n^i)$, which is not necessarily the same as $(\triangle x_n, \triangle y_n)$. Moreover, there is considerable freedom in the choice of G as long as the differentiability condition needed for end-to-end training is satisfied. We will discuss several candidate numerical interpolation methods in Section 2.5.3. Finally, inspired by modulated deformable convolution [27], we rewrite Eq. (2.3.3) in the following equivalent form:

$$I'_t(x, y) = \sum_{n=1}^{N} W_n \cdot I(x+\triangle x_n, y+\triangle y_n, z_n) \cdot \triangle m_n(x, y), \tag{2.3.5}$$

where $\triangle m_n(x, y) \in [0, 1]$ is an adaptive modulation term.

As illustrated in Fig. 2.2, three types of feature maps are generated in GDCM via three different convolution layers. The first $2(T+1)N$ feature maps represent the spatial offsets (horizontal and vertical) for the support points (i.e., $\triangle x_n^i$, $\triangle y_n^i$), and the next $3N$ feature maps represent the spatial offsets and temporal parameters for

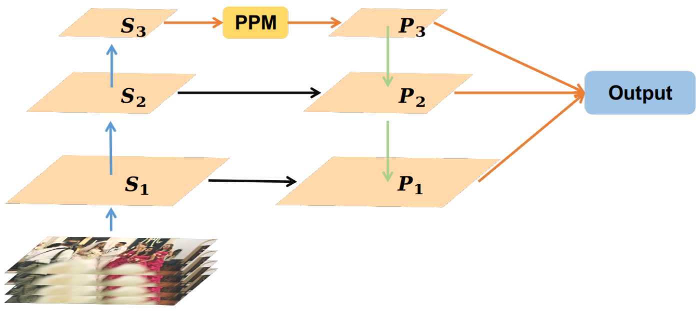


Figure 2.5: Illustration of the architecture of SEM.

the sampling points (i.e., $\triangle x_n$, $\triangle y_n$, z_n), and the last N feature maps represent the modulation terms (i.e., $\triangle m_n$). We set the initial values of the adaptive parameters $\triangle x_n$, $\triangle y_n$, z_n, $\triangle m_n$, $\triangle x_n^i$ and $\triangle y_n^i$ as 0, 0, 0, 1, 0, and 0, respectively.

2.3.2 Other Modules

Now we proceed to give a brief description of the remaining modules in the proposed GDConvNet.

Source Extraction Module: As shown in Fig. 2.5, we adopt the FPN backbone [28] to generate hierarchical features. In the bottom-up pathway, there are three levels (each consisting of two residual blocks and one convolution layer) and the associated feature maps (which are of different scales) are denoted as S_1, S_2, and S_3. The input P_3 to the top level of the top-down pathway is generated from S_3 through a pyramid pooling module [29]. P_3 is then upsampled and merged with S_2 via element-wise addition to generate P_2, which is further upsampled and merged with S_1 to generate P_1. Finally, P_2 and P_3 are upsampled and concatenated with P_1 to form the output.

Context Extraction Module: It is demonstrated in [30] that context information is very important for VFI. We use one convolution layer and two residual blocks [31] to sequentially extract contextual features. A SEblock [32] is then used to rearrange these feature maps, and finally its output is smoothed by a convolution layer.

Post-Processing Module: To refine the warped image, we adopt the GridDehazeNet architecture [33], where each row is associated with a different scale and contains five RDB blocks [34], while each column can be considered as a bridge connecting different scales through downsampling or upsampling modules. (which decrease or increase the size of feature maps by a factor of two.) Instead of employing the hard attention mechanism in [33], we use SEBlocks [32] to adaptively rebalance the incoming information flows at the junctions of GridDehazeNet.

2.4 Understanding Generalized Deformable Convolution in VFI

In this section, we shall place generalized deformable convolution in a board context and explain why it is an effective mechanism for VFI.

2.4.1 Related Works

Generalized deformable convolution is conceptually related to several existing ideas in the literature.

Deformable Convolution: There are many works on variants of conventional convolution with improved performance, including active convolution [35], dynamic filter [36], atrous convolution [37], among others. A culminating achievement of this line of research is deformable convolution [26, 27]. Our generalized deformable convolution degenerates to conventional deformable convolution [26, 27] if the temporal dimension is not present, and its basic form, shown in Fig. 2.3(c), can be viewed as a $3D$-version of deformable convolution.

Non-Local Network: In deep learning, non-locality means that the receptive field is not restricted to a certain local region and can capture long-range context information. The receptive field of conventional convolution is typically a fixed grid and consequently is local in nature. Significant efforts have been devoted to addressing this issue [29,37–39]. Arguably the most successful one is [39], which takes all possible spatial positions into consideration. However, this comes at the cost of high memory usage. In contrast, generalized deformable convolution is memory-efficient as it is able to achieve non-local coverage and capture long-range context information with a relatively small kernel by adaptively and intelligently selecting sampling points in space-time.

Attention Mechanism: An attention mechanism enables differentiated treatment of different input features according to their relative importance, which has shown to yield significant performance gain in many vision tasks. Traditionally, it can be divided into spatial-wise attention [40] and channel-wise/temporal-wise attention [32]. Recently, there have also been attempts [41, 42] to combine these two types of attention. Nevertheless, in these approaches the spatial-wise and channel-wise/temporal-wise attention maps are still generated separately. It is interesting to note that generalized deformable convolution offers a natural way to consolidate these two types of attention by suitably modulating the sampling points at different locations in space-time.

Non-Linearity: The conventional approach to increasing the non-linearity of convolutional neural networks [43–45] is by stacking more non-linear modules [45, 46].

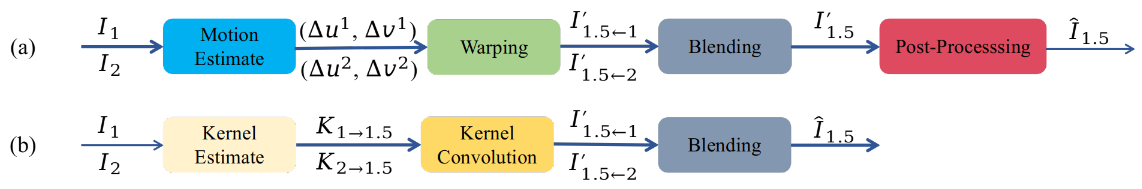


Figure 2.6: Illustration of (a) flow-based VFI pipeline and (b) kernel-based VFI pipeline.

However, it has been recognized that a more effective approach is to allow the functionalities of constituent modules to be input-dependent [32, 40, 47]. From this perspective, generalized deformable convolution converts a linear convolution operation into a highly non-linear operation by adaptively adjusting its kernel according to the input, and by doing so it yields enhanced learning capabilities.

2.4.2 Comparison with State-of-the-Art VFI Algorithms

The state-of-the-art VFI methods can be divided into two categories: flow-based methods and kernel-based methods. For illustrative purposes, we shall consider the simple scenario where two source frames I_1 and I_2 are used to predict one target frame $I_{1.5}$, unless specified otherwise.

Flow-based: These methods admit a common mathematical formulation as follows:

$$I'_{1.5\leftarrow 1}(x, y) = I_1(x + \triangle u^1, y + \triangle v^1), \tag{2.4.1}$$

or

$$I'_{1.5\leftarrow 2}(x, y) = I_2(x + \triangle u^2, y + \triangle v^2), \tag{2.4.2}$$

where $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ are respectively optical flow fields from $I_{1.5}$ to I_1 and I_2, while $I'_{1.5\leftarrow 1}$ and $I'_{1.5\leftarrow 2}$ denote the warped images from each direction. The pipeline of flow-based methods is illustrated in Fig. 2.6(a). First, two input frames are used to estimate optical flow maps, typically with the help of traditional optical flow estimation methods [14–16] or convolution neural network [6, 17–19]. The input frames are then warped according to these optical flow maps. Finally, blending and post-processing operations are performed to generate the final output. The linear motion model is widely adopted in flow map estimation. However, this model is not accurate for describing accelerated and curvilinear motions. To handle such complex motions, a quadratic model is proposed in [23], where $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ are estimated based on four frames I_0, I_1, I_2, and I_3 instead of just I_1 and I_2. To understand the connection with our method, it is instructive to consider a special case of Eq. (2.3.5) with $N = 1$, where $z_1 = 1$, $(\triangle x_1, \triangle y_1) = (\triangle x_1^1, \triangle y_1^1)$, or $z_1 = 2$, $(\triangle x_1, \triangle y_1) = (\triangle x_1^2, \triangle y_1^2)$:

$$\begin{aligned} I'_{1.5\leftarrow 1}(x, y) &= W_1 \cdot I(x + \triangle x_1, y + \triangle y_1, 1) \cdot \triangle m_1 \\ &= W_1 \cdot I_1(x + \triangle x_1^1, y + \triangle y_1^1) \cdot \triangle m_1, \end{aligned} \tag{2.4.3}$$

or

$$\begin{aligned} I'_{1.5\leftarrow 2}(x, y) &= W_1 \cdot I(x + \triangle x_1, y + \triangle y_1, 2) \cdot \triangle m_1 \\ &= W_1 \cdot I_2(x + \triangle x_1^2, y + \triangle y_1^2) \cdot \triangle m_1. \end{aligned} \tag{2.4.4}$$

One can readily recover Eq. (2.4.1) and Eq. (2.4.2) from Eq. (2.4.3) and Eq. (2.4.4) by setting $W_1 = \triangle m_1 = 1$ and interpreting $(\triangle x_1^i, \triangle y_1^i)$ as $(\triangle u^i, \triangle v^i)$, $i = 1, 2$. Similarly to the case with $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ in [23], the estimation of the offsets $(\triangle x_1^1, \triangle y_1^1)$ and $(\triangle x_1^2, \triangle y_1^2)$ can also benefit from more than two source frames. More importantly, in our method, the offset estimation does not directly resort to any predetermined mathematical model and is carried out in a completely data-driven manner. As such, it can cope with real-world motions more flexibly and accurately. Furthermore, for the general version of our method, the number of sampling points can be set to be greater than 1 (i.e., $N > 1$), which, together with the freedom in choosing the space-time coordinates of the sampling points and the relaxation of the constraint $(\triangle x_n, \triangle y_n) = (\triangle x_n^i, \triangle y_n^i)$, makes it possible to capture complex diffusion and dispersion effects. Finally, we would like to point out that the space-time numerical interpolation operation in our method plays a role similar to that of the blending operation in some existing flow-based methods [13, 20, 22] (see also Fig. 2.6(a)), but requires fewer parameters, as it is performed at the sampling point level.

Kernel-based: These methods [12, 24, 25] generate two sets of spatially-adaptive convolution kernels and use them to convolve with source frame patches to get the predicted target frames $I^{'}_{1.5\leftarrow 1}$, $I^{'}_{1.5\leftarrow 2}$ from two sides, which are then blended at the pixel level to get final VFI result:

$$\begin{aligned} \hat{I}_{1.5}(x,y) &= I^{'}_{1.5\leftarrow 1}(x,y) + I^{'}_{1.5\leftarrow 2}(x,y) \\ &= K_1(x,y) * I_1(x,y) + K_2(x,y) * I_2(x,y). \end{aligned} \tag{2.4.5}$$

The pipeline of kernel-based methods is shown in Fig. 2.6(b). Note that in the presence of complex motions, the technique in [24] and [25] need to adopt large kernels (specifically, the size of convolutional kernels used in [24] and [25] are 41×41 and 51×51, respectively) to ensure sufficient coverage, which is inflexible and memory-inefficient. AdaCoF [12] addresses this issue by adopting deformable convolution. Nevertheless, the sampling points in AdaCoF are only spatially adaptive. In contrast, the proposed method can make more effective use of the sampling points by freely exploring space-time (not just in the spatial domain). As such, it often suffices to employ small kernels, even when dealing with very complex motions. Our method also has the additional advantage of blending images at the sampling point level (in the form of space-time numerical interpolation), which is more efficient than blending at the pixel level in kernel-based methods.

2.5 Four-Frame VFI Experiments

Due to its flexibility, our method can leverage an arbitrary number of frames for VFI. Here we focus on the four-frame VFI case. The experimental results for two-frame VFI will be presented in Section 2.6.

2.5.1 Implementation Details

We use four source frames I_0, I_1, I_2, and I_3 to synthesize the target frame $I_{1.5}$. In GDConv, the number of sampling points for each warped pixel is set to 25. The loss function, the training dataset, and the training strategy are described below.

Loss Function: In addition to the supervision provided at the output end, we introduce intermediate supervision to ensure proper training of the GDCM (which is the key component of GDConvNet). Note that without intermediate supervision, we have no direct control of the training of the GDCM due to the fact that the downstream post-processing module, which is a relatively large and complex network, tends to dilute the impact of the supervisory signal. The overall loss function can be formulated as:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_r + \lambda\mathcal{L}_w \\ &= \sum_x ||\hat{I}_t(x) - I_{GT}(x)||_1 + \lambda \sum_x ||I_t^{'}(x) - I_{GT}(x)||_1,\end{aligned} \tag{2.5.1}$$

where I_{GT} is the ground-truth frame, and λ is a hyper-parameter to balance the warped loss $\mathcal{L}_w$ and the refined loss $\mathcal{L}_r$. (Experimentally, we found that $\lambda = 0.5$ yields the best performance.) We use the ℓ_1 norm instead of the ℓ_2 norm because the latter is known to produce blurry results in image synthesis tasks. Following [12, 21, 22, 48], we use the Charbonnier Function $\Phi(x) = \sqrt{x^2 + \epsilon^2}$ to smoothly approximate the ℓ_1 norm and set $\epsilon = 10^{-6}$.

Training DataSet: The Vimeo90k Septuplet training dataset [13] is used to train our model. This training dataset is composed of 64, 612 seven-frame sequences with a resolution of 256×448. We use the first, the third, the fifth, and the seventh frames (corresponding to I_0, I_1, I_2, and I_3 in our notation, respectively) of each sequence to predict the fourth one (corresponding to $I_{1.5}$). We randomly crop image patches of size 256×256 for training. Horizontal and vertical flipping, as well as temporal order reversal, are performed for data augmentation.

Training Strategy: Different from [13, 21, 22], our network can be trained from scratch without relying on any pre-trained model. We adopt the Adam optimizer [49], where β_1 and β_2 are set as the default values 0.9 and 0.999, respectively. We set the training batch size as 8 and train our network for 14 epochs (nearly 11, 300 iterations) in total. The initial learning rate is set as 10^{-3}, and the learning rate is reduced by a factor of two every 4 epochs for the first 8 epochs and by a factor of five every 2 epochs for the last 6 epochs. The training is carried out on four NVIDIA GTX

Table 2.1: The statistics of pixel displacement within different datasets. This table shows the average pixel displacement, the percentage of pixels with displacement larger than 15, 20 and 25 respectively for three datasets.

Dataset	avg disp.	> 15	> 20	> 25
Vimeo90k Dataset	6.1	9.1%	5.0%	3.0%
Gopro Dataset	6.1	7.0%	2.7%	1.0%
Adobe240 Dataset	8.2	13.0%	9.0%	6.1%

1080Ti GPUs, and takes about 58 hours to converge.

2.5.2 Evaluation Datasets

The following three datasets are used for performance evaluation.

Vimeo90K Septuplet Test Set [13]: This dataset consists of $7,824$ video sequences, each with 7 frames. As in the case of the Vimeo90K Septuplet training dataset, the first, the third, the fifth, and the seventh frames of each sequence are leveraged to synthesize the fourth one. The image resolution of this dataset is 256×448.

Gopro Dataset [50]: This dataset is composed of 33 high-resolution videos recorded by hand-held cameras. The frame rate of each video is 240 fps, and the image resolution is $720 \times 1,280$. The dataset was released in an image format, consisting of a total of $35,782$ images. We successively group every 25 consecutive images as a test sequence, and resize the images to 360×480. Finally, $1,392$ test sequences are selected. For each sequence, the first, the ninth, the seventeenth, and the twenty-fifth frames (corresponding to I_0, I_1, I_2, and I_3, respectively) are used to synthesize the thirteenth frame (corresponding to $I_{1.5}$). This dataset is rich with non-linear camera motions and dynamic object motions, posing significant challenges to VFI methods in these respects.

Adobe240 Dataset [51]**:** This dataset consists of 133 240 fps videos in total, where the resolution of each video is $720 \times 1,280$. These videos are recorded by hand-held cameras, and mainly contain outdoor scenes. Different from the Gopro dataset, this dataset is released in a video format. We extract $7,479$ non-overlapped test sequences, each with 25 frames. This dataset is rich with large motions. Indeed, it has the largest average pixel displacement among the three datasets under consideration according to Table. 2.1. Therefore, it can be used to examine the strength of a VFI method in handling such motions.

2.5.3 Numerical Interpolation Methods

As described in Section 2.3.1, a numerical interpolation function G is used to specify the value $s_n = I(x + \triangle x_n, y + \triangle y_n, z_n)$ of a sampling point in accordance with its position and the corresponding support points $s_n^i = I(x + \triangle x_n^i, y + \triangle y_n^i, i)$, $i \in \{0, 1, \cdots, T\}$, when it does not exactly lie on an input frame (i.e., when z_n is not an integer). In principle, any numerical interpolation function satisfying the differentiability condition can be leveraged for this purpose. However, different numerical interpolation functions may generate different values for the same sampling point and consequently lead to different final outputs. Therefore, it is important to understand how the choice of the numerical interpolation function affects the overall system performance. To this end, we investigate the following representatives: linear interpolation, 3D and 1D versions of inverse-distance-weighted interpolation, and polynomial interpolation.

Table 2.2: Quantitative comparisons of GDConvNet with different numerical interpolation methods on Viemo-90k test dataset, Gopro dataset, and Adobe240 dataset.

Method	#Parameters	Vimeo-90k		Gopro		Adobe240	
	(million)	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ours-Linear	5.1	34.96	0.9534	30.06	0.9092	34.20	0.9422
Ours-3D Inv	5.1	35.01	0.9535	30.12	0.9099	34.27	0.9427
Ours-1D Inv	5.1	35.08	0.9541	30.16	0.9099	34.36	0.9436
Ours-Poly	5.1	**35.58**	**0.9580**	**30.49**	**0.9180**	**34.53**	**0.9456**
Ours-Poly-clamping	5.1	35.10	0.9548	30.18	0.9072	34.33	0.9442

2.5.3.1 Linear Interpolation

This is one of the simplest interpolation methods. It can be be formulated as:

$$s_n = \sum_{i=0}^{T} max(0, 1 - |z_n - i|) \cdot s_n^i. \tag{2.5.2}$$

Note that even if $T > 1$, only two adjacent support points are taken into consideration in Eq. (2.5.2) for interpolating s_n. (The maximum operation suppresses the contribution of other support points.) We regard this interpolation method as the baseline in comparisons.

2.5.3.2 3D Version of Inverse-Distance-Weighted Interpolation (3D Inv)

In contrast to linear interpolation, this method makes use of all support points (see Fig. 2.4) as follows:

$$s_n = \frac{\sum_{i=0}^{T} w_i \cdot s_n^i}{\sum_{i=0}^{T} w_i}, \tag{2.5.3}$$

where $w_i = 1/((d_x^i)^2 + (d_y^i)^2 + (d_z^i)^2)$, $d_x^i = |\triangle x_n - \triangle x_n^i|/H$, $d_y^i = |\triangle y_n - \triangle y_n^i|/W$, and $d_z^i = |z_n - i|/T$. The quantitative comparisons in Table 2.2 indicate that leveraging all support points instead of just two adjacent points yields better performance. Table 2.3 shows the means of $(d_x^i)^2$, $(d_y^i)^2$, and $(d_z^i)^2$ (averaged over i), denoted as $(d_x)^2$, $(d_y)^2$ and $(d_z)^2$, respectively. It is clear that $(d_x)^2$ and $(d_y)^2$ are about two orders of magnitude smaller than $(d_z)^2$. This implies that it might suffice to set the weights based on the temporal information alone, which naturally suggests the following interpolation method.

Table 2.3: Mean of the squared distance.

$(d_x)^2$	$(d_y)^2$	$(d_z)^2$
0.0025	0.0010	0.2009

2.5.3.3 1D Version of Inverse-Distance-Weighted Interpolation (1D Inv)

Setting $w_i = 1/(d_z^i)^2$ in Eq. (2.5.3) leads to the 1D version of inverse distance weighted interpolation (see Fig. 2.7(a) for an example with $T = 3$). The quantitative results of this interpolation method are shown in Table 2.2. Somewhat surprisingly, the 1D version slightly outperforms its 3D counterpart. The reason is that focusing on the dominant dimension enables more effective use of the training data and consequently yields more accurate VFI results. This suggests that it might be possible to further improve the performance by employing more advanced 1D interpolation methods.

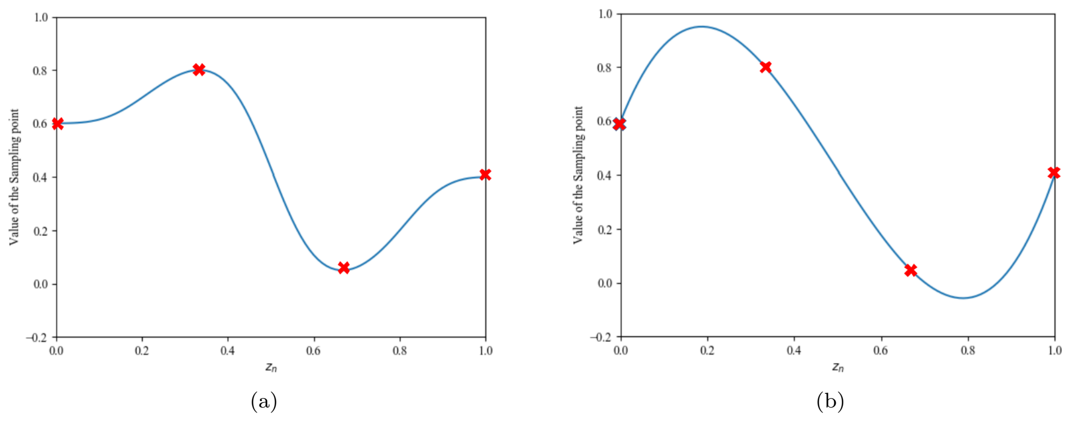


Figure 2.7: Illustration of (a) 1D version of inverse distance weighted interpolation and (b) polynomial interpolation with support points highlighted in red. Here $s_n^0 = 0.6$, $s_n^1 = 0.8$, $s_n^2 = 0.05$, and $s_n^3 = 0.4$, respectively.

2.5.3.4 Polynomial Interpolation (Poly)

This method uses a polynomial function of degree T to perform interpolation. More specifically, we have:

$$G = a_0 + a_1 z_n + \cdots + a_T z_n^T, z \in [0, T], \tag{2.5.4}$$

where the coefficients a_0, a_1, $\cdots$, and a_T can be uniquely determined by jointly solving $T+1$ linear equations $G|_{z_n=i} = s_n^i$, $i \in \{0, 1, \cdots, T\}$. It should be emphasized that sampling points and their associated support points are still selected in 3D space-time even if a 1D interpolation method is adopted; as such, the overall method is intrinsically 3D.

Fig. 2.7(b) provides an example of polynomial interpolation with $T = 3$. In contrast to 1D Inv, polynomial interpolation is able to generate values beyond the upper and lower limits of s_n^i, $i \in \{0, 1, \cdots, T\}$. This extra freedom might be the reason why polynomial interpolation leads to 0.5 dB improvement over 1D Inv as

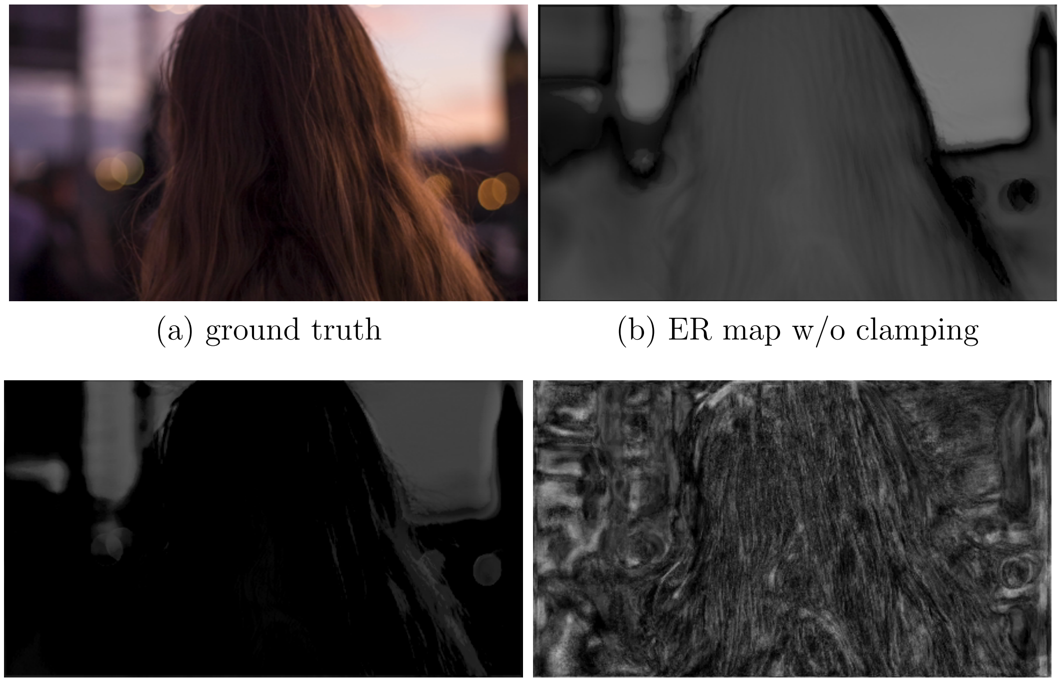
Table 2.4: The statistical distribution of sampling points beyond limits.

Method	Beyond Upper	Beyond Lower
Upper GDCM	10.8%	22.6%
Lower GDCM	9.3%	10.0%

shown in Table 2.2.

To provide supporting evidence for our conjecture, we count the number of sampling points whose values are beyond the upper or lower limit. As shown in Table 2.4, for the upper GDCM used to synthesize intermediate frame I'_t, there are 10.8% and 22.6% sampling points beyond the upper limit and lower limit respectively. As for the lower GDCM used to predict the context map C'_t, 9.3% and 10.0% points are beyond the upper limit and the lower limit, respectively. We then clamp those values to their associated limits and reevaluate the model on the test datasets. As shown in Table 2.2, indeed, forcing the values of sampling points to stay in the range set by support points jeopardizes the performance.

Fig. 2.8 provides visual examples of the results. It can be seen that compared to 1D inv, standard polynomial interpolation provides a better reconstruction in the texture regions, which usually contain a fair amount of sampling points beyond limits. In contrast, clamped polynomial interpolation performs considerably worse than the standard one in these regions. Similar phenomena can be observed for images in different datasets. In summary, polynomial interpolation is able to generate sampling points beyond upper and lower limits, and these sampling points contribute positively to the synthesis of the texture regions of the images, which helps to improve the overall performance.



(a) ground truth (b) ER map w/o clamping

(c) ER map w clamping (d) Distribution map

Figure 2.8: Visualization of (a) ground truth, (b) error residual (ER) map generated by GDConvNet with standard polynomial (c) error residual map generated by GDConvNet with clamped polynomial, (d) distribution map of sampling points beyond limits. The error residual map is calculated by $ER = MSE_{1\_inv} - MSE_{poly}$, where MSE_{1_inv} denotes the mean squared error map between the ground truth and the result generated by GDConvNet with 1D Inv, and MSE_{poly} is similarly defined for polynomial interpolation.

2.5.4 Comparison with the State-of-the-Art

We compare our best-performing GDConvNet (Ours-Poly) with the state-of-the-art VFI algorithms on the aforementioned three evaluation datasets. Specifically, the following ones are chosen for comparison: the phase-based method (Phase) [52], separable adaptive convolution (SepConv) [25], deep voxel flow (DVF) [19], SuperSlomo (Slomo) [20], quadratic video interpolation (QVI) [23], and adaptive collaboration of flows (AdaCoF) [12]. Since these methods just use two frames (I_1, I_2) to synthesize

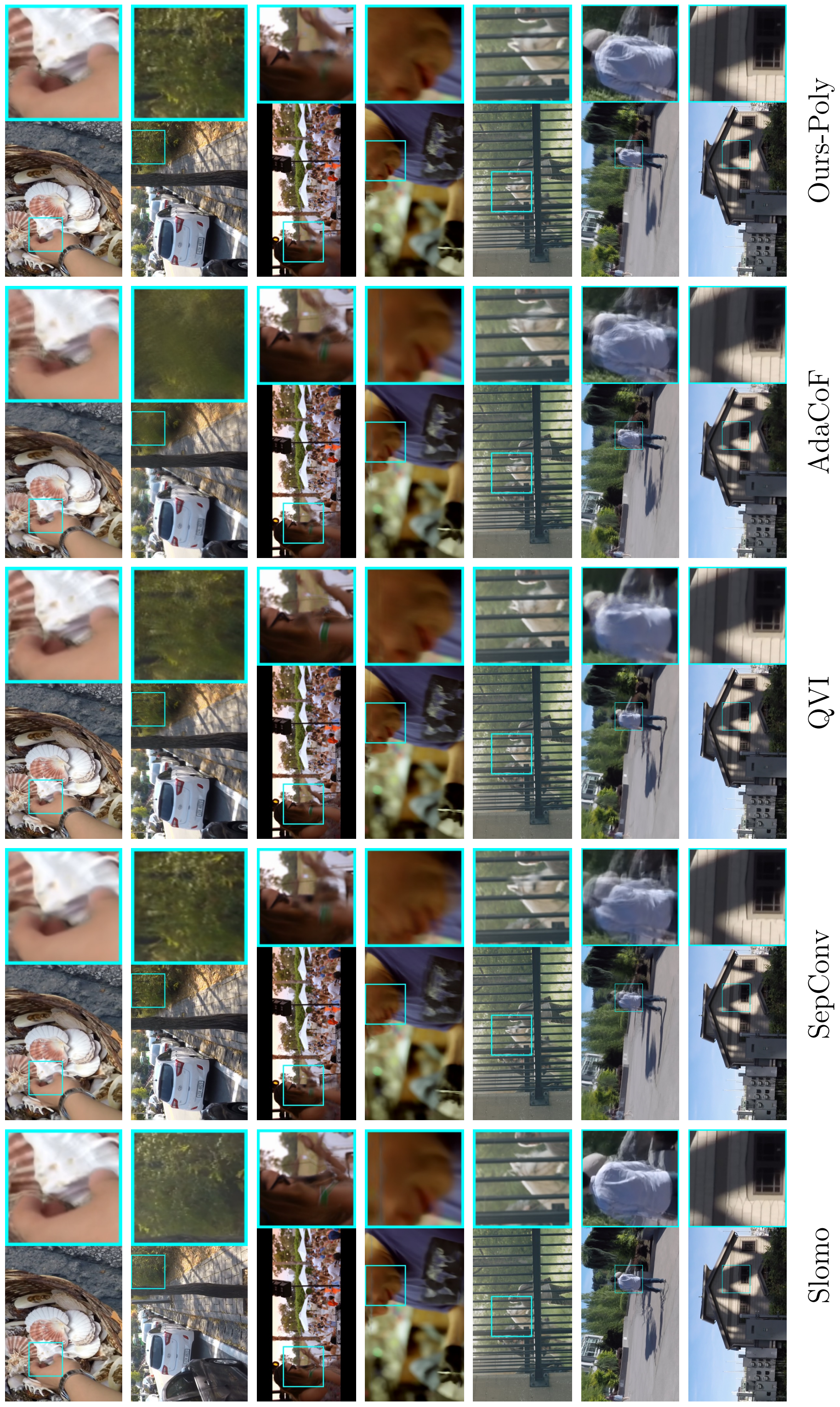


Figure 2.9: Qualitative comparisons of different VFI algorithms.

Table 2.5: Quantitative comparisons of different VFI methods on Vimeo90K Septulet test set, Gopro dataset and Adobe240 dataset, where the first place and second place are highlighted in red and blue, respectively.

Method	#Parameters	Vimeo90K		Gopro		Adobe240	
	(million)	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ours-Poly	5.1	**35.58**	**0.9580**	**30.49**	0.9180	**34.53**	**0.9456**
Ours-Poly*	5.1	35.01	0.9558	30.12	0.9100	34.12	0.9422
AdaCoF	21.8	33.92	0.9453	28.45	0.8734	33.17	0.9305
QVI	29.2	35.19	0.9563	30.24	**0.9230**	33.06	0.9393
Slomo	39.6	33.73	0.9453	28.50	0.8827	31.94	0.9264
SepConv	21.6	33.65	0.9435	28.66	0.8798	33.41	0.9349
DVF	3.8	30.79	0.8912	25.13	0.7633	22.33	0.6159
Phase	–	30.52	0.8854	26.17	0.8135	31.20	0.8930

the target frame[3], we also provide a degraded version of our method (Ours-Poly*) with 4 frames (I_0, I_1, I_2, I_3) for offset generation and 2 frames (I_1, I_2) for target frame prediction. For fair comparison, DVF, Slomo, QVI, and AdaCoF are retrained on our training dataset. As the SepConv training code is not available, we choose to directly evaluate the original SepConv model.

In Table 2.5, we quantitatively compare our method with the state-of-the-art methods on the evaluation datasets under two well-known objective image quality metrics, PSNR and SSIM. It can be seen that although it suffers from some performance degradation with respect to Ours-Poly, Ours-Poly* still performs on par with QVI (which is 6 times as large as Ours-Poly* in terms of model size) and surpasses other methods by a visible margin. As for Ours-Poly, it shows a significant improvement over its degraded counterpart due to the complete freedom in exploiting the given frames, and ranks consistently at the top in Table 2.5 (except for the Gopro dataset on which it comes in a close second in terms of the SSIM value). Overall, our

[3]Although 4 frames are employed in QVI, only 2 of them are directly involved in predicting the target frame.

method has a clear advantage under joint consideration of cost and performance.

Fig. 2.9 shows some qualitative comparisons. It can be seen that our method produces clearer and sharper results. For example, on the first row, our method is capable of generating smooth edges around the hand compared with that of Phase, DVF, SepConv, Slomo, QVI, and AdaCoF.

2.5.5 Ablation Study

In our ablation studies, we adopt polynomial interpolation and consider a simplified version of GDConvNet in which the CEM and the associated GDCM, as well as the PM, are removed. This simplification greatly reduces the training time and, more importantly, enables us to focus on the most essential aspects of GDConvNet.

2.5.5.1 Generalized Deformable Convolution Module

In order to validate the effectiveness of our design, we compare the proposed GDConv with DConv (more precisely, spatially-adaptive DConv or modulated DConv) adopted by [12], as well as several variants of GDConv.

Table 2.6: Comparisons of DConv, GDConv with different numbers of sampling points, and some variants of GDConv.

Method	Sampling points	PSNR	SSIM
DConv	25	32.82	0.9236
GDConv	1	33.40	0.9342
GDConv	9	33.98	0.9414
GDConv	25	**34.20**	**0.9436**
GDConv	36	34.17	0.9430
Variant (a)	25	32.99	0.9274
Variant (b)	25	33.24	0.9310
Variant (c)	25	33.92	0.9410
Variant (d)	25	34.06	0.9418
Variant (e)	25	**34.20**	**0.9436**

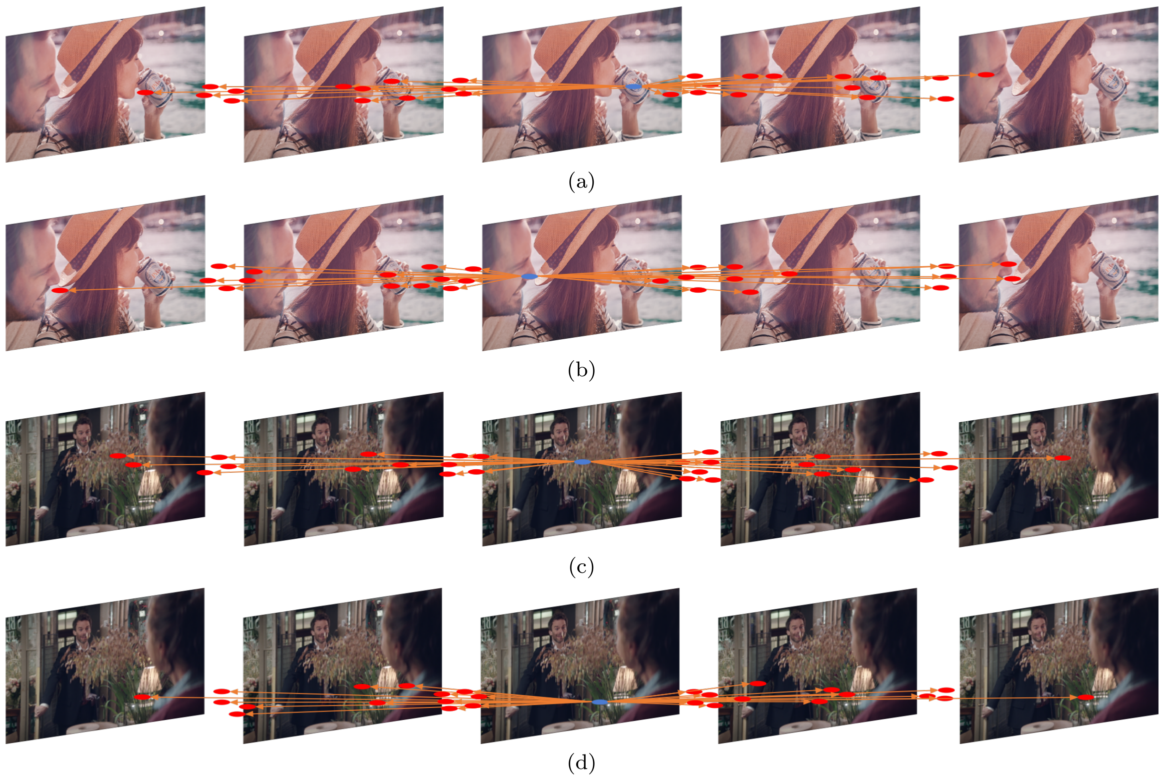


Figure 2.10: Visualization of sampling points in GDCM when $t = 1.5$. Here ((a), (b)) and ((c), (d)) illustrate two different pixels in a same target intermediate frame and their associated sampling points respectively. It can be seen that sampling points are not exclusively located between I_1 and I_2. Indeed, there are some between I_1 and I_2, and some between I_2 and I_3. This indicates that the information from I_1 and I_2 is more significant for synthesizing I_t, but I_0 and I_3 also contribute to the synthesized result.

Superiority of GDConv over DConv: As mentioned earlier, the proposed GDConv is able to fully exploit the given source frames in accordance with their relevance to the target intermediate frame in terms of temporal distance. In contrast, the performance of DConv is limited by the inflexibility in choosing the number of sampling points from each source frame. For instance, consider the case where 4 consecutive frames are used for VFI and the convolution kernel size is set to 3. DConv is constrained to select 9 sampling points from each frame. This is inefficient from the perspective of resource allocation since the source frames closer to the target intermediate frame in time are conceivably more informative and should receive more attention. In this sense, the proposed GDConv is more desirable as it is endowed with complete freedom to select sampling points in space-time. Specifically, in GDConv, the number of sampling points in each frame is adjustable according to the significance of that frame in synthesis. More importantly, sampling points are not even required to lie exactly on the source frames, and are allowed to be anywhere in the spatio-temporal domain specified by their associated parameters $\triangle x_n$, $\triangle y_n$ and z_n (see Fig. 2.10 for some visual results). This mechanism is especially important for VFI since it is better suited to cope with complex and irregular inter-frame motions. In Table 2.6, we provide quantitative comparisons of DConv and GDConv. Here the number of input source frames is 4. In GDConv, the number of sampling points is set to 36. For fair comparison, the kernel size in DConv is chosen to be 3; thus, there are $3 \times 3 \times 4 = 36$ sampling points in total, as well. It is evident that the proposed GDConv achieves better performance in terms of the PSNR and SSIM metrics.

Importance of Spatio-Temporal Freedom: We consider the following 4 variants of GDConv to illustrate the importance of spatio-temporal freedom for sampling points.

(a) No spatio-temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, are identical and fixed to be a distinct point in a 5×5 grid $\{(-2,-2), (-2,-1), \cdots, (2,2)\}\}$, and $z_n = 1.5$.

(b) Limited spatial freedom, no temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, are identical but adaptive, and $z_n = 1.5$.

(c) Limited spatial freedom, complete temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, are identical but adaptive, and z_n is adaptive.

(d) Complete spatial freedom, no temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, can be different from each other and are individually adaptive, and $z_n = 1.5$.

(e) Complete spatio-temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, can be different from each other and are individually adaptive, and z_n is also adaptive.

The results of the experiment are shown in Table 2.6. One can easily find that the performance rises progressively with the availability of every additional degree of freedom. It is worth noting that the temporal parameter z_n is better interpreted as being effective time instead of physical time. Indeed, forcing $z_n = 1.5$ limits the degrees of freedom and jeopardizes the performance.

Choice of the Number of Sampling Points: We further investigate how to choose the number of sampling points in GDConv. As shown in Table 2.6, as the number of sampling points increases, the performance improves initially, but becomes saturated eventually. In particular, using more than 36 sampling points does not further enhance the quality of synthesized frames.

2.5.5.2 Input Length and Offset Generation

Table 2.7: Comparisons for different numbers of reference frames (with the number of generation frames set to be the same as that of reference frame).

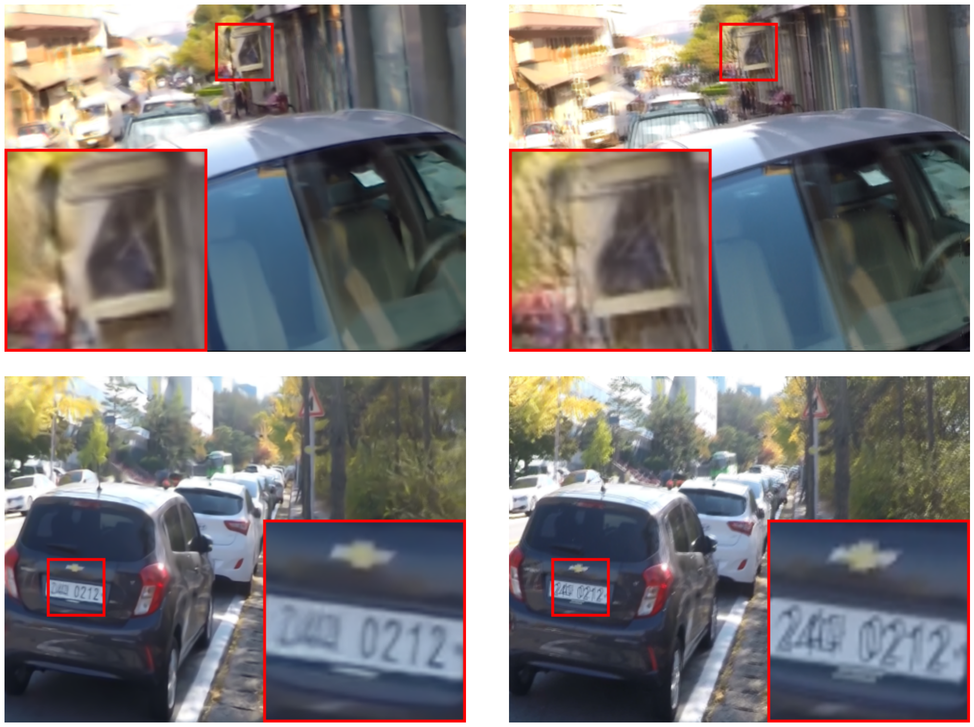
Reference Frames	PSNR	SSIM
I_1, I_2	33.69	0.9416
I_0, I_1, I_2	33.97	0.9427
I_0, I_1, I_2, I_3	**34.20**	**0.9436**

So far, except for the degraded version in Section 2.5.4, we have assumed that all 4 source frames I_0, I_1, I_2, and I_3 participate in generating offsets (as well as z_n and $\triangle m_n$) and in predicting the target intermediate frame $I_{1.5}$. It is interesting to study how the proposed method performs if one only utilizes a subset of source frames. In

Table 2.8: Comparisons for different numbers of generation frames (with the reference frames fixed to be I_1 and I_2).

Generation Frames	PSNR	SSIM
I_1, I_2	33.69	0.9416
I_0, I_1, I_2	33.84	0.9418
I_0, I_1, I_2, I_3	**34.05**	**0.9434**

fact, our framework is flexible enough to allow the use of different subsets of source frames for offset generation and frame prediction separately. For clarity, we shall refer to source frames used for generating offsets as generation frames and those directly involved in predicting the target intermediate frame as reference frames. For example, if we use I_0, I_1, I_2 to generate offsets for I_1 and I_2, which are subsequently leveraged to predict $I_{1.5}$, then I_0, I_1, I_2 are generation frames while the latter two are reference frames. We first study the scenario with the same subset of source frames used for both purposes. It is clear from Table 2.7 that the VFI result improves progressively with the increase in the number of reference frames (as well as generation frames). We further investigate the scenario where reference frames and generation references are not necessarily the same. Specifically, we fix I_1 and I_2 to be reference frames, and consider various combinations of generation frames. It can be seen from Table 2.8 that increasing the number of generation frames leads to better performance. This is consistent with a similar finding regarding flow-based methods: namely, it is profitable to have three or more generation frames as that opens the door for exploiting higher-order approximation of motion trajectories (instead of relying on linear approximation, which is basically the only available choice in the case with just two generation frames). Finally, comparing the corresponding rows in Table 2.7 and Table 2.8 reveals that VFI can also benefit from an increase in the number of reference frames (when the number of generation frames is fixed).



(a) Ours (b) QVI

Figure 2.11: Visualization of failure cases.

2.5.6 Failure Case Analysis

Our method is trained in a purely data-driven manner to learn motion estimation. As such, it is able to handle complex motion patterns that cannot be characterized by simple mathematical models. On the other hand, the success of our method depends critically on the quality of the training dataset, which should ideally contain extensive motion patterns to ensure sufficient coverage. The performance of our method tends to degenerate when the motion patterns encountered in the evaluation dataset deviate significantly from those in the training dataset. Fig. 2.11 provides some examples where object motions are atypical with respect to the training dataset. It can be seen that the VFI results produced by our method are somewhat blurry (albeit still slightly better than those of QVI, which is the best known mathematical-model-based method).

2.6 Two-Frames VFI Experiments

As described earlier, our method is able to handle an arbitrary number of frames. To substantiate this claim, here we conduct two-frame VFI experiments (i.e., using I_0 and I_1 to predict $I_{0.5}$).

2.6.1 Implementation Details

We adopt polynomial interpolation (or linear interpolation) and set the number of sampling points for each warped pixel to be 25 in GDConv. The training dataset and the training strategy are described below.

Training Dataset: The Vimeo90k interpolation training dataset [13] is used to train our model. This training dataset is composed of $51,312$ triplets with resolution 256×448. We use the first frame and the third frame (corresponding to I_0 and I_1, respectively) of each triplet to predict the second one (corresponding to $I_{0.5}$). We randomly crop image patches of size 256×256 for training. Horizontal and vertical flipping, as well as temporal order reversal, are performed for data augmentation.

Training Strategy: This is the same as the four-frame case, except that we train our network for 20 epochs in total. The initial learning rate remains to be 10^{-3}, and the learning rate is reduced by a factor of two every 4 epochs for the first 12 epochs and by a factor of five every 4 epochs for the last 8 epochs. The whole training process takes about 3 days on our hardware.

2.6.2 Evaluation Datasets

Following [21], we evaluate the proposed GDConvNet on three public datasets (Vimeo90k Interpolation Test Set [13], UCF101 Test Dataset [53], and Middlebury-Other Dataset [54]) and compare it with the state-of-the-art.

Vimeo90k Interpolation Test Set [13]: This dataset consists of $3,782$ video sequences, each with 3 frames. As in the case of the Vimeo90K interpolation training dataset, the first frame and the third frame of each sequence are leveraged to synthesize the second one. The image resolution of this dataset is 256×448.

UCF101 Test Dataset [53]: The UCF101 dataset contains 379 triplets with a large variety of human actions. The image resolution of this dataset is 256×256.

Middlebury-Other Dataset [54]: The Middlebury-Other dataset is another commonly used benchmark for VFI, which contains 12 triplets in total. Most of the images in this dataset are of resolution 640×480. Again, we use the first frame and the third frame to predict the second one.

2.6.3 Experimental Results

We compare our GDConvNet with the state-of-the-art VFI algorithms on the aforementioned datasets. Specifically, the following ones are chosen for comparison: MIND [55], DVF [19], SepConv [25], CtxSyn [30], ToFlow [13], SuperSlomo [20], MEMC-Net [22], DAIN [21], and AdaCoF [12].

In Table 2.9, we quantitatively compare our method with the state-of-the-art on Vimeo90k and UCF101 under PSNR and SSIM, while Interpolation Error [44] (IE) is used as the performance measure for the Middlebury-Other dataset. It can be

Table 2.9: Quantitative comparisons on Vimeo90K interpolation test set, UCF101 dataset and Middlebury-Other dataset, where the first place and second place are highlighted in red and blue, respectively.

Method	#Parameters	UCF101		Vimeo90K		Middlebury
	(million)	PSNR	SSIM	PSNR	SSIM	IE
MIND	7.60	33.93	0.9661	33.50	0.9429	3.35
DVF	3.80	34.12	0.9631	31.54	0.9462	7.75
ToFlow	1.07	34.58	0.9667	33.73	0.9682	2.51
SepConv-Lf	21.6	34.69	0.9655	33.45	0.9674	2.44
SepConv-L1	21.6	34.78	0.9669	33.79	0.9702	2.27
MEMC-Net	70.3	34.96	0.9682	34.29	0.9739	2.12
DAIN	24.0	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**
AdaCoF	21.8	**34.99**	0.9682	33.43	0.9677	2.43
Ours	5.6	**35.16**	**0.9683**	**34.99**	**0.9750**	**2.03**

seen that the proposed method performs favorably against those under consideration. Overall, our method has a clear advantage under joint consideration of cost and performance. In particular, although DAIN [21] also shows very competitive performance, its model size is about 5 times that of our model. In addition, our method can be trained from scratch, while DAIN [21] needs to rely on a pre-trained model.

2.7 Conclusion

In this paper, a new mechanism named generalized deformable convolution is proposed to tackle the VFI problem. This mechanism unifies the essential ideas underlying flow-based and kernel-based methods and resolves some performance-limiting issues. It should be noted that the proposed mechanism is largely generic in nature, and is potentially applicable to a wide range of problems, especially those involving video data (e.g., video super-resolution, enhancement, and quality mapping). Exploring such applications is an endeavor well worth undertaking.

Bibliography

[1] G.-J. Qi, H. Larochelle, B. Huet, J. Luo, and K. Yu, "Guest editorial: Deep learning for multimedia computing," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1873–1874, 2015.

[2] X. Yang, H. Mei, J. Zhang, K. Xu, B. Yin, Q. Zhang, and X. Wei, "Drfn: Deep recurrent fusion network for single-image super-resolution with large factors," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 328–337, 2018.

[3] Z. He, Y. Cao, L. Du, B. Xu, J. Yang, Y. Cao, S. Tang, and Y. Zhuang, "Mrfn: Multi-receptive-field network for fast and accurate single image super-resolution," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 1042–1054, 2019.

[4] X. Liu, K. Shi, Z. Wang, and J. Chen, "Exploit camera raw data for video super-resolution via hidden markov model inference," *arXiv preprint arXiv:2008.10710*, 2020.

[5] P. Hu, G. Wang, and Y.-P. Tan, "Recurrent spatial pyramid cnn for optical flow estimation," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2814–2823, 2018.

[6] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[7] C. Li, C. Guo, J. Guo, P. Han, H. Fu, and R. Cong, "Pdr-net: Perception-inspired single image dehazing network with refinement," *IEEE Transactions on Multimedia*, vol. 22, no. 3, pp. 704–716, 2019.

[8] Y. Song, J. Li, X. Wang, and X. Chen, "Single image dehazing using ranking convolutional neural network," *IEEE Transactions on Multimedia*, vol. 20, no. 6, pp. 1548–1560, 2017.

[9] K. Zhu, R. Wang, Q. Zhao, J. Cheng, and D. Tao, "A cuboid cnn model with an attention mechanism for skeleton-based action recognition," *IEEE Transactions on Multimedia*, 2019.

[10] M. Usman, X. He, K.-M. Lam, M. Xu, S. M. M. Bokhari, and J. Chen, "Frame interpolation for cloud-based mobile video streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 831–839, 2016.

[11] M. Ma, O. C. Au, L. Guo, S.-H. G. Chan, and P. H. Wong, "Error concealment for frame losses in mdc," *IEEE transactions on multimedia*, vol. 10, no. 8, pp. 1638–1647, 2008.

[12] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee, "Adacof: Adaptive collaboration of flows for video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5316–5325.

[13] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[14] C. Zhang, L. Ge, Z. Chen, M. Li, W. Liu, and H. Chen, "Refined tv-l 1 optical flow estimation using joint filtering," *IEEE Transactions on Multimedia*, vol. 22, no. 2, pp. 349–364, 2019.

[15] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, vol. 281. International Society for Optics and Photonics, 1981, pp. 319–331.

[16] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[17] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[18] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[19] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[20] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video

interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.

[21] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3703–3712.

[22] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang, "Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[23] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang, "Quadratic video interpolation," in *NeurIPS*, 2019.

[24] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[25] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[26] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[27] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[29] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[30] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1701–1710.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[33] X. Liu, Y. Ma, Z. Shi, and J. Chen, "Griddehazenet: Attention-based multi-scale network for image dehazing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7314–7323.

[34] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2472–2481.

[35] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4201–4209.

[36] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 667–675.

[37] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*. Springer, 1990, pp. 286–297.

[38] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 267–283.

[39] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[40] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 270–12 279.

[41] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[42] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[47] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[48] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4463–4471.

[49] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization. corr abs/1412.6980 (2014)," 2014.

[50] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3883–3891.

[51] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep video deblurring for hand-held cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1279–1288.

[52] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1410–1418.

[53] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[54] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[55] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *European Conference on Computer Vision*. Springer, 2016, pp. 434–450.

The following chapter is reproduced from an IEEE published paper:

Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. "Video Frame Interpolation Transformer." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

Contribution Declaration: Zhihao Shi (the author of this thesis) is the first author and the main contributor to this paper (more than 50%). He proposed the algorithm, wrote the corresponding program, conduct all experiments, and composed the manuscript. Xiangyu Xu participated in the discussions of the key idea of the paper and helped revise the manuscript. Xiaohong Liu helped prepare presentation figures and demo video; Jun Chen and Ming-Hsuan Yang helped revise the manuscript.

Chapter 3

Video Frame Interpolation Transformer

3.1 Abstract

Existing methods for video interpolation heavily rely on deep convolution neural networks, and thus suffer from their intrinsic limitations, such as content-agnostic kernel weights and restricted receptive field. To address these issues, we propose a Transformer-based video interpolation framework that allows content-aware aggregation weights and considers long-range dependencies with the self-attention operations. To avoid the high computational cost of global self-attention, we introduce the concept of local attention into video interpolation and extend it to the spatial-temporal domain. Furthermore, we propose a space-time separation strategy to save memory usage, which also improves performance. In addition, we develop a multi-scale frame synthesis scheme to fully realize the potential of Transformers. Extensive experiments

demonstrate the proposed model performs favorably against the state-of-the-art methods both quantitatively and qualitatively on a variety of benchmark datasets. The code and models will be released at: https://github.com/zhshi0816/Video-Frame-Interpolation-Transformer

3.2 Introduction

VFI aims to temporally upsample an input video by synthesizing new frames between existing ones. It is a fundamental problem in computer vision that involves the understanding of motions, structures, and natural image distributions, which facilitates numerous downstream applications, such as image restoration [1,2], virtual reality [3], and medical imaging [4].

Most of the state-of-the-art VFI methods are based on deep CNNs [5–12]. While achieving the state-of-the-art performance, these CNN-based architectures usually suffer from two major drawbacks. First, the convolution layer is content-agnostic, where the same kernels are used to convolve with different locations of different inputs. While this design can serve as a desirable inductive bias for image recognition models to acquire translational equivalence [13], it is not always suitable for video interpolation which involves a complex motion-compensation process that is spatially-variant and content-dependent. Thus, adopting CNN backbones may restrict the ability of adaptive motion modeling and potentially limits further development of video interpolation models.

Second, capturing long-range dependencies is of central importance in video interpolation for which large motion fields pose the most prominent challenges. However,

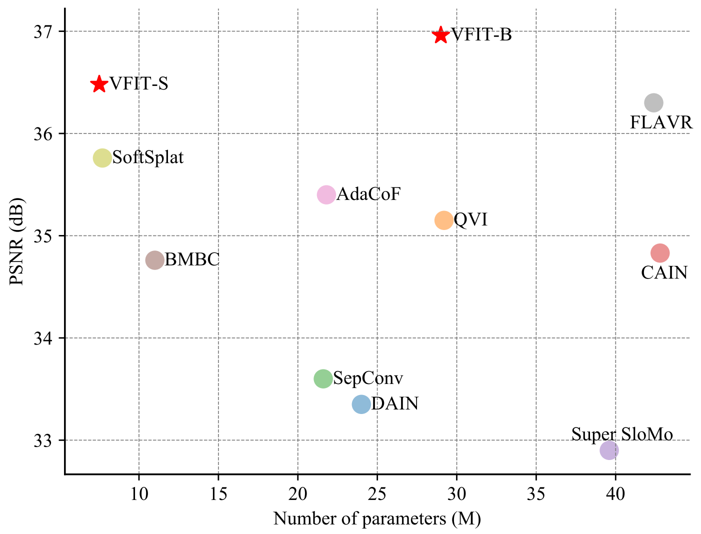


Figure 3.1: Comparison of performance and model size using the Vimeo-90K dataset [14]. VFIT outperforms the state-of-the-art methods with fewer parameters. VFIT-S and VFIT-B denote the proposed small and base models.

most CNNs [7, 12] usually employ small convolution kernels (typically 3×3 as suggested by VGG [15]), which is inefficient in exploiting long-range information and thus less effective in synthesizing high-quality video frames. While it seems an easy fix to use larger kernels in the convolution layer, it significantly increases the number of model parameters and computational cost, thereby leading to poor local minimums in training without proper regularizations. Moreover, simply stacking multiple small kernel layers for a larger receptive field does not fully resolve this problem either, as distant dependencies cannot be effectively learned in a multi-hop fashion [16].

On the other hand, Transformers [17], which are initially designed for NLP to efficiently model long-range dependencies between input and output, naturally overcome the above drawbacks of CNN-based algorithms, and are in particular suitable for the task of video interpolation. Motivated by the success in NLP, several methods recently adapt Transformers to computer vision and demonstrate promising results on various tasks, such as image classification [18, 19], semantic segmentation [20], object detection [21], and 3D reconstruction [22]. Nevertheless, how to effectively apply Transformers to video interpolation that involves an extra temporal dimension remains an open yet challenging problem.

In this work, we propose the Video Frame Interpolation Transformer (VFIT) for effective video interpolation. Compared with typical Transformers [18, 21, 23] where the basic modules are largely borrowed from the original NLP model [17], there are three distinguished designs in the proposed VFIT to generate photo-realistic and temporally-coherent frames. First, the original Transformer [17] is based on a self-attention layer that interacts with the input elements (e.g., pixels) globally. As this global operation has quadratic complexity with regard to the number of elements, directly applying it to our task leads to extremely high memory and computational cost due to the high-dimensionality nature of videos. Several methods [23, 24] circumvent this problem by dividing the feature maps into patches and treating each patch as a new element in the self-attention. However, this strategy cannot model fine-grained dependencies between pixels inside each patch which are important for synthesizing realistic details. Moreover, it may introduce edge artifacts around patch borders. In contrast, we introduce the local attention mechanism of Swin [25] into VFIT to address the complexity issue while retaining the capability of modeling long-range dependencies with its shift-window scheme. We demonstrate that with proper development and adaptation, the local attention mechanism originally used for image recognition can effectively improve the video interpolation performance with a smaller amount of parameters as shown in Figure 3.1.

Second, the original local attention mechanism [25] is only suitable for image input and cannot be easily used for the video interpolation task where an extra temporal dimension is involved. To address this issue, we generalize the concept of local attention to spatial-temporal domain, which leads to the Spatial-Temporal Swin attention layer (STS) that is compatible with videos. However, this simple extension could lead to memory issues when using large window sizes. To make our model more memory efficient, we further devise a space-time separable version of STS, called Sep-STS, by factorizing the spatial-temporal self-attention. Interestingly, Sep-STS not only effectively reduces memory usage but also considerably improves video interpolation performance.

To exploit the full potential of our Sep-STS, we propose a new multi-scale kernel-prediction framework which can better handle multi-scale motion and structures in diverse videos, and generates high-quality video interpolation results in a coarse-to-fine manner. The proposed VFIT is concise, flexible, light-weight, high-performing, fast, and memory-efficient. As shown in Figure 3.1, a small model (VFIT-S) already outperforms the state-of-the-art FLAVR method [26] by 0.18 dB with only **17.7%** of its parameters, while our base model (VFIT-B) achieves **0.66 dB** improvement with 68.4% of its parameters.

3.3 Related Work

Video frame interpolation. Existing VFI methods can be broadly classified into three categories: flow-based [9–12], kernel-based [5–7, 27], and direct-regression-based methods [4].

The flow-based methods [9–12] generate intermediate frames by warping pixels from the source images according to predicted optical flow. Although these methods perform well, they are usually based on simplified motion assumptions such as linear [9] and quadratic [12], limiting their performance in many real-world scenarios where the assumptions are violated.

Unlike the flow-based approaches, the kernel-based methods [5–7,27] do not rely on any prescribed assumptions and thus generalize better to diverse videos. For example, SepConv [6] predicts adaptive separable kernels to aggregate source pixels of the input, and AdaCoF [7] learns deformable spatially-variant kernels that are used to convolve with the input frames to produce the target frame. However, these approaches usually apply the kernel prediction modules at one scale and thereby cannot effectively handle complex motions and structures that could appear in different scales. Moreover, these CNN-based methods do not account for long-range dependency among pixels. In contrast, we propose a multi-scale Transformer-based kernel prediction module, which achieves higher-quality results for video interpolation as will be shown in Section 3.5.

Recently, Kalluri *et al.* [26] propose a CNN model to directly regress the target frame, which achieves the state-of-the-art results. As shown in Figure 3.1, the proposed VFIT outperforms this method by a clear margin with fewer parameters, which clearly shows the advantages of Transformers in video interpolation.

Vision Transformer. Transformers have recently been applied to various vision tasks, such as image classification [18, 25], object detection [21], semantic segmentation [20], 3D reconstruction [22], and image restoration [23]. Nevertheless, it has not been exploited for VFI. In this work, we propose VFIT that achieves the state-of-the-art performance with a light-weight model. To overcome the high computational cost caused by global self-attention, we introduce the local attention mechanism of

Swin [25] to avoid the complexity issues while retaining the ability of long-range dependency modeling. One recent work [28] also uses the local attention for low-level vision tasks. However, it only considers image input and cannot deal with videos which are more challenging to handle due to the extra temporal dimension. In contrast, we extend the concept of local attention to the spatial-temporal domain to enable Transformer-based video interpolation, and propose a space-time separation strategy which not only saves memory usage but also acts as an effective regularization for performance gains.

3.4 Proposed Method

Figure 3.2 shows an overview of the proposed model. Similar to existing methods [5, 6, 12, 26], to synthesize an intermediate frame $I_{0.5}$, we use its T neighboring frames $I_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, 0, 1, \cdots, \lceil \frac{T}{2} \rceil\}}$ as the input. Specifically, the input frames are I_{-1}, I_0, I_1, I_2 when T is 4.

The proposed VFIT consists of three modules: shallow feature embedding, deep feature learning, and final frame synthesis. First, the embedding layer takes the input frames and generates shallow features for the deep feature learning module. Similar to [25], the shallow embedding is realized with a convolution layer, where we adopt the 3D convolution rather than its 2D counterpart in [25] to better encode the spatial-temporal features of the input sequence. Next, we feed the shallow features to the deep module to extract hierarchical feature representations $\{F^l, l = 0, 1, 2\}$ to capture the multi-scale motion information (Section 3.4.1). Finally, an intermediate frame $\hat{I}_{0.5}$ can be generated by the frame synthesis blocks (SynBlocks in Figure 3.2) using the deep features F^l (Section 3.4.2).

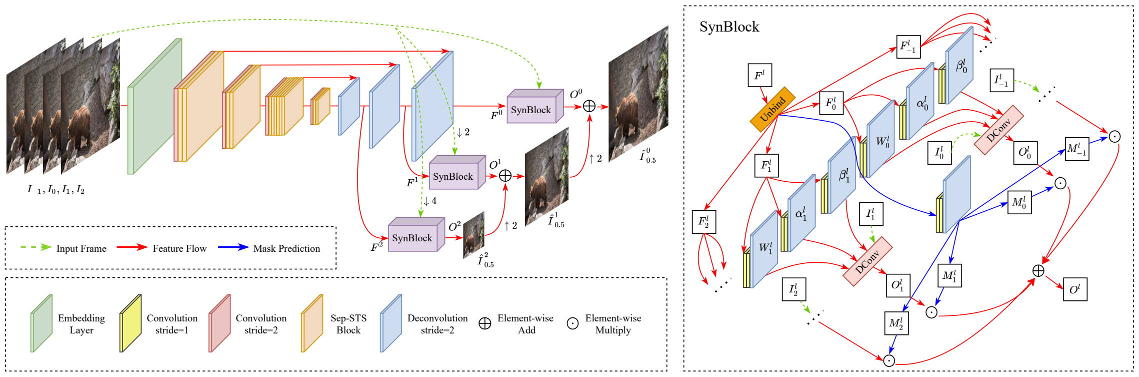


Figure 3.2: Overview of the proposed VFIT. We first use an embedding layer to transform input frames into shallow features, followed by a Transformer-based encoder-decoder network to extract deep hierarchical features. These features together with the input frames are fed to a multi-scale frame synthesis network that is composed of three SynBlocks to obtain the final output. "$\downarrow n$" and "$\uparrow n$" denote downsampling and upsampling by a factor of n, respectively. "DConv" represents deformable convolution [29]. Please find more detailed explanations in the text of Section 3.4.

3.4.1 Learning Deep Features

As shown in Figure 3.2, we use a Transformer-based encoder-decoder architecture for learning features. The encoder is composed of four stages, where each stage starts with a 3D convolution layer using a stride of 2 to downsample the input features, and the downsampling layer is followed by several Sep-STS blocks which are the main components of our framework. For the decoder, we use a light-weight structure that only has three 3D deconvolution layers with a stride of 2 to upsample the low-resolution feature maps. Note that we only resize the spatial dimension of the features throughout our network and leave the temporal size unchanged. Next, we provide more explanations about the proposed Sep-STS block.

Local attention. Existing Transformers [17, 18, 21] mainly adopt a global attention mechanism to aggregate information from the input, which could cause extremely high memory and computational cost for VFI. A straightforward solution to this

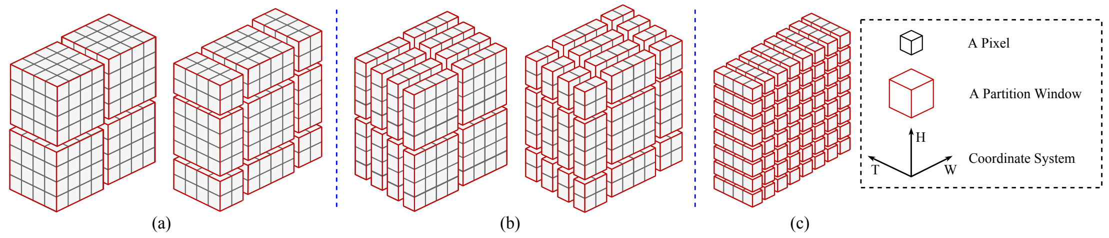


Figure 3.3: Illustration of different local partition strategies. (a) Regular and shifted partitions for spatial-temporal cubes of STS. (b) Regular and shifted partitions for spatial windows of Sep-STS. (c) Temporal vector partition for Sep-STS.

problem is to directly divide the feature maps into patches and treat each patch as a new element in the global attention [23,24]. This strategy is equivalent to aggressively downsampling the input with pixel shuffle [30] (downsampling factor equals to the patch size) and cannot well reconstruct high-quality image details which require fine-grained dependency modeling between pixels.

In this work, we introduce the local attention mechanism of Swin Transformer [25], which can effectively address the above issues. First, as the self-attention of Swin is computed inside local windows, it naturally avoids the heavy computational burden of global attentions. Second, Swin employs a shifted-window partition strategy to connect different local regions, and alternatively using regular and shifted-window partitions enables long-range dependency modeling. Nevertheless, this method is designed for image applications and cannot be easily applied to videos.

Spatial-temporal local attention. To make the Swin Transformer compatible with video inputs, we generalize the local attention mechanism to spatial-temporal space and propose the STS attention. As shown in Figure 3.3(a), the STS is conceptually similar to Swin but involves an extra temporal dimension.

Given an input feature of size $C \times T \times H \times W$ where C, T, H, W respectively represent the channel, time, height, and width dimensions, we first partition it to $\frac{HW}{M^2}$

non-overlapped 3D sub-cubes with the shape of each cube as $T \times M \times M$ (Figure 3.3(a)-left) and then perform standard multi-head self-attention (MSA) on each sub-cube. Note that each element of this cube is a C-dimensional feature vector, and we omit the channel dimension when describing the partition strategies for simplicity. Once all the sub-cubes are processed, we merge them back to recover the original shape of the input. In order to bridge connections across neighboring cubes, we adopt a shifted-cube partition strategy, which shifts the cubes to top-left by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels (Figure 3.3 (a-right).

Separation of space and time. Although the above STS can handle video inputs, it may suffer from memory issues when dealing with large cube sizes, *i.e.*, large T or M. To alleviate this issue, we propose the Sep-STS by separating the spatial-temporal computations into space and time.

First, for the computation in space, given an input feature map with a size of $C \times T \times H \times W$, we first partition it into $\frac{THW}{M^2}$ non-overlapped 2D sub-windows with a size of $M \times M$ as shown in Figure 3.3(b)-left, and then perform the standard MSA for each sub-window. For connecting different windows, as we restrict our computations in 2D here, we simply use the shifted window partition strategy of the Swin for each frame as shown in Figure 3.3(b)-right.

Second, for the computation in the temporal dimension, we reshape the input feature map into HW temporal vectors with a length of T as shown in Figure 3.3(c) and perform MSA inside each vector such that the dependencies across frames can be modeled. This step complements the self-attention in the spatial domain, and thus the two operations need to be used together to process videos.

Sep-STS block. Based on the Sep-STS attention, we devise our main component,

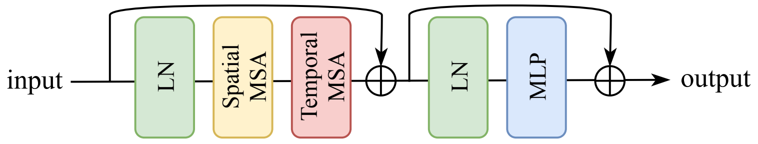


Figure 3.4: Illustration of the Sep-STS block. The Spatial MSA and Temporal MSA represent the multi-head self-attentions in spatial and temporal local windows, respectively (Section 3.4.1).

the Sep-STS block, which is composed of separated spatial and temporal attention modules as well as an MLP (Figure 3.4). The MLP adopts a two-layer structure and uses the GELU function [31] for activation. Similar to [25], we apply Layer Normalization (LN) [32] and residual connections [33] in this block to stabilize training. Similar to Swin, the regular and shifted partitions are employed alternatively for consecutive Sep-STS blocks to model long-range spatial-temporal dependencies.

Memory usage. The Sep-STS attention factorizes a computationally expensive operation into two lighter operations in space and time, which effectively lessens the memory usage reducing from $\mathcal{O}((TMM)\cdot THW)$ of the STS to $\mathcal{O}((T+MM)\cdot THW)$ of our Sep-STS.

During training, compared to the STS baseline, we observe a 26.2% GPU memory reduction by using our Sep-STS. As the window size MM is usually much larger than the number of input frames T, this reduction ratio essentially relies on T which we set as 4 by default following the settings of the state-of-the-art algorithms [12, 26, 34]. Since the proposed framework is flexible and can be used for arbitrary number of frames, the Sep-STS can potentially give more significant memory reduction for a larger T. In addition, the space-time separation strategy can also reduce the computational cost similar to the memory usage. However, as the Sep-STS is naively implemented with two separate PyTorch [35] layers in our experiments, its run-time is in fact similar to that of the STS. Optimizing its implementation with a customized

CUDA kernel will be part of our future work.

Discussions. In this work, we explore the concept of local attention for Transformer-based video interpolation. Similar concepts have been adopted in other recent methods but with different forms and goals. For image recognition, Hu *et al*. [36] first propose a local relation network to adaptively determine aggregation weights in a local region for the center query pixel. Ramachandran *et al*. [37] propose a stand-alone self-attention network which uses vector-based attention to improve the aggregation weights but still shares the same spatial arrangement as the local relation network for image classification and object detection. Further, Liu *et al*. [25] build upon the success of local attention models and develop an efficient spatial arrangement Swin where the local window does not need to be centered at the query.

Nevertheless, these algorithms are designed for images, and less attention is paid to exploiting local attention mechanisms for videos due to difficulties caused by the extra temporal dimension. In addition, existing methods mainly focus on image recognition tasks that are generally seen as high-level vision tasks, while in this work we emphasize more on motion modeling and appearance reconstruction. In this work, we focus on the temporal extension of local attention modules for effective VFI. We explore the space-time separable local attention, which is in spirit similar to MobileNet [38] that improves a standard convolution by factorizing it into a depthwise convolution and a pointwise convolution. Furthermore, we propose a multi-scale kernel-prediction framework to fully exploit the features learned by local attention, as introduced next.

3.4.2 Frame Synthesis

With the features from the proposed encoder-decoder network, our VFIT synthesizes the output image by predicting spatially-variant kernels to adaptively fuse the source frames. Different from existing kernel-based video interpolation methods [5–7], we propose a multi-scale kernel-prediction framework using the hierarchical feature $\{F^l, l = 0, 1, 2\}$ as shown in Figure 3.2.

The frame synthesis network of VFIT is composed of three SynBlocks that make predictions at different scales, and each SynBlock is a kernel prediction network. VFIT fuses these multi-scale predictions to generate the final result by:

$$\hat{I}^l_{0.5} = f_{\text{up}}(\hat{I}^{l+1}_{0.5}) + O^l, \tag{3.4.1}$$

$$O^l = f^l_{\text{syn}}(F^l, I^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}), \tag{3.4.2}$$

where $l = 0, 1, 2$ represent different scales from fine to coarse, and f_{up} denotes the bilinear upsampling function. The synthesized frame at a finer scale $\hat{I}^l_{0.5}$ can be obtained by merging the upsampled output from the coarse scale ($f_{\text{up}}(\hat{I}^{l+1}_{0.5})$) and the prediction of the current SynBlock (O^l). The output at the finest scale $\hat{I}^0_{0.5}$ is the final result of our VFIT, *i.e.*, $\hat{I}_{0.5} = \hat{I}^0_{0.5}$, and the initial value $\hat{I}^3_{0.5} = 0$. Here, f^l_{syn} is the l-th SynBlock which takes the spatial-temporal feature F^l and the frame sequence $I^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}$ as input, and I^l_t represents a frame I_t downsampled by a factor of 2^l with bilinear interpolation, where I^0_t is equivalent to the original frame without downsampling.

SynBlock. Given the input feature map $F^l \in \mathbb{R}^{C \times T \times H \times W}$, the SynBlock generates its prediction at the l-th scale by estimating a set of deformable kernels [29] to aggregate the information from the source frames.

As illustrated in Figure 3.2, we first unbind F^l at the temporal dimension to get T separate feature maps for all input frames, denoted as $F^l_{\{-(\lfloor\frac{T}{2}\rfloor-1),\cdots,\lceil\frac{T}{2}\rceil\}}$, and for each frame t, $F^l_t \in \mathbb{R}^{C\times H\times W}$. Then we feed F^l_t into three small 2D CNNs to obtain the per-pixel deformable kernels for frame I^l_t, including the kernel weights $W^l_t \in \mathbb{R}^{K\times H\times W}$, horizontal offsets $\alpha^l_t \in \mathbb{R}^{K\times H\times W}$, and vertical offsets $\beta^l_t \in \mathbb{R}^{K\times H\times W}$, where K is the number of sampling locations of each kernel.

With the predicted kernels we obtain the output of the SynBlock at location (x, y) for frame t as:

$$O^l_t(x,y) = \sum_{k=1}^{K} W^l_t(k,x,y) I^l_t(x+\alpha^l_t(k,x,y), y+\beta^l_t(k,x,y)),$$

which aggregates neighboring pixels around (x, y) with adaptive weights W similar to [29].

Finally, we generate the output at scale l by blending O^l_t of all frames with learned masks. Specifically, we concatenate the feature maps $F^l_{\{-(\lfloor\frac{T}{2}\rfloor-1),\cdots,\lceil\frac{T}{2}\rceil\}}$ at the channel dimension and send the concatenated features to a small 2D CNN to produce T blending masks $M^l_{\{-(\lfloor\frac{T}{2}\rfloor-1),\cdots,\lceil\frac{T}{2}\rceil\}}$. Note that we use a softmax function as the last layer of the CNN to normalize the masks along the temporal dimension. The final output of the SynBlock f^l_{syn} is generated by:

$$O^l = \sum_t M^l_t \cdot O^l_t. \tag{3.4.3}$$

3.5 Experimental Results

We show the main experimental results in this section.

3.5.1 Implementation Details

Network. As shown in Figure 3.2, the VFIT encoder consists of four stages that have 2, 2, 6, and 2 Sep-STS blocks, respectively. The skip connections between the encoder and decoder are realized with concatenation. For all three SynBlocks, we set the deformable kernel size as $K = 5 \times 5$. We present two variants of VFIT: a base model VFIT-B and a small one VFIT-S, where the model size of VFIT-S is about 25% of VFIT-B. The two models use the same architecture, and the only difference is the channel dimension of each stage, where we shrink the channels by half for VFIT-S.

Training. For training our network, we employ a simple ℓ_1 loss: $||I_{0.5} - \hat{I}_{0.5}||$, where $I_{0.5}$ is the ground truth. We use the AdaMax optimizer [39] with $\beta_1 = 0.9, \beta_2 = 0.999$. The training batch size is set as 4. We train the models for 100 epochs, where the learning rate is initially set as $2e^{-4}$ and gradually decayed to $1e^{-6}$.

Dataset. Similar to [26], we adopt the Vimeo-90K septuplet training set [14] to learn our models, which consists of 64612 seven-frame sequences with a resolution of 448×256. The first, third, fifth, and seventh frames of each sequence correspond to I_{-1}, I_0, I_1, I_2 in Figure 3.2 and are used to predict the fourth frame corresponding to $I_{0.5}$. For data augmentation, we randomly crop 256×256 image patches from frames, and perform horizontal and vertical flipping, as well as temporal order reverse.

We evaluate the models on the widely-used benchmark datasets, including the Vimeo-90K septuplet test set [14], UCF101 dataset [40], and DAVIS dataset [41].

Table 3.1: Quantitative comparisons on the Vimeo-90K, UCF101, and DAVIS datasets. Numbers in bold indicate the best performance and underscored numbers indicate the second best.

Method	# Parameters (M)	Vimeo-90K		UCF101		DAVIS	
		PSNR (↑)	SSIM (↑)	PSNR (↑)	SSIM (↑)	PSNR (↑)	SSIM (↑)
SuperSloMo [9]	39.6	32.90	0.957	32.33	0.960	25.65	0.857
DAIN [11]	24.0	33.35	0.945	31.64	0.957	26.12	0.870
SepConv [6]	21.6	33.60	0.944	31.97	0.943	26.21	0.857
BMBC [10]	11.0	34.76	0.965	32.61	0.955	26.42	0.868
CAIN [42]	42.8	34.83	0.970	32.52	0.968	27.21	0.873
AdaCoF [7]	21.8	35.40	0.971	32.71	0.969	26.49	0.866
QVI [12]	29.2	35.15	0.971	32.89	0.970	27.17	0.874
SoftSplat [43]	<u>7.7</u>	35.76	0.972	32.89	0.970	27.42	0.878
FLAVR [26]	42.4	36.30	0.975	33.33	**0.971**	27.44	0.874
VFIT-S	**7.5**	<u>36.48</u>	<u>0.976</u>	<u>33.36</u>	**0.971**	<u>27.92</u>	<u>0.885</u>
VFIT-B	29.0	**36.96**	**0.978**	**33.44**	**0.971**	**28.09**	**0.888**

Following [12, 26], we report performance on 100 quintuples generated from UCF101 and 2847 quintuples from DAVIS.

3.5.2 Evaluation against the State of the Arts

We evaluate the proposed algorithm against the state-of-the-art video interpolation methods: SepConv [6], DAIN [11], SuperSloMo [9], CAIN [42], BMBC [10], AdaCoF [7], SoftSplat [43], QVI [12], and FLAVR [26]. Among these methods, SuperSloMo, DAIN, CAIN, QVI, AdaCoF, and FLAVR are trained on the same training data as our models. For SepConv and BMBC, as the training code is not available, we directly use the pre-trained models for evaluation. The results of SoftSplat [43] are kindly provided by the authors.

We show quantitative evaluations in Table 3.1 where the PSNR and SSIM [44] are used for image quality assessment similar to previous works. Thanks to the learning capacity of the Sep-STS block, the proposed VFIT achieves better performance than

Table 3.2: Run-time of evaluated methods in seconds per frame. The models are tested on a desktop with an Intel Core i7-8700K CPU and an NVIDIA GTX 2080 Ti GPU. The results are averaged on the Vimeo-90K dataset.

Method	BMBC	QVI	FLAVR	VFIT-S	VFIT-B
Run-time	0.57	0.08	0.15	0.08	0.14

the evaluated CNN-based methods, demonstrating the superiority of using Transformers for video interpolation. Specifically, with only 7.5M parameters, the VFIT-S is able to outperform FLAVR, the best video interpolation method to date, on all the evaluated datasets. Furthermore, the VFIT-B achieves more significant improvements over FLAVR (0.66 dB on Vimeo-90K and 0.65 dB on DAVIS). Since the videos of UCF101 have relatively low qualities with low image resolutions and slow motion as explained in [12], our performance gain is less significant. Note that the large improvement of the VFIT comes solely from the architecture design without relying on any external information, which differs sharply from several prior works [11,12,43] that use pre-trained optical flow and/or depth models and thus implicitly benefit from additional motion and/or depth labels.

In addition, we provide qualitative comparisons in Figure 3.5, where the proposed VFIT generates visually more pleasing results with clearer structures and fewer distortions than the baseline approaches. Moreover, to evaluate the accuracy of the interpolation results, we show overlaps of the interpolated frame and the corresponding ground truth in Figure 3.6. The overlapped images of VFIT are much clearer than the baselines, *i.e.* closer to the ground truth, indicating better capabilities of VFIT in motion modeling.

We also present the run-time of our method in Table 3.2. The run-time performance of the VFIT is on par with the best performing CNN-based algorithms, which

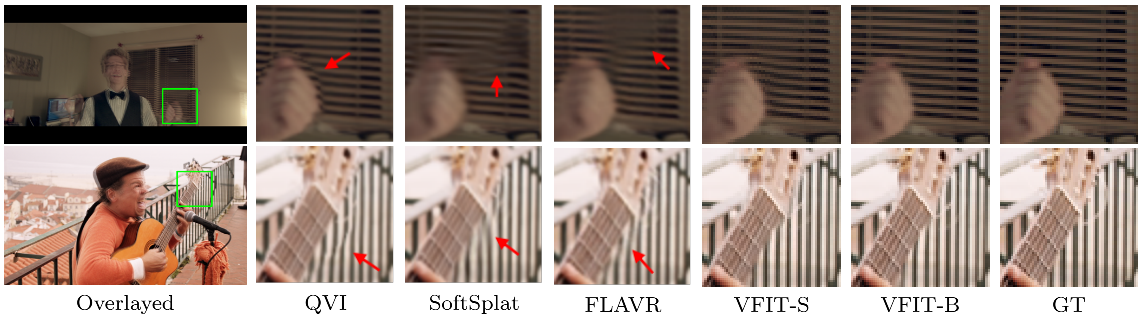


Figure 3.5: Qualitative comparisons against the state-of-the-art video interpolation algorithms. The VFIT generates higher-quality results with clearer structures and fewer distortions.

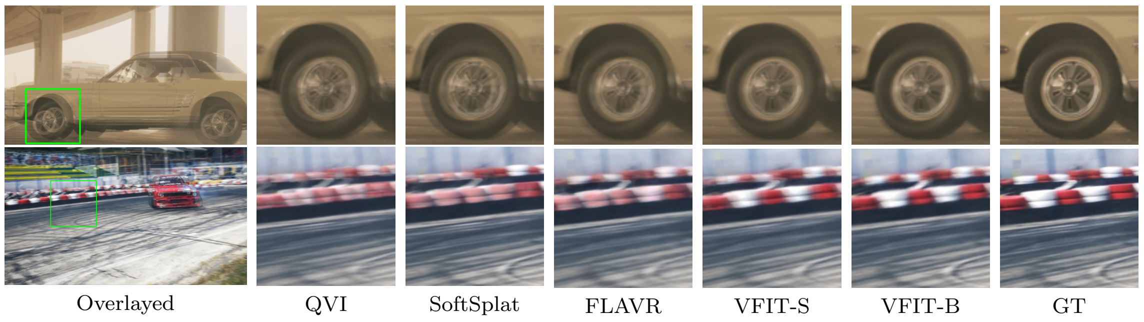


Figure 3.6: Overlap of interpolated frames and the corresponding ground truth, where a clearer overlapped image indicates a more accurate prediction. Note that for the second example, as the predictions of the baseline methods and the ground truth are not well aligned, the overlap of the red and white regions presents a blurry pink color.

facilitates its deployment in vision applications.

3.5.3 Ablation Study

We conduct the ablation studies on the Vimeo-90K dataset. As we notice that the training process converges quickly in the early training stage where differences between models can already be distinguished, we train all models in this study for 20 epochs to accelerate the development and concentrate on the most essential parts of VFIT.

Local attention. In contrast to our model which introduces the local attention

Table 3.3: Effectiveness of the proposed Sep-STS block.

Method	PSNR	SSIM	#Parameters (M)
VFIT-B	**36.02**	**0.975**	29.0
VFIT-STS	35.84	0.974	29.1
VFIT-CNN	35.82	0.973	65.4
VFIT-Global	35.18	0.971	42.4
$M = 4$	35.82	0.974	29.0
$M = 6$	35.90	0.974	29.0
$M = 8$	**36.02**	**0.975**	29.0
$M = 10$	35.93	0.974	29.0

Table 3.4: Comparison with the base models under different motion conditions.

Method	Fast	Medium	Slow
VFIT-B	**33.23/0.954**	**35.91/0.976**	**38.36/0.987**
VFIT-STS	32.91/0.950	35.77/0.975	38.27/0.987
VFIT-CNN	32.80/0.950	35.75/0.975	38.26/0.987
VFIT-Global	32.15/0.945	35.10/0.972	37.62/0.985

mechanism, several recent methods [23, 24] follow the basic structure of conventional Transformers in NLP to use global attention for vision applications, where the high computational cost of the global attention is circumvented by dividing input into patches and redefining each patch as a new element in self-attention. In our experiments, we also try this strategy by replacing each Sep-STS block of VFIT-B with a patch-based global-attention block, which is called VFIT-Global. As shown in Table 3.3, the result of VFIT-Global is lower than VFIT-B by as large as 0.84 dB, which emphasizes the essential role of local attention in Transformer-based VFI.

Sep-STS. To further validate the effectiveness of the Sep-STS block, we compare our VFIT-B with its two variants: 1) VFIT-CNN where all the Sep-STS blocks are replaced by convolutional ResBlocks [33], and each ResBlock is composed of two 3D convolution layers; and 2) VFIT-STS where the Sep-STS block is replaced by its inseparable counterpart, *i.e.*, the STS block.

As shown in Table 3.3, while VFIT-CNN uses more than two times parameters of VFIT-STS, these two models achieve similar results, demonstrating the advantages of using Transformers for video interpolation. Further, our base model VFIT-B, which uses the proposed Sep-STS as the building block, obtains even better performance than the VFIT-STS. It should be emphasized that the performance gain is significant as the Sep-STS block is initially designed to reduce memory usage as discussed in Section 3.4.1. This can be attributed to that the self-attention of the large-size sub-cubes in STS is relatively difficult to learn, and the space-time separation in Sep-STS can serve as a low-rank regularization [45] to remedy this issue.

To better analyze the performance of our models, we further compare with the baselines under different motion conditions. Following [46, 47], we split the Vimeo-90K test set into fast, medium, and slow motions, respectively. Table 3.4 shows VFIT-B outperforms VFIT-CNN by 0.43 dB on fast motion, 0.16 dB on medium motion, and 0.10 dB on slow motion, highlighting the exceptional capability of the proposed Sep-STS in handling challenging large-motion scenarios. We also provide interpolated frames from a video with fast motion in Figure 3.7 for comparisons.

To analyze the effect of different window size of the Sep-STS, we evaluate the VFIT-B with $M = 4, 6, 8, 10$, respectively. Table 3.3 shows, our model performs better as the window size is increased until $M > 8$. Thus, we choose $M = 8$ as our default setting in this work.

Multi-scale frame synthesis. As introduced in Section 3.4.2, we propose a multi-scale kernel-prediction network for final frame synthesis. To verify the effectiveness of this design, we experiment with a single-scale variant of the VFIT, called VFIT-Single, by removing the second and third SynBlocks in Figure 3.2. This single-scale

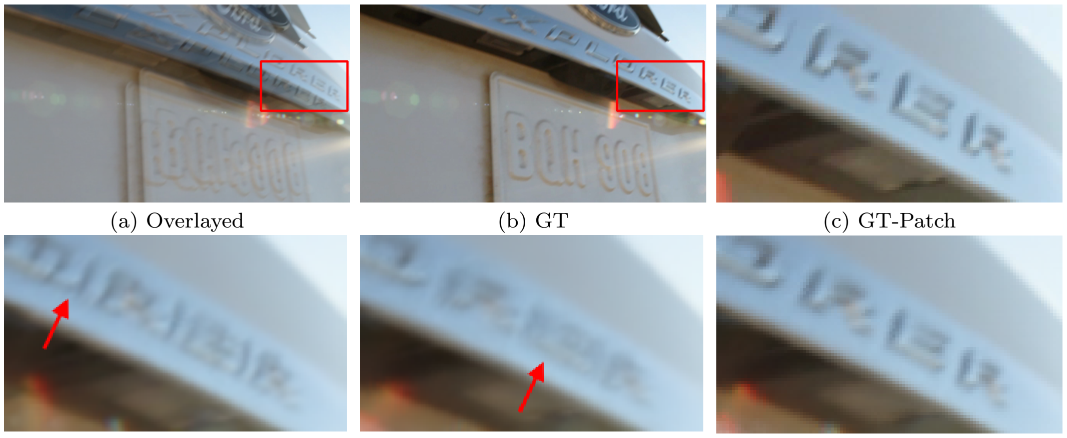


Figure 3.7: Interpolated frames from a video with fast motion. VFIT-CNN produces severe ghosting artifacts due to it incapability of handling large motion, while the result of VFIT-STS appears blurry. In contrast, the VFIT-B generates a higher-quality intermediate frame closer to the ground truth.

strategy is essentially similar to the ordinary kernel-prediction networks in [5–7]. The PSNR achieved by VFIT-Single is 35.54 dB, which is 0.48 dB lower than our base model VFIT-B. The large performance gap shows the importance of the multi-scale framework for fully realizing the potential of Transformers.

Note that we only apply the loss function to the final output, *i.e.*, the finest level output $\hat{I}^0_{0.5}$ of the multi-scale framework as introduced in Section 3.5.1. Alternatively, one may consider adding supervision to all-scale outputs of the network. However, we empirically find this scheme does not perform well.

Resizing modules. As illustrated in Figure 3.2, we use 3D convolution and deconvolution layers for downsampling and upsampling the feature maps. Motivated by the performance gain of our Sep-STS over CNN-based models, it is of great interest to explore the use of Transformer layers as the resizing modules for VFI.

Table 3.5: Comparison with Transformer-based resizing modules.

Method	PSNR	SSIM	Run-time (s)
VFIT-B	**36.02**	**0.975**	0.14
VFIT-TD	35.92	0.974	0.17
VFIT-TU	35.97	0.974	0.20

To answer this question, we adopt the method in [48] which introduces a Transformer-based resizing module for video classification by downsampling the query of the self-attention layer. To enable Transformer-based upsampling, we extend the idea in [48] by upsampling the query with bilinear interpolation. We respectively replace the convolution and deconvolution layers of VFIT-B with these Transformer-based downsampling and upsampling modules, and refer to the two variants as VFIT-TD and VFIT-TU. As shown in Table 3.5, both VFIT-TD and VFIT-TU perform slightly worse than our base model with degraded run-time performance, indicating that the current designs of Transformer-based resizing operations in computer vision are less effective for complex motion modeling. This is a limitation of our current work, which will be an interesting problem for future research.

3.6 Conclusion

In this paper, we propose a parameter, memory, and runtime efficient VFIT framework for VFI with the state-of-the-art performance. A significant part of our effort focuses on extending the local attention mechanism to the spatial-temporal space, and this module can be integrated in other video processing tasks. In addition, we demonstrate the effectiveness of a novel space-time separation scheme, which implies the necessity of well-structured regularizations in video Transformers. The architecture of VFIT is simple and compact, which can be effectively applied to numerous downstream vision tasks.

Bibliography

[1] T. Brooks and J. T. Barron, “Learning to synthesize motion blur,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[2] X. Xu, J. Pan, Y.-J. Zhang, and M.-H. Yang, “Motion blur kernel estimation via deep learning,” *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 194–205, 2017.

[3] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, “Jump: virtual reality video,” *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–13, 2016.

[4] A. Karargyris and N. Bourbakis, “Three-dimensional reconstruction of the digestive wall in capsule endoscopy videos using elastic video interpolation,” *IEEE Transactions on Medical Imaging*, vol. 30, no. 4, pp. 957–971, 2010.

[5] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive convolution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive separable convolution,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[7] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee, “Adacof: Adaptive collaboration of flows for video frame interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[8] Z. Shi, X. Liu, K. Shi, L. Dai, and J. Chen, "Video frame interpolation via generalized deformable convolution," *IEEE Transactions on Multimedia*, 2021.

[9] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[10] J. Park, K. Ko, C. Lee, and C.-S. Kim, "Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation," in *Proceedings of the European Conference on Computer Vision*, 2020.

[11] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[12] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang, "Quadratic video interpolation," in *Advances in Neural Information Processing Systems*, 2019.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[14] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.

[16] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the International Conference on Learning Representations*, 2020.

[19] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of the International Conference on Machine Learning*, 2021.

[20] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *Proceedings of the European Conference on Computer Vision*, 2020.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision*, 2020.

[22] X. Xu and C. C. Loy, "3D human texture estimation from a single image with transformers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[23] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[24] J. Cao, Y. Li, K. Zhang, and L. Van Gool, "Video super-resolution transformer," *arXiv preprint arXiv:2106.06847*, 2021.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the International Conference on Computer Vision*, 2021.

[26] T. Kalluri, D. Pathak, M. Chandraker, and D. Tran, "Flavr: Flow-agnostic video representations for fast frame interpolation," *arXiv preprint arXiv:2012.08512*, 2020.

[27] S. Niklaus, L. Mai, and O. Wang, "Revisiting adaptive convolutions for video frame interpolation," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2021.

[28] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," *arXiv preprint arXiv:2108.10257*, 2021.

[29] X. Xu, M. Li, W. Sun, and M.-H. Yang, "Learning spatial and spatio-temporal pixel aggregations for image and video denoising," *IEEE Transactions on Image Processing*, 2020.

[30] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[32] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[34] Z. Chi, R. M. Nasiri, Z. Liu, J. Lu, J. Tang, and K. N. Plataniotis, "All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling," in *Proceedings of the European Conference on Computer Vision*, 2020.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[36] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[37] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems*, 2019.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2014.

[40] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[41] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[42] M. Choi, H. Kim, B. Han, N. Xu, and K. M. Lee, "Channel attention is all you need for video frame interpolation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[43] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[44] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[45] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, pp. 1–37, 2011.

[46] M. Haris, G. Shakhnarovich, and N. Ukita, "Recurrent back-projection network for video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[47] X. Xiang, Y. Tian, Y. Zhang, Y. Fu, J. P. Allebach, and C. Xu, "Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[48] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

The following chapter is reproduced from an IEEE published paper:

Zhihao Shi, Xiaohong Liu, Chengqi Li, Linhui Dai, Jun Chen, Timothy N. Davidson, and Jiying Zhao. "Learning for Unconstrained Space-Time Video Super-Resolution." In IEEE Transactions on Broadcasting, 2021..

Contribution Declaration: Zhihao Shi (the author of this thesis) is the first author and the main contributor of this paper (more than 75%). He proposed the algorithm, wrote the corresponding program, conduct all experiments, and composed the manuscript. Xiaohong Liu helped revise the manuscript; Chengqi Li and Linhui Dai helped organize the experimental data; Jun Chen, Timothy N. Davidson, and Jiying Zhao helped revise the manuscript.

Chapter 4

Learning for Unconstrained Space-Time Video Super-Resolution

4.1 Abstract

Recent years have seen considerable research activities devoted to video enhancement that simultaneously increases temporal frame rate and spatial resolution. However, the existing methods either fail to explore the intrinsic relationship between temporal and spatial information or lack flexibility in the choice of final temporal/spatial resolution. In this work, we propose an unconstrained space-time video super-resolution network, which can effectively exploit space-time correlation to boost performance. Moreover, it has complete freedom in adjusting the temporal frame rate and spatial resolution through the use of the optical flow technique and a generalized pixelshuffle operation. Our extensive experiments demonstrate that the proposed method not

only outperforms the state-of-the-art, but also requires far fewer parameters and less running time.

4.2 Introduction

Recently, we have witnessed the popularization of Ultra-High-Definition TeleVision (UHDTV) and the rising of UHD TV shows in broadcasting. However, despite new media contents can be filmed by the advanced UHD recorder, remaking a large quantity of existed ones is impractical, leading to the overall short supply. VSR technologies provide a promising way to reconstruct HR videos from their LR counterparts. Furthermore, while watching sport events on TV, one may playback the fleeting moments with slow motion. VFI is one of the solutions that can temporally increase the frame rate of the broadcast videos.

In this paper, STVSR, the combination of VSR [1–7] and VFI [8–19], is mainly researched that aims at increasing spatial resolution and temporal frame rate simultaneously. The traditional approaches to STVSR [20–23] typically rely on strong assumptions or hand-crafted priors, and consequently are only suited to specific scenarios. The advent of deep learning has revolutionized many areas in computer vision, including, among others, image super-resolution [24,25], image quality assessment [26], image deblurring [27], image compression [28], and video coding [29]. In particular, it enables the development of data-driven approaches to VFI and Super-Resolution (SR) that can capitalize on the learning capability of neural networks as opposed to relying on prescribed rules. STVSR also naturally benefits from this advancement since it can be realized via a direct combination of VFI and SR. Specifically, one can first use VFI to increase the temporal frame rate, then leverage SR to enhance the spatial resolution. Moreover, the State-Of-The-Art (SOTA) VFI and SR methods (*e.g.*, the flow-based VFI methods [11–16] and the meta-learning-based SR methods [30]) have the freedom to adjust the frame rate and the spatial resolution, respectively. As a consequence, the resulting two-stage scheme is able to perform unconstrained STVSR. However, as pointed out in [31, 32], this two-stage scheme does not take advantage of the intrinsic relationship between temporal and spatial information, which limits the highest resolution that can be potentially achieved. In addition, performing STVSR in a two-stage fashion tends to be highly inefficient since VFI and SR are computationally intensive by themselves and likely involve many operations that can be shared.

To tackle these problems, two recent works [31, 32] have proposed a one-stage approach to STVSR by consolidating VFI and SR. This boosts performance by a large margin, while involving far fewer parameters and incurring less computational cost. However, this gain comes at a price. Indeed, compared to its two-stage counterpart, the new approach in [31,32] lacks flexibility in the choice of the final temporal/spatial resolution. Specifically, in the temporal domain, the CNN employed to synthesize the intermediate frame (based on two input frames) is tailored to a particular target time. As for the spatial domain, due to the use of the pixelshuffle layer [33] or deconvolution layer, it is impossible to adjust the up-sampling factor without modifying or retraining the network. Besides, the intrinsic limitation of these two layers renders fractional up-sampling factors unrealizable.

A natural question that arises here is whether the performance of the one-stage scheme can be retained without compromising flexibility? We offer an affirmative answer in this work by proposing an Unconstrained Space-Time Video Super-Resolution

Network (USTVSRNet), which is able to increase the temporal/spatial resolution of a given video by an arbitrary factor. For temporal interpolation, the optical flow technique is adopted to ensure the desired flexibility in the temporal resolution. Moreover, different from [31, 32], where the intermediate frame is synthesized at the feature level, we make predictions at both the image and feature levels, which leads to a noticeable performance improvement. As to spatial up-sampling, we introduce a Generalized Pixelshuffle Layer (GPL) that can project low-dimensional features to a high-dimensional space with the dimension ratio freely chosen. In addition, we construct a Scale-Attentive Residual Dense Block (SARDB) to generate scale-aware features. Due to the innovative features of our design, USTVSRNet is capable of up-sampling frames by an arbitrary factor with a single model. Our experimental results will show that the proposed method outperforms the SOTA two-stage methods, and does so with significantly lower computational cost.

The main contributions of this paper are as follows: (1) We propose a novel unconstrained STVSR method, which possesses the strengths of the SOTA one-stage and two-stage approaches while avoiding their drawbacks. (2) To realize unconstrained STVSR, several new mechanisms are introduced, including, integrating image-level and feature-level information to improve the quality of the synthesized intermediate frame, generalizing the Standard Pixelshuffle Layer (SPL) to increase the degrees of freedom in terms of up-sampling factor, and generating scale-aware features to make the network more adaptive. (3) Even when evaluated for particular temporal/spatial resolutions, the performance of the proposed unconstrained STVSR method remains highly competitive and outperforms the SOTA one-stage methods on various datasets.

4.3 Related Work

4.3.1 Video Frame Interpolation

The goal of VFI is to increase the frame rate by synthesizing intermediate frames while maintaining spatial and temporal consistencies with the given video frames. There are two major categories of video interpolation methods: kernel-based and flow-based methods.

As a pioneer of the kernel-based method, reference [18] employs a rigid spatially-adaptive convolution kernel to generate each target pixel. Naturally, very large kernels are needed for covering large motions, which leads to a substantial memory overhead. Reference [19] replaces regular 2D convolution kernels with pairs of 1D kernels to reduce the memory overhead. Even though that reduction is significant, the method cannot handle motions that are larger than the kernel size. To solve this problem, AdaCoF [17] breaks the rigid limitation of the regular convolution kernel and proposes a 2D deformable spatially-adaptive convolution scheme for VFI. Later, GDConvNet, introduced in [34], further exploited the degrees of freedom available in the three dimension of space-time, which improves the performance significantly. While kernel-based methods show promise, the time-oblivious nature of the convolution kernels means that the temporal information in the intermediate frames needs to be built into kernel-based methods in the design phase and cannot be easily adjusted during implementation.

In contrast, flow-based methods [12–16] generate the value of each pixel in the target intermediate frame according to an associated optical flow. Specifically, they first use the input frames to estimate source optical flows with the help of an optical flow estimation network [35–38]. They then convert the source optical flows into the associated ones with respect to the intermediate time t. Finally, the input frames are warped to the target frame according to these optical flows. As such, these methods have the inherent ability to perform interpolation with respect to an arbitrary time. Flow-based methods [12–15] typically adopt a linear model to convert the source optical flows. Recently, a quadratic model was proposed in [16], and preliminary results suggest that it may be able to better estimate the optical flows by exploiting four consecutive frames. For simplicity, in the present paper we will focus on the linear model, which involves two consecutive frames. It is straightforward to extend our work to incorporate higher-order models.

4.3.2 Super Resolution

SR has two main branches: SISR and video super-resolution, which aim at recovering a visually pleasing high-resolution image and video, respectively.

In terms of SISR, an end-to-end network which maps the interpolated LR images to HR ones was proposed in [39], and was enhanced by increasing network depth or stacking more complicated modules in [40–43]. However, all of these methods need to pre-compute an interpolated LR image before applying convolution neural networks, which significantly increases the computational complexity. To avoid the inefficient pre-computing process, the deconvolution layer and SPL, proposed by [44] and [33] receptively, enable the networks to directly output HR images from LR images, which dramatically reduces the computational complexity, and contributes to recovery of more fine-grained details.

On the other hand, the deconvolution layer and SPL also make it possible for VSR networks [2, 4–6, 45, 46] to output HR videos from LR ones directly. The processing pipeline of the SOTA VSR methods is roughly as follows: extract features from the reference frame and neighboring frames, then feed them (after proper alignment and fusion) into a reconstruction network to generate a super-resolved frame. By employing a deconvolution layer or an SPL in the reconstruction network, the SOTA VSR methods have been shown to generate satisfactory results in terms of efficiency and effectiveness on various datasets.

Although the SOTA SISR and VSR methods have performed satisfactorily on many datasets, they lack flexibility in adjusting the resolution of the final output. This is due to the intrinsic limitations of the deconvolution layer and the SPL. Recently, the meta-up-sample module proposed by [30] enables up-sampling by an arbitrary factor using a single model. Its refined version, known as the scale-aware up-sampling module [47], can better address the resulting memory overhead issues, but the underlying mechanism remains the same.

Unlike [30, 47], in the method proposed herein, we will generalize the SPL to release it from the constraints on the up-sampling factors. It will be shown that the new mechanism performs on par with, or slightly better than, SPL in terms of fixed scale up-sampling, and delivers better performance than that in [30, 47] in terms of up-sampling by arbitrary factors.

4.3.3 Space-Time Video Super-Resolution

Distinct from the separated operations of VFI and VSR, in a STVSR system we seek to simultaneously increase the temporal frame rate and the spatial resolution of a given video. This line of research was initiated in [48]. As the STVSR operation is a highly ill-posed inverse problem, due to the inadequacy of the available information, traditional methods [20–23] often resort to some hand-crafted priors or artificially-imposed constraints. For instance, reference [20] adopts a space-time directional smoothness prior and reference [21] makes a hypothesis that there is no great change in illumination for the static regions. As a result, these methods cannot cope with many real-world scenarios. In addition, the optimization for these methods is extremely computational inefficient (e.g., the processing speed for [21] is about 1 min/frame).

With the aid of deep learning, it is now possible to develop data-driven assumption-free STVSR methods. One simple way to do that is to realize STVSR via sequential execution of deep-learning-based VFI and SR. However, this two-stage scheme is sub-optimal since it is susceptible to error accumulation and makes no use of space-time correlation. In addition, a direct combination of VFI and SR without any consolidation is clearly inefficient in terms of the running cost.

In view of the problems with the two-stage approach, some one-stage STVSR methods [31,32] have been proposed, which are able to offer improved performance at a reduced cost. While they are highly innovative, these newly-proposed methods [31, 32], have two major limitations. Firstly, due to the use of a CNN to directly synthesize the intermediate frames, the temporal position of such frames is not adjustable after training. Secondly, there is no freedom to choose the spatial up-sampling factors to be different from those set in the training phase, nor to accommodate fractional factors. The main motivation of the present work is to remove these two limitations and realize unconstrained STVSR.

After posting a preprint of this submission on arXiv [49], we become aware of a concurrent and independent work [50], which addresses the temporal inflexibility

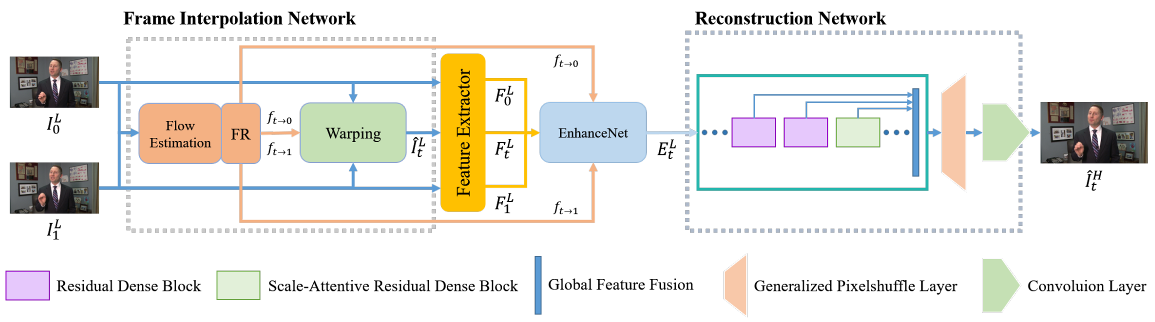


Figure 4.1: Illustration of the architecture of USTVSRNet.

problem using so-called TMBlock. But the spatial domain issue remains unsolved in [50]. A performance comparison is included in Section 4.6 of this submission.

4.4 Unconstrained Space-Time Video Super-Resolution Network

The goal of the proposed USTVSRNet is to transform a low-resolution low-frame-rate (LFR) video into a high-resolution high-frame-rate (HFR) one. Specifically, given two LR input frames (I_0^L and I_1^L), an arbitrary target time $t \in [0, 1]$, and an arbitrary spatial up-sampling factor $s \in [1, +\infty)$, the goal is to synthesize an intermediate HR frame I_t^H with $H = sL$. The overall architecture of USTVSRNet is shown in Fig. 4.1, which mainly consists of 4 sub-networks: a Frame Interpolation Network (FINet), a Feature Extractor, a Enhancement Network (EnhanceNet), and a Reconstruction Network.

As illustrated in Fig. 4.1, first a LR intermediate frame $\hat{I}_t^L$ is constructed by the FINet based on neighboring source frames (I_0^L and I_1^L) and bidirectional optical flows ($f_{t\to0}$ and $f_{t\to1}$). Then the features F_0^L, F_t^L and F_1^L are generated through the feature extractor from I_0^L, $\hat{I}_t^L$ and I_1^L respectively. Next, F_t^L is enhanced to E_t^L at the feature level through the enhancement network, and, finally, E_t^L is fed into the reconstruction network to produce a high-resolution frame $\hat{I}_t^H$ as an approximation of I_t^H. The details of each steps are outlined below.

4.4.1 Frame Interpolation Network

Given I_0^L and I_1^L, the FINet is employed to generate a coarse prediction $\hat{I}_t^L$ as the reference frame, which will be used in conjunction with the feature-level prediction to produce the final reconstruction. In principle, any flow-based VFI algorithm can serve this purpose. However, the SOTA systems [12–14, 16] often involve complex designs (e.g., depth information [13], quadratic model [16]), and consequently are not very efficient as a component of a larger system. For this reason, we consider a simple design for the FINet.

First a light-weight optical flow estimation network (PWCNet [38]) is utilized to estimate the bidirectional flows $f_{0\to1}$ and $f_{1\to0}$. They are then passed to the flow reverse layer [16] to predict backward flows $f_{t\to0}$ and $f_{t\to1}$. Specifically, we have

$$f_{t\to0} = \mathrm{FR}(f_{0\to t}), \tag{4.4.1}$$

where $f_{0\to t} = t * f_{0\to1}$, and FR denotes the flow reverse operation [16]; $f_{t\to1}$ can be computed in a similar way. Finally the reference frame is synthesized as:

$$\hat{I}_t^L = \frac{(1-t)\cdot B\cdot g(I_0^L, f_{t\to0}) + t\cdot(1-B)\cdot g(I_1^L, f_{t\to1})}{(1-t)\cdot B + t\cdot(1-B)}, \tag{4.4.2}$$

where B is a blending mask generated by a small CNN [12, 16, 51], and $g(\cdot)$ denotes the warping function.

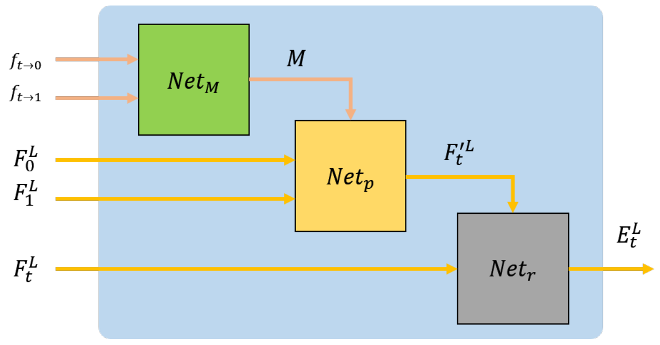


Figure 4.2: Illustration of the architecture of EnhanceNet.

4.4.2 Feature Extractor

The frame features F_0^L, F_t^L and F_1^L are extracted from I_0^L, $\hat{I}_t^L$ and I_1^L, respectively, through a feature extractor, which is composed of a convolution layer and several residual blocks [52].

4.4.3 Enhancement Network

As illustrated in Fig. 4.2, the inputs to the enhancement network consist of the three extracted feature maps F_0^L, F_t^L, F_1^L as well as the pre-computed bidirectional optical flows $f_{t\to0}$, $f_{t\to1}$. The goal of this sub-network is threefold: 1) leverage the source frame features (F_0^L and F_1^L) and the bidirectional optical flows ($f_{t\to0}$ and $f_{t\to1}$) to predict the features of the intermediate frame $F_t'^L$; 2) refine the generated reference frame at the feature level to alleviate the error accumulation problem as the coarse prediction $\hat{I}_t^L$ obtained in the first stage tends to have many artifacts; 3) fuse the source frames to the intermediate frame for better reconstruction under the guidance of $f_{t\to0}$, $f_{t\to1}$. The operation of the enhancement network can be expressed as:

$$M = \mathrm{Net}_M(f_{t\to0}, f_{t\to1}), \tag{4.4.3}$$

$$F_t^{'L} = \mathrm{Net}_p(F_0^L, F_1^L, M), \tag{4.4.4}$$

$$E_t^L = \mathrm{Net}_r(F_t^L, F_t^{'L}), \tag{4.4.5}$$

where Net_M, Net_p, and Net_r are composed of several residual blocks and a convolution layer, and M denotes the motion features extracted from $f_{t\to 0}$ and $f_{t\to 1}$ through Net_M.

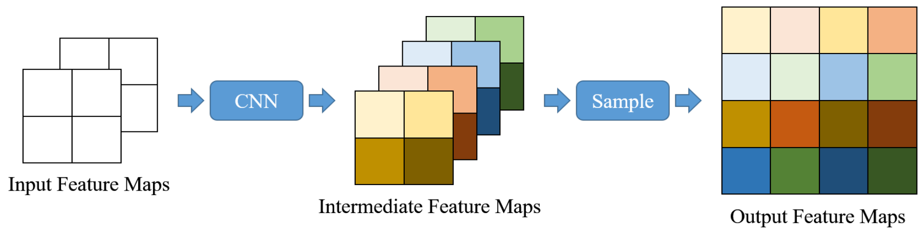
4.4.4 Reconstruction Network

The reconstruction network is designed using the residual dense network [43] as the backbone. We replace the SPL with a novel GPL described below, making it possible to up-sample low-resolution features by an arbitrary scale factor s. Moreover, we substitute one out of every K RDBs with our newly constructed SARDB, which is able to generate scale-adaptive features and contribute positively to the overall performance.

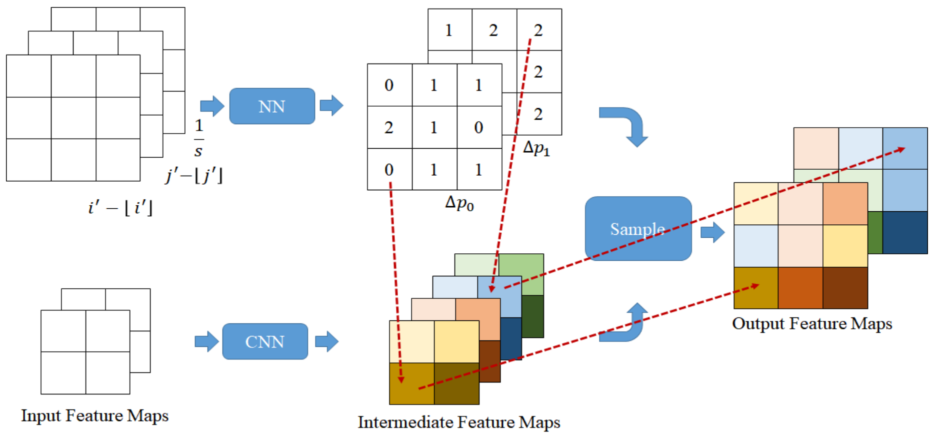
4.4.4.1 Generalized Pixelshuffle Layer

a new GPL is proposed to address the lack of flexibility in the SPL. Here we describe both the SPL and the GPL in parallel and highlight their differences.

The goal of the SPL and the GPL is to convert input feature maps of size $C_{\text{in}} \times H \times W$ to output feature maps of size $C_{\text{out}} \times sH \times sW$ for some scale factor s (s is allowed to be fractional for GPL but not for SPL). They both proceed in three steps:
Widen Input Features: The input feature maps are transformed via convolution to the intermediate feature maps T of size $C_{\text{mid}} \times H \times W$. Note that C_{mid} must be



(a)



(b)

Figure 4.3: Examples of the standard and generalized pixelshuffle layers, where (a) shows the standard layer while $C_{\text{in}} = 2, C_{\text{int}} = 4, C_{\text{out}} = 1$, and $s = 2$; (b) shows the generalized pixshuffle layer while $C_{\text{in}} = 2, C_{\text{int}} = 4, C_{\text{out}} = 2$, $s = 1.5$, and $p_c = 0$.

equal to s^2C_{out} for SPL, but can be an arbitrary positive integer for GPL.

Location Projection: Each spatial position on the output feature maps (i, j), $i \in [0, sH - 1], j \in [0, sW - 1]$ is projected to $(i', j') = (\frac{i}{s}, \frac{j}{s})$ on the intermediate feature maps.

Feature Mapping: Sample features from the intermediate feature maps T for each 3D output position $(i, j, c), c \in [0, C_{\text{out}} - 1]$ on the output feature maps according to a certain rule. Specifically, for SPL, the rule can be formulated as follows according to [33]:

$$\text{SPL}(T)_{i,j,c} = T_{\lfloor i'\rfloor,\lfloor j'\rfloor,C_{\text{out}}\cdot s\cdot\text{mod}(i,s)+C_{\text{out}}\cdot s\cdot\text{mod}(j,s)+c}. \tag{4.4.6}$$

A concrete example can be found in Fig. 4.3 (a). In contrast, for GPL, we propose to sample using

$$\mathrm{GPL}(T)_{i,j,c} = T_{\lfloor i' \rfloor, \lfloor j' \rfloor, p_c + \triangle p_c}, \tag{4.4.7}$$

where p_c is a pre-determined channel position and $\triangle p_c$ denotes an adaptive offset predicted by a small fully connected network with $(i' - \lfloor i' \rfloor, j' - \lfloor j' \rfloor, 1/s)$ as input (which is inspired by [30]). Note that we associate each 3D output position with a $\triangle p_c$, resulting in $sH \cdot sW \cdot C_{\text{out}}$ offsets in total. In the case where $p_c + \triangle p_c$ is not an integer, the sampling value $T_{\lfloor i' \rfloor, \lfloor j' \rfloor, p_c + \triangle p_c}$ can be computed using a linear interpolation function:

$$T_{\lfloor i' \rfloor, \lfloor j' \rfloor, p_c + \triangle p_c} = \sum_{i=0}^{C_{\text{out}}-1} \max(0, 1 - |p_c + \triangle p_c - i|) \cdot T_{\lfloor i' \rfloor, \lfloor j' \rfloor, i}. \tag{4.4.8}$$

By designing so, the sampling position $(\lfloor i' \rfloor, \lfloor j' \rfloor, p_c + \triangle p_c)$ on the intermediate feature maps is capable of moving along the channel direction to sample the needed feature. We provide a concrete example in Fig. 4.3 (b)

From Eqs. (4.4.6)-(4.4.7) and Fig. 4.3, we have two observations: 1) the proposed GPL not only achieves unconstrained up-sampling of feature maps but also has the capability to freely specify the channel dimension of the intermediate feature maps; 2) the GPL degenerates to the SPL if we set $C_{\text{mid}} = s^2 C_{\text{out}}$, $p_c = C_{\text{out}} \cdot r \cdot \mathrm{mod}(i, s) + C_{\text{out}} \cdot r \cdot \mathrm{mod}(j, s) + c$, and force $\triangle p_c = 0$. From these two points, it can be seen that the proposed GPL is a generalized version of the SPL with more degrees of freedom that can be fruitfully explored.

In our implementation, we set $p_c = c \cdot \frac{C_{\text{mid}}}{C_{\text{out}}} + \frac{C_{\text{mid}}/C_{\text{out}} - 1}{2}$. As such, the initial sampling positions are evenly distributed along the channel direction, which makes

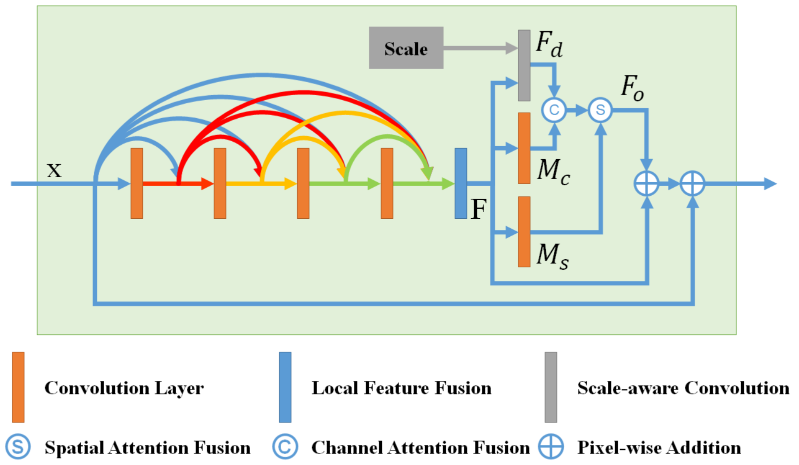


Figure 4.4: Illustration of the SARDB architecture.

it possible to capture features as diverse as possible. We initialize $\triangle p_c$ with 0 and set the learning rate of the small fully connected layer to be the same as the global learning rate.

4.4.4.2 Scale-Attentive Residual Dense Block

As pointed out in [47], the features generated by SR networks can be divided into scale-independent ones and scale-dependent ones, and the latter should be adapted to different scales. However, the scale-aware adaptation module introduced by [47] is built solely upon the spatial-wise attention mechanism, and makes no use of channel-wise attention [53]. With this observation, we propose SARDB to exploit the available degrees of freedom more thoroughly.

The architecture of the proposed SARDB is shown in Fig. 4.4. The features F output by the LFF [43] are fed into several convolution layers to generate spatial attention map M_s and channel attention map M_c respectively. Then, the scale-aware convolution [47] is employed to convert the features F into scale-dependent features F_d, which are then modulated by M_s and M_c by broadcasting and multiplication.

The above operations can be expressed as follows:

$$\begin{aligned}
F_d &= \mathrm{Sconv}(F), \\
M_c &= \mathrm{Net_c}(F), \\
M_s &= \mathrm{Net_s}(F), \\
F_o &= F_d \odot M_c \odot M_s + F.
\end{aligned} \tag{4.4.9}$$

Finally, the results from the upper branch F_o and lower branches x are merged to produce scale-adaptive features.

4.5 Experiments For Unconstrained Space-Time Video Super-Resolution

Unconstrained STVSR methods can flexibly adjust the temporal frame rate and the spatial resolution of the output video. In this section, we discuss the unconstrained STVSR. The experiment for fixed STVSR will be presented in Section 4.6.

4.5.1 Implementation Details

In our experiments, we explore the performance for different values of the target time t and the up-sampling factor s. We let t vary from 0 to 1 with a step size of 0.125, and s vary from 1 to 4 with a step of 0.5. During the implementation, we set $K = 4$ and $C_{\mathrm{mid}} = 5C_{\mathrm{in}} = 5C_{\mathrm{out}} = 5 \times 64$ respectively. The adopted loss function, training dataset, and training strategy are described below.

4.5.1.1 Loss Function

We employ two loss terms to train our network, being L_1 loss and perceptual loss [54], respectively:

Reconstruction Loss: The L_1 loss is used to measure the difference between the prediction and the ground-truth in a per-pixel manner, and can be formulated as follows:

$$\mathcal{L}_1 = \sum_x ||\hat{I}_t^H(x) - I_t^H(x)||_1. \tag{4.5.1}$$

The L_2 loss can also be used, but it is widely known in the image synthesis area that the L_2 loss could lead to blurry results to a certain degree. Following [17,55], we adopt Charbonnier penalty function [56] to optimize L_1 loss function and set $\epsilon = 10^{-6}$.

Perceptual Loss: Different from the per-pixel loss, the perceptual loss seeks to measure the difference from a global visual view, which has been shown effective in generating visually realistic images. The perceptual loss often leverages multi-scale feature maps extracted from a pre-trained network to quantify the difference. Here, we adopt VGG-16 [57] as the pre-trained network, and use feature maps from the last layer of each of the first three stages to measure the difference (*i.e.*, Conv1_2, Conv2_2 and Conv3_3). The loss can be expressed in the following form:

$$\mathcal{L}_p = \sum_{i=1}^{3} ||\Phi_i(\hat{I}_t^H) - \Phi_i(I_t^H)||_2^2, \tag{4.5.2}$$

where $\Phi_i(I_t^H), i = 1, 2, 3$ are the aforementioned three feature maps corresponding to I_t^H while $\Phi_i(\hat{I}_t^H)$ corresponds to $\hat{I}_t^H$.

Overall Loss: By combining the L_1 loss and the perceptual loss, the overall loss can be defined as:

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_p, \tag{4.5.3}$$

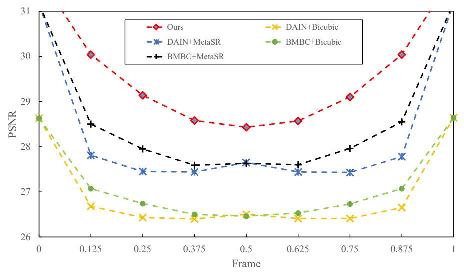
where λ is a hyper-parameter to balance the L_1 loss term and the perceptual loss term. Experimentally, we find setting $\lambda = 0.04$ reaches the best performance.

4.5.1.2 Training Dataset

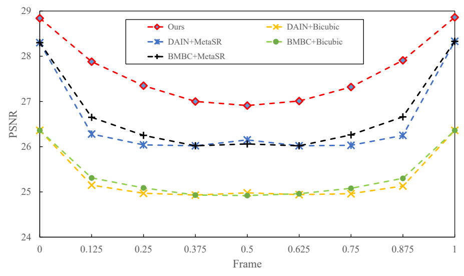
Adobe-240 dataset [58] consists of 133 handheld recorded videos, which mainly contain outdoor scenes. The frame rate of each video is 240 fps, with spatial resolution as $720 \times 1,280$. From this set, 103 videos are randomly selected to construct our training dataset. That set is formed by successively grouping every 9 consecutive frames, and resizing them to 360×640 to form a training sequence $I_0^H, I_{0.125}^H, \cdots, I_1^H$. In this way, we obtain $10,895$ sequences in total. The LR frames are generated through bicubic down-sampling from the HR frames. We randomly crop image patches of size 56×56 from the LR frames for training. Horizontal/vertical flipping as well as temporal order reversal is performed for data augmentation.

4.5.1.3 Training Strategy

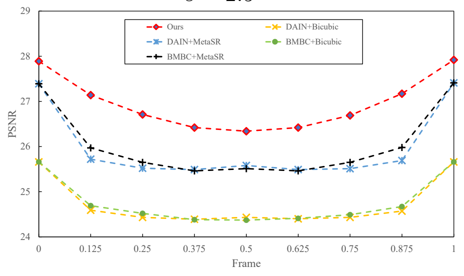
During the training phase, t and s are randomly selected to build each training batch. The image patches within a single batch share the same t and s. We adopt the Adam optimizer [59] with a batch size of 18, where β_1 and β_2 are set to the default values 0.9 and 0.999, respectively. We train our network for 30 epochs in total with the initial learning rate set to 10^{-4}, and the learning rate is reduced by a factor of 10 at epoch 20. The training is carried out on two NVIDIA GTX 2080Ti GPUs, which takes about one day to converge.



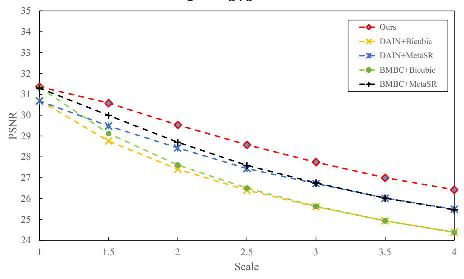
(a) PSNR for different temporal positions with $s = 2.5$



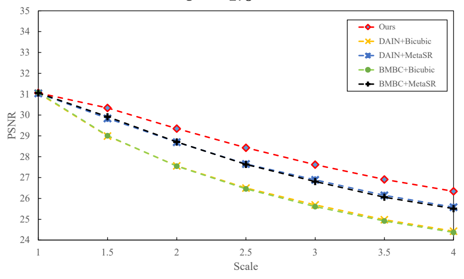
(b) PSNR for different temporal positions with $s = 3.5$



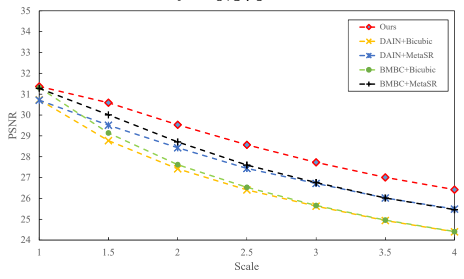
(c) PSNR for different temporal positions with $s = 4.0$



(d) PSNR for different scale factors with $t = 0.375$



(e) PSNR for different scale factors with $t = 0.500$



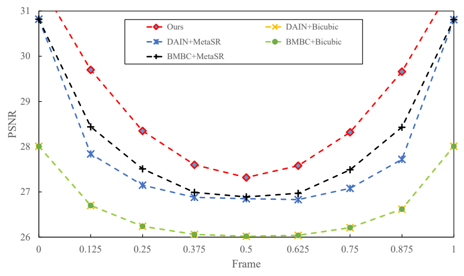
(f) PSNR for different scale factors with $t = 0.625$

Figure 4.5: Quantitative comparisons of unconstrained STVSR methods on Adobe240 dataset.

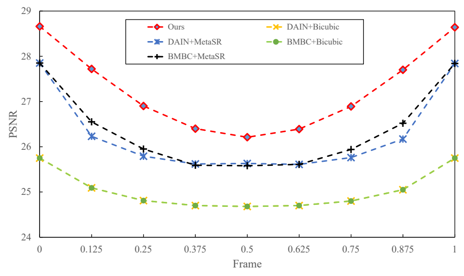
4.5.2 Evaluation Dataset

4.5.2.1 Adobe Testing Dataset [58]

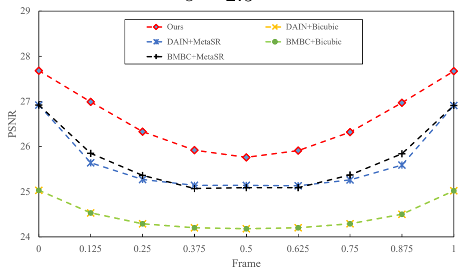
We treat the remaining 30 videos of the Adobe-240 dataset as an evaluation dataset. As in the case of the training dataset, we successively group every 9 consecutive frames (resized to 360×640), resulting in $2,560$ test sequences. For each sequence, the LR frames are generated from the HR ones via bicubic down-sampling.



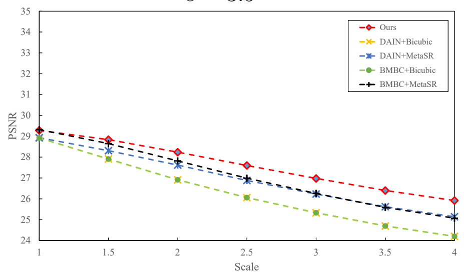
(a) PSNR for different temporal positions with $s = 2.5$



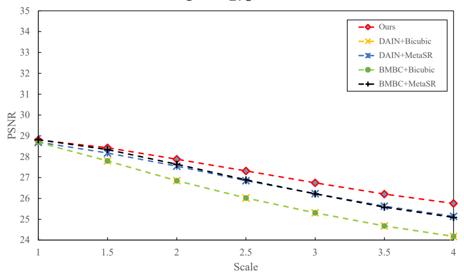
(b) PSNR for different temporal positions with $s = 3.5$



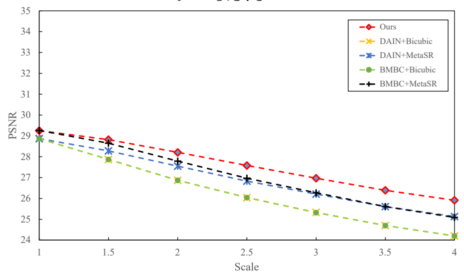
(c) PSNR for different temporal positions with $s = 4.0$



(d) PSNR for different scale factors with $t = 0.375$



(e) PSNR for different scale factors with $t = 0.500$



(f) PSNR for different scale factors with $t = 0.625$

Figure 4.6: Quantitative comparisons of unconstrained STVSR methods on Gopro dataset.

4.5.2.2 Gopro Testing Dataset [60]

This dataset contains 11 videos recorded by a hand-held camera. The frame rate of each video is 240 fps, and the image resolution is $720 \times 1,280$. The dataset is released in image format with a total of 12,221 images. We successively group every 9 consecutive images as a test sequence. In this way, 1,355 test sequences are generated.

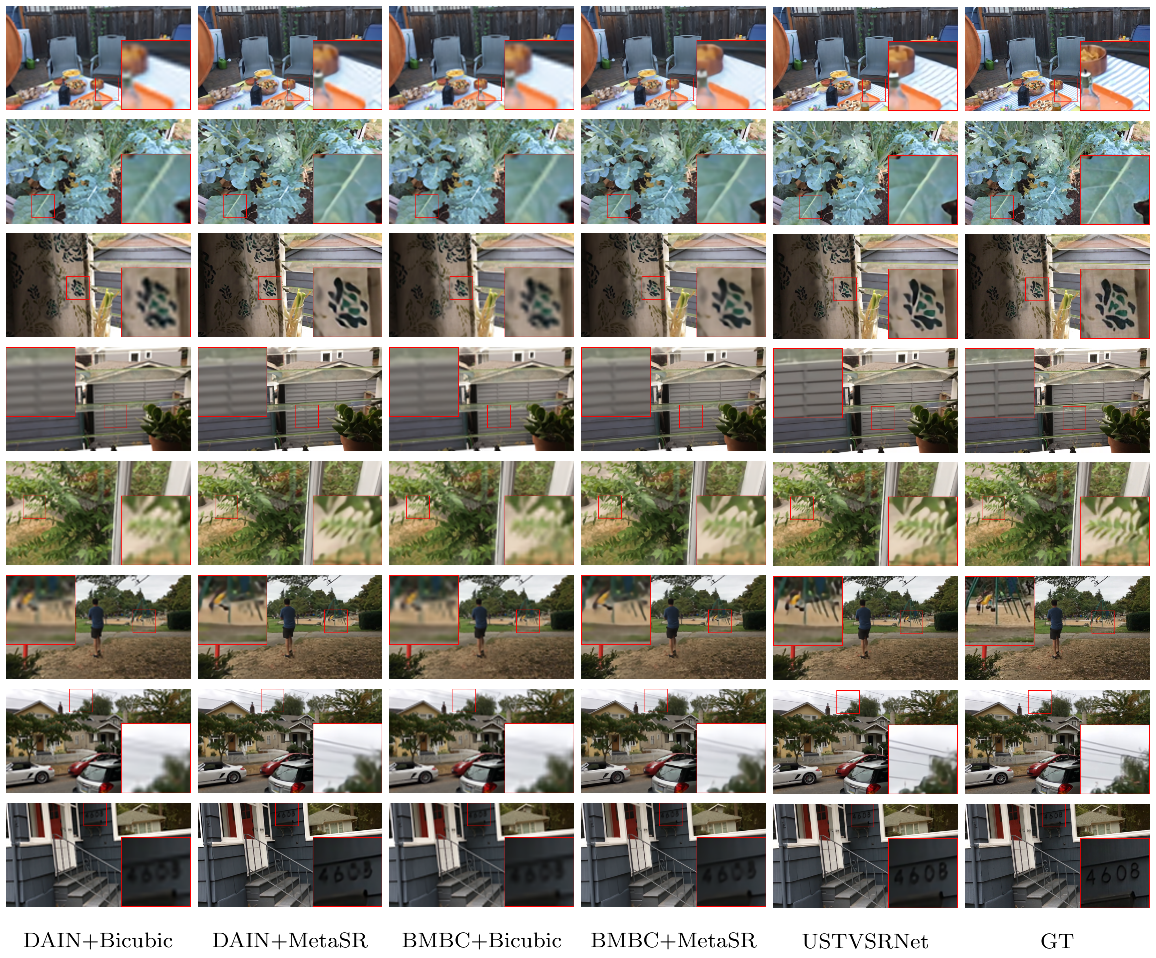


Figure 4.7: Qualitative comparisons of different Unconstrained STVSR algorithms.

4.5.3 Comparisons to SOTA methods

To the best of our knowledge, there is no one-stage method of this kind in the literature. So we only consider two-stage methods composed of SOTA unconstrained VFI methods (BMBC [61] and DAIN [13]) and SOTA SISR methods (since the code of [47] is not publicly available, we choose to use Meta-SR [30]). Here we set $t = 0, 0.125, \cdots, 1$ and $s = 1, 1.5, \cdots, 4$ respectively.

Fig. 4.5 and Fig. 4.6 show (a)-(c) PSNR scores for different temporal positions with

Table 4.1: Model size and running time comparisons with $s = 4$, where the model size is reported in millions (M) and the running time is reported in second (s) per frame.

Method	#Parameters (M)	Runtime (s)
USTVSRNet	12	0.12
BMBC + Meta-SR	33	0.34
BMBC + Bicubic	11	0.21
DAIN+Meta-SR	46	0.35
DAIN+Bicubic	24	0.18

$s = 2.5, 3.5, 4.0$, (d)-(f) PSNR scores for different scale factors with $t = 0.375, 0.500, 0.625$, on the Adobe240 and Gopro testing datasets, respectively. More results can be found in the supplementary materials. According to the experimental results, we make two observations: 1) the usage of more advanced VFI or SR methods contribute to better predicted results. For example, although BMBC+MetaSR and DAIN+MetaSR are both equipped with the same SR method, the former performs better than the latter due to the fact that BMBC is more advanced than DAIN; 2) compared to two-stage methods, the proposed method has better performance and is much more stable. This is due to the fact that the components of the two-stage methods work in isolation and cannot exploit the relationships between behavior in space and time.

Fig. 4.7 shows some qualitative comparisons with $t = 0.5$ and $s = 4$. It can be seen that the proposed method tends to generate more visually appealing results than the others. For instance, the proposed USTVSRNet yields sharper and clearer strips in the first row of Fig. 4.7; the leaves and the flower pattern generated by our method are much clearer than others in the second and the third rows, respectively.

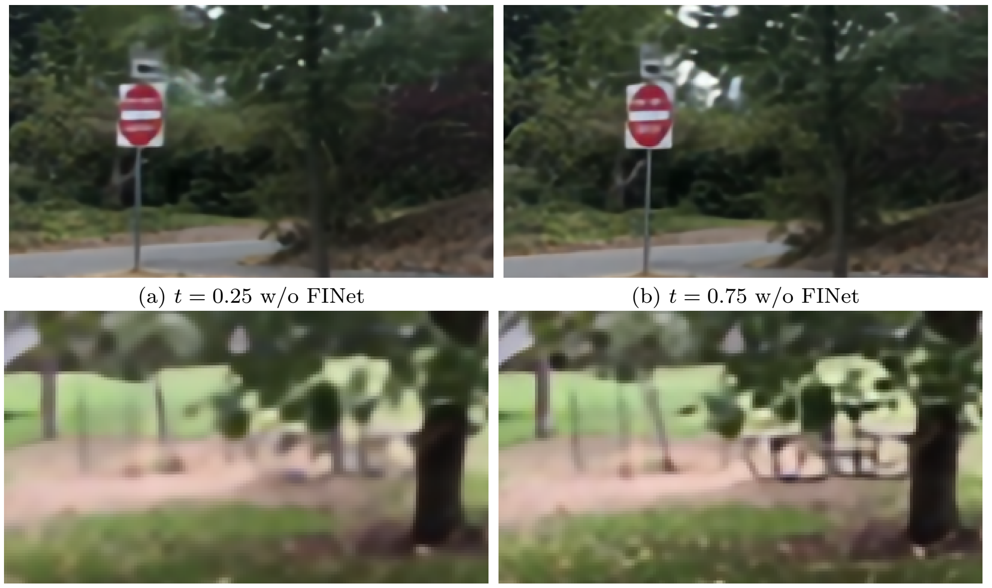
Tab. 4.1 provides comparisons between these methods in terms of model size and running time. Since two-stage methods are simple concatenations of the VFI and SR algorithms, they tend to be overweight and slow. In contrast, the proposed method is

Table 4.2: Quantitative results of ablation study regarding FINet and EnhanceNet with $s = 1, 2, 3, 4$, where PSNR and SSIM scores are averaged over t.

Method	$s = 1$		$s = 2$		$s = 3$		$s = 4$	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
(a)	31.44	0.9140	30.09	0.8949	27.82	0.8357	26.42	0.7767
(b)	32.02	0.9249	29.81	0.8882	27.38	0.8182	25.99	0.7550
(c)	32.39	0.9270	30.85	0.9078	28.44	0.8542	26.83	0.7945

more compact and efficient. Specifically, compared to the best performing two-stage method, namely BMBC + Meta-SR, the proposed USTVSRNet only has about 1/4 of size and takes half time to reconstruct one frame.

4.5.4 Ablation Study


(a) $t = 0.25$ w/o FINet (b) $t = 0.75$ w/o FINet

(c) w/o EnhanceNet (d) w/ EnhanceNet

Figure 4.8: Qualitative results of ablation study regarding FINet and EnhanceNet.

4.5.4.1 Effectiveness of image-level and feature-level interpolation

To validate the effectiveness of the image-level and feature-level interpolation, we consider the following three variants: a) USTVSRNet without FINet (where the flow estimation and reverse network are still preserved to generate optical flows for EnhanceNet). For this variant, we directly pass $F_t^{\prime L}$ to the reconstruction network; b) USTVSRNet without EnhanceNet. For this variant, F_t^L is directly fed into the reconstruction network; c) the complete network. We only provide the average scores over t with $s = 1, 2, 3, 4$ in Tab. 4.2 due to paper space limitation (Noted we excluded $t = 0, 1$ for $s = 1$). However, we observe similar results for different values of s.

From Tab. 4.2 (a) and (c), we can make the following two observations: 1) interpolating at the image level in addition to the feature level does contribute positively to the final reconstruction. It is expected that the performance can be improved further if more advanced VFI methods are adopted; 2) even without the explicit image-level prediction by FINet, the network still retains some, albeit reduced, ability to generate intermediate frames for different target times, owing to the implicit feature-level synthesis in EnhanceNet. A visual example is illustrated in Figs. 4.8 (a-b), in which two frames for different times are generated through USTVSRNet without FINet.

As can be seen from Tab. 4.2 (b-c), removing EnhanceNet degrades the performance in terms of PSNR and SSIM. Indeed, in addition to the loss of feature-level interpolation, removing EnhanceNet deprives the system of the ability to refine the reference frame and gain information from the source frames for reconstruction, which leads to unsatisfactory results, as shown in Figs. 4.8 (c-d).

4.5.4.2 Effectiveness of GPL and SARDB

We next demonstrate the effectiveness of the GPL and SARDB. Since the reconstruction network is functionally orthogonal to the other three sub-networks, we repurpose it as an SISR network. The following experiments are based on the SISR network and the RDN [43] is adopted as the backbone. The Vimeo90K dataset is commonly used in the SR area, thus we adopt this dataset in this part. We first demonstrate the effectiveness of the GPL and then SARDB.

Comparison with SPL: As we stated in the previous section, the proposed GPL is a generalized version of the SPL. Here we compare them in terms of the fixed scale up-sampling. The baseline is RDN, which employs the SPL at the end of the network to upscale features by a fixed scale factor. We will denote this system by S-RDN. Then, we replace the SPL by the proposed GPL to obtain a system called G-RDN. We evaluate each method on up-sampling factors $r = 2, 3, 4$, respectively. For each scale factor, the baseline RDN needs to be modified and re-trained. In contrast, for G-RDN, there is no need to modify the network structure. For fair comparisons, we also train it on each scale factor. The experiment results can be found in Tab. 4.3.

Table 4.3: Comparisons between SPL and GPL in terms of the fixed scale up-sampling on the Vimeo90k dataset.

Method	scale-factor	PSNR	SSIM	Runtime
S-RDN	×2	40.81	0.9780	0.13
G-RDN	×2	**40.84**	**0.9780**	0.15
S-RDN	×3	36.27	0.9465	0.07
G-RDN	×3	**36.30**	**0.9468**	0.09
S-RDN	×4	33.88	**0.9162**	0.05
G-RDN	×4	**33.90**	0.9161	0.07

As we can see from Tab. 4.3, G-RDN achieves slightly better results than S-RDN at all scales with a negligible running time increase (except for ×4 up-sampling on which G-RDN performs a little worse in terms of the SSIM value), which implies the performance of the GPL is on par or marginally better than that of the SPL in the scenario of fixed scale up-sampling. More importantly, compared to the SPL, the GPL enables the network to have the capability to perform unconstrained up-sampling task (will be demonstrated in the next), instead of restricted to certain specific scaling factors. Therefore, from these two aspects, the GPL can be considered as a generalized version of the SPL.

Comparison with unconstrained up-sampling methods: Different from constrained counterparts, unconstrained methods are able to upscale an image by an arbitrary factor within a single model. For this part, we compare the GPL with certain unconstrained up-sampling modules. Since only a few methods concentrate on arbitrary scale factor upsampling, we need to design several baseline systems. The following three baselines are taken into consideration: 1) the first baseline directly adopt the bicubic interpolation technique to up-sample images, denoted as Bicubic; 2) we first use a standard RDN to up-sample a image by k times (k is a fixed integer), then resize the up-sampled image to the desired size with bicubic interpolation, denoted as I-RDN($\times k$); 3) we replace the SPL of RDN with the bicubic interpolation method, which means that bicubic interpolation is used to upscale the feature maps, denoted as Bi-RDN. In addition, we also compare with the SOTA unconstrained up-sampling method, Meta-RDN [30], which is the same as Bi-RDN and G-RDN except for the final up-sampling module. Bi-RDN, Meta-RDN, and G-RDN are trained with the same unconstrained training strategy. Tab. 4.4 shows the evaluation results, which are averaged over $s \in [1, 4]$.

Tab. 4.4 illustrates the effectiveness of the proposed GPL. It has a clear advantage

Table 4.4: Comparisons for different unconstrained upscale methods on Vimeo90k dataset.

Method	PSNR	SSIM	Runtime
Bicubic	37.41	0.9282	0.02
I-RDN(×2)	38.25	0.9548	0.16
I-RDN(×4)	38.02	0.9578	0.17
Bi-RDN	40.37	0.9586	0.17
Meta-RDN	40.72	0.9578	0.20
G-RDN	**40.90**	**0.9590**	0.18

over other methods in terms of PSNR and SSIM. In particular, GPL outperforms the SOTA up-sampling module, Meta-Upscale, by about 0.18 dB and has a faster running speed.

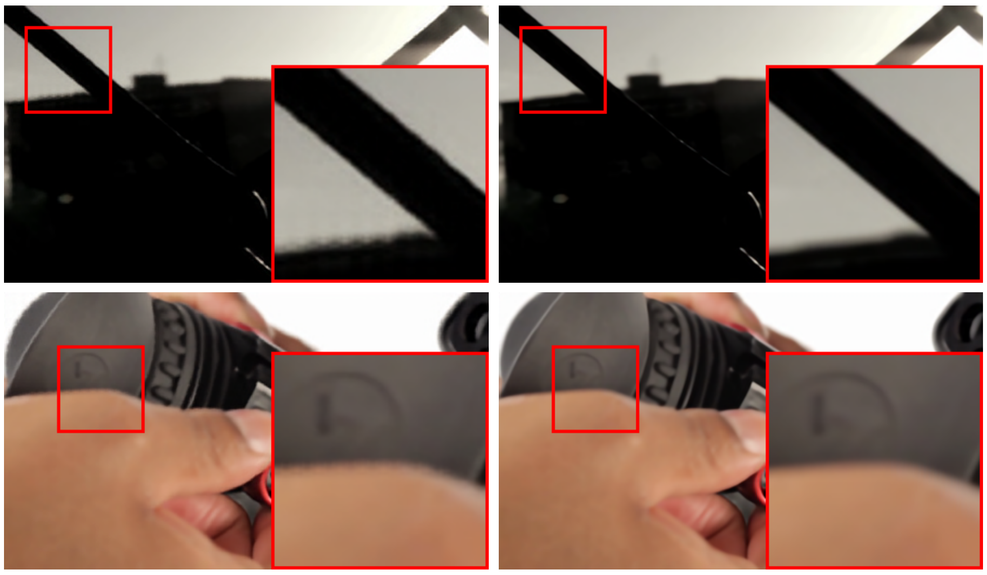

(a) FG-RDN (b) G-RDN

Figure 4.9: Qualitative results of ablation study regarding channel dimension freedom.

Importance of channel direction freedom: To illustrate the importance of channel direction freedom, we consider the following variant: we keep other components the same as G-RDN except for forcing $\triangle p_c = 0$, in which the sampling position cannot move along the channel direction. This system is denoted by FG-RDN. The

Table 4.5: Comparisons of FG-RDN and G-RDN with different C_{mid}.

Method	dimension	PSNR	SSIM	Runtime
FG-RDN	5×64	39.37	0.9525	0.17
G-RDN	1×64	40.78	0.9587	0.18
G-RDN	3×64	40.86	0.9589	0.18
G-RDN	5×64	**40.90**	**0.9590**	0.18
G-RDN	7×64	40.90	0.9590	0.18

experimental results can be found in Tab. 4.5, where one can easily find freedom of channel direction that leads to better reconstruction results. Indeed, if $\Delta p_c = 0$, then the local feature vectors of the output feature maps will become identical, which tends to generate blurry results or jagged edges. A visual example can be found in Fig. 4.9. Specifically, consider two different output positions on the output feature maps (i_1, j_1) and (i_2, j_2). If they are projected to the same location on the intermediate feature maps $\lfloor i'_1 \rfloor, \lfloor j'_1 \rfloor = \lfloor i'_2 \rfloor, \lfloor j'_2 \rfloor$, their output feature vectors $\text{GPL}(T)_{i_1,j_1}$ and $\text{GPL}(T)_{i_2,j_2}$ will be exactly the same according to Eq. (4.4.7) (due to $\Delta p_c = 0$), which limits the diversity of the output feature maps. From another point of view, forcing $\triangle p_c = 0$ corresponds to using the nearest interpolation to up-sample the feature maps. Naturally, its performance is not as good as that of G-RDN, and it is even worse than Bi-RDN (since bicubic interpolation is superior to nearest interpolation in nature). Therefore, the channel direction freedom plays an important role in the GPL.

Choice of C_{mid}: For the all experiments above, we set $C_{\text{mid}} = 5C_{\text{in}} = 5C_{\text{out}} = 5 \times 64$. Now, we investigate how to choose the channel dimension of intermediate feature maps C_{mid}. We fix $C_{\text{in}} = C_{\text{out}} = 64$ and vary C_{mid}. Tab. 4.5 shows the results, which are averaged over $s \in [1, 4]$. As shown in Tab. 4.5, as the dimension increases, the performance improves initially, but eventually becomes saturated. In particular, setting C_{mid} to more than 7×64 does not further improve the quality of the reconstructed HR image.

Effectiveness of SARDB: We finally investigate the contribution of the SARDB. Two networks are trained and evaluated on scale factor $r \in [1, 4]$: one with SARDB; the other one with RDB. We experimentally find that the scale-dependent features generated by SARDB improve the performance by 0.28 dB and 0.0012 in terms of PSNR and SSIM, respectively, with negligible increasing in the running cost.

4.6 Experiments For Fixed Space-Time Video Super-Resolution

Different from unconstrained STVSR, in fixed STVSR the temporal frame rate and the spatial resolution are not adjustable without retraining or modifying the network. Some experimental comparisons with fixed STVSR are provided below.

4.6.1 Implementation Details

In this section, t can only vary among $\{0, 0.5, 1\}$ and s is set to 4, which means the network can only up-sample a video by $\times 2$ and $\times 4$ times in terms of temporal and spatial resolutions, respectively. As in the previous section, we set $K = 4$ and $C_{\text{mid}} = 5C_{\text{in}} = 5C_{\text{out}} = 5 \times 64$. The training dataset and training strategy are described below.

4.6.1.1 Training Dataset

Same as [31], the Vimeo90k Triplet Training Dataset [11] is adopted to train our model, where we have $51,312$ sequences in total and the image resolution is 256×448.

Within each sequence, the first, the second and the third frames are treated as I_0^H, $I_{0.5}^H$ and I_1^H, respectively. Similarly, we use the bicubic down-sampling method to generate LR images from HR ones. We also perform horizontal and vertical flips, as well as temporal order reversal, for data augmentation.

4.6.1.2 Training Strategy

For each training iteration, t is randomly selected from $\{0, 0.5, 1\}$ and s is set as 4 to construct the corresponding training batch. The Adam optimizer is adopted with a batch size of 24. We train the network for 25 epochs in total, with the initial learning rate as 10^{-4}. The learning rate is reduced by $\times 2$ times at every 8 epochs for the first 16 epochs and by $\times 5$ times every 3 epochs for the last 9 epochs. The training is carried out on two NVIDIA GTX 2080Ti GPUs, which takes about one day to converge.

4.6.2 Evaluation Dataset

4.6.2.1 Adobe Testing Dataset and Gopro Testing Dataset

The Adobe Testing Dataset and the Gopro Dataset from the previous section are directly used as the two evaluation datasets for this section. There are $2,560$ and $1,355$ sequences in the Adobe dataset and the Gopro dataset, respectively, each with 9 frames. We only use the first, the fifth and the last frame of each sequence to compare different algorithms.

4.6.2.2 Vimeo90k Triplet Testing Dataset [11]

This dataset consists of $3,782$ video sequences, each with 3 frames. The image resolution of this dataset is 256×448. The first, second, and third frames in each video sequence are treated as I_0^H, $I_{0.5}^H$, and I_1^H respectively.

4.6.3 Comparisons to SOTA Methods

Here the STARNet [31], Zooming SloMo [32], and TMNet [50] are chosen as representatives of one-stage fixed STVSR methods. For fair comparison, they are retrained on our training dataset using the same strategy. As to two-stage methods, we combine pre-trained SOTA VFI methods (AdaCoF [17] and BMBC [61]) and SR methods (RSDN [6], RBPN [3] and DBPN [42] are chosen as representatives of the VSR and SISR methods, respectively).

We quantitatively compare our method with the chosen one-stage and two-stage methods under two well-known objective image quality metrics (PSNR and SSIM). The scores of the center frame and the average scores over all three frames are provided in Tab. 4.6. It can be seen that the proposed method ranks consistently at the top performance-wise, and comes in a close second in terms of the number of parameters. The two-stage methods not only suffer from large model size, but also lack competitiveness in performance since the constituent VFI and SR techniques are constrained to work in isolation. Although STARNet, Zooming SloMo, and TMNet are capable of handling diverse space-temporal patterns and improve the performance significantly, they are still behind the proposed method by a visible gap.

Table 4.6: Quantitative comparisons on Adobe240 dataset, Gopro dataset, and Vimeo90K dataset with $s = 4$, where 'Center' denotes the PSNR and SSIM value of the center frame $\hat{I}^H_{0.5}$ while 'Whole' is for average values of $\hat{I}^H_0$, $\hat{I}^H_{0.5}$, and $\hat{I}^H_1$. We highlight the first place and the second place in red and blue, respectively.

Method	#Parameters	Adobe240		Gopro		Vimeo90k	
	(million)	Center	Whole	Center	Whole	Center	Whole
USTVSRNet	12	26.64/0.7944	27.43/0.8102	25.94/0.8093	26.91/0.8347	30.32/0.8866	31.20/0.8956
TMNet	12	26.38/0.7890	27.28/0.8086	25.88/0.8083	26.90/0.8352	30.20/0.8847	31.09/0.8942
Zooming SloMo	11	26.36/0.7874	27.26/0.8069	25.84/0.8066	26.85/0.8333	30.11/0.8830	31.02/0.8928
STARNet	58	26.01/0.7639	27.01/0.7895	25.80/0.8053	26.80/0.8324	30.11/0.8828	30.92/0.8907
AdaCoF+RSDN	55	25.77/0.7583	26.90/0.7888	25.27/0.7825	26.46/0.8187	29.34/0.8654	30.69/0.8843
AdaCoF+RBPN	35	25.68/0.7501	26.64/0.7766	25.27/0.7773	26.25/0.8074	29.32/0.8612	30.48/0.8781
AdaCoF+DBPN	32	25.44/0.7322	26.46/0.7630	25.03/0.7612	26.01/0.7945	29.13/0.8543	30.30/0.8732
BMBC+RSDN	44	25.82/0.7575	26.95/0.7883	25.30/0.7808	26.54/0.8195	29.46/0.8670	30.75/0.8849
BMBC+RBPN	24	25.73/0.7504	26.67/0.7767	25.29/0.7762	26.26/0.8069	29.36/0.8636	30.50/0.8793
BMBC+DBPN	32	25.51/0.7329	26.50/0.7633	25.07/0.7608	26.03/0.7643	29.28/0.8568	30.38/0.8745

4.7 Conclusion

In summary, we have proposed an unconstrained STVSR method that has the freedom to arbitrarily adjust the temporal frame rate and spatial resolution of the output video. Beyond using the optical flow technique for temporal interpolation, several new ideas are introduced, which include the generalized pixelshuffle operation for upsampling, a refined mechanism to generate scale-adaptive features, and the integration of image-level and feature-level representations. Despite their excellent performance, it is conceivable that these new ideas could be further developed to yield even better performance. Moreover, there could well be likely alternative approaches to realizing unconstrained STVSR. In this sense, our work should be viewed as a stepping-stone towards a full-fledged framework for AI-enabled STVSR.

Bibliography

[1] A. Zhang, Q. Li, Y. Chen, X. Ma, L. Zou, Y. Jiang, Z. Xu, and G.-M. Muntean, "Video super-resolution and caching–an edge-assisted adaptive video streaming solution," *IEEE Transactions on Broadcasting*, 2021.

[2] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[3] M. Haris, G. Shakhnarovich, and N. Ukita, "Recurrent back-projection network for video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3897–3906.

[4] P. Yi, Z. Wang, K. Jiang, J. Jiang, and J. Ma, "Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3106–3115.

[5] Y. Tian, Y. Zhang, Y. Fu, and C. Xu, "Tdan: Temporally-deformable alignment network for video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3360–3369.

[6] T. Isobe, S. Li, X. Jia, S. Yuan, G. Slabaugh, C. Xu, Y.-L. Li, S. Wang, and Q. Tian, "Video super-resolution with temporal group attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Springer, 2020, pp. 8008–8017.

[7] X. Liu, K. Shi, Z. Wang, and J. Chen, "Exploit camera raw data for video super-resolution via hidden markov model inference," *IEEE Transactions on Image Processing*, vol. 30, pp. 2127–2140, 2021.

[8] D. Wang, A. Vincent, P. Blanchfield, and R. Klepko, "Motion-compensated frame rate up-conversionpart ii: New algorithms for frame interpolation," *IEEE Transactions on Broadcasting*, vol. 56, no. 2, pp. 142–149, 2010.

[9] K.-C. Yang, A.-M. Huang, T. Q. Nguyen, C. C. Guest, and P. K. Das, "A new objective quality metric for frame interpolation used in video compression," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 680–11, 2008.

[10] B. Yan, W. Tan, C. Lin, and L. Shen, "Fine-grained motion estimation for video frame interpolation," *IEEE Transactions on Broadcasting*, vol. 67, no. 1, pp. 174–184, 2020.

[11] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[12] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang, "MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[13] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3703–3712.

[14] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, “Super slomo: High quality estimation of multiple intermediate frames for video interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.

[15] S. Niklaus and F. Liu, “Context-aware synthesis for video frame interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1701–1710.

[16] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang, “Quadratic video interpolation,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[17] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee, “Adacof: Adaptive collaboration of flows for video frame interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5316–5325.

[18] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive convolution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[19] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive separable convolution,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[20] E. Shechtman, Y. Caspi, and M. Irani, “Space-time super-resolution,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 531–545, 2005.

[21] U. Mudenagudi, S. Banerjee, and P. K. Kalra, "Space-time super-resolution using graph-cut optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 995–1008, 2010.

[22] H. Takeda, P. Van Beek, and P. Milanfar, "Spatiotemporal video upscaling using motion-assisted steering kernel (mask) regression," in *High-Quality Visual Experience*. Springer, 2010, pp. 245–274.

[23] O. Shahar, A. Faktor, and M. Irani, "Space-time super-resolution from a single video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3353–3360.

[24] Z. Jiang, H. Zhu, Y. Lu, G. Ju, and A. Men, "Lightweight super-resolution using deep neural learning," *IEEE Transactions on Broadcasting*, vol. 66, no. 4, pp. 814–823, 2020.

[25] J. Wan, H. Yin, Z. Liu, A. Chong, and Y. Liu, "Lightweight image super-resolution by multi-scale aggregation," *IEEE Transactions on Broadcasting*, vol. 67, no. 2, pp. 372–382, 2020.

[26] Q. Qu, X. Chen, V. Chung, and Z. Chen, "Light field image quality assessment with auxiliary learning based on depthwise and anglewise separable convolutions," *IEEE Transactions on Broadcasting*, 2021.

[27] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, "Updresnn: A deep light-weight image upsampling and deblurring residual neural network," *IEEE Transactions on Broadcasting*, vol. 67, no. 2, pp. 538–548, 2021.

[28] T. Tian, H. Wang, L. Zuo, C.-C. J. Kuo, and S. Kwong, "Just noticeable difference level prediction for perceptual image compression," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 690–700, 2020.

[29] J. Lei, Y. Shi, Z. Pan, D. Liu, D. Jin, Y. Chen, and N. Ling, "Deep multi-domain prediction for 3d video coding," *IEEE Transactions on Broadcasting*, 2021.

[30] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-SR: A magnification-arbitrary network for super-resolution," 2019.

[31] M. Haris, G. Shakhnarovich, and N. Ukita, "Space-time-aware multi-resolution video enhancement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2859–2868.

[32] X. Xiang, Y. Tian, Y. Zhang, Y. Fu, J. P. Allebach, and C. Xu, "Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3370–3379.

[33] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[34] Z. Shi, X. Liu, K. Shi, L. Dai, and J. Chen, "Video interpolation via generalized deformable convolution," *arXiv preprint arXiv:2008.10680*, 2020.

[35] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with

convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[36] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[37] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[38] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[39] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199.

[40] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[41] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3929–3938.

[42] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1664–1673.

[43] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2472–2481.

[44] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conference on Computer Vision*. Springer, 2016, pp. 391–407.

[45] X. Yang, H. Mei, J. Zhang, K. Xu, B. Yin, Q. Zhang, and X. Wei, "DRFN: Deep recurrent fusion network for single-image super-resolution with large factors," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 328–337, 2018.

[46] J. Chen, X. Tan, C. Shan, S. Liu, and Z. Chen, "Vesr-net: The winning solution to youku video enhancement and super-resolution challenge," *arXiv preprint arXiv:2003.02115*, 2020.

[47] L. Wang, Y. Wang, Z. Lin, J. Yang, W. An, and Y. Guo, "Learning for scale-arbitrary super-resolution from scale-specific networks," *arXiv preprint arXiv:2004.03791*, 2020.

[48] E. Shechtman, Y. Caspi, and M. Irani, "Increasing space-time resolution in video," in *Proceedings of the European Conference on Computer Vision*, 2002, pp. 753–768.

[49] Z. Shi, C. Li, L. Dai, X. Liu, J. Chen, and T. N. Davidson, "Learning for unconstrained space-time video super-resolution," *arXiv preprint arXiv:2102.13011*, 2021.

[50] G. Xu, J. Xu, Z. Li, L. Wang, X. Sun, and M.-M. Cheng, "Temporal modulation network for controllable space-time video super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6388–6397.

[51] Z. Chi, R. M. Nasiri, Z. Liu, J. Lu, J. Tang, and K. N. Plataniotis, "All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling," in *Proceedings of the European Conference on Computer Vision*, 2020.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[53] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 3–19.

[54] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[55] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4463–4471.

[56] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2. IEEE, 1994, pp. 168–172.

[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2014.

[58] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep video deblurring for hand-held cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1279–1288.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[60] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3883–3891.

[61] J. Park, K. Ko, C. Lee, and C.-S. Kim, "BMBC: Bilateral motion estimation with bilateral cost volume for video interpolation," *arXiv preprint arXiv:2007.12622*, 2020.

Chapter 5

Conclusion

5.1 Conclusion

In this thesis, deep learning based video enhancement has been researched, where the first two articles focus on video frame interpolation and the last one deals with space-time video super-resolution.

More specifically, in the first article, a new mechanism named generalized deformable convolution is proposed, which unifies the essential ideas underlying flow-based and kernel-based methods and resolves some performance-limiting issues. Furthermore, a new VFI network is developed based on the mechanism, which surpasses the performance of the state-of-the-arts.

The second article successfully adapts Transformer to VFI, where a novel space-time separation window-based self-attention scheme is devised. Based on this scheme, a parameter, memory, and run-time efficient VFIT framework with state-of-the-art performance is presented. The architecture of VFIT is simple and compact, which can be effectively applied to numerous downstream vision tasks.

Last but not least, the third article proposes a one-staged unconstrained STVSR method that has complete freedom to arbitrarily adjust the temporal frame rate and spatial resolution of the output video. Meanwhile, the network is able to fully exploit the intrinsic relations between space and time, offering significant diversity gain over two-stage methods. The proposed USTVSRNet is unified, flexible, and performance-wise competitive, which sets a solid cornerstone for realizing full-fledged unconstrained STVSR.