GEM-VFI: Guided and Enhanced Motion Video Frame Interpolation

Teewalee Asawaniwed

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Sciencein Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Teewalee Asawaniwed, Ottawa, Canada, 2026

Members of the Jury

The following served on the Members of the Jury for this thesis.

External Member:	Peter Liu Professor, Department of Systems and Computer Engineering, Carleton University
Internal Member:	Abdulmotaleb El Saddik Professor, School of Electrical Engineering and Computer Science, University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Video Frame Interpolation (VFI) methods often optimize for pixel-based metrics such as Peak Signal-to-Noise Ratio (PSNR), which can favor over-smoothed results and fail to capture human visual perception. While high-performing VFI frameworks achieve strong visual quality under large and complex motion, their heavy computational overhead severely limits practical deployment. In this work, we introduce GEM-VFI, an enhanced framework designed to simultaneously improve feature representation and runtime efficiency. GEM-VFI incorporates three key proposed components: a Motion Detection Network (MD-Net) leveraging Squeeze-and-Excitation (SE) attention, Guided Flow Upsampling (GFU) for accelerated inference, and a perceptual loss optimization strategy to enhance fine structural details and reduce over-smoothing. Experimental results demonstrate that GEM-VFI achieves a 10–20% improvement in perceptual quality metrics while reducing inference latency by up to 13.4% compared to recent state-of-the-art baselines, successfully delivering superior visual synthesis alongside lower computational cost.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Jiying Zhao, for giving me the opportunity to pursue this research. This experience has been invaluable in allowing me to learn and grow, and to produce something tangible and complete through my own effort.

I am equally grateful to my co-supervisor, Dr. Yu Liu, for generously sharing his knowledge and industry experience in video processing. His perspective from outside academia broadened my understanding of the field and helped me see this work from angles I would not have otherwise considered.

I would also like to thank the other graduate research students in our MCRlab, Jiaqi Yang, Yanbo Feng, and Sihan Zhang, for sharing their own research experiences, which inspired and informed my own work. Their encouragement pushed me to go beyond my original plan and continually refine and improve what I had built. I am especially grateful for their candid feedback throughout this process, which gave me clear direction on where my work needed improvement and helped guide its development.

Finally, I want to thank my family and friends for their constant support and encouragement throughout this journey.

Table of Contents

List of Tables x

List of Figures xi

List of Algorithms xiii

List of Abbreviations xiv

1 Introduction 1

1.1 Background 1

1.2 Challenges and Motivation 3

1.3 Objective 4

1.4 Contributions 4

1.5 Structure of the Thesis 5

2 Background 7

2.1 Convolutional Neural Networks 7

2.2 Downsampling and Upsampling 9

2.2.1 Downsampling: constructing coarse levels 9

2.2.2 Upsampling: propagating features to finer levels 10

2.3 Deep Feature Representation 12

2.3.1 Residual Networks (ResNet) for Feature Extraction 13

2.3.2 Feature Space Robustness for VFI . . . 14
2.4 VGG Network . . . 14
2.4.1 VGG Features as Perceptual Embedding Space . . . 15
2.4.2 VGG16 . . . 15
2.5 Perceptual Metrics . . . 16
2.5.1 LPIPS (Learned Perceptual Image Patch Similarity) . . . 18
2.5.2 STLPIPS (Shift-Tolerant Perceptual Similarity Metric) . . . 21
2.5.3 NIQE (Natural Image Quality Evaluator) . . . 22
2.6 Cost Volume for Dense Motion Estimation . . . 24

3 Literature Review **26**
3.1 Approaches to Video Frame Interpolation . . . 26
3.2 Optical Flow Estimation . . . 28
3.2.1 Unidirectional vs. Bidirectional Formulations . . . 30
3.3 Core Challenges in Flow-based VFI . . . 30
3.4 Warping Artifacts in Flow-Based VFI . . . 31
3.5 Time-to-Location Ambiguity and Motion Descriptors . . . 32
3.6 Motion and Context Feature Extractors . . . 36
3.7 State-of-the-art Flow Upsampling Strategies in VFI . . . 37
3.7.1 The Limitation of Naive Flow Upsampling . . . 39
3.7.2 Convex Upsampling . . . 40
3.8 Knowledge Distillation in VFI . . . 41
3.8.1 Motivation and Problem Setting . . . 41
3.8.2 Fundamentals of Knowledge Distillation . . . 41
3.8.3 KD Mechanisms and Taxonomy in VFI . . . 41
3.8.4 BiM-VFI and Task-Specific Distillation . . . 42
3.9 Justifying the BiM-VFI Baseline . . . 43
3.10 Evaluation Practices in VFI Literature . . . 45

4 Methodology **46**
4.1 GEM-VFI: Guided and Enhanced Motion Video Frame Interpolation . . . 46
4.2 Image Pyramid Structure and State Propagation . . . 51
4.3 Multi-Scale Feature Extraction Architecture . . . 52
4.4 Symmetrical Teacher-Student Distillation Framework . . . 56
4.4.1 Parameter Sharing: One Network, Two Information Regimes . . . 57
4.4.2 Asymmetrical Operational Roles . . . 58
4.4.3 Knowledge Transfer Channels . . . 58
4.5 Motion Detection Network (MD-Net) . . . 59
4.5.1 Channel-wise Feature Recalibration via SE Attention . . . 65
4.6 Guided Flow Upsampling (GFU) . . . 67
4.7 Synthesis Network (SN) . . . 72
4.8 Loss Functions . . . 75

5 Experiments **78**
5.1 Experiment Setup . . . 78
5.1.1 Training Strategy and Implementation Details . . . 78
5.1.2 Dataset . . . 79
5.2 Evaluation Metrics . . . 81
5.2.1 Qualitative Results (Visual Comparisons) . . . 82
5.2.2 Perceptual Quality . . . 90
5.2.3 Fidelity metrics . . . 93
5.2.4 Computational complexity . . . 95
5.3 Ablation test . . . 98
5.3.1 Effect of Motion Detection Network (MD-Net) . . . 98
5.3.2 Effect of Perceptual Loss ($\mathcal{L}_{\text{perc}}$) . . . 100
5.3.3 Parameter Efficiency Analysis of GFU . . . 101
5.3.4 Coarse-to-Fine vs. Fine-Only Pyramid Processing . . . 102

6 Conclusions and Future Work **104**

6.1 Thesis Contributions 104

6.2 Discussion: Component transferability to Other Models 105

6.3 Limitations 106

6.4 Future Work 107

References **108**

List of Tables

2.1 Evaluation metrics for VFI. 18

3.1 Taxonomy of VFI Methods by Motion Representation 27
3.2 Taxonomy of optical flow estimation strategies 29
3.3 Taxonomy of optical flow estimation strategies in VFI 31
3.4 Comparison of Flow Upsampling Strategies in Optical Flow and VFI Methods 39
3.5 Knowledge Distillation in VFI 42

4.1 Comparison of Components Used During Training and Inference 48
4.2 Adaptive Pyramid Depth (L) and Input Scaling Configuration Across Operating Modes 52
4.3 Summary of Photometric and Flow-Centric Loss Functions 75

5.1 Summary of the Datasets Used in This Thesis. 80
5.2 Perceptual metrics comparison on Vimeo90K and SNU-Film 92
5.3 Fidelity metrics comparison (PSNR/SSIM) 93
5.4 Comparison of total parameters and FLOPs across evaluated models . . . 95
5.5 Module parameter counts and relative changes compared to BiM-VFI . . . 97
5.6 FLOPs comparison between BiM-VFI and Ours 97
5.7 Ablation study on MD-Net perceptual metrics 99
5.8 Ablation study on perceptual loss 100
5.9 Parameter comparison of CAUN and proposed GFU 102
5.10 Ablation study on coarse-to-fine vs. fine-only modes 103

List of Figures

1.1 General representation of video frame interpolation 2

2.1 CNN feature extraction pipeline and feature hierarchy 8
2.2 Corner-aligned vs. bilinear downsampling 10
2.3 Bilinear upsampling strategies for spatial resolution expansion 12
2.4 Residual block with skip connection 13
2.5 VGG16 feature backbone for perceptual learning 16
2.6 The pixel-wise metric dilemma 17
2.7 Two-stream computation flow of LPIPS with clean, independent side labels. 20
2.8 Natural Image Statistics (NSS) 22
2.9 NIQE Training Pipeline for Pristine Natural Scene Statistics Model 23
2.10 Patch partitioning and sharpness-based filtering under the NSS framework. 24

3.1 Comparison of motion estimation techniques under uniform, non-uniform, and bidirectional motion fields (BiM) 33
3.2 Geometric proof of uniqueness 34
3.3 Geometric interpretation of the motion prior $M = (r, \phi)$ 35
3.4 Geometric interpretation of the uniform motion prior $M^{\text{uni}} = (0.5, \pi)$ . . . 35
3.5 Motion and Context Feature Extractors of BiM-VFI 37
3.6 Multi-scale feature maps extracted by MFE and CFE branches. 38
3.7 BiM-VFI KDVCF motion distillation 43

4.1 Overview of the proposed multi-scale VFI framework 47
4.2 Multi-scale hierarchical architecture for coarse-to-fine motion estimation . . 53
4.3 Adaptive image pyramid construction with padding 54
4.4 Multi-scale ResNetPyramid architecture 55
4.5 Shared-parameter teacher–student framework 58
4.6 MD-Net Diagram 62
4.7 Channel indexing layout of the 81-channel cost volume 63
4.8 Channel attention mechanism via Squeeze-and-Excitation block 66
4.9 Overview of spatial mapping for $\times 4$ upsampling 67
4.10 Detailed convex mask upsampling process 68
4.11 GFU Diagram 70

5.1 Example Test Cases for Uniform and Non-Uniform Motion 82
5.2 License plate details 84
5.3 Watermelon case: 7-frame interpolation 85
5.4 Swimmer case: 7-frame interpolation 86
5.5 Soccer case: 7-frame interpolation 87
5.6 Kite lines case: 7-frame interpolation 88
5.7 Water splash case: 7-frame interpolation 89
5.8 Visual comparison at $t = 0.5$ 91
5.9 GPU memory and inference time comparison 96
5.10 Ablation study on MD-Net 99
5.11 Qualitative ablation study on perceptual loss ($\mathcal{L}_{\text{perc}}$). 101

6.1 Dense repetitive pattern 107

List of Algorithms

4.1 Teacher Process ($\mathcal{P}_{\mathcal{T}}$) (Training only) 49
4.2 Student Process ($\mathcal{P}_{\mathcal{S}}$) (Training / Inference) 49
4.3 Multi-Scale Feature Extraction via Shared ResNet Pyramid. 55
4.4 Motion Detection Network (Training Mode). 60
4.5 Motion Detection Network (Inference Mode). 61
4.6 Guided Flow Upsampling (GFU). 69
4.7 Synthesis Network (SN). 73

List of Abbreviations

BAPPS Berkeley–Adobe Perceptual Patch Similarity 19

BiM Bidirectional Motion Field 34, 36, 43

BiMFN BiM-guided FlowNet 29, 36, 46, 59, 65

CAUN Content-Aware Upsampling Network 32, 36, 40, 44, 100, 104

CFE Context Feature Extractor 36, 48, 50, 52

FID Fréchet Inception Distance 28, 44, 104

FPS Frames Per Second 95

GFU Guided Flow Upsampling 5, 32, 44, 48, 67, 74, 96, 100, 104

KD Knowledge Distillation 41

KDVCF Knowledge Distillation for VFI-centric Flow supervision 42, 43

LPIPS Learned Perceptual Image Patch Similarity 15, 18, 22, 45, 100

MD-Net Motion Detection Network 4, 5, 44, 46, 59, 98, 102, 104, 105

MFE Motion Feature Extractor 36, 48, 50, 52

MLP Multi-Layer Perceptron 66

MSE Mean Squared Error 17, 41

NIQE Natural Image Quality Evaluator 22, 23, 45, 100

NSS Natural Scene Statistic 22

PSNR Peak Signal-to-Noise Ratio 16–18, 45, 90

SE Squeeze-and-Excitation 4, 48, 64–66, 104, 105

SN Synthesis Network 36, 48, 72

SSIM Structural Similarity Index Measure 16, 18, 45, 90

STLPIPS Shift-Tolerant Perceptual Similarity Metric 21, 22, 100

TTL time-to-location 32, 36, 43, 44, 46, 81

VFI Video Frame Interpolation 1, 26, 28, 30, 31, 41, 43, 45, 46, 81, 105

VGG Visual Geometry Group 5, 14, 15

Chapter 1

Introduction

1.1 Background

Video Frame Interpolation (VFI) generates intermediate frames between consecutive frames to enable smoother motion and higher frame rates, and is widely used in applications such as slow-motion generation [1], video streaming, virtual reality, and video editing systems. In addition to improving visual smoothness, VFI is also an important component in video restoration pipelines, frame rate up-conversion, and computational photography systems where temporal resolution is limited by hardware constraints. However, achieving high-quality interpolation remains challenging due to complex motion patterns, occlusions, and temporal ambiguity.

As illustrated in Figure 1.1, the general pipeline of video frame interpolation involves taking two consecutive original frames, denoted as I_0 at temporal position $t = 0$ and I_1 at $t = 1$, as inputs. Mathematically, the interpolation process can be formulated as:

$$I_t = F(I_0, I_1, t) \tag{1.1.1}$$

where $t \in (0, 1)$ denotes the target temporal position of the intermediate frame to be synthesized. The interpolation function $F(\cdot)$, typically parameterized by a deep neural network, takes the reference keyframes and the target timestamp to predict the pixel values of the intermediate frame I_t, effectively modeling the underlying continuous motion between the boundaries. Specifically, when the continuous temporal domain is divided into n uniform temporal intervals, the model is evaluated at discrete time steps $t_k = \frac{k}{n}$, where $k = 1, 2, \ldots, n-1$ represents the index of each generated frame.

Beyond visual quality, modern VFI methods are also expected to satisfy practical constraints such as low latency, memory efficiency, and real-time performance on consumer hardware. These requirements are particularly important for deployment in mobile devices, edge AI systems, and online video platforms, where computational resources are limited and inference speed directly impacts user experience.

Existing deep learning-based methods can be categorized by their core mechanism. Flow-based methods provide a good balance between accuracy and efficiency by explicitly modeling pixel correspondences, while kernel-based approaches [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] learn adaptive spatial aggregation to synthesize intermediate frames. More recent transformer-based [13, 14, 15] and diffusion-based methods achieve higher perceptual quality and improved realism, but often require significantly higher computational cost, making them less practical for real-time applications.

Despite these advances, there remains a persistent trade-off between visual fidelity and efficiency, especially in scenarios involving complex motion or strict runtime constraints. This trade-off motivates the development of models that can retain the representational power of high-performance architectures while improving computational feasibility.

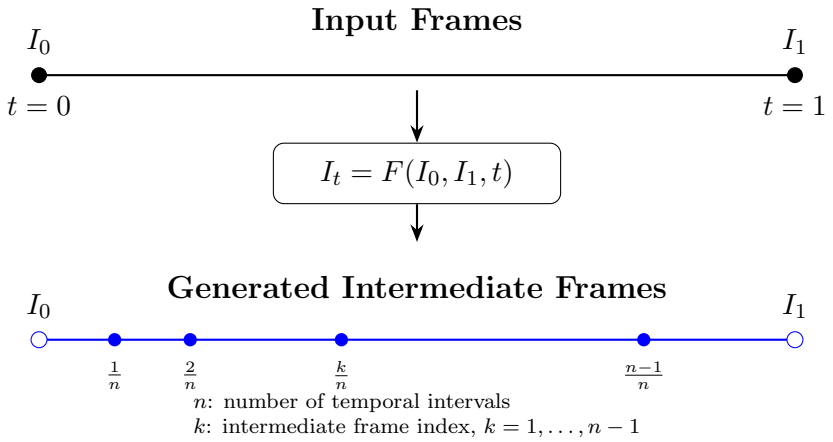


Figure 1.1: General representation of video frame interpolation. Two input frames I_0 and I_1 are used to estimate intermediate frames at temporal positions $t_k = \frac{k}{n}$, where n denotes the number of temporal intervals and k represents the index of each intermediate frame.

1.2 Challenges and Motivation

Despite recent progress, achieving high-quality frame interpolation remains difficult due to several fundamental challenges that consistently limit real-world performance [16]: large motion, occlusion, lighting variation, and non-linear motion.

Large motion occurs when objects undergo substantial displacement between consecutive frames, such as fast-moving subjects or abrupt camera motion. This makes it difficult for models to establish accurate correspondences over long spatial ranges, especially when feature matching is performed at a limited resolution. Lighting variation, including shadows, reflections, or exposure changes, further violates the brightness constancy assumption underlying many optical flow formulations and can degrade motion estimation even when geometric motion is small.

In addition to these well-known issues, two challenges are particularly critical in modern VFI systems. Occlusion arises when certain regions in the intermediate frame are not visible in either input frame, leaving the model without direct correspondence cues. In such cases, naive warping or blending often leads to ghosting, texture duplication, or structural inconsistency. Non-linear motion, including acceleration, deceleration, and directional changes, further complicates the problem by introducing motion ambiguity: multiple plausible trajectories can explain the same observed displacement when only two frames are available.

Another important but often underemphasized issue is temporal consistency under uncertainty. Since VFI models infer unseen intermediate states, small prediction errors can accumulate and manifest as flickering or temporal instability when extended to video sequences. This issue becomes more severe in high-frequency texture regions or under fast motion conditions.

These challenges frequently co-occur in real-world scenarios. For example, the SportsSloMo benchmark [17] shows that human-centric sports footage is particularly difficult due to highly deformable structures, frequent self-occlusion, and abrupt motion changes. Such conditions combine large non-rigid motion with persistent occlusion, making accurate motion estimation and frame synthesis especially challenging.

From a practical standpoint, another limitation of existing methods is their computational inefficiency in high-accuracy models. While recent architectures such as BiM-VFI [18] improve performance through bidirectional motion representation, they introduce additional computational overhead and memory consumption. This creates a gap between models that are accurate enough for challenging motion scenarios and models that are efficient enough for deployment.

BiM-VFI [18] addresses non-linear motion and occlusion through an explicit bidirectional motion descriptor that encodes angular and magnitude differences relative to the intermediate time, achieving strong performance under complex motion conditions. However, this design increases computational complexity due to bidirectional processing and dense motion estimation.

This reveals a broader research gap: how to preserve the representational benefits of bidirectional motion modeling while reducing redundancy in feature computation and upsampling operations. In particular, existing approaches often rely on heavy decoding or iterative refinement stages, which are not suitable for latency-sensitive applications.

Therefore, the central motivation of this work is to design a more efficient yet motion-aware framework that reduces computational redundancy while maintaining strong performance in challenging scenarios involving occlusion and non-linear motion. Achieving this balance is essential for enabling practical deployment of VFI systems beyond controlled benchmarks into real-world applications such as streaming, conferencing, and interactive media.

1.3 Objective

This thesis investigates the limitations of current state-of-the-art VFI methods. In particular, it addresses their struggles with challenging motion patterns, such as non-linear trajectories and large displacements (e.g., fast-moving body parts), alongside their high computational costs. Based on these insights, this work develops a compact VFI framework designed for faster inference and higher computational efficiency. Crucially, performance is evaluated by prioritizing satisfactory visual fidelity alongside standard quantitative metrics.

1.4 Contributions

In summary, we make the following contributions:

- We propose Motion Detection Network (MD-Net) with Squeeze-and-Excitation (SE) attention that enhances motion feature representation by adaptively emphasizing informative channels, improving robustness under large motion and occlusion compared to BiM-VFI.

- We introduce Guided Flow Upsampling (GFU), a lightweight upsampling strategy that replaces conventional decoding stages with a guidance-based convex combination mechanism, significantly reducing inference cost while preserving structural fidelity.
- We incorporate perceptual supervision using a pre-trained Visual Geometry Group (VGG) network to encourage the model to produce sharper and more visually realistic interpolated frames, improving alignment with human perceptual quality beyond pixel-wise losses.
- We further analyze the trade-off between accuracy and efficiency, providing insights into how architectural simplification affects motion representation and reconstruction quality in VFI systems.
- Integrating these components, we present our proposed framework, a unified model that achieves improved perceptual quality and computational efficiency over the BiM-VFI baseline while maintaining robustness under challenging motion conditions.

In this work, we extend the BiM-VFI framework by introducing a Motion Detection Network (MD-Net) for enhanced motion modeling and a Guided Flow Upsampling (GFU) module for computationally efficient optical flow resolution expansion. We evaluate our method using a comprehensive set of pixel-based, perceptual, qualitative, and computational complexity metrics, including runtime and parameter efficiency, to demonstrate its suitability for real-world deployment.

1.5 Structure of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 (Background) presents the fundamental concepts required to understand the remainder of the thesis, including optical flow, frame warping, and the core principles underlying flow-based video frame interpolation.

Chapter 3 (Literature Review) examines existing state-of-the-art approaches to video frame interpolation. This chapter analyzes the design principles that inform the proposed method, as well as the limitations and gaps in current approaches that motivate the contributions of this thesis.

Chapter 4 (Methodology) presents the proposed method, describing the end-to-end design of our model. It details each core component, including MD-Net and GFU, and

explains how they extend the BiM-VFI framework to achieve superior motion estimation and computational efficiency.

Chapter 5 (Experiment) presents the experimental results, including training and inference outcomes, and compares the proposed models against state-of-the-art baselines across multiple evaluation perspectives.

Chapter 6 (Conclusions) summarizes the thesis and discusses potential directions for future research.

Chapter 2

Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep learning models designed for grid-like data such as images. They have become a fundamental backbone in computer vision due to their ability to learn hierarchical feature representations directly from raw pixels, without manual feature engineering [19, 20]. Their core operation, convolution, applies learnable filters across local receptive fields. This enables parameter sharing and sparse connectivity while preserving spatial structure and reducing the number of parameters compared to fully connected networks [21]. A typical CNN consists of convolutional layers, non-linear activations (e.g., ReLU), and downsampling operations such as pooling. This structure allows the network to progressively transform low-level patterns, such as edges and textures, into abstract semantic representations in deeper layers [22]. These properties have made CNNs a standard backbone for various vision tasks, including classification [20, 23], detection, and restoration. In this work, CNNs serve as the fundamental feature extraction framework for learning representations from image frames for the VFI setting discussed in Section 4.3.

Fig. 2.1 illustrates a simplified example of a CNN feature extraction pipeline using a ResNet-style backbone. The example helps illustrate the concept of "hierarchical feature representation" used later in this thesis. At the earliest stage (`conv1`), the network produces high-resolution feature maps ($64 \times 112 \times 112$). These features mainly capture low-level structures such as edges, color contrast, and local gradients. As the image passes through the residual stages (`layer1` to `layer4`), the spatial resolution gradually decreases while the number of channels increases. The representation changes from local patterns ($64 \times 56 \times 56$)

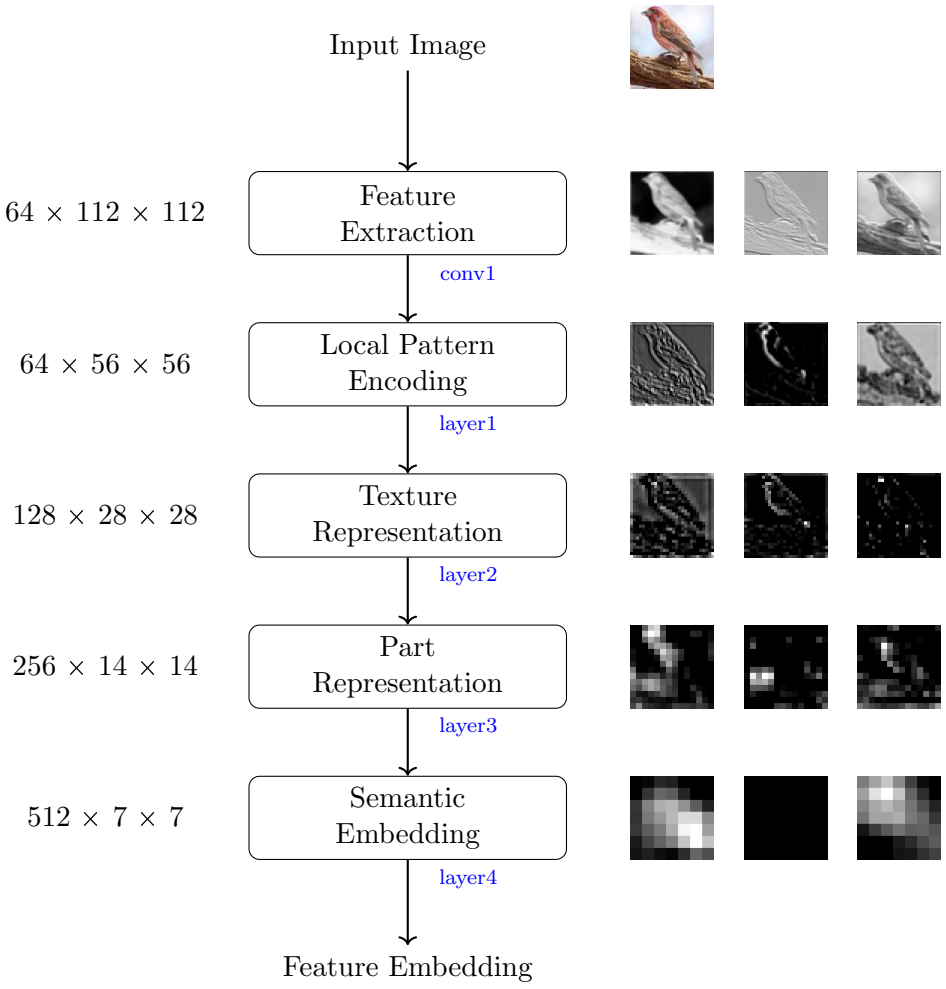


Figure 2.1: CNN feature extraction pipeline with structured feature hierarchy, aligned numerical annotations, and multi-view feature visualization.

to texture-level features ($128 \times 28 \times 28$), object-part features ($256 \times 14 \times 14$), and finally high-level semantic features ($512 \times 7 \times 7$). The channel visualizations show this transition clearly. Shallow layers mainly highlight fine boundaries and repeated textures, while deeper layers respond more strongly to larger object regions. This progression illustrates a key trade-off in CNN feature extraction. Shallow layers preserve spatial details but contain less semantic information. Deep layers provide richer semantic information but have lower spatial resolution. Note that Fig. 2.1 is only illustrative. Different CNN architectures may have different structures, but they follow the same principle of learning increasingly high-level feature representations.

This trade-off is also important for video frame interpolation. VFI uses features at different levels to handle large motion and occlusion. Coarse features provide useful semantic information for matching regions that are far apart but belong to the same object. Fine features preserve spatial details, which helps recover accurate object boundaries and reduce blurring during warping and synthesis. The use of hierarchical features in the proposed VFI framework is described in Section 4.3.

2.2 Downsampling and Upsampling

2.2.1 Downsampling: constructing coarse levels

Downsampling is a process that reduces the spatial resolution of an image by decreasing the number of pixels while preserving the important visual information. Given an input image I with spatial resolution of $H \times W$, the downsampled image I' can be represented as

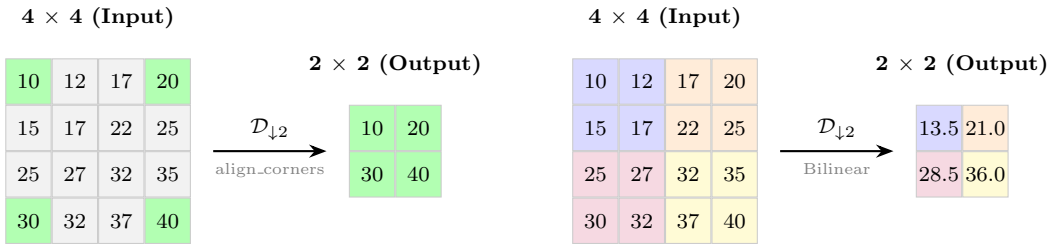
$$I' = \text{Downsample}(I, s), \tag{2.2.1}$$

where s denotes the scaling factor. For example, when $s = 1/2$, the spatial resolution of the image is reduced by half in both height and width:

$$I \in \mathbb{R}^{H \times W \times C} \rightarrow I' \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times C}, \tag{2.2.2}$$

where C represents the number of image channels.

In practice, downsampling is commonly performed by first applying a low-pass filter to remove high-frequency components and reduce aliasing artifacts, followed by a subsampling operation that selects pixels at a lower sampling rate. The process can be expressed as



(a) $\mathcal{D}_{\downarrow 2}$ (Align Corners) **(b) $\mathcal{D}_{\downarrow 2}$ (Bilinear Average)**

Figure 2.2: Illustration of downsampling from a 4×4 input to a 2×2 output. (a) Corner-aligned downsampling, where the output values correspond to the input corners. (b) Bilinear downsampling, where each output value is computed from neighboring input pixels.

$$I'(x, y) = \tilde{I}(sx, sy), \tag{2.2.3}$$

where $\tilde{I}$ denotes the filtered image and (sx, sy) represents the sampling locations after resolution reduction.

Downsampling is widely used in multi-scale methods to generate lower-resolution representations of an image. These coarse representations provide larger effective receptive fields and allow algorithms to capture large-scale structures or motions more efficiently.

For example, when $s = 1/2$, the spatial resolution of the image is reduced by half in both height and width, as illustrated in Fig. 2.2:

2.2.2 Upsampling: propagating features to finer levels

Upsampling is the inverse process of downsampling, which increases the spatial resolution of an image or feature map. In multi-scale architectures, such as coarse-to-fine pyramids, upsampling is critical for propagating low-resolution predictions (e.g., coarse optical flows or occlusion maps) back to finer resolutions to guide detail reconstruction. Given a low-resolution input $I' \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times C}$, the upsampled image $I'' \in \mathbb{R}^{H \times W \times C}$ with an upscaling factor of $s = 2$ is denoted as:

$$I'' = \text{Upsample}(I', s). \tag{2.2.4}$$

Among various upsampling methods, bilinear interpolation is widely adopted due to its simplicity and smooth gradient propagation. The intensity of an upsampled pixel at

continuous coordinates (x, y) is interpolated from its four nearest neighbors on the low-resolution grid:

$$I''(x, y) = \sum_{i=1}^{2} \sum_{j=1}^{2} w_{ij} I'(x_i, y_j), \tag{2.2.5}$$

where (x_i, y_j) are the integer coordinates of the neighboring pixels in I', and w_{ij} represents the bilinear weights calculated based on the relative distances:

$$w_{ij} = (1 - |x - x_i|)(1 - |y - y_j|). \tag{2.2.6}$$

The mapping from the target high-resolution coordinate $(x_{\text{high}}, y_{\text{high}})$ to the source low-resolution coordinate $(x_{\text{low}}, y_{\text{low}})$ heavily depends on the coordinate alignment configuration, which alters the interpolation behavior near the boundaries. To illustrate this, consider a non-uniform 2×2 input matrix $I \in \mathbb{R}^{2\times 2}$:

$$I = \begin{bmatrix} 10 & 30 \\ 50 & 70 \end{bmatrix}. \tag{2.2.7}$$

Here, W_{in} and W_{out} denote the width of the input and output grids, respectively, while x_{high} and x_{low} denote a discrete coordinate in the output grid and its corresponding source coordinate in the input grid. Two conventions exist for defining this coordinate mapping, differing in how the grids are geometrically aligned:

- **Align Corners (`align_corners=True`):** The corner pixels of the input and output grids are aligned. The coordinate transformation is formulated as:

$$x_{\text{low}} = x_{\text{high}} \times \frac{W_{\text{in}} - 1}{W_{\text{out}} - 1} = \frac{x_{\text{high}}}{3}. \tag{2.2.8}$$

  This alignment maps $x_{\text{high}} \in \{0, 1, 2, 3\}$ to $x_{\text{low}} \in \{0, \frac{1}{3}, \frac{2}{3}, 1\}$, preserving the exact original corner values at the outer boundaries (as illustrated in Fig. 2.3(a)). The resulting output matrix is:

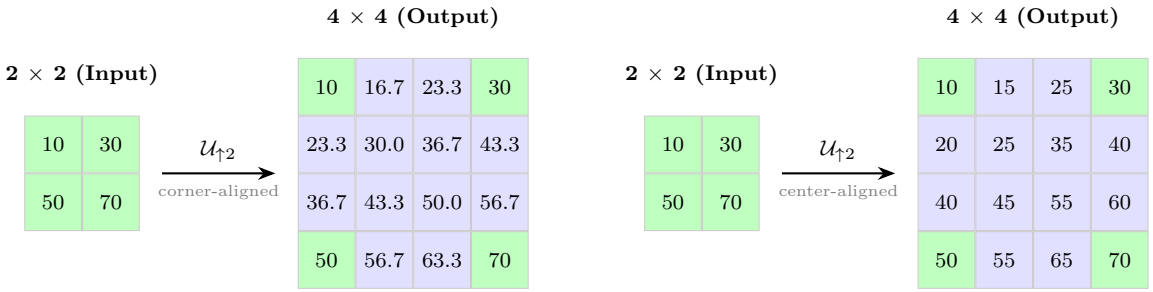
$$I''_{\text{true}} = \begin{bmatrix} \mathbf{10.0} & 16.7 & 23.3 & \mathbf{30.0} \\ 23.3 & 30.0 & 36.7 & 43.3 \\ 36.7 & 43.3 & 50.0 & 56.7 \\ \mathbf{50.0} & 56.7 & 63.3 & \mathbf{70.0} \end{bmatrix}. \tag{2.2.9}$$

- **Align Centers (`align_corners=False`):** The pixels are treated as patches with centers, aligning the corner points of the outer boundaries. The transformation is defined as:

$$x_{\text{low}} = (x_{\text{high}} + 0.5) \times \frac{W_{\text{in}}}{W_{\text{out}}} - 0.5 = 0.5x_{\text{high}} - 0.25. \tag{2.2.10}$$

This maps $x_{\text{high}} \in \{0, 1, 2, 3\}$ to continuous source coordinates $x_{\text{low}} \in \{-0.25, 0.25, 0.75, 1.25\}$, where values out of bounds are clipped to the edges. This prevents boundary shift and maintains scale-invariant geometric relationships (as illustrated in Fig. 2.3(b)). The resulting output matrix is:

$$I''_{\text{false}} = \begin{bmatrix} \mathbf{10.0} & 15.0 & 25.0 & \mathbf{30.0} \\ 20.0 & 25.0 & 35.0 & 40.0 \\ 40.0 & 45.0 & 55.0 & 60.0 \\ \mathbf{50.0} & 55.0 & 65.0 & \mathbf{70.0} \end{bmatrix}. \tag{2.2.11}$$



(a) Align Corners (True) **(b) Align Centers (False)**

Figure 2.3: Bilinear upsampling with a scale factor of 2 on a non-uniform matrix. (a) Under corner-aligned spatial grid mapping, interpolation step sizes are divided into non-uniform fractional intervals $(1/3, 2/3)$. (b) Under center-aligned spatial grid mapping, pixel values are symmetrically distributed with respect to grid centers using uniform step sizes (0.5), yielding smoother spatial transitions.

2.3 Deep Feature Representation

Deep feature representation refers to learning-based embeddings that transform raw pixel inputs into high-dimensional feature spaces where semantic and structural similarities are more explicitly represented. Unlike handcrafted descriptors, these representations are

learned directly from data through deep neural networks, enabling hierarchical abstraction of visual information [22, 24].

While convolutional neural networks (CNNs) provide the mechanism for learning such representations (as described in Section 2.1), the resulting feature space itself plays an important role in many vision tasks. Instead of operating on pixel intensities, images are mapped into embedding spaces where similarity reflects perceptual and structural closeness.

In video frame interpolation (VFI), this property is particularly important because intermediate frames are not directly observed and must be inferred from neighboring frames. Pixel-wise comparison is often sensitive to motion, occlusion, and spatial misalignment, which limits its effectiveness in modeling temporal relationships. Feature representations provide a more stable alternative by reducing sensitivity to small spatial transformations, making them more suitable for motion-aware modeling in video sequences [25, 26].

2.3.1 Residual Networks (ResNet) for Feature Extraction

Residual Networks introduce skip connections that reformulate feature learning as residual mapping, enabling more stable optimization in very deep architectures [23]. This design improves gradient propagation and allows networks to scale to greater depth without degradation in performance.

Beyond optimization benefits, ResNet also enhances the quality of learned feature representations. By preserving identity information through skip connections, ResNet-based encoders maintain both local details and global structure.

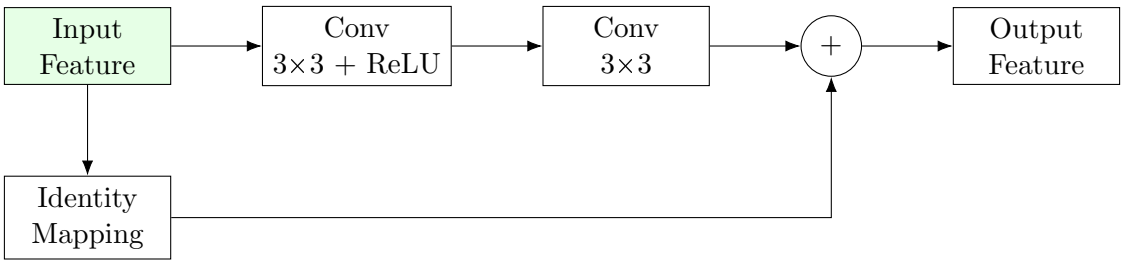


Figure 2.4: Residual block architecture with identity skip connection. Feature transformation is learned via convolutional layers while the original input is directly added to the output, enabling residual learning and improved gradient flow.

Fig. 2.4 illustrates the fundamental structure of a residual block used in ResNet-based architectures. The key idea is the introduction of a skip (identity) connection that by-

passes the convolutional transformation path and is combined with the transformed features through element-wise addition. This formulation enables the network to learn residual mappings instead of directly learning the desired underlying function.

In practice, this design improves gradient flow during backpropagation and alleviates the degradation problem observed in very deep networks. As a result, ResNet can be scaled to greater depths while maintaining stable optimization and strong feature learning capability.

2.3.2 Feature Space Robustness for VFI

In VFI, the main challenge arises from motion-induced misalignment between adjacent frames. Direct pixel-wise comparison often fails under occlusion, large displacement, or non-linear motion. Feature representations mitigate this issue by embedding frames into a space where structural correspondence is more linear and robust.

As a result, interpolation models often rely on feature-level supervision or feature-based matching to guide motion estimation and frame synthesis, rather than relying solely on pixel reconstruction error.

2.4 VGG Network

VGG is a deep convolutional neural network originally proposed for large-scale image classification [27]. It was trained on ImageNet [20], a dataset containing millions of labeled natural images across a thousand categories. The network follows a simple, uniform architecture by repeatedly stacking 3×3 convolutional filters. These convolutional layers are interleaved with max-pooling operations to progressively reduce spatial resolution [27]. This design allows VGG to build deep, hierarchical feature representations. Earlier layers capture low-level features such as edges and textures. In contrast, deeper layers extract higher-level structures like object parts and shapes.

This hierarchical property makes VGG particularly useful for perceptual loss. Instead of comparing images pixel by pixel, perceptual loss compares their feature representations extracted from a pretrained VGG network [28]. Two images that are perceptually similar to humans tend to produce similar VGG features, even when their pixel values differ. In this work, a VGG16 network pretrained on ImageNet is used to compute perceptual loss, with features extracted from multiple intermediate layers to capture both low-level texture

and high-level structural information. This encourages the model to produce interpolated frames that are not only pixel-accurate, but also visually and perceptually consistent with the ground truth.

2.4.1 VGG Features as Perceptual Embedding Space

VGG is trained purely for image classification on ImageNet [20], a task that forces its intermediate layers to learn representations general enough to discriminate between a very wide range of object categories, textures, and scene types, rather than representations specialized to any single downstream task. It is precisely this generality that makes VGG features transferable to tasks the network was never trained for, including perceptual similarity assessment. Because the network's hierarchy of edges, textures, and object-part structure loosely mirrors the coarse-to-fine organization believed to exist in the human visual system, a distance computed between VGG activations of two images tends to track how different a human observer would judge the two images to be, even without the network ever having seen a human similarity judgment during training.

This idea was first used in neural style transfer [29]. It combines the content of one image with the artistic style of another by matching VGG feature statistics between the two.

Johnson et al. [28] later extended this idea into a general training signal by measuring the difference between VGG feature representations and using it as a loss. Networks trained with this loss produced sharper and more realistic images than those trained with pixel-wise loss alone.

Using a fixed, pretrained VGG network to define a loss function is commonly referred to as a *perceptual loss*. It is closely related to, but distinct from, its use as an evaluation metric such as Learned Perceptual Image Patch Similarity (LPIPS) (Section 2.5.1). LPIPS additionally learns calibration weights on top of a VGG-like backbone, so that feature distances better match human similarity judgments [30].

In this work, VGG serves the former role: its frozen, pretrained activations define the feature space in which our perceptual loss is computed during training (Section 4.8).

2.4.2 VGG16

Among the VGG variants, VGG16 is the most widely adopted backbone for perceptual loss and perceptual similarity applications [28, 29, 30], offering a favorable balance between representational depth and computational cost, and we adopt it for this reason.

VGG16 consists of 13 convolutional layers and 3 fully connected layers, forming a 16-layer deep architecture. The convolutional layers are organized into five sequential blocks, where each block applies a series of 3×3 convolutions followed by ReLU activations and a max-pooling operation that progressively reduces spatial resolution while increasing the number of feature channels.

The fully connected layers originally designed for ImageNet classification are discarded in this work. Instead, only the convolutional backbone is retained and initialized with ImageNet-pretrained weights, which are kept fixed during training to preserve the learned feature hierarchy.

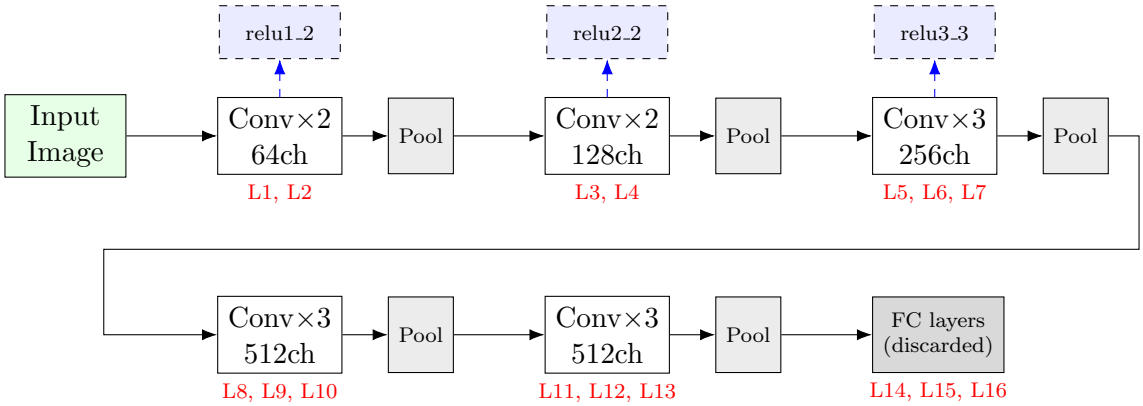


Figure 2.5: VGG16 feature extraction backbone used for perceptual learning. The network consists of stacked convolutional blocks with progressively increasing channel dimensions and decreasing spatial resolution. Intermediate activations at selected layers are used as feature representations.

Fig. 2.5 summarizes the VGG16 backbone used in this work. Intermediate feature representations from selected layers are utilized in the following to define a feature-based learning objective, as described in Section 2.5.

2.5 Perceptual Metrics

Pixel-wise metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [31] are the most widely reported evaluation criteria in video frame interpolation, yet both were designed for general-purpose image fidelity assessment and carry assumptions that are poorly suited to the interpolation setting specifically. PSNR

is a monotonic function of the Mean Squared Error (MSE) between the predicted and ground-truth frame, and therefore implicitly assumes that a good prediction is one that minimizes per-pixel intensity error. This assumption breaks down precisely in the regions where interpolation is hardest: near occlusion boundaries, in areas of fast or non-linear motion, and wherever the correct intermediate appearance is genuinely ambiguous given only the two input frames. In such regions, a network trained to minimize a pixel-wise loss such as ℓ_1, ℓ_2, or Charbonnier loss has a well-documented tendency to hedge between plausible outcomes by producing a blurred average, since averaging several plausible sharp solutions minimizes expected pixel-wise error even though the averaged result does not resemble any single plausible sharp outcome [32]. Because PSNR is computed on exactly the same pixel-wise basis as this training objective, it systematically rewards this blurring behavior rather than penalizing it: as illustrated in Fig. 2.6, a blurry frame can achieve a higher PSNR than a sharp but slightly misaligned one, even though a human observer would judge the sharp result as more visually convincing.

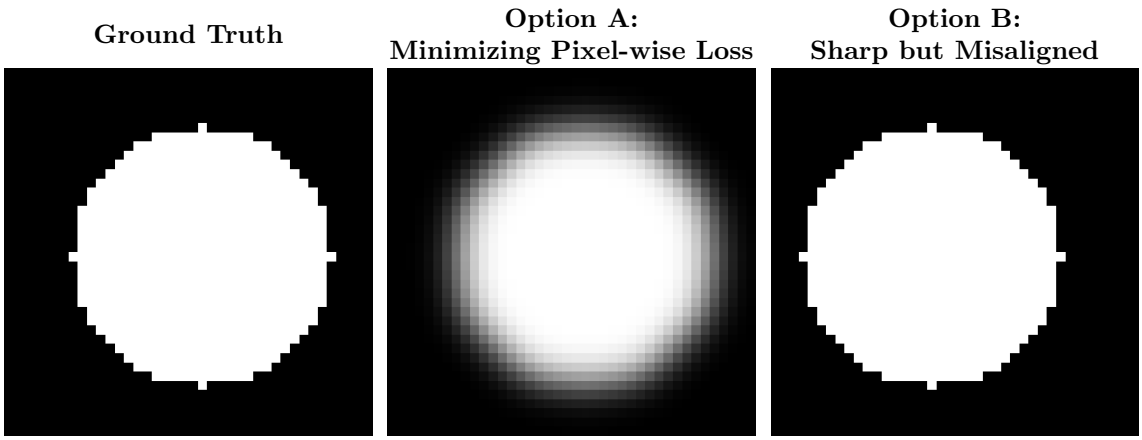


Figure 2.6: An illustration of the pixel-wise metric dilemma: comparing Option A (pixel-wise loss) and Option B (sharp misaligned) against the Ground Truth.

SSIM improves on PSNR by comparing local luminance, contrast, and structural statistics within a sliding window rather than raw pixel intensities, making it somewhat more robust to uniform brightness or contrast shifts. However, its structural comparison is still computed over small, fixed-size local windows and therefore remains fundamentally a low-level, local statistic. It does not capture higher-level semantic information such as whether

a texture is physically plausible, whether an edge corresponds to a true object boundary or an interpolation artifact, or whether fine details have been faithfully reconstructed or hallucinated. Both PSNR and SSIM are also strictly full-reference metrics, requiring a pixel-aligned ground-truth frame, which limits their applicability to benchmark datasets and prevents their use on real-world video where ground truth is unavailable.

Table 2.1: Evaluation metrics for VFI.

Metric	Interpolation Quality	Reference Image Required?
PSNR	Higher = Better ↑	Yes✓
SSIM [31]	Higher = Better ↑	Yes✓
LPIPS [30]	Lower = Better ↓	Yes✓
STLPIPS [33]	Lower = Better ↓	Yes✓
NIQE [34]	Lower = Better ↓	No ×

Beyond PSNR and SSIM, recent VFI research increasingly adopts perceptual image quality metrics to better reflect visual quality and human perception. These metrics evaluate reconstructed frames from different perspectives, including deep feature similarity, robustness to spatial misalignment, and natural scene statistics. Table 2.1 summarizes the perceptual evaluation metrics discussed in the following subsections, together with their optimization objectives and whether a reference image is required.

2.5.1 LPIPS (Learned Perceptual Image Patch Similarity)

LPIPS [30] is a learned perceptual metric designed to measure similarity between two images based on deep feature representations rather than pixel-wise differences. Unlike traditional metrics such as PSNR or SSIM, LPIPS aims to align more closely with human perception by operating in a learned feature space extracted from pretrained convolutional neural networks such as AlexNet [20], VGG [35], or SqueezeNet [36].

Given a reference image x and a reconstructed image x', both images are first passed through a pretrained network to extract multi-layer feature maps. Let $F_l(x) \in \mathbb{R}^{H_l \times W_l \times C_l}$ denote the **raw feature map** at layer l, where H_l, W_l, and C_l are the spatial height, width, and number of channels, respectively. The features are then unit-normalized along the channel dimension via the **Channel-wise L2 Norm** block to obtain the **normalized features** $\hat{F}_l(x)$.

For each spatial location (h, w), we denote the feature vector as $\hat{F}_l^{h,w}(x) \in \mathbb{R}^{C_l}$. LPIPS then computes a weighted Euclidean distance between corresponding features of the two images across all spatial locations and layers.

Formally, LPIPS is defined as:

$$\text{LPIPS}(x, x') = \sum_l \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \left\| w_l \odot \left(\hat{F}_l^{h,w}(x) - \hat{F}_l^{h,w}(x') \right) \right\|_2^2, \tag{2.5.1}$$

where l indexes the selected convolutional layers, $w_l \in \mathbb{R}^{C_l}$ are learned channel-wise weights, and $\odot$ denotes element-wise multiplication. These weights are learned using a linear calibration model trained on the Berkeley–Adobe Perceptual Patch Similarity (BAPPS) dataset, which consists of human perceptual similarity judgments.

The comprehensive internal computation of the LPIPS metric is systematically structured into a two-stream processing architecture, as illustrated in Fig. 2.7. The procedure begins with two parallel streams operating independently. The ground truth image x and the reconstructed image x' are fed into separate branches to execute identical preprocessing and feature extraction steps. Specifically, the pretrained backbone networks independently extract intermediate **raw feature maps** across L layers. These extracted features are subsequently passed through the **Channel-wise L2 Norm** block to enforce unit length on the per-pixel feature vector, yielding the **normalized features** $\hat{F}_l(x)$ and $\hat{F}_l(x') \in \mathbb{R}^{C_l \times H_l \times W_l}$ for the respective branches.

The outputs from these parallel streams then merge at the **"Feature Difference"** node, where the mathematical formulation is mapped directly onto a centralized, sequential pipeline. At this junction, the spatial feature difference $(\hat{F}_l^{h,w}(x) - \hat{F}_l^{h,w}(x'))$ is computed. This raw difference vector is then modulated via element-wise multiplication $(\odot)$ by the **Learned Channel-wise Scaling** weights $w_l \in \mathbb{R}^{C_l}$. Next, the **Per-pixel L_2 Distance Squared** $(\|\cdot\|_2^2)$ evaluates the per-pixel distance across the tensor dimensions. To aggregate these spatial differences, the resulting values are processed through the **Average Across Spatial Dimensions** stage $(H_l \times W_l)$ to derive a scalar distance d_l for each individual layer. Finally, the per-layer distances are combined via a **Sum over Selected Layers** $(\sum_l d_l)$ to produce the final LPIPS score. Crucially, as a distance-based metric, a lower final LPIPS score indicates a smaller discrepancy in the deep feature space, directly signifying higher perceptual similarity between the two images.

Intuitively, LPIPS measures how different two images are in a deep feature space that captures hierarchical visual information such as edges, textures, and object-level semantics. Because these features are learned from large-scale vision models and calibrated using

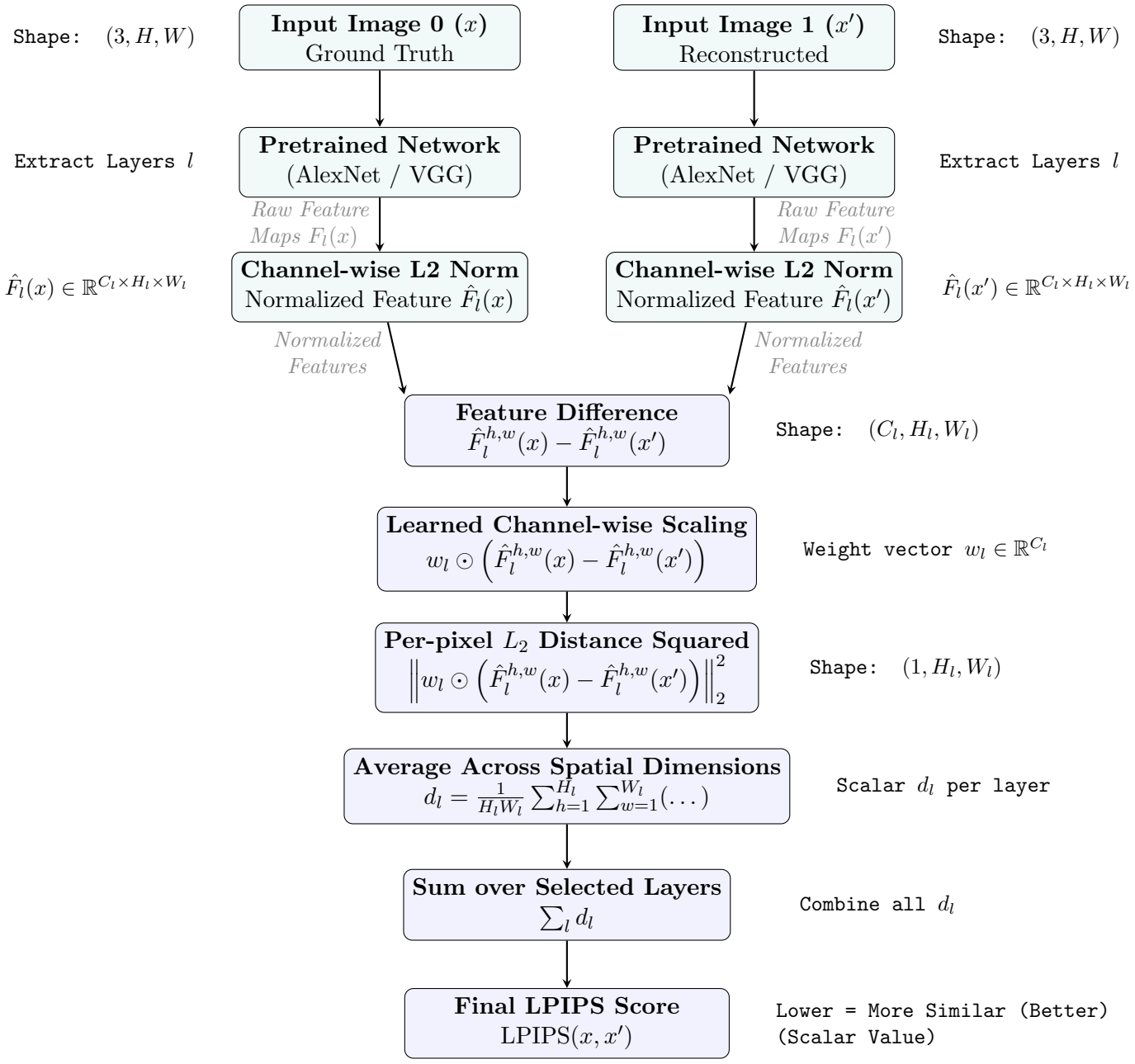


Figure 2.7: Two-stream computation flow of LPIPS with clean, independent side labels.

human judgments, LPIPS correlates significantly better with perceptual similarity than pixel-based metrics.

As a result, LPIPS has become a standard benchmark for assessing perceptual quality across recent image and video synthesis tasks. In VFI specifically, it is regularly adopted by state-of-the-art architectures—including RIFE [37], EMA-VFI [38], BiM-VFI [18], and Shu et al. [39]—to demonstrate superior visual fidelity. The same metric is equally standard in single-image super-resolution, where perceptual quality is often prioritized over pixel-wise fidelity; ESRGAN [40] and Real-ESRGAN [41] are among the most widely cited examples that adopt LPIPS as a primary evaluation criterion alongside PSNR and SSIM.

2.5.2 STLPIPS (Shift-Tolerant Perceptual Similarity Metric)

Shift-Tolerant Perceptual Similarity Metric (STLPIPS) [33] extends LPIPS by making the feature representation tolerant to small spatial shifts. Standard LPIPS can be sensitive to even one- or two-pixel misalignments that are barely noticeable to humans. This is particularly relevant to VFI because interpolated frames are generated from estimated motion and may not be perfectly aligned with the ground truth, especially near occlusion boundaries and under fast motion. STLPIPS addresses this limitation using anti-aliased downsampling and other architectural modifications to improve shift tolerance, followed by retraining on the BAPPS dataset [30, 42]. It therefore provides a more robust perceptual measure for interpolated frames with small motion-induced misalignments.

STLPIPS complements rather than replaces LPIPS. Reporting both metrics can help distinguish perceptual errors caused by spatial misalignment from genuine content differences. STLPIPS is also different from FloLPIPS, which incorporates optical-flow information to measure temporal distortions across consecutive frames [43]. STLPIPS instead focuses on spatial shift tolerance within an individual image pair. The two metrics therefore address different limitations of standard LPIPS and can be used together for VFI evaluation.

STLPIPS remains a relatively new metric in VFI evaluation, but its adoption in BiM-VFI [18] as one of three core perceptual metrics for state-of-the-art comparison [18], together with its inclusion in the AceVFI survey's summary of standard perceptual metrics for the field [16], motivates its inclusion in this thesis's evaluation protocol. Reporting STLPIPS alongside LPIPS follows the same practice as this thesis's primary baseline, allowing for a direct and consistent comparison.

2.5.3 NIQE (Natural Image Quality Evaluator)

Unlike LPIPS and STLPIPS, Natural Image Quality Evaluator (NIQE) [34] is a *no-reference* (blind) metric that does not require a ground-truth image. It is also *opinion-* and *distortion-unaware*, as it is never trained on human quality scores or distorted images. Instead, NIQE learns only the statistical properties of undistorted natural images and measures how far a test image deviates from this pristine natural image model. This makes it particularly suitable for video frame interpolation, where reference intermediate frames are unavailable in real-world deployment.



Figure 2.8: Illustration of natural scene statistics (NSS), where locally normalized pixel intensities in natural images follow stable statistical distributions.

As shown in Fig. 2.8, NIQE evaluates image quality by testing adherence to Natural Scene Statistic (NSS). The test image is divided into $P \times P$ patches, and mean-subtracted contrast-normalized (MSCN) coefficients are computed in each patch following Ruderman and Bialek [44]. These coefficients are modeled using a generalized Gaussian distribution (GGD), while pairwise products are modeled using an asymmetric GGD, producing a 36-dimensional feature vector per patch [34].

Importantly, this feature extraction is applied uniformly to all patches, including smooth regions. The sharpness-based patch selection in NIQE is used only during training of the pristine model, not at test time, to avoid removing blur or structural artifacts that are crucial for quality assessment.

The patch-level features are then aggregated into a multivariate Gaussian (MVG) representation, and the final NIQE score is computed as the distance between this MVG and the learned pristine NSS distribution.

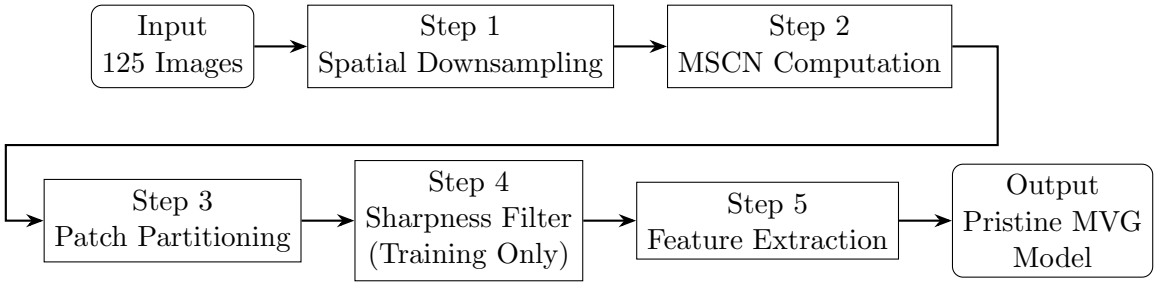


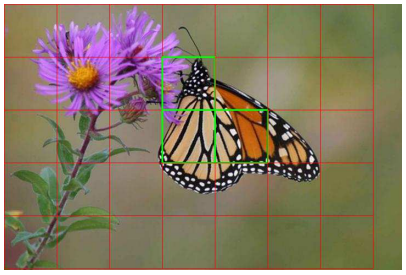
Figure 2.9: NIQE Training Pipeline for Pristine Natural Scene Statistics Model

The pristine model is constructed from a relatively small set of 125 high-quality natural images [34], following the overall training pipeline illustrated in Fig. 2.9. This dataset size is sufficient because NIQE does not model semantic content, but instead captures universal low-order statistics of natural images, which are highly consistent across different scenes due to physical image formation processes [44]. As a result, even a limited dataset can reliably estimate the mean and covariance of this statistical space.

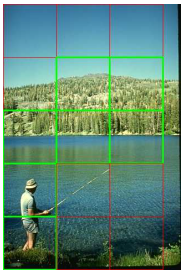
This design makes NIQE robust and widely applicable for video frame interpolation. Since it is not trained on specific distortion types, it is difficult to artificially optimize for NIQE without genuinely improving perceptual quality. Therefore, NIQE is commonly used alongside full-reference metrics in VFI literature, including the AceVFI survey [16], velocity disambiguation methods [45], and BiM-VFI [18], evaluated on datasets such as Vimeo90K and SNU-FILM.

To mathematically substantiate the selection mechanism within the Natural Scene Statistics (NSS) framework, the patch partitioning and sharpness-based filtering process is visualized across three distinct image specimens in Figure 2.10. The image domain is systematically segmented into non-overlapping blocks of size $P \times P$ pixels, where $P = 96$. For each standalone patch, a localized sharpness index $\delta(b)$ is calculated by taking the mean of its corresponding local variance field σ, mapped directly from the spatial domain calculations. A strict quality-control threshold T is evaluated dynamically for each image as a fraction $p = 0.75$ of its maximum recorded block sharpness ($\max(\delta(b))$).

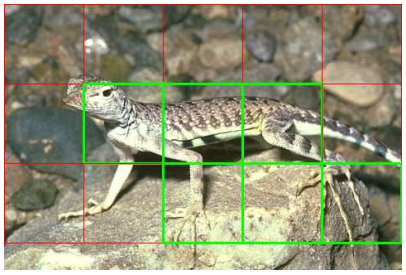
As quantitatively validated by the empirical sharpness matrix mapping, patches satisfying the condition $\delta(b) > T$ are highlighted with thick green borders. These regions heavily coincide with high-contrast structures, edge boundaries, and detailed textures—elements



(a) Example 1



(b) Example 2



(c) Example 3

Figure 2.10: Visual exemplification of the patch partitioning and sharpness-based filtering process under the Natural Scene Statistics (NSS) framework across three distinct image samples. The matrix indicates blocks of size $P = 96$ pixels, filtered by a dynamically evaluated threshold $T = 0.75 \times$ peak sharpness. Thick green borders denote selected high-sharpness patches reserved for model fitting, whereas thin red borders signify discarded low-texture areas.

that represent rich structural statistics essential for building the pristine multivariate Gaussian (MVG) distribution model. Conversely, smooth blocks containing minor spatial variance (such as uniform sky or flat surfaces) fail to pass the threshold and are marked with thin red borders, meaning they are excluded from the statistical model training pool. Boundary fragments extending beyond the integer grid are cleanly omitted via floor division. This distribution of green and red boxes explicitly demonstrates how the NIQE pipeline separates informative structural natural scenes from trivial, low-frequency content.

2.6 Cost Volume for Dense Motion Estimation

In computer vision, establishing dense pixel-level correspondence between temporally adjacent frames is a fundamental problem for motion estimation and optical flow. Traditional approaches rely on global patch matching, which is computationally prohibitive. To address this, modern deep-learning architectures, pioneered by networks like PWC-Net [46], compute a localized correlation representation known as a *Cost Volume*.

Given two normalized latent feature maps $\bar{F}_0, \bar{F}_1 \in \mathbb{R}^{C \times H \times W}$, where C is the number of feature channels, each feature vector has unit length. The cost volume aggregates matching similarities between them within a constrained search window of radius d. For a pixel $\mathbf{x} = (x, y)$ in the reference frame, this similarity is measured against a displaced coordinate $\mathbf{x} + \mathbf{d}$ in the target frame, where the displacement vector $\mathbf{d} = (u, v)$ is bounded

by $[-d, d]^2$. The resulting inner product is defined in Eq. (2.6.1):

$$\mathbf{C}(\mathbf{x}, \mathbf{d}) = \langle \bar{F}_0(\mathbf{x}),\ \bar{F}_1(\mathbf{x} + \mathbf{d}) \rangle = \sum_{c=1}^{C} \bar{F}_0(\mathbf{x}, c) \cdot \bar{F}_1(\mathbf{x} + \mathbf{d}, c) \tag{2.6.1}$$

The resulting tensor has spatial resolution $H \times W$ and $(2d+1)^2$ channels, with each channel representing a possible displacement. The cost volume therefore provides information about how well each possible displacement matches the input features. This information helps the network estimate sub-pixel motion, especially near complex motion boundaries and under large displacements.

For example, when $d = 1$, the neighborhood contains $3 \times 3 = 9$ possible displacements, where $\mathbf{d} \in -1, 0, 1^2$. The resulting cost volume therefore has 9 channels, with each channel storing the match score for one displacement. A larger search radius provides more possible displacements but also increases the computational cost. This formulation forms the basis of the correspondence estimation in the proposed MD-Net. Its application is described in Section 4.5.

Chapter 3

Literature Review

3.1 Approaches to Video Frame Interpolation

Video Frame Interpolation (VFI) methods can be broadly organized by how they model motion and synthesize the intermediate frame: kernel-based, flow-based, hybrid kernel-and-flow, Transformer-based, and diffusion-based approaches [16]. Table 3.1 summarizes representative methods in each category. The remainder of this section discusses their motion-modeling assumptions and trade-offs, which motivate this thesis's choice of paradigm.

Kernel-based methods predict spatially-adaptive convolutional kernels that are applied directly to input patches. They implicitly encode motion in the kernel weights without explicit correspondence estimation [4, 11]. These methods are typically designed for center-time interpolation at $t = 0.5$ and are constrained by fixed receptive fields. Their effective motion range is therefore bounded by kernel size, making them less effective under large motion [47].

Flow-based methods, in contrast, explicitly estimate dense optical flow to warp and blend input frames. Because the flow can be scaled according to the target time t, flow-based methods naturally support arbitrary-time interpolation. They are also the most widely adopted paradigm in the field [18, 37, 48, 49, 50].

Hybrid methods combine both mechanisms. They use flow for coarse alignment and learned kernels to correct residual misalignment and occlusion. This improves robustness to flow errors but reduces some of the interpretability of flow-based methods [51].

More recent directions include Transformer-based methods, which model motion through attention rather than explicit correspondence. They can capture long-range dependencies

but incur quadratic attention complexity [52]. Diffusion-based methods improve perceptual realism under ambiguous motion through generative denoising [53], but their high training complexity [54] and iterative inference sampling [55] prevent real-time deployment [16].

Table 3.1: Taxonomy of Video Frame Interpolation Methods by Motion Representation

Category	Motion Representation	Inference Cost	Output Quality
Kernel-based	Predicts spatially-adaptive convolutional kernels applied directly to input patches; motion is implicit in the kernel weights, with no explicit correspondence or flow field estimated	Low	Blurs under large motion
Flow-based	Explicitly estimates a dense optical flow field between input frames, then warps and blends pixels along that field toward the target time	Low–Medium	Sharp, but flow-error sensitive
Transformer-based	Uses attention to compute soft correspondences between patch tokens across frames, aggregating motion-relevant context without an explicit displacement field	High	Strong on long-range motion
Diffusion-based	Reframes synthesis as iterative denoising conditioned on the input frames, generating the target frame through a learned reverse diffusion process rather than direct regression	High	Most perceptually realistic

This thesis builds on the flow-based paradigm. Its explicit and interpretable motion representation, together with native support for arbitrary-time interpolation, makes it well suited to the accuracy-efficiency trade-off targeted by this work. Kernel-based methods lack temporal flexibility, while Transformer- and diffusion-based methods incur substantially higher computational costs, which conflicts with the efficiency objective of this thesis.

Within the flow-based paradigm, BiM-VFI [18] stands out as the strongest candidate for this thesis. Quantitatively, it achieves the best average scores across multiple benchmark datasets on perceptual quality metrics, including LPIPS, STLPIPS, and NIQE. This advantage is also supported by subjective human evaluation, where BiM-VFI achieves a 61% preference rate, indicating that human raters find its outputs more visually convincing than those of competing flow-based methods.

BiM-VFI's standing is further supported by independent work. Kye et al. [16] identifies it as a foundational method for modeling non-linear motion, distinguishing it from methods that assume linear motion between frames. Subsequent works have also adopted it as a

primary point of comparison. Both LDF-VFI [56] and TLB-VFI [57] use BiM-VFI as their state-of-the-art baseline. Notably, SNM-VFI [55] reports that BiM-VFI outperforms diffusion-based methods on DAVIS, achieving an Fréchet Inception Distance (FID) score of 35.93. FID measures the similarity between the distributions of generated and real images, with lower values indicating better perceptual similarity. This result shows that a flow-based method can achieve competitive perceptual realism while retaining the lower inference cost of the flow-based paradigm. Taken together, its leading perceptual-quality scores, strong human preference, and consistent adoption as a state-of-the-art reference point motivate this thesis's use of BiM-VFI as its base architecture.

3.2 Optical Flow Estimation

Optical flow is the dominant motion representation in VFI, modeling apparent pixel-wise displacement between consecutive frames [58, 59]. Early learning-based VFI methods directly integrated general-purpose flow estimators into an end-to-end pipeline of motion estimation, warping, and frame synthesis [4, 5, 8, 60, 61, 62]. In this pipeline, flow errors directly degrade the interpolated frame, causing ghosting, blurring, or structural artifacts [59, 63, 64]. This dependency motivates a closer look at how flow estimators themselves have evolved.

Early coarse-to-fine models like SpyNet [65] estimate flow via spatial pyramids [66] by predicting residual updates without explicit correlation volumes. While lightweight, this approach struggles with large or complex displacements compared to correlation-based methods [59, 63, 64]. Modern estimators instead build dense correlation volumes, typically at reduced resolutions ($\frac{H}{8}$ in RAFT [67] and SEA-RAFT [68], or $\frac{H}{4}$ in BiM-VFI [18] and MS-RAFT [69]) to balance spatial detail against memory constraints. MS-RAFT+ [70] extends this trade-off to $\frac{H}{2}$ using on-demand sampling, while ReCoVEr [71] demonstrates that correlation volumes can even be dropped at inference without compromising accuracy.

Since estimated flow fields are produced at reduced resolutions, they must be upsampled before warping full-resolution frames. As summarized in Table 3.3, the trade-offs among pyramid structure, correlation resolution, and computational cost remain an active, evolving design choice in modern optical flow architectures.

This comparison highlights three core trends. First, general-purpose estimators have evolved from SpyNet's simple pyramid toward correlation-volume methods, refined either at a single scale (RAFT, SEA-RAFT) or hierarchically (MS-RAFT+). Second, context networks play an increasingly critical role: SEA-RAFT regresses an initial flow to cut

Table 3.2: Taxonomy of optical flow estimation strategies across general-purpose estimators and VFI architectures.

Method	Coarse-to-Fine Pyramid	Correlation / Cost Volume	Correlation Resolution	Refinement Strategy
General-purpose optical flow estimators				
SpyNet [65]	Yes	No	N/A	Residual flow prediction per pyramid level
RAFT [67]	No	Yes	$\frac{H}{8} \times \frac{W}{8}$	Iterative GRU refinement via cost volume (24–32 iters)
SEA-RAFT [68]	No	Yes	$\frac{H}{8} \times \frac{W}{8}$	Direct initial flow prediction + GRU refinement (4 iters)
MS-RAFT [69]	Yes (3-scale)	Yes	$\frac{H}{4} \times \frac{W}{4}$	Coarse-to-fine initialization with shared update block
MS-RAFT+ [70]	Yes (4-scale)	Yes	$\frac{H}{2} \times \frac{W}{2}$	High-resolution estimation using on-demand cost sampling
ReCoVEr [71]	No	Used in training only	$\frac{H}{8} \times \frac{W}{8}$ (training only)	Context net predicts flow at inference (cost volume dropped without retraining)
VFI-integrated flow architectures				
RIFE [37]	Yes	No	N/A	Stacked IFBlocks predict residual updates; privileged teacher block discarded at inference
UPR-Net [49]	Yes	Yes (partial volume from warped features)	$\frac{1}{4}$ resolution per level	Shared weights across scales; joint flow and intermediate frame refinement
AMT [50]	Yes	Yes (bidirectional all-pairs built once)	$\frac{H}{8} \times \frac{W}{8}$	Per-level update blocks refine flow/content; finest level fuses multiple flow groups
XVFI [48]	Yes	No (purely convolutional)	N/A	Recursive bidirectional flow estimation followed by full-scale target-time flow reversal
BiM-VFI [18]	Yes	Yes (bidirectional)	$\frac{H}{4} \times \frac{W}{4}$	Recurrent refinement via BiM-guided FlowNet (BiMFN)

refinement iterations from 32 to 4, while ReCoVEr omits the cost volume entirely at inference, achieving up to 2.5× lower latency and 8× lower memory. Third, VFI-integrated models overwhelmingly favor coarse-to-fine structures but split on correlation volumes: lightweight models (RIFE, XVFI) skip them, whereas others retain them at fixed scales (AMT), per-level (UPR-Net), or bidirectionally (BiM-VFI). This divergence shows VFI designs have not converged on a single correlation strategy when co-trained with frame synthesis.

Consequently, flow-based VFI methods inherit this resolution bottleneck, estimating flow ($\tilde{V}_{t\to 0}$, $\tilde{V}_{t\to 1}$) at reduced resolution and relying heavily on downstream upsampling for full-resolution warping [18]. Thus, upsampling fidelity directly governs interpolation quality—an issue further detailed in Section 3.4.

3.2.1 Unidirectional vs. Bidirectional Formulations

A fundamental design choice in VFI lies in flow directionality, as summarized in Table 3.3. Early architectures rely on *unidirectional flow* ($F_{0\to 1}$ or $F_{1\to 0}$) scaled to target time t. However, this requires forward warping, which is prone to pixel collisions and occlusion holes [25, 62], while implicitly assuming linear uniform motion.

To overcome these limits, modern frameworks adopt *bidirectional flow* ($V_{t\to 0}$ and $V_{t\to 1}$) anchored at I_t. This enables backward warping for dense, hole-free frame synthesis [37, 49, 50]. Furthermore, advanced models like BiM-VFI [18] incorporate explicit motion descriptors to resolve time-to-location ambiguity under complex, non-uniform trajectories.

3.3 Core Challenges in Flow-based VFI

Flow-based VFI performance is bounded by four recurring challenges: large motion, occlusion, lighting variation, and non-linear motion [16]. Large motion and occlusion have been extensively addressed through coarse-to-fine pyramid estimation [67] and occlusion-aware blending masks [18], and are now reasonably well handled by modern architectures. Non-linear motion, however, remains comparatively under-addressed: because two input frames provide only two temporal samples, any assumption about how an object accelerates, decelerates, or changes direction between them is fundamentally underdetermined. This limitation directly motivates the choice of baseline for this thesis, discussed next.

Table 3.3: Taxonomy and comparison of optical flow estimation formulations in VFI architectures.

Optical Flow Formulation	Representative Models	Flow Direction & Estimation	Key Advantages	Limitations
Bidirectional Flow	BiM-VFI [18] EMA-VFI [38] UPR-Net [49] AMT [50] IFRNet [72]	Estimates $V_{t\to 0}$ and $V_{t\to 1}$ directly anchored at intermediate time t	• Guarantees dense, hole-free synthesis via backward warping • Handles non-uniform motion and complex trajectories well	• Requires predicting flow at I_t without accessing target frame during inference
Unidirectional Flow	Super SloMo [1] Softmax Splatting [25] RIFE [37] DAIN [62]	Estimates $V_{0\to 1}$ or $V_{1\to 0}$ between input frames, then scales to time t	• Easily integrates pre-trained flow estimators (e.g., RAFT) • Straightforward motion modeling	• Uses forward warping; prone to pixel collisions and holes • Struggles with non-uniform acceleration/deceleration

3.4 Warping Artifacts in Flow-Based VFI

Given coarse, low-resolution flow, flow-based VFI methods synthesize $\hat{I}_t$ by warping I_0 and I_1 toward time t and blending the results. This warping step is where estimation error and resolution loss stop being abstract and become visible pixel-level defects. Two failure modes dominate the literature. Disocclusion holes occur where content visible at t is absent from both source frames (e.g., background uncovered as a foreground object moves); naive backward warping leaves such regions undefined or fills them with incorrect, duplicated content, producing ghosting [73]. Motion-boundary blur, meanwhile, arises specifically from upsampling: bilinear or otherwise naive upsampling of low-resolution flow smooths the field across object boundaries, so pixels adjacent to a motion discontinuity are warped with an incorrect, averaged displacement, producing halo-like blur or doubled edges exactly where accuracy matters most [74].

Splatting-based, forward-warping methods such as M2M [75] address the first failure mode by pushing source pixels forward toward t and resolving collisions with learned weighting. This avoids the undefined-region problem of backward warping but introduces a different issue, namely overlapping pixels in occluded areas.

The second failure mode, motion-boundary blur caused by flow upsampling, is addressed more directly by guided-upsampling methods, which condition the upsampling process on additional context rather than upsampling the flow field in isolation. BiM-VFI uses a Content-Aware Upsampling Network (CAUN) to guide flow upsampling with warped multi-scale image features across two stages. This improves boundary fidelity but requires three additional backward-warping operations per forward pass [18]. Wu et al. [74] take a related approach at high output resolutions, refining bilinearly-upsampled flow using convolutional features extracted from the input frames themselves, then pairing this refined flow with an explicit texture-mapping step that further corrects boundary regions using texture blocks sourced directly from the input frames.

These guided-upsampling approaches share a lesson directly relevant to this thesis. Boundary fidelity in the upsampled flow strongly affects whether warping artifacts remain visible in the synthesized frame. Stronger guidance, such as additional warping passes or richer conditioning features, can improve fidelity but also increases computational cost.

This trade-off is evident in BiM-VFI's CAUN (Section 3.9). It motivates this thesis's search for a lighter-weight upsampling mechanism that retains guided, boundary-aware behavior without the multi-stage warping overhead of CAUN. The redesigned mechanism, GFU, is presented in Chapter 4.

3.5 Time-to-Location Ambiguity and Motion Descriptors

A specific manifestation of the non-linear motion problem is what Seo et al. [18] term *time-to-location (TTL) ambiguity*: given only two source frames, infinitely many trajectories are consistent with the same pair of endpoints, so a model supervised only by the target timestep t has no way to distinguish between them during training. In practice, this forces the network to average over plausible trajectories, producing blurred interpolations precisely in the regions where motion is most complex [18].

Two motion descriptors have been proposed to resolve this ambiguity during training rather than at inference. Zhong et al. [45] introduce *distance indexing*, which encodes the relative magnitude between the motion from I_0 to I_1 and from I_0 to I_t as a pixel-wise ratio map. This resolves ambiguity in motion *speed*, but because it discards directional information, it cannot distinguish between two trajectories that reach the same intermediate position via different *directions* — for example, two objects accelerating toward the same point along different curved paths remain indistinguishable under distance indexing alone.

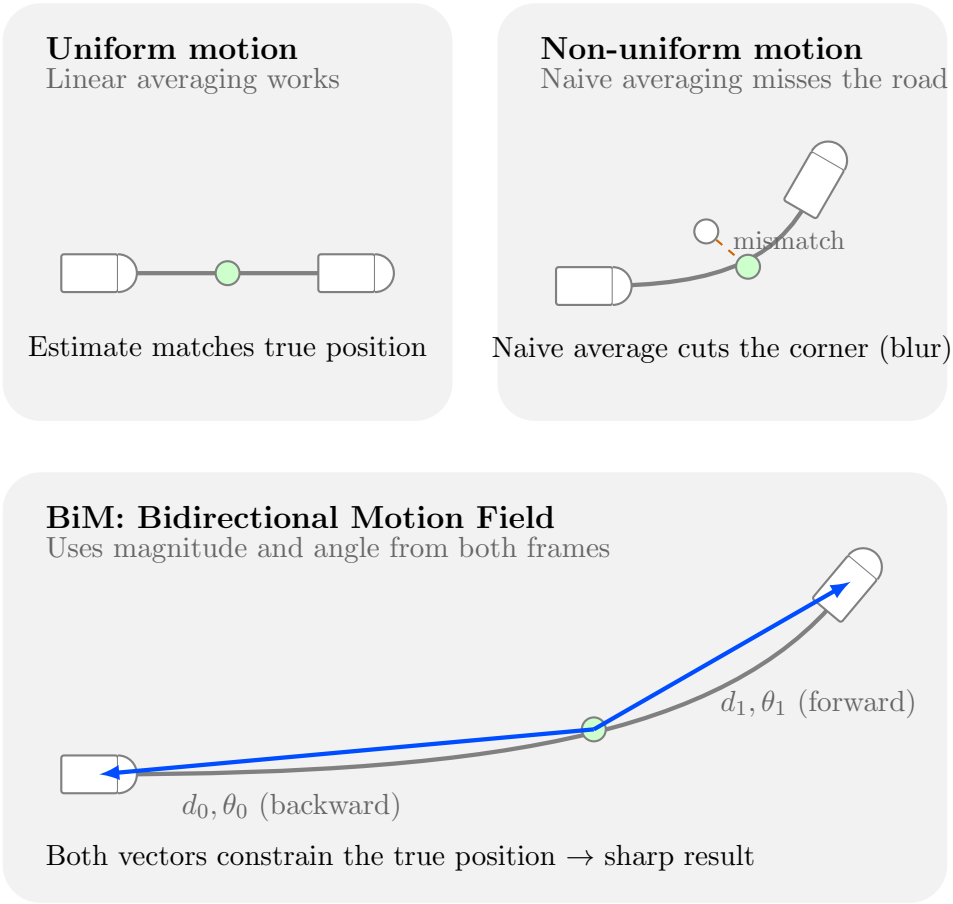
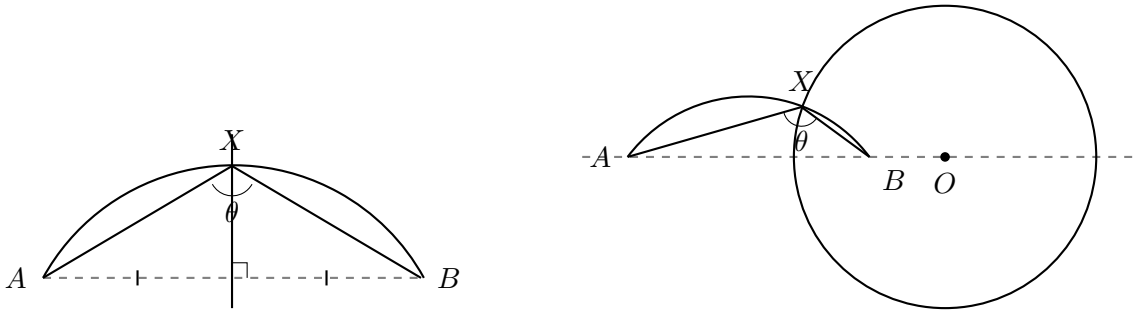


Figure 3.1: Comparison of motion estimation techniques: The top row illustrates how linear averaging succeeds under uniform motion but fails during non-uniform motion. The bottom row demonstrates the BiM approach, utilizing bidirectional vectors to accurately constrain the true position.

Seo et al. [18] address this remaining gap with the *Bidirectional Motion Field (BiM)*, which augments the same magnitude-ratio information with an explicit angular component ϕ between the two directional flows $V_{t\to 0}$ and $V_{t\to 1}$. This additional angular term allows BiM to disambiguate motions that distance indexing cannot. Consider two objects that happen to reach the same relative position between I_0 and I_1 at the same ratio, one moving at a constant, unchanging direction and the other undergoing a directional change partway through its trajectory (e.g., a curved or accelerating path). Distance indexing, which only encodes the magnitude ratio, describes both cases identically and therefore cannot tell them apart. BiM, by contrast, captures the difference in angular change between the two bidirectional flows, and so assigns the two motions distinct descriptors even though their magnitude ratios are the same [18].

Fig. 3.1 illustrates this contrast conceptually: linear averaging under uniform motion recovers the true intermediate position, but the same naive averaging cuts the corner under non-uniform motion, whereas the bidirectional vectors used by BiM jointly constrain the true position regardless of motion linearity.



(a) $k = 1$ (b) $k \neq 1$

Figure 3.2: Geometric proof of the uniqueness of the Bidirectional Motion Field representation for (a) $k = 1$ and (b) $k \neq 1$.

Formally, for each pixel location (x, y), the BiM descriptor is represented as a two-channel vector field containing the normalized magnitude ratio R and the angular difference ϕ, computed between the orientations of the two bidirectional optical flows $V_{t\to 1}$ and $V_{t\to 0}$ [18]:

$$M_{t\to 0,1}(x, y) = [R(x, y), \Phi(x, y)]^T = \left[\frac{r_0}{r_0 + r_1}, \phi\right]^T \tag{3.5.1}$$

where $r_0 = \|V_{t\to 0}(x, y)\|$ and $r_1 = \|V_{t\to 1}(x, y)\|$ denote the bidirectional optical flow

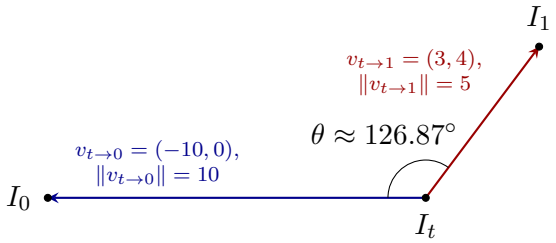


Figure 3.3: Geometric interpretation of the motion prior $M = (r, \phi)$. The magnitude ratio $r = \|v_{t\to0}\|/(\|v_{t\to0}\| + \|v_{t\to1}\|) = 0.667$ reflects non-uniform motion speed ($r \neq t$). The direction vector $\phi = [\cos\theta, \sin\theta]$ encodes the relative orientation between backward and forward flows via the enclosed angle $\theta \approx 126.87°$.

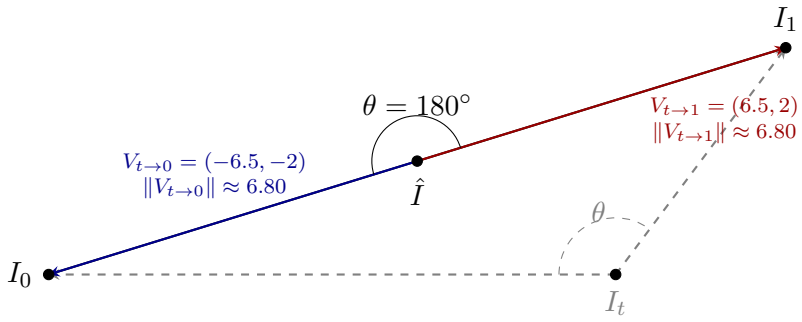


Figure 3.4: Geometric interpretation of the uniform motion prior $M^{\text{uni}} = (0.5, \pi)$ used at inference. The dashed gray vectors represent the true I_t position from Fig. 3.3 for direct comparison. Without true I_t at inference, placing $\hat{I}$ at the midpoint enforces symmetric magnitude ratio $r = 0.5$ and opposite flow directions $\theta = 180°$ ($\phi = \pi$).

magnitudes, and $\phi = \angle V_{t\to 1}(x,y) - \angle V_{t\to 0}(x,y)$ represents their angular difference [18]. Fig. 3.3 illustrates this computation numerically for a non-uniform case ($r \neq 0.5$), while Fig. 3.4 shows the constant-velocity prior $M^{\text{uni}} = (0.5, \pi)$ used when no ground-truth target frame is available at inference.

The uniqueness of this representation is mathematically guaranteed via geometric proof in Fig. 3.2. For pixel positions A (I_0) and B (I_1), the locus of all spatial locations X (I_t) maintaining a constant magnitude ratio $k = r_0/r_1$ forms a perpendicular bisector ($k = 1$, Fig. 3.2(a)) or an Apollonian circle ($k \neq 1$, Fig. 3.2(b)). Concurrently, the locus of locations sharing a constant angle $\theta = \phi$ forms a circular arc $\widehat{AB}$. Because these two loci intersect at exactly a single, unique spatial location X, BiM completely resolves TTL ambiguity.

This unique, disambiguated formulation is what this thesis builds on directly: Chapter 4 incorporates BiM as a prior guiding feature representations toward sharp intermediate-frame synthesis.

3.6 Motion and Context Feature Extractors

RAFT [67] employs separate feature and context encoders. The feature encoder builds correlation volumes from both frames. Meanwhile, the context encoder initializes the recurrent update operator. Similarly, BiM-VFI [18] relies on two distinct extractors. Specifically, it employs a Motion Feature Extractor (MFE) alongside a Context Feature Extractor (CFE). Both are built on the same ResNet-style pyramid backbone [23]. Trained with independent weights ($W_{\text{MFE}} \neq W_{\text{CFE}}$), they specialize in motion estimation and frame synthesis, respectively.

This separation allows each extractor to specialize in a different task. MFE features are used by the BiMFN to estimate the bidirectional flows $\tilde{V}_{t\to 0}$ and $\tilde{V}_{t\to 1}$. CFE features are then passed to the CAUN and Synthesis Network (SN). These features are used to warp the source frames and synthesize the output, as shown in Fig. 3.5, where ∇ denotes the direction of gradient flow during backpropagation. The two extractors are also trained with different objectives. MFE is mainly supervised by a flow-centric loss ($\mathcal{L}_{\text{flow}}$) that encourages its estimated flows to match teacher-generated flows, whereas CFE is mainly supervised by a photometric loss ($\mathcal{L}_{\text{photo}}$) that compares the synthesized frame with the target frame (I_t).

As shown in Fig. 3.6, each encoder outputs a three-level feature pyramid $[C_0, C_1, C_2]$ with base channel width $C_{\text{feat}} = 32$:

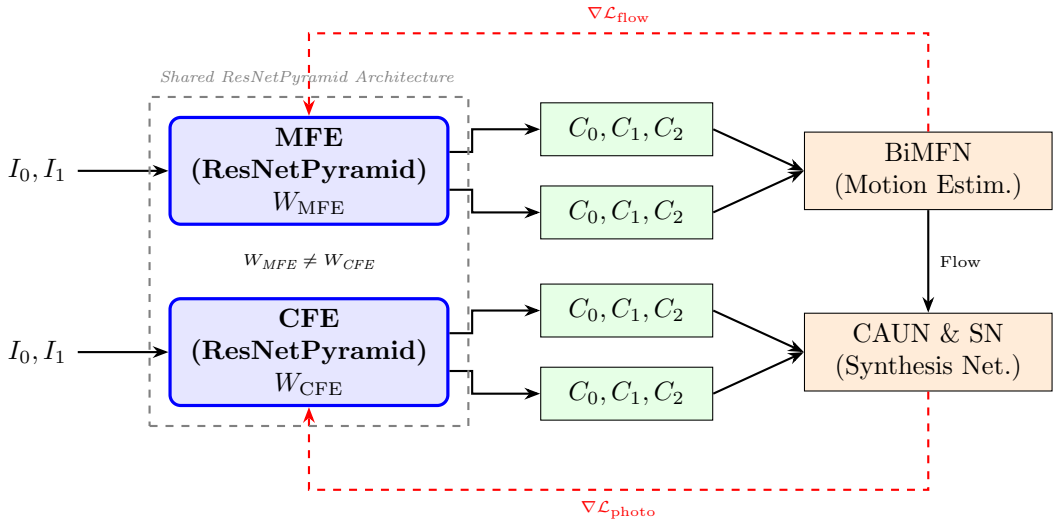


Figure 3.5: Motion and Context Feature Extractors of BiM-VFI

- C_0**:** $32 \times H \times W$ — full resolution, fine detail.
- C_1**:** $64 \times \frac{H}{2} \times \frac{W}{2}$ — half resolution, balanced context.
- C_2**:** $128 \times \frac{H}{4} \times \frac{W}{4}$ — quarter resolution, high-level semantics.

The MFE pyramids feed into BiMFN for coarse-to-fine flow estimation, while the CFE pyramids feed into the Context-Aware Upsampling Network (CAUN) and Synthesis Network (SN) to guide frame generation.

3.7 State-of-the-art Flow Upsampling Strategies in VFI

Modern video frame interpolation (VFI) methods commonly avoid estimating optical flow at full resolution due to its high computational and memory cost [46, 67]. Instead, flow is typically predicted at a lower spatial resolution and later upsampled to the target resolution. This design improves efficiency but introduces a new challenge: recovering accurate high-resolution motion details from a coarse flow representation. As summarized in Table 3.4, existing methods address this challenge through four broad upsampling strategies, ranging from simple fixed-kernel interpolation to learned, content-aware refinement.

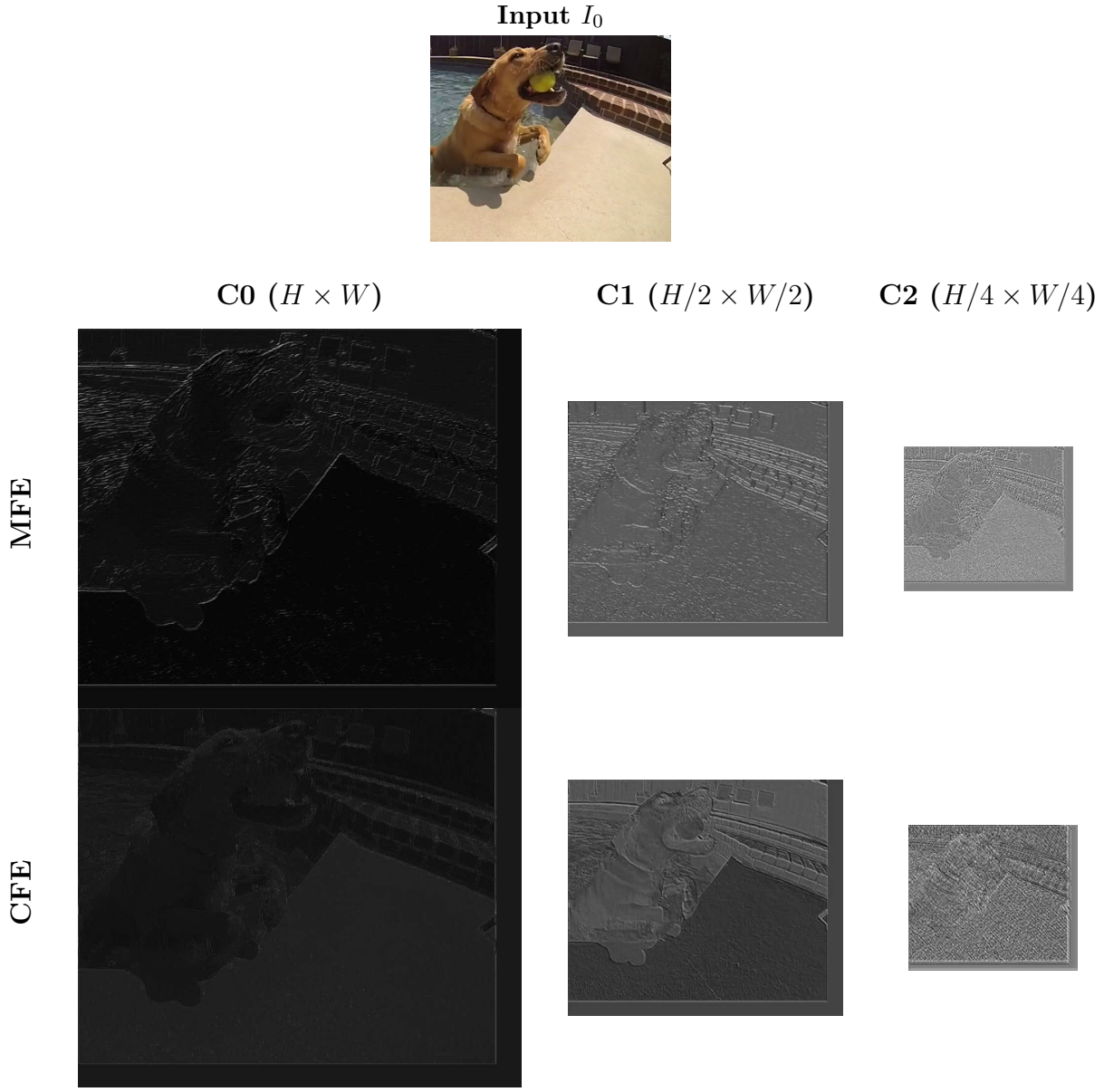


Figure 3.6: Multi-scale feature maps (C_0, C_1, C_2) extracted from I_0 by the MFE and CFE branches, illustrating the hierarchical structural abstraction within the Symmetrical Shared Feature Encoder.

Table 3.4: Comparison of Flow Upsampling Strategies in Optical Flow and VFI Methods

Upsampling Strategy	Description	Example Methods
Bilinear / Nearest Interpolation	Flow is directly resized to the target resolution using fixed interpolation kernels, without learned or content-aware refinement.	PWC-Net [46], SpyNet [65]
Convex Upsampling	A network predicts per-pixel convex weights over a local neighbourhood, reconstructing high-resolution flow as a softmax-weighted combination of low-resolution neighbours [67].	RAFT [67], BiM-VFI [18]
Flow Refinement at Full Resolution	Coarse flow is first upsampled and then refined by a dedicated network that corrects residual motion errors at full resolution.	UPR-Net [49], CRAFT [76]
Deformable / Kernel-based Upsampling	Spatially varying kernels or deformable sampling offsets are learned to aggregate context-aware motion information for each output location.	DAIN [77], AMT [50]

3.7.1 The Limitation of Naive Flow Upsampling

In modern optical flow and VFI pipelines, flow fields are typically estimated at a fraction of the full image resolution, commonly 1/4 or 1/8, to reduce memory and computational cost [46, 67]. The estimated low-resolution flow must then be upsampled to match the original image dimensions before it can be used to warp input frames for synthesis.

The most straightforward approach is bilinear or bicubic interpolation, which computes each high-resolution output as a fixed weighted average of surrounding low-resolution values. However, this approach is fundamentally ill-suited for optical flow: at motion boundaries, pixels on either side may belong to different objects moving in entirely different directions. Interpolating between them produces a spurious intermediate value that does not correspond to any real motion [67]. As noted by Seo et al. [18], bilinear and bicubic upsampling can incur flow leakages along object boundaries and cause small objects to lose their motion identity by blending with surrounding flows. The alternative, nearest-neighbour upsampling, avoids blending but introduces blocky, staircase artefacts that misalign flow edges with actual object contours [78].

3.7.2 Convex Upsampling

To overcome the limitations of naive flow upsampling, Teed and Deng [67] proposed *convex upsampling*, which replaces the fixed interpolation kernel with a *content-adaptive* one. The core idea is that each high-resolution output pixel is expressed as a convex combination of a local 3×3 neighbourhood of low-resolution flow values:

$$\mathbf{V}^{\mathrm{HR}}(p) = \sum_{k \in \mathcal{N}(p)} w_k \, \mathbf{V}^{\mathrm{LR}}(k), \quad \sum_k w_k = 1, \quad w_k \geq 0 \tag{3.7.1}$$

where $\mathbf{V}^{\mathrm{HR}}(p)$ is the upsampled flow at high-resolution pixel p, $\mathcal{N}(p)$ denotes the 9 low-resolution neighbours in a 3×3 grid, and $\{w_k\}$ are per-pixel weights satisfying the convexity constraint (enforced via softmax). Crucially, the weights are *predicted dynamically* at inference time by a small convolutional network conditioned on the image features, meaning each pixel receives a unique blending kernel tailored to the local content, rather than a generic fixed one [67]. The non-negativity and sum-to-one constraints ensure the upsampled flow at any pixel remains bounded by its low-resolution neighbours, preventing the spurious extrapolation that plagues unconstrained upsampling methods.

Adoption of Convex Upsampling in VFI and Its Limitations

Although convex upsampling was originally proposed for optical flow estimation [67], its benefits apply equally to VFI, where inaccurate upsampled flow leads directly to warping artefacts in the synthesised frame. Seo et al. (2025) [18] observe that adaptive upsampling is commonly employed in optical flow models but has not been widely adopted in VFI pipelines. They address this gap by introducing the Content-Aware Upsampling Network (CAUN), which extends convex upsampling for bidirectional VFI flow. CAUN constructs its adaptive kernels by warping multi-scale features from both input frames at three pyramid levels, progressively fusing them through stacked encoders with pixel-shuffle upsampling to produce separate masks for $\times2$ and $\times4$ upsampling stages. This multi-scale feature aggregation enables CAUN to preserve fine object boundaries and small motion details that single-scale upsampling cannot recover [18].

However, this design introduces non-trivial complexity. CAUN requires three backward warping operations per forward pass, one at each pyramid level. It also uses approximately 605K parameters for the upsampling module alone. Furthermore, the masks depend on warped features and are therefore affected by flow accuracy. This can be problematic during early training or under challenging motion, when flow estimates are less reliable. These

limitations motivate a simpler alternative that retains the core convex combination mechanism. The proposed method removes the warping dependency and multi-scale overhead and is presented in Chapter 4.

3.8 Knowledge Distillation in VFI

3.8.1 Motivation and Problem Setting

Video Frame Interpolation (VFI) aims to synthesize intermediate frames between consecutive inputs while maintaining high visual fidelity. However, lightweight models are often constrained by limited representational capacity, making it difficult to handle complex and non-uniform motion. As a result, models trained solely with pixel-wise supervision, such as L_1 or MSE losses, frequently suffer from blurring, ghosting, and temporal inconsistency [30, 37]. These limitations motivate the use of additional supervisory signals beyond direct reconstruction.

3.8.2 Fundamentals of Knowledge Distillation

Knowledge Distillation (KD) mitigates this issue by transferring knowledge from a high-capacity Teacher model to a lightweight Student model [79, 80]. The Teacher, often trained with access to privileged information such as ground-truth intermediate frames, provides richer supervisory signals in the form of soft outputs, feature representations, or motion cues. This enables the Student to learn more informative priors while maintaining a low inference cost. Importantly, distillation is applied only during training, ensuring no additional computational overhead at inference time.

3.8.3 KD Mechanisms and Taxonomy in VFI

In the context of VFI, knowledge distillation methods can be broadly categorized into response-based, feature-based, relation-based, and flow-based approaches [81]. Response-based methods align final outputs such as interpolated frames or optical flow, while feature-based methods transfer intermediate representations across network layers. Relation-based approaches focus on preserving structural dependencies such as temporal consistency and motion correlation [82]. In addition, VFI-specific methods emphasize motion- or flow-based distillation, where optical flow or bidirectional motion fields are explicitly transferred to

better model dynamic scene changes. Table 3.5 summarizes representative methods under each category.

Table 3.5: Taxonomy of Knowledge Distillation in VFI and representative methods.

KD Type	Transferred Knowledge	Representative Methods in VFI
Response-based KD	Final outputs such as interpolated frames or predicted optical flow	RIFE [37], ST-MFNet Mini [83]
Feature-based KD	Intermediate feature representations and multi-scale embeddings	FitNets-style approaches [84] (rarely explicitly used in VFI)
Relation-based KD	Structural relationships such as temporal consistency or motion correlation	General VFI consistency learning methods [82]
Motion / Flow-based KD (VFI-specific)	Optical flow, bidirectional motion fields, or pseudo flow supervision	ST-MFNet [85], BiM-VFI [18]

3.8.4 BiM-VFI and Task-Specific Distillation

Fig. 3.7 illustrates the task-specific knowledge distillation framework in BiM-VFI, referred to as KDVCF. During training, the teacher network processes the privileged input triplet (I_0, I_1, I_t) to estimate accurate bidirectional motion fields and a per-pixel motion prior (r, ϕ). This prior represents the magnitude ratio and relative direction of the true trajectory. It is detached and directly provided to the student network to guide motion estimation.

In contrast, the student uses only (I_0, I_1) and does not access I_t. At inference time, the distilled prior is replaced with a fixed uniform prior $(r = 0.5, \phi = \pi)$. The distillation loss $\mathcal{L}_{\text{distill}}$ aligns the student's residual motion fields with the teacher's at the corresponding stages. Both teacher and student share network parameters, while standard reconstruction losses are omitted from the diagram for visual clarity.

The training objective uses the knowledge distillation loss $\mathcal{L}_{\text{distill}}$ to maintain consistency between the teacher's and student's motion representations. This allows the student to learn motion-aware priors from the teacher while remaining computationally efficient at inference time.

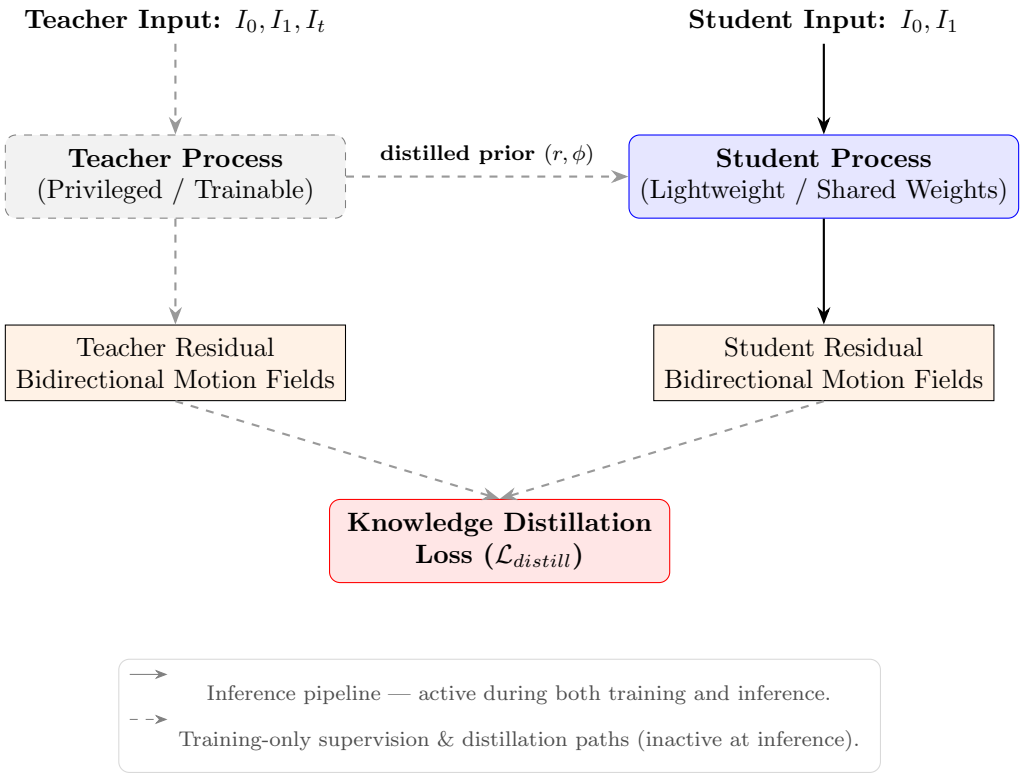


Figure 3.7: Bidirectional-motion knowledge distillation framework in BiM-VFI (Knowledge Distillation for VFI-centric Flow supervision (KDVCF)).

3.9 Justifying the BiM-VFI Baseline

Flow-based VFI still faces two unresolved challenges. First, non-linear motion remains underdetermined with only two input frames (Section 3.3). Second, guided flow upsampling requires balancing boundary fidelity against computational cost (Section 3.4). BiM-VFI is the strongest available response to the first of these. By jointly encoding bidirectional magnitude ratio and angular difference, BiM resolves TTL ambiguity outright rather than merely mitigating it (Section 3.5). Building on this formulation, BiM-VFI improves LPIPS and STLPIPS by 26% and 45% over prior baselines, including RIFE [37], UPR-Net [49], and AMT [50], on arbitrary-time interpolation benchmarks. It also remains competitive or superior on fixed-time datasets despite not being explicitly trained for them [18]. These gains are not incidental to the architecture. BiM-VFI's ablation study confirms they stem specifically from the angular component of BiM. Replacing the full descriptor with

a ratio-only variant degrades all reported perceptual metrics (LPIPS [30], STLPIPS [33], NIQE [34]). This confirms the angular term's advantage over magnitude-only descriptors like distance indexing [18].

Beyond numerical metrics, the superior perceptual quality of BiM-VFI is strongly supported by subjective human evaluations, where it achieves a 61% user preference rate over competing methods [18]. Its significance in handling non-linear motion is also widely recognized across the VFI literature. Specifically, it is highlighted as a foundational baseline in recent survey papers such as *AceVFI* and serves as the primary baseline for state-of-the-art follow-up architectures, including *LDF-VFI* [56], *TLB-VFI* [57], and *SNM-VFI* [55].

BiM-VFI's response to the second challenge sits at the expensive end of the trade-off described in Section 3.4. Its dual independently-weighted encoders and multi-stage CAUN upsampling achieve strong boundary fidelity, but only at substantial structural cost (Section 3.7.2). The architecture is well suited to resolving TTL ambiguity but was not designed for lightweight deployment. This gap, a strong motion descriptor paired with an unaddressed efficiency problem, is precisely what this thesis targets: retaining BiM-VFI's Bidirectional Motion Field formulation while redesigning its two heaviest components, the feature encoder and the flow upsampling stage, into MD-Net and GFU (Chapter 4).

A natural question is whether BiM-VFI remains the strongest available baseline given more recent work. Jeong et al. [55] propose SNM-VFI, which pairs a symmetric nonlinear motion prior with a pre-trained video diffusion model, and reports higher SSIM and LPIPS than BiM-VFI on several benchmarks. Two considerations limit the relevance of this comparison. First, the settings differ. SNM-VFI is evaluated on short, low-resolution clips at fixed $\times 2/\times 4$ ratios, not the arbitrary-time, non-uniform-motion setting BiM-VFI's motion descriptor was designed for. Second, and more directly, BiM-VFI still wins on the metric best suited to judging non-uniform motion. On the DAVIS benchmark, BiM-VFI achieves an FID score [86] of 35.93, the best of all methods compared, including SNM-VFI (43.04) [55]. This matters because FID measures the distributional realism of generated frames, rather than per-pixel agreement with the ground truth; under ambiguous motion, PSNR and SSIM can reward blurred, averaged-out predictions, while FID does not. BiM-VFI's lead in FID therefore indicates that its outputs remain the most perceptually realistic of the compared methods, precisely where pixel-wise metrics are least trustworthy.

SNM-VFI's gains also come with a cost this thesis cannot accept: it depends on an off-the-shelf video diffusion model and iterative denoising at inference, making it substantially more expensive than BiM-VFI's single forward pass. Among end-to-end trainable, non-generative architectures, BiM-VFI therefore remains both the most perceptually accurate and the most practical foundation to build on. This thesis adopts it as the baseline

and targets its two heaviest components for efficiency-oriented redesign, as detailed in Chapter 4.

3.10 Evaluation Practices in VFI Literature

Consistent with the shift toward perceptually-aware training objectives discussed above, recent VFI literature has moved away from relying solely on pixel-centric metrics such as PSNR and SSIM, which are known to reward blur under ambiguous motion [16]. Full-reference perceptual metrics such as LPIPS and its shift-tolerant variant STLPIPS, together with the no-reference metric NIQE, are now standard alongside PSNR/SSIM in state-of-the-art comparisons, including in BiM-VFI's own evaluation protocol and in the AceVFI survey's summary of field-wide evaluation practice [16, 18]. This thesis adopts the same combination of metrics for consistency with the literature; their technical definitions are given in Section 2.5.

Chapter 4

Methodology

4.1 GEM-VFI: Guided and Enhanced Motion Video Frame Interpolation

An overview of the proposed multi-scale Video Frame Interpolation (VFI) framework is schematically illustrated in Fig. 4.1. Using distilled parameters (r, ϕ) from Section 3.5, our framework resolves TTL ambiguity [18]. Architecturally, the system is structured into three cascaded phases: hierarchical image pyramid state propagation, multi-scale feature extraction, and a symmetrical teacher-student knowledge distillation pipeline. During training, the framework utilizes both the teacher process ($\mathcal{P}_\mathcal{T}$), which leverages the intermediate ground-truth frame (I_t), and the student process ($\mathcal{P}_\mathcal{S}$), whereas during inference, the model relies exclusively on the student process ($\mathcal{P}_\mathcal{S}$) without requiring ground-truth guidance. The unified execution sequence across both operating regimes is formally codified in Algorithm 4.1 for the teacher process ($\mathcal{P}_\mathcal{T}$) and Algorithm 4.2 for the student process ($\mathcal{P}_\mathcal{S}$).

As summarized in Table 4.1, the framework uses both Teacher and Student processes during training, but only the Student process during inference. Since both processes share the same network parameters, the Teacher introduces no additional parameters or inference overhead.

To address the primary bottlenecks in non-uniform motion modeling and detail degradation, our proposed framework introduces three key technical contributions:

1. **Motion Detection Network (MD-Net):** Extended from the BiM-guided FlowNet (BiMFN) baseline [18], MD-Net introduces channel-wise feature recalibration via

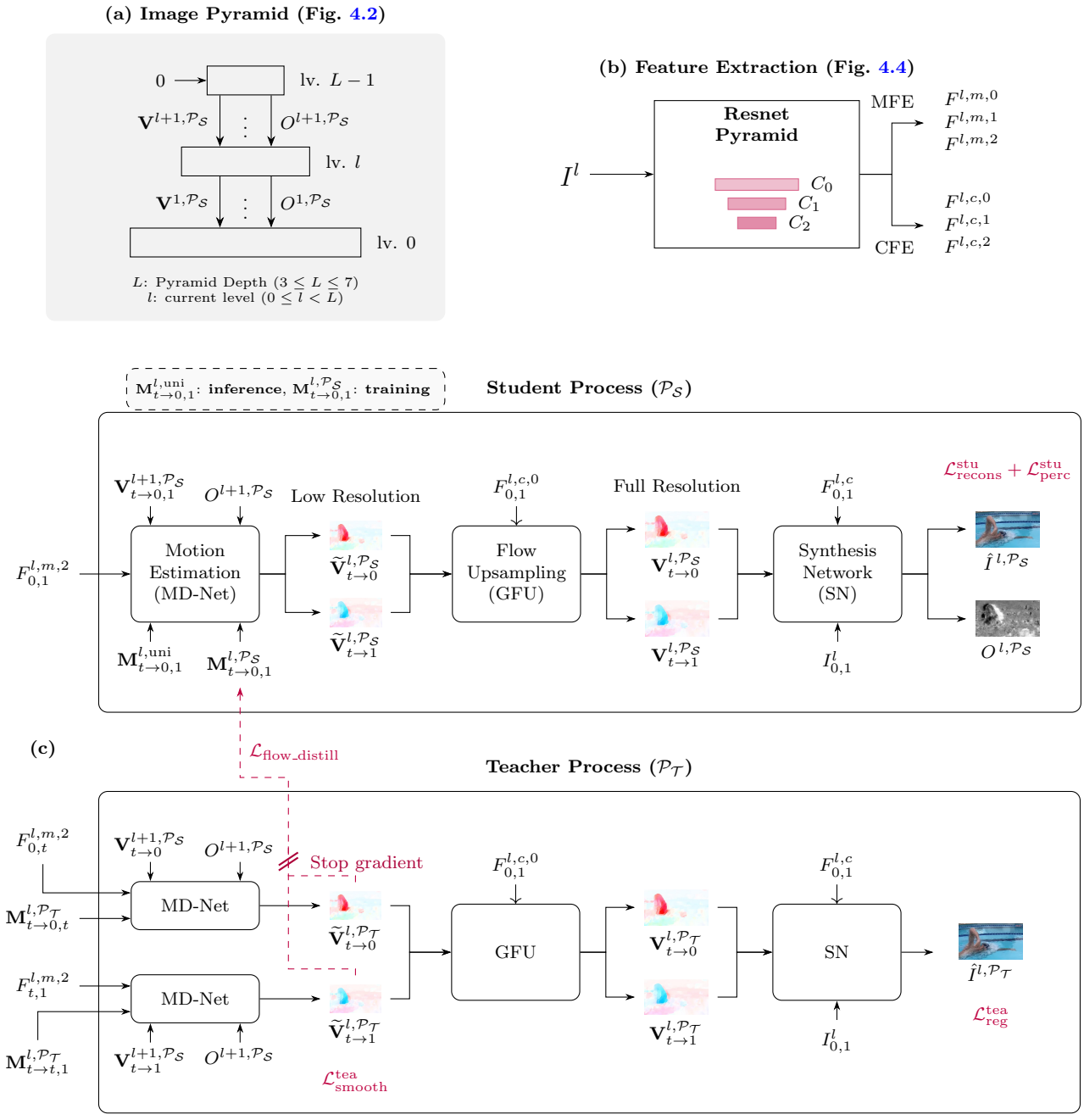


Figure 4.1: Overview of the proposed multi-scale VFI framework

Table 4.1: Comparison of Components Used During Training and Inference

Component	Training Phase	Inference Phase
Input Data	Frame triplet (I_0, I_1, I_t)	Frame pair (I_0, I_1) and target time t
Network Execution	Teacher ($\mathcal{P}_T$) and Student ($\mathcal{P}_S$)	Student ($\mathcal{P}_S$) only
Motion Guidance Prior	$\mathbf{M}_{t\to 0,1}^{l,\mathcal{P}_S}$	$\mathbf{M}_{t\to 0,1}^{l,\text{uni}}$
Loss Functions	Active	Not used

Squeeze-and-Excitation (SE) attention [87] prior to correlation matching, constraining non-uniform motion search spaces and sharpening cost volume representations.

2. **Guided Flow Upsampling (GFU) (Primary Novelty):** A novel context-guided convex upsampling module. Inspired by the convex upsampling concept in RAFT [67] and CAUN in BiM-VFI [18], GFU advances these designs by conditioning mask prediction jointly on bilateral context features ($F_0^{l,c,0}, F_1^{l,c,0}$) and low-resolution flow fields using a single lightweight Guidance Network, eliminating feature-warping dependencies and preserving sharp motion boundaries during $\times 4$ spatial expansion.

3. **Late-Stage Perceptual Loss Integration ($\mathcal{L}_{\text{perc}}$):** A targeted optimization strategy incorporating pre-trained VGG-16 feature discrepancies [35] during late-stage refinement, resolving the over-smoothing tendency of standard photometric losses without distorting underlying motion fields.

The auxiliary feature extraction networks (MFE, CFE) and synthesis network (SN) complete the end-to-end pipeline.

The primary mathematical notations, tensors, and algorithmic components deployed within the unified coarse-to-fine execution sequence (Algorithm 4.1 for $\mathcal{P}_\mathcal{T}$ and Algorithm 4.2 for $\mathcal{P}_\mathcal{S}$) are formally defined as follows:

- $I_{0,1}^l$ denote the source keyframes ($t = 0$ and $t = 1$) resampled to pyramid level l.
- I_t^l denotes the privileged target ground-truth intermediate frame at level l, active exclusively during training.

Algorithm 4.1 Teacher Process ($\mathcal{P}_\mathcal{T}$) (Training only)

Input: $I^l_{0,1,t}, F^{l,m}_{0,1,t}, F^{l,c}_{0,1}$ ▷ input images and features at level l
$\mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}_{t\to 0,1}, O^{l+1,\mathcal{P}_\mathcal{S}}$ ▷ coarse flow and occlusion priors
Output: $\hat{I}^{l,\mathcal{P}_\mathcal{T}}$ ▷ teacher's interpolated frame at level l
for $l = L-1$ **to** 0 **do**
  $\mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,t}, \mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to t,1} \leftarrow$ Motion Guidance ▷ privileged guidance, uses I^l_t
  $\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0} \leftarrow \text{MD-Net}\big(F^{l,m,2}_{0,t}, \mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,t}, \mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}_{t\to 0}, O^{l+1,\mathcal{P}_\mathcal{S}}\big)$ ▷ low-res flow $I_0 \to I_t$
  $\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{T}}_{t\to 1} \leftarrow \text{MD-Net}\big(F^{l,m,2}_{t,1}, \mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to t,1}, \mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}_{t\to 1}, O^{l+1,\mathcal{P}_\mathcal{S}}\big)$ ▷ low-res flow $I_t \to I_1$
  $\mathbf{V}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1} \leftarrow \text{GFU}\big(\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1}, F^{l,c,0}_{0,1}\big)$ ▷ upsample flow to full resolution
  $\hat{I}^{l,\mathcal{P}_\mathcal{T}} \leftarrow \text{SN}\big(I^l_{0,1}, F^{l,c}_{0,1}, \mathbf{V}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1}\big)$ ▷ synthesize teacher's interpolated frame
end for

Algorithm 4.2 Student Process ($\mathcal{P}_\mathcal{S}$) (Training / Inference)

Input: $I^l_{0,1}, F^{l,m}_{0,1}, F^{l,c}_{0,1}$ ▷ input images and features at level l; $I^l_t, F^{l,m}_t$ used only if training
$\mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1}$ (if training) or $\mathbf{M}^{l,\text{uni}}_{t\to 0,1}$ (if inference) ▷ motion guidance (r, ϕ)
$\mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}_{t\to 0,1}, O^{l+1,\mathcal{P}_\mathcal{S}}$ ▷ coarse flow and occlusion priors
Output: $\hat{I}^{\,l,\mathcal{P}_\mathcal{S}}, O^{\,l,\mathcal{P}_\mathcal{S}}$ ▷ student outputs at level l
for $l = L-1$ **to** 0 **do**
  $\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1} \leftarrow \text{MD-Net}\big(F^{l,m,2}_{0,1}, \{\mathbf{M}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1} \text{ or } \mathbf{M}^{l,\text{uni}}_{t\to 0,1}\}, \mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}_{t\to 0,1}, O^{l+1,\mathcal{P}_\mathcal{S}}\big)$ ▷ low-res bidirectional flow
  $\mathbf{V}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1} \leftarrow \text{GFU}\big(\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1}, F^{l,c,0}_{0,1}\big)$ ▷ upsample flow to full resolution
  $\hat{I}^{\,l,\mathcal{P}_\mathcal{S}}, O^{\,l,\mathcal{P}_\mathcal{S}} \leftarrow \text{SN}\big(I^l_{0,1}, F^{l,c}_{0,1}, \mathbf{V}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1}\big)$ ▷ synthesize final frame & occlusion
end for

- $\hat{I}^{l,\mathcal{P}_\mathcal{T}}$ represents the auxiliary intermediate frame synthesized at pyramid level l by the teacher process ($\mathcal{P}_\mathcal{T}$).
- $\hat{I}^{l,\mathcal{P}_\mathcal{S}}$ represents the intermediate frame synthesized at pyramid level l by the student process ($\mathcal{P}_\mathcal{S}$) that is active during inference.
- L denotes the total number of spatial image pyramid levels.
- $l \in \{0, 1, \ldots, L-1\}$ denotes the spatial image pyramid level, where $l = L-1$ corresponds to the coarsest bottleneck resolution and $l = 0$ represents the finest full-resolution tier.
- $F_{0,1}^{l,m}, F_{0,1}^{l,c}$ designate the motion feature (F^m) and context feature (F^c) pyramids extracted from source frames (I_0, I_1) at level l via MFE and CFE, respectively.
- $F_{0,t}^{l,m}, F_{t,1}^{l,m}$ specify the teacher-side motion feature pyramids extracted between the privileged target frame I_t and source frames (I_0, I_1).
- $F^{l,m,2}$ represents the coarsest motion features (C_2) supplied directly into MD-Net for correlation matching.
- $F^{l,c,0}$ represents the finest context features (C_0) utilized by GFU to guide convex upsampling.
- $\mathbf{V}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}$ identifies the full-resolution/upsampled bidirectional optical flow field ($t \to 0$ and $t \to 1$) parameterized at pyramid level l within the student process ($\mathcal{P}_\mathcal{S}$).
- $\mathbf{V}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$ represents the corresponding upsampled bidirectional optical flow field estimated within the teacher process ($\mathcal{P}_\mathcal{T}$).
- $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}, \widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$ denote the un-upsampled, low-resolution raw bidirectional flows estimated directly by MD-Net before GFU processing.
- $O^{l,\mathcal{P}_\mathcal{S}}$ specifies the localized pixel-wise occlusion mask prior at pyramid level l for the student process.
- $\mathbf{M}_{t\to 0,t}^{l,\mathcal{P}_\mathcal{T}}, \mathbf{M}_{t\to t,1}^{l,\mathcal{P}_\mathcal{T}}$ identify the teacher-side motion guidance priors conditioning the two unidirectional teacher streams ($I_0 \to I_t$ and $I_t \to I_1$).
- $\mathbf{M}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$ denotes the teacher-distilled motion guidance supplied to the student process during training.

- $\mathbf{M}_{t\to 0,1}^{l,\text{uni}}$ signifies the constant-velocity baseline motion guidance used by the student process during autonomous inference deployment (no teacher available).
- $\text{MFE}(\cdot), \text{CFE}(\cdot)$ denote the Motion Feature Extractor and Context Feature Extractor networks, built upon a multi-scale ResNetPyramid backbone.
- MD-Net$(\cdot)$ represents the Motion Detection Network responsible for localized optical flow estimation and correlation matching.
- GFU$(\cdot)$ identifies the Guided Flow Upsampling module that projects low-resolution flows via context-guided convex combination.
- SN$(\cdot)$ designates the Synthesis Network configured to execute joint pixel-and-feature warping for final frame reconstruction.

4.2 Image Pyramid Structure and State Propagation

The multi-resolution pipeline begins by building a hierarchical Image Pyramid. Fig. 4.2 shows how processing proceeds across resolution levels. The pyramid depth L is not fixed. It depends on the input resolution as summarized in Table 4.2. During training, input frames are cropped to fixed 256×256 patches. This crop size gives a fixed pyramid depth of $L = 3$. During inference, the input resolution varies, so L is chosen adaptively based on the frame height h. Low-resolution inputs ($h < 1080$) use $L = 5$. Full HD inputs ($1080 \leq h < 2160$) use $L = 6$. Ultra HD/4K inputs ($h \geq 2160$) use $L = 7$.

Following spatial rescaling, non-standard frame dimensions are adaptively zero-padded to the nearest spatial dimensions divisible by 2^{L+1} (e.g., padding a 1080×2048 input to 1152×2048 for $L = 5$), as illustrated in Fig. 4.3. This adaptive pre-padding preserves structural aspect ratios, avoids rounding artifacts, and enables patch-free evaluation on arbitrary video inputs. Each pyramid level is resampled in a single pass from the padded reference frames (I_0, I_1) via bilinear interpolation with anti-aliasing to prevent compounding decimation artifacts.

Each pyramid level does not estimate flow from scratch: it refines a prior carried down from the coarser level above it. This prior consists of the previous level's flow field and occlusion map, downsampled and rescaled to match the current level's resolution.

At the coarsest level ($lv.\ L-1$), no such prior exists, so the initial bidirectional flow and occlusion map are zero-initialized. At every finer level l, the flow and occlusion estimates

Table 4.2: Adaptive Pyramid Depth (L) and Input Scaling Configuration Across Operating Modes

Operating Mode	**Input Resolution / Height (h)**	**Input Scaling Factor**	**Pyramid Depth (L)**
Training	Fixed Crop (256×256 px)	$1.0\times$	3
Inference	Low-Res ($h < 1080$ px)	$1.0\times$	5
Inference	Full HD ($1080 \leq h < 2160$ px)	$0.5\times$	6
Inference	Ultra HD / 4K ($h \geq 2160$ px)	$0.25\times$	7

inherited from the previous level are bilinearly downsampled by a factor of 0.5; the flow field is additionally scaled by 0.5 so that displacement magnitudes remain correct on the halved coordinate grid:

$$\mathbf{V}^l_{\text{prior}} = \mathcal{D}_{0.5}\left(\mathbf{V}^{l+1,\mathcal{P}_\mathcal{S}}\right) \times 0.5 \tag{4.2.1}$$

$$O^l_{\text{prior}} = \mathcal{D}_{0.5}\left(O^{l+1,\mathcal{P}_\mathcal{S}}\right) \tag{4.2.2}$$

where $\mathcal{D}_{0.5}(\cdot)$ denotes bilinear downsampling by a factor of 0.5. MD-Net then predicts a residual flow $\mathbf{V}^l_{\text{res}}$ at the current level, and the level's flow is obtained by adding this residual to the propagated prior, $\mathbf{V}^l = \mathbf{V}^l_{\text{res}} + \mathbf{V}^l_{\text{prior}}$. GFU then refines this into the level's final flow and occlusion map.

Only estimates from the student process ($\mathcal{P}_\mathcal{S}$) are carried through this chain. The teacher process ($\mathcal{P}_\mathcal{T}$) runs in parallel during training but is deliberately excluded from it, so that the propagation path matches exactly what is available at inference, when no teacher exists.

Fig. 4.2 illustrates this propagation across all levels, from the zero-initialized coarsest level ($lv.\ L-1$) to the full-resolution output ($lv.\ 0$, $H \times W$).

4.3 Multi-Scale Feature Extraction Architecture

To capture rich spatial correlations and geometric structures across varying visual frequencies, the framework incorporates a shared deep Convolutional Neural Network (CNN) encoder structure, designated as the Motion Feature Extractor (MFE) and the Context Feature Extractor (CFE). At each active pyramid level l, an input image I^l_i (where $i \in \{0, 1, t\}$ signifies the frame index) is mapped into a high-dimensional feature space. The MFE isolates temporal motion cues for correspondence matching, while the CFE extracts rich contextual priors to guide the subsequent image synthesis phase.

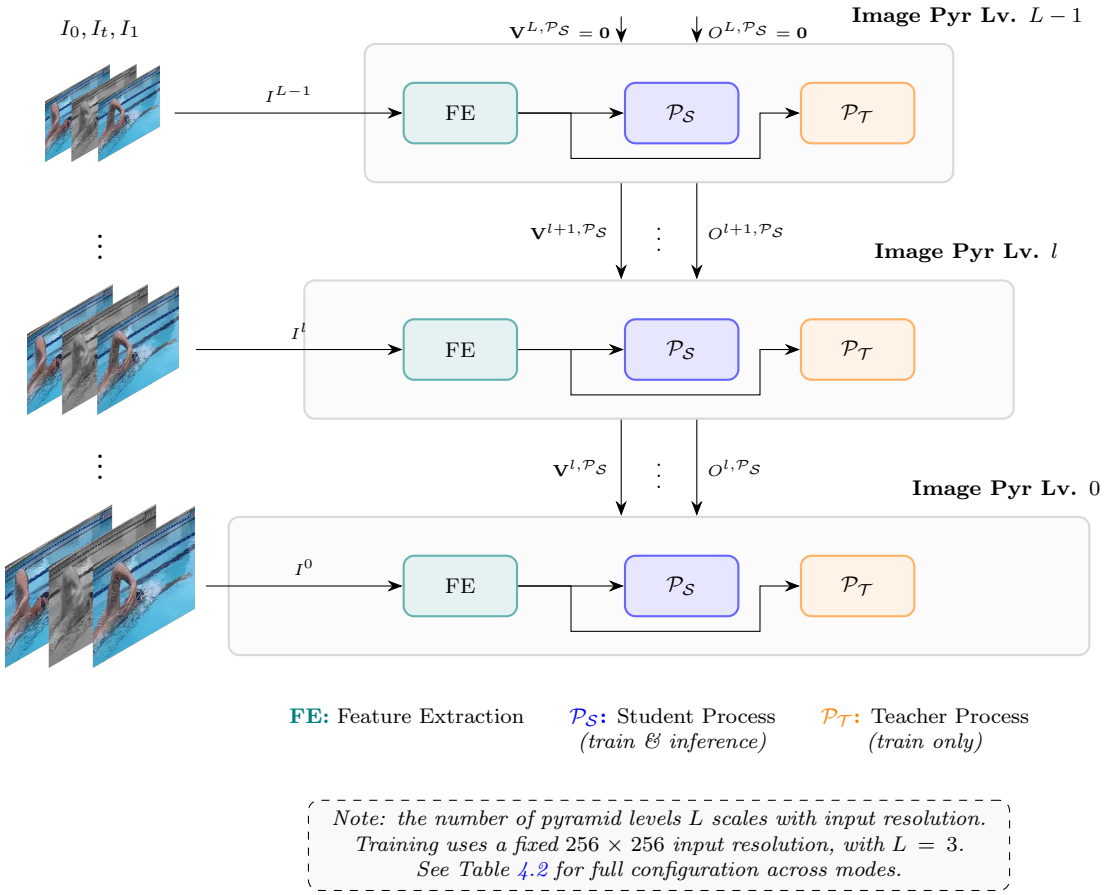


Figure 4.2: Hierarchical Image Pyramid architecture. Feature Extraction (FE), Student Process ($\mathcal{P}_S$), and Teacher Process ($\mathcal{P}_T$) are applied independently at each resolution level. Coarse-to-fine optical flow and occlusion information are propagated from lower-resolution levels to guide finer-level synthesis.

Crucially, the internal feature extraction pipeline generates a multi-scale representation across exactly three fixed scales: C_0 (finest), C_1 (mid-level), and C_2 (coarsest). This three-scale design is structurally decoupled from the external image pyramid depth L, maintaining its fixed multi-scale hierarchy within every active pyramid level. While L dictates the spatial recursion depth across the macro-image pyramid based on input resolution, the three-scale extraction hierarchy (C_0, C_1, C_2) remains fixed and executes internally within every active pyramid layer l. This design ensures that multi-frequency spatial contexts are systematically extracted at each pyramid tier regardless of the total depth L. The shared

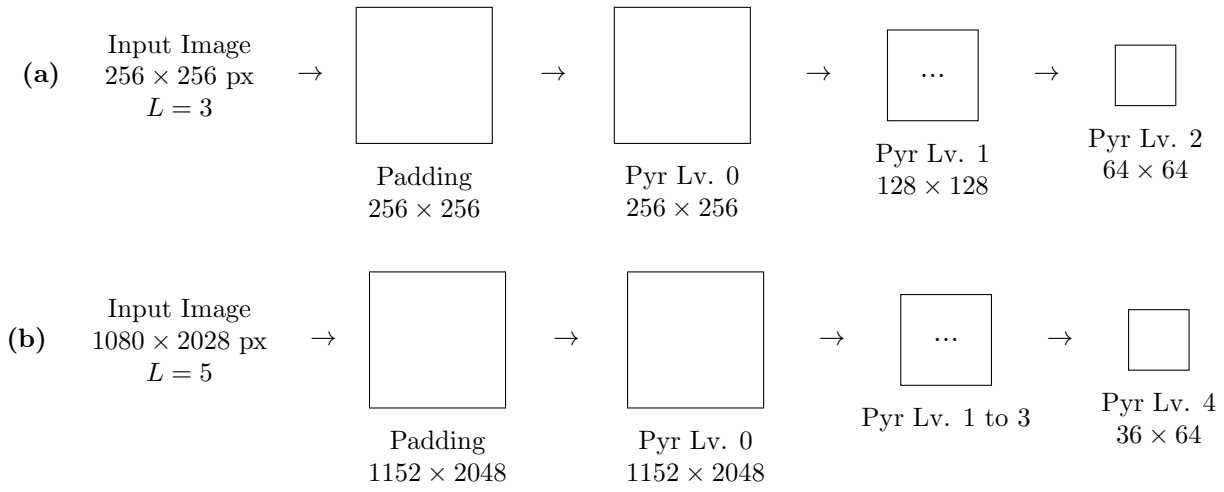


Figure 4.3: Illustration of the adaptive input image pyramid construction with a pre-padding mechanism across (a) standard low-resolution and (b) non-standard high-definition resolution boundaries.

encoder configuration and its internal execution steps are detailed in Fig. 4.4 and formally structured in Algorithm 4.3.

As illustrated in Fig. 4.4(a) and detailed in Algorithm 4.3, the backbone projects an input image I^l through an initial 3×3 convolutional layer (Conv_{in}) and a sequence of residual blocks (B_0, B_1, B_2) to extract a three-scale feature pyramid $\mathbf{F}^{l,x} = \{F^{l,x,0}, F^{l,x,1}, F^{l,x,2}\}$, representing dense ($C_0$), mid-level ($C_1$), and coarse abstract resolutions (C_2):

$$F^{l,x,0} = B_0\big(\text{Conv}_{\text{in}}(I^l)\big) \tag{4.3.1}$$

$$F^{l,x,1} = B_1\big(F^{l,x,0}\big) \tag{4.3.2}$$

$$F^{l,x,2} = \text{Conv}_{\text{out}}\Big(B_2\big(F^{l,x,1}\big)\Big) \tag{4.3.3}$$

where $x \in \{m, c\}$ distinguishes between the motion extraction (m) and context extraction (c) modes.

The specific operational flow across the pyramid tiers in Fig. 4.4(a) proceeds as follows:

- **Finest Scale (C_0):** The input image $I^l \in \mathbb{R}^{3\times H\times W}$, where $H = H_0/2^l$, $W = W_0/2^l$ Algorithm 4.3, is first projected into a high-dimensional feature representation via the initial convolutional layer $\text{Conv}_{\text{in}}(\cdot)$, followed by a primary residual block $B_0(\cdot)$. The features are then passed through the first downsampling residual block $B_1(\cdot)$ to produce the finest-scale feature map $F^{l,x,0} \in \mathbb{R}^{C\times H\times W}$ as defined in Equation (4.3.1).

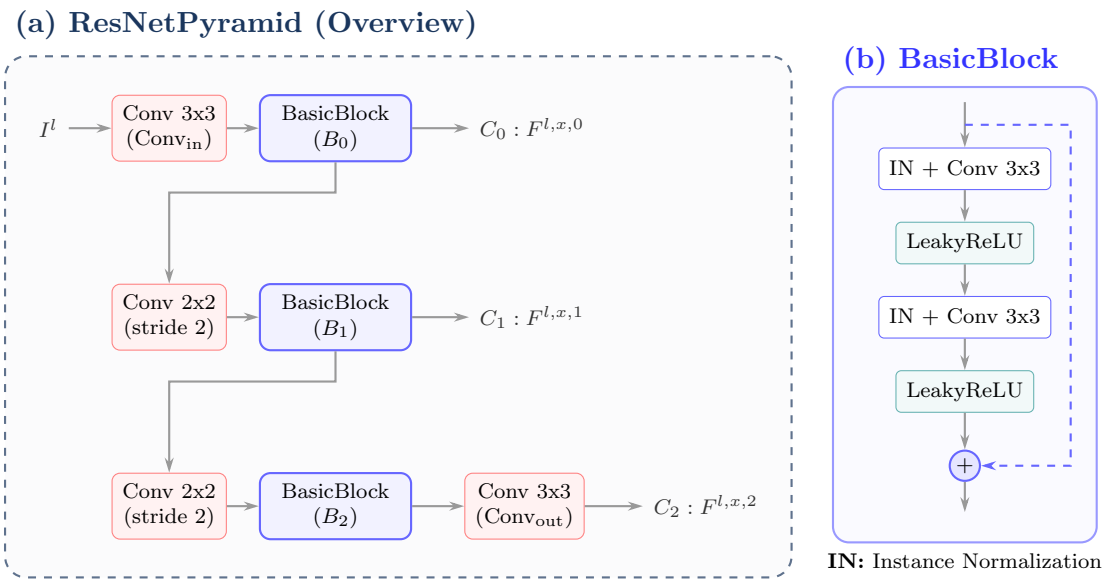


Figure 4.4: Multi-scale ResNetPyramid architecture and its constituent BasicBlock components.

Algorithm 4.3 Multi-Scale Feature Extraction via Shared ResNet Pyramid.

Input:
- I^l (input image at pyramid level l, where $I^l \in \mathbb{R}^{3\times H\times W}$, $H = H_0/2^l$, $W = W_0/2^l$) ▷ $H_0 \times W_0$: padded full input resolution; $l = 0, \ldots, L-1$

Output:
- $\mathbf{F}^{l,x} = \{F^{l,x,0}, F^{l,x,1}, F^{l,x,2}\}$ (3-scale feature pyramid, where $x \in \{m, c\}$)

Initialize Network: $\text{Extractor}_x \leftarrow \text{ResNetPyramid}()$ ▷ Instantiated for MFE or CFE

▷ **Feature Extraction Pipeline (C0, C1, C2 scales)**

$F^{l,x,0} \leftarrow B_0\big(\text{Conv}_{\text{in}}(I^l)\big)$ ▷ Extract finest scale (C_0), no downsampling

$F^{l,x,1} \leftarrow B_1(F^{l,x,0})$ ▷ Extract mid scale (C_1), B_1 downsamples $\times\frac{1}{2}$

$F^{l,x,2} \leftarrow \text{Conv}_{\text{out}}\Big(B_2(F^{l,x,1})\Big)$ ▷ Extract coarsest scale (C_2), B_2 downsamples $\times\frac{1}{2}$

Aggregate multi-scale feature pyramid: $\mathbf{F}^{l,x} \leftarrow \{F^{l,x,0}, F^{l,x,1}, F^{l,x,2}\}$

return $\mathbf{F}^{l,x}$

- **Mid Scale (C_1):** To expand the receptive field and capture intermediate spatial semantics, $F^{l,x,0}$ is downsampled through a 2×2 stride-2 convolution and refined by the shared residual block $B_1(\cdot)$, yielding the mid-scale feature map $F^{l,x,1} \in \mathbb{R}^{2C \times \frac{H}{2} \times \frac{W}{2}}$ as defined in Equation (4.3.2).

- **Coarsest Scale (C_2):** The mid-scale representation $F^{l,x,1}$ undergoes further spatial downsampling via a second 2×2 stride-2 convolution into the residual block $B_2(\cdot)$, followed by a final 3×3 convolutional layer (Conv_{out}). This produces the coarsest-scale feature map $F^{l,x,2} \in \mathbb{R}^{4C \times \frac{H}{4} \times \frac{W}{4}}$ as defined in Equation (4.3.3).

As shown in Fig. 4.4(b), each constituent BasicBlock (B_0, B_1, B_2) consists of two sequential 3×3 convolutional layers coupled with Instance Normalization (IN) and LeakyReLU activations. A residual skip connection bridges the input directly to the output addition node after the second LeakyReLU activation, guaranteeing stable gradient flow during end-to-end training.

This generalized pipeline is applied symmetrically across the source frames to construct the motion feature pyramids ($\mathbf{F}_0^{l,m}, \mathbf{F}_1^{l,m}$) and context feature pyramids ($\mathbf{F}_0^{l,c}, \mathbf{F}_1^{l,c}$). During training, it is additionally extended to extract the privileged teacher-side motion feature pyramid ($\mathbf{F}_t^{l,m}$) directly from the ground-truth intermediate frame I_t^l, ensuring subsequent tracking and synthesis modules dynamically ingest both high-resolution local details and wide-range global contexts.

4.4 Symmetrical Teacher-Student Distillation Framework

Inspired by the Knowledge Distillation for VFI-centric Flow supervision (KDVCF) strategy [18], our training framework deploys a symmetrical distillation paradigm comprising two parallel pathways at each pyramid level l: the Student Process ($\mathcal{P}_\mathcal{S}$) and the Teacher Process ($\mathcal{P}_\mathcal{T}$). While both pathways share identical network weights, they operate under asymmetrical temporal constraints: the teacher leverages the privileged ground-truth target frame (I_t) to compute accurate optical flows and motion guidance parameters (r, ϕ), which are subsequently distilled to supervise the student process.

4.4.1 Parameter Sharing: One Network, Two Information Regimes

Conventionally, knowledge distillation frameworks in video processing use a dual-network setup. This includes model compression for video frame interpolation (VFI) [83] and privileged flow/trajectory guidance [88, 89, 90]. The teacher network has its own parameter set Θ_{tea}, while the student network uses a distinct parameter set ($\Theta_{\text{stu}} \neq \Theta_{\text{tea}}$) [83, 88, 89, 90]. The teacher is often larger, pre-trained, or updated via Exponential Moving Average. In sharp contrast, our framework operates under a unified single-model paradigm: the teacher and student calls share one and the same set of parameters, with no separate teacher copy at all.

Specifically, let Θ denote the single set of trainable weights of the motion-and-synthesis backbone, comprising the Motion Feature Encoder MFE_Θ, the Context Feature Encoder CFE_Θ, the Motion Detection Network MD-Net_Θ, the Guided Flow Upsampling module GFU_Θ, and the Synthesis Network SN_Θ. Each of these five modules is instantiated *exactly once* for the whole model. As shown in Fig. 4.5, both $\mathcal{P}_\mathcal{T}$ and $\mathcal{P}_\mathcal{S}$ invoke the same module instances and underlying parameter tensors Θ at every layer l. The two pathways differ only in the *arguments* supplied to these shared functions, not in the functions themselves.

- $\mathcal{P}_\mathcal{T}$ additionally encodes the privileged frame, $\text{MFE}_\Theta(I_t)$, and conditions MD-Net_Θ on (I_0, I_t, I_1) to recover ground-truth-consistent bidirectional trajectories, yielding the teacher output $\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{T}}_{t\to 0,1}$.
- $\mathcal{P}_\mathcal{S}$ never receives I_t; it conditions the same MD-Net_Θ on (I_0, I_1) together with the motion guidance parameters (r^l, ϕ^l) distilled from $\mathcal{P}_\mathcal{T}$ at that layer, producing the student motion field $\mathbf{M}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1}$ and, ultimately, the student output $\widetilde{\mathbf{V}}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1}$.

The two pathways are connected through the detached motion guidance (r^l, ϕ^l). The stop-gradient is applied when these quantities leave $\mathcal{P}_\mathcal{T}$ and before they enter $\mathcal{P}_\mathcal{S}$. Similarly, the detached teacher flow is used as the target for $\mathcal{L} * \text{distill}$. This prevents gradients from $\mathcal{L}_{\text{student}}$ and $\mathcal{L}_{\text{distill}}$ from flowing through the teacher pathway. However, both $\mathcal{L}_{\text{teacher}}$ and $\mathcal{L}_{\text{student}} + \mathcal{L}_{\text{distill}}$ still update the shared parameters Θ through their respective forward passes. Thus, the framework uses a single shared-parameter network with separate gradient paths, requiring no parameter copying, synchronization, or Exponential Moving Average updates.

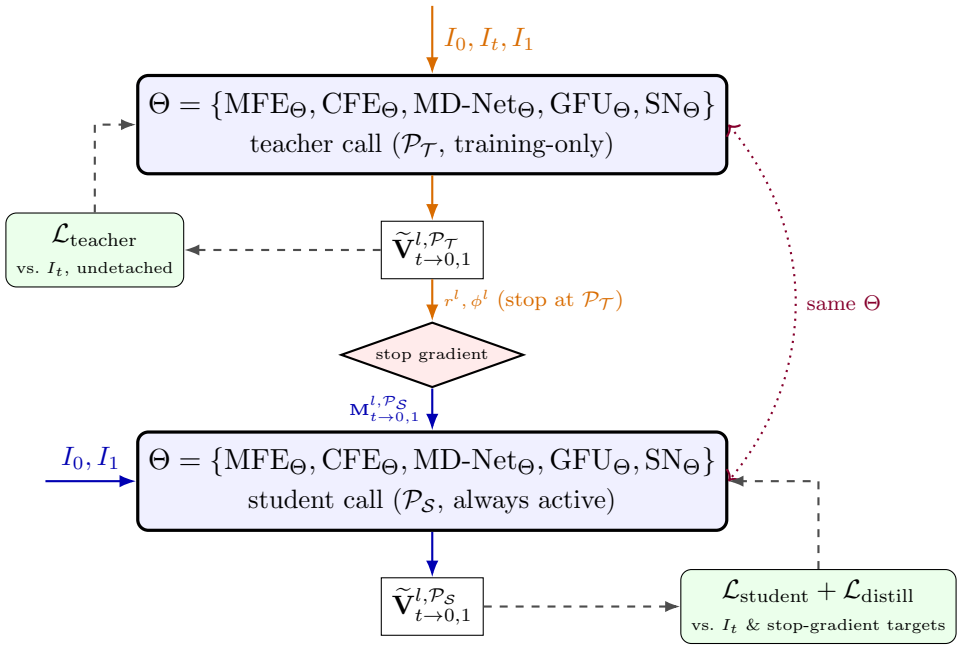


Figure 4.5: Shared-parameter teacher–student framework with a single parameter set Θ.

4.4.2 Asymmetrical Operational Roles

The student process ($\mathcal{P}_\mathcal{S}$) simulates the inference environment, receiving only the adjacent anchor key frames (I_0 and I_1). It must independently estimate bidirectional motion fields and synthesize the intermediate frame $\hat{I}^{l,\mathcal{P}_\mathcal{S}}$ without structural cues from the target timestamp. Conversely, the teacher process ($\mathcal{P}_\mathcal{T}$) acts as an asymmetric temporal oracle active exclusively during training. Leveraging the ground-truth intermediate frame (I_t), the teacher establishes a motion-consistent reference pathway to guide the student's optimization. At inference, the teacher pathway is completely deactivated, avoiding computational overhead or reliance on privileged information at test time.

4.4.3 Knowledge Transfer Channels

The teacher process guides the optimization of the deployable student via two distinct one-way channels:

- **Privileged Motion Guidance ($\mathbf{M}^{l,\mathcal{P}_\mathcal{S}}$):** The teacher decomposes the motion estimation into two unidirectional passes ($I_t \to I_0$ and $I_t \to I_1$). Since the search

originates from the actual target frame, it yields highly stable local tracking. The extracted velocity scaling ratio r and direction-aware angular attributes ϕ are bundled into the motion guidance tensor $\mathbf{M}^{l,\mathcal{P}_S}$ and injected into the student's motion detection module. Gradients are detached (detach$(\cdot)$) at the teacher boundary to prevent student state corruption.

- **Residual Distillation Loss ($\mathcal{L}_{\text{distill}}$):** During the backward pass, a dedicated distillation loss penalizes spatial discrepancies between the student's bidirectional predicted motion residuals and the teacher's stable unidirectional residuals. The teacher effectively provides a dense motion target sequence derived from I_t. During training, this sequence is used to supervise the student network. The goal is to allow the student to approximate the teacher's latent motion distribution during inference.

4.5 Motion Detection Network (MD-Net)

As the first primary technical contribution of this work, the proposed Motion Detection Network (MD-Net) serves as the localized motion estimation engine of our framework, active across both the student and teacher processes to decode bidirectional flow residuals and displacement fields at each active pyramid level l. As detailed in Section 3.9, MD-Net extends BiMFN [18] by integrating the prior $\mathbf{M}_{t\to 0,1}$ for residual flow decoding. Operating on a shared-weight architecture across both pathways with temporally asymmetrical feature inputs during training, MD-Net ingests the coarsest motion feature representations $F_i^{l,m,2} \in \mathbb{R}^{C\times\frac{H}{4}\times\frac{W}{4}}$ (extracted at scale C_2 of the MFE ResNet pyramid) to capture wide-range velocity fields. The granular mathematical procedures and parallel execution pathways of this network are formally defined for training in Algorithm 4.4 and for deployment in Algorithm 4.5.

The compact dataflow of this block under student evaluation is visually summarized in Fig. 4.6, which depicts the pipeline using inference-mode variables ($F_0^{l,m,2}$ and $F_1^{l,m,2}$). During training, the overall network structure remains identical, but accepts feature representations from temporal pairs $(F_t^{l,m,2}, F_0^{l,m,2})$ and $(F_t^{l,m,2}, F_1^{l,m,2})$ rather than solely between frame pairs.

First, prior to dense correspondence estimation, the coarse bidirectional flow prior propagated from the parent level, $\mathbf{V}_{t\to 0,1}^{l+1,\mathcal{P}_S}$, is resampled to the spatial resolution of level l through a plain, resolution-matched pyramid downsample: its spatial extent is halved via bilinear interpolation and its magnitude correspondingly scaled by a factor of 0.5 to remain consistent with the coarser pixel grid, applied identically and intact to both directions:

Algorithm 4.4 Motion Detection Network (Training Mode).

Input:
- $F_0^{l,m,2}, F_1^{l,m,2}$ (Motion features), $F_t^{l,m,2}$ (Teacher-side GT motion feature)
- $\mathbf{V}_{t\to 0,1}^{l+1,\mathcal{P}_\mathcal{S}}$ (Coarse student prior), $O^{l+1,\mathcal{P}_\mathcal{S}}$ (Coarse occlusion prior)
- $\mathbf{M}_{t\to 0,t}^{l,\mathcal{P}_\mathcal{T}}, \mathbf{M}_{t\to t,1}^{l,\mathcal{P}_\mathcal{T}}$ (Teacher perturbation priors)

Output: $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}$, and $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$ Low-resolution bidirectional flow

▷ **Phase I: Teacher Branches ($k \in \{0,1\}$: k=0 $\Rightarrow t\to 0$, k=1 $\Rightarrow t\to 1$; $\bar{k}$: opposite index)**

$\mathbf{V}_{t\to 0}^{l+1,\mathcal{P}_\mathcal{T}}, \mathbf{V}_{t\to 1}^{l+1,\mathcal{P}_\mathcal{T}}, O^l \leftarrow \text{Downsample}(\mathbf{V}_{t\to 0,1}^{l+1,\mathcal{P}_\mathcal{S}}, O^{l+1,\mathcal{P}_\mathcal{S}})$

$F_{k,\text{warp}}^{l,m,2} \leftarrow \mathcal{W}(F_k^{l,m,2}, \mathbf{V}_{t\to k}^{l+1,\mathcal{P}_\mathcal{T}})$; $\quad F_{t,\text{warp},k}^{l,m,2} \leftarrow F_t^{l,m,2}$ ▷ $k=0,1$; F_t unwarped: already at t

$\hat{F}_{k,\text{warp}}^{l,m,2} \leftarrow \text{SE}(F_{k,\text{warp}}^{l,m,2})$; $\quad \hat{F}_{t,\text{warp},k}^{l,m,2} \leftarrow \text{SE}(F_{t,\text{warp},k}^{l,m,2})$

$\mathbf{C}_0^{\mathcal{P}_\mathcal{T}} \leftarrow \text{CostVolume}(\hat{F}_{0,\text{warp}}^{l,m,2}, \hat{F}_{t,\text{warp},0}^{l,m,2}, \text{rad}=4)$; $\quad \mathbf{C}_1^{\mathcal{P}_\mathcal{T}} \leftarrow \text{CostVolume}(\hat{F}_{t,\text{warp},1}^{l,m,2}, \hat{F}_{1,\text{warp}}^{l,m,2}, \text{rad}=4)$

$\mathbf{C}_{\bar{k}}^{\mathcal{P}_\mathcal{T}} \leftarrow \text{CostVolume}(\hat{F}_{t,\text{warp},k}^{l,m,2}, \hat{F}_{k,\text{warp}}^{l,m,2}, \text{rad}=4)$ ▷ reverse-direction term; unused by design, kept for shared FlowHead

$\hat{\mathbf{C}}_k^{\mathcal{P}_\mathcal{T}}, \hat{\mathbf{C}}_{\bar{k}}^{\mathcal{P}_\mathcal{T}} \leftarrow \text{Conv}_{\text{corr}}(\mathbf{C}_k^{\mathcal{P}_\mathcal{T}}), \text{Conv}_{\text{corr}}(\mathbf{C}_{\bar{k}}^{\mathcal{P}_\mathcal{T}})$

$\mathbf{H}_{\text{flow},k}^{\mathcal{P}_\mathcal{T}} \leftarrow \text{Conv}_{\text{flow}}(\mathbf{V}_{t\to k}^{l+1,\mathcal{P}_\mathcal{T}})$; $\quad \mathbf{H}_{\text{flow},\bar{k}}^{\mathcal{P}_\mathcal{T}} \leftarrow \text{Conv}_{\text{flow}}(\mathbf{0})$; $\quad \mathbf{H}_{\text{occ},k}^{\mathcal{P}_\mathcal{T}} \leftarrow \text{Conv}_{\text{occ}}(O^l)$

$\mathbf{X}_{\text{concat},k} \leftarrow [\hat{\mathbf{C}}_k^{\mathcal{P}_\mathcal{T}} \parallel \hat{F}_{k,\text{warp}}^{l,m,2} \parallel \mathbf{H}_{\text{flow},\bar{k}}^{\mathcal{P}_\mathcal{T}} \parallel \mathbf{H}_{\text{occ},k}^{\mathcal{P}_\mathcal{T}}]$

$\widetilde{\mathbf{V}}_{t\to k}^{l,\mathcal{P}_\mathcal{T}}, \Delta\mathbf{V}_{t\to k}^{l,\mathcal{P}_\mathcal{T}} \leftarrow \text{FlowHead}(\mathbf{X}_{\text{concat},k}, \mathbf{M}_{t\to k}^{l,\mathcal{P}_\mathcal{T}})$ ▷ combine $\to \widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$

▷ **Phase II: Symmetrical Student Process**

$r \leftarrow \dfrac{\|\Delta\mathbf{V}_{t\to 0}^{l,\mathcal{P}_\mathcal{T}}\|}{\|\Delta\mathbf{V}_{t\to 0}^{l,\mathcal{P}_\mathcal{T}}\| + \|\Delta\mathbf{V}_{t\to 1}^{l,\mathcal{P}_\mathcal{T}}\|}$; $\quad \phi \leftarrow [\cos\theta, \sin\theta]$ ▷ $\theta = \angle(\Delta\mathbf{V}_{t\to 0}^{l,\mathcal{P}_\mathcal{T}}, \Delta\mathbf{V}_{t\to 1}^{l,\mathcal{P}_\mathcal{T}})$; grad. blocked

$F_{0,\text{warp}}^{l,m,2} \leftarrow \mathcal{W}(F_0^{l,m,2}, \mathbf{V}_{t\to 0}^{l+1,\mathcal{P}_\mathcal{T}})$; $\quad F_{1,\text{warp}}^{l,m,2} \leftarrow \mathcal{W}(F_1^{l,m,2}, \mathbf{V}_{t\to 1}^{l+1,\mathcal{P}_\mathcal{T}})$

$\hat{F}_{0,\text{warp}}^{l,m,2} \leftarrow \text{SE}(F_{0,\text{warp}}^{l,m,2})$; $\quad \hat{F}_{1,\text{warp}}^{l,m,2} \leftarrow \text{SE}(F_{1,\text{warp}}^{l,m,2})$

$\mathbf{C}_0^{\mathcal{P}_\mathcal{S}} \leftarrow \text{CostVolume}(\hat{F}_{0,\text{warp}}^{l,m,2}, \hat{F}_{1,\text{warp}}^{l,m,2}, \text{rad}=4)$; $\quad \mathbf{C}_1^{\mathcal{P}_\mathcal{S}} \leftarrow \text{CostVolume}(\hat{F}_{1,\text{warp}}^{l,m,2}, \hat{F}_{0,\text{warp}}^{l,m,2}, \text{rad}=4)$

$\hat{\mathbf{C}}_j^{\mathcal{P}_\mathcal{S}} \leftarrow \text{Conv}_{\text{corr}}(\mathbf{C}_j^{\mathcal{P}_\mathcal{S}})$; $\quad \mathbf{H}_{\text{flow},j}^{\mathcal{P}_\mathcal{S}} \leftarrow \text{Conv}_{\text{flow}}(\mathbf{V}_{t\to j}^{l+1,\mathcal{P}_\mathcal{T}})$; $\quad \mathbf{H}_{\text{occ}}^{\mathcal{P}_\mathcal{S}} \leftarrow \text{Conv}_{\text{occ}}(O^{l+1,\mathcal{P}_\mathcal{S}})$ ▷ $j=0,1$

$\mathbf{X}_{\text{concat}} \leftarrow [\hat{\mathbf{C}}_0^{\mathcal{P}_\mathcal{S}} \parallel \hat{\mathbf{C}}_1^{\mathcal{P}_\mathcal{S}} \parallel \hat{F}_{0,\text{warp}}^{l,m,2} \parallel \hat{F}_{1,\text{warp}}^{l,m,2} \parallel \mathbf{H}_{\text{flow},0}^{\mathcal{P}_\mathcal{S}} \parallel \mathbf{H}_{\text{flow},1}^{\mathcal{P}_\mathcal{S}} \parallel \mathbf{H}_{\text{occ}}^{\mathcal{P}_\mathcal{S}}]$

$\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}, \mathbf{V}_{\text{res}}^l \leftarrow \text{FlowHead}(\mathbf{X}_{\text{concat}}, r, \phi)$

return $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}$, $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{T}}$

Algorithm 4.5 Motion Detection Network (Inference Mode).

Input:
- $F_0^{l,m,2}, F_1^{l,m,2}$ (Motion features)
- $\mathbf{V}_{t\to 0,1}^{l+1,\mathcal{P}_S}$ (Coarse student prior flow), $O^{l+1,\mathcal{P}_S}$ (Coarse student prior occlusion)
- $\mathbf{M}_{t\to 0,1}^{\text{linear}}$ (Constant-velocity baseline prior)

Output:
- $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_S}$ (Low-resolution bidirectional flow output)

Extract $r \leftarrow t$ from $\mathbf{M}_{t\to 0,1}^{\text{linear}}$ ▷ Inference straight-line assumption
$\tilde{\mathbf{V}}_{t\to 0}^{l} \leftarrow -r \cdot \mathbf{V}_{t\to 0}^{l+1,\mathcal{P}_S}$; $\tilde{\mathbf{V}}_{t\to 1}^{l} \leftarrow (1-r) \cdot \mathbf{V}_{t\to 1}^{l+1,\mathcal{P}_S}$ ▷ Scale course priors
$F_{0,\text{warp}}^{l,m,2} \leftarrow \mathcal{W}(F_0^{l,m,2}, \tilde{\mathbf{V}}_{t\to 0}^{l})$; $F_{1,\text{warp}}^{l,m,2} \leftarrow \mathcal{W}(F_1^{l,m,2}, \tilde{\mathbf{V}}_{t\to 1}^{l})$ ▷ Warp features toward t
$\hat{F}_{0,\text{warp}}^{l,m,2} \leftarrow \text{SE}(F_{0,\text{warp}}^{l,m,2})$; $\hat{F}_{1,\text{warp}}^{l,m,2} \leftarrow \text{SE}(F_{1,\text{warp}}^{l,m,2})$ ▷ Channel attention via SEBlock
$\mathbf{C}_0^{\mathcal{P}_S} \leftarrow \text{CostVolume}(\hat{F}_{0,\text{warp}}^{l,m,2}, \hat{F}_{1,\text{warp}}^{l,m,2}, \text{radius} = 4)$
$\mathbf{C}_1^{\mathcal{P}_S} \leftarrow \text{CostVolume}(\hat{F}_{1,\text{warp}}^{l,m,2}, \hat{F}_{0,\text{warp}}^{l,m,2}, \text{radius} = 4)$
$\hat{\mathbf{C}}_0^{\mathcal{P}_S} \leftarrow \text{Conv}_{\text{corr}}(\mathbf{C}_0^{\mathcal{P}_S})$; $\hat{\mathbf{C}}_1^{\mathcal{P}_S} \leftarrow \text{Conv}_{\text{corr}}(\mathbf{C}_1^{\mathcal{P}_S})$
$\mathbf{H}_{\text{flow},0}^{\mathcal{P}_S} \leftarrow \text{Conv}_{\text{flow}}(\tilde{\mathbf{V}}_{t\to 0}^{l})$; $\mathbf{H}_{\text{flow},1}^{\mathcal{P}_S} \leftarrow \text{Conv}_{\text{flow}}(\tilde{\mathbf{V}}_{t\to 1}^{l})$; $\mathbf{H}_{\text{occ}}^{\mathcal{P}_S} \leftarrow \text{Conv}_{\text{occ}}(O^{l+1,\mathcal{P}_S})$
$\mathbf{X}_{\text{concat}} \leftarrow [\hat{\mathbf{C}}_0^{\mathcal{P}_S} \parallel \hat{\mathbf{C}}_1^{\mathcal{P}_S} \parallel \hat{F}_{0,\text{warp}}^{l,m,2} \parallel \hat{F}_{1,\text{warp}}^{l,m,2} \parallel \mathbf{H}_{\text{flow},0}^{\mathcal{P}_S} \parallel \mathbf{H}_{\text{flow},1}^{\mathcal{P}_S} \parallel \mathbf{H}_{\text{occ}}^{\mathcal{P}_S}]$
$\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_S}, \mathbf{V}_{\text{res}}^{l} \leftarrow \text{FlowHead}(\mathbf{X}_{\text{concat}})$
return $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_S}$

$$\tilde{\mathbf{V}}_{t\to 0}^{l} = 0.5 \cdot \text{Downsample}\Big(\mathbf{V}_{t\to 0}^{l+1,\mathcal{P}_S},\, 0.5\Big) \tag{4.5.1}$$

$$\tilde{\mathbf{V}}_{t\to 1}^{l} = 0.5 \cdot \text{Downsample}\Big(\mathbf{V}_{t\to 1}^{l+1,\mathcal{P}_S},\, 0.5\Big) \tag{4.5.2}$$

Notably, this resolution adjustment is purely geometric. It is independent of the target timestamp t and is not yet modulated by the motion guidance pair (r, ϕ). During training, (r, ϕ) is derived from distillation-based teacher guidance. During inference, it is fixed to the constant-velocity baseline ($r = t$, $\phi = \pi$). This pair is reserved for the later BiM-guided modulation stage, after correspondence estimation. The source motion features $F_0^{l,m,2}$ and $F_1^{l,m,2}$ are mapped toward the target timestamp via the bilinear backward-warping operator ($\mathcal{W}$), using the corresponding half of the downsampled prior, yielding the warped representations $F_{0,\text{warp}}^{l,m,2} = \mathcal{W}(F_0^{l,m,2}, \tilde{\mathbf{V}}_{t\to 0}^{l})$ and $F_{1,\text{warp}}^{l,m,2} = \mathcal{W}(F_1^{l,m,2}, \tilde{\mathbf{V}}_{t\to 1}^{l})$. These features are subsequently recalibrated through a channel-wise Squeeze-and-Excitation (SE) attention block, producing the refined motion representations $\hat{F}_{0,\text{warp}}^{l,m,2}$ and $\hat{F}_{1,\text{warp}}^{l,m,2}$.

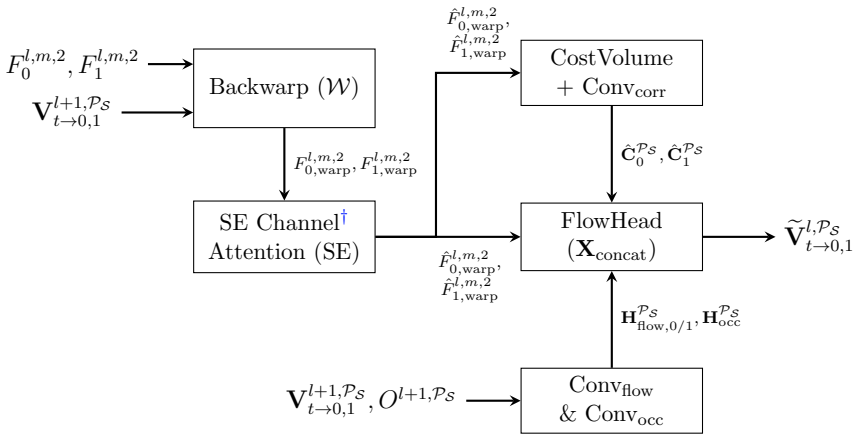


Figure 4.6: Architectural diagram of MD-Net. †In this work, the baseline motion estimation module of BiM-VFI is augmented with Squeeze-and-Excitation (SE) channel attention [87].

Second, to construct the localized cost volumes, the attention-refined features ($\hat{F}_{0,\text{warp}}^{l,m,2}$ and $\hat{F}_{1,\text{warp}}^{l,m,2}$) are first normalized along the channel dimension using L_2 normalization to ensure numerical stability. Subsequently, a localized spatial correlation search is conducted within a fixed window defined by a maximum search radius $d = 4$. This establishes a square search grid of dimensions $(2d+1) \times (2d+1) = 9 \times 9$ pixels. Symmetrical cost volumes $\mathbf{C}_0$ and $\mathbf{C}_1$ are computed via inner-product matching over this neighborhood:

$$\mathbf{C}_0(\mathbf{x}, \mathbf{d}) = \langle \hat{F}_{0,\text{warp}}^{l,m,2}(\mathbf{x}),\ \hat{F}_{1,\text{warp}}^{l,m,2}(\mathbf{x}+\mathbf{d}) \rangle \tag{4.5.3}$$

$$\mathbf{C}_1(\mathbf{x}, \mathbf{d}) = \langle \hat{F}_{1,\text{warp}}^{l,m,2}(\mathbf{x}),\ \hat{F}_{0,\text{warp}}^{l,m,2}(\mathbf{x}+\mathbf{d}) \rangle \tag{4.5.4}$$

where $\mathbf{x} = (x, y)$ denotes the absolute 2D spatial pixel coordinate on the feature map grid, $d = 4$ specifies the scalar search radius, and $\mathbf{d} = (d_x, d_y) \in \{-d, \ldots, d\}^2 = \{-4, \ldots, 4\}^2$ represents the relative 2D spatial displacement offset vector (where d_x and d_y are horizontal and vertical pixel shifts relative to the anchor coordinate $\mathbf{x}$).

As schematically illustrated in Fig. 4.7, this localized correspondence operation explicitly maps the 2D spatial search neighborhood onto the channel dimension of the correlation tensor. Each discrete displacement pair (d_x, d_y) is uniquely index-mapped to a specific channel C_k using a row-major raster layout:

$$k = (d_y + d) \times (2d + 1) + (d_x + d) \tag{4.5.5}$$

yielding $k \in \{0, 1, \ldots, 80\}$ channels. Here, C_0 corresponds to the maximum negative displacement $(-d, -d) = (-4, -4)$, while C_{80} represents the maximum positive displacement

d_y \ d_x	−4	−3	−2	−1	0	1	2	3	4
−4	C0	C1	C2	C3	C4	C5	C6	C7	C8
−3	C9	C10	C11	C12	C13	C14	C15	C16	C17
−2	C18	C19	C20	C21	C22	C23	C24	C25	C26
−1	C27	C28	C29	C30	C31	C32	C33	C34	C35
0	C36	C37	C38	C39	C40	C41	C42	C43	C44
1	C45	C46	C47	C48	C49	C50	C51	C52	C53
2	C54	C55	C56	C57	C58	C59	C60	C61	C62
3	C63	C64	C65	C66	C67	C68	C69	C70	C71
4	C72	C73	C74	C75	C76	C77	C78	C79	C80

Figure 4.7: Structural mapping of the 2D neighborhood displacement search space onto the 81-channel cost volume architecture (9×9 grid, where $d_x, d_y \in [-4, 4]$). The zero-displacement vector maps directly onto the central anchor channel C_{40}.

$(d, d) = (4, 4)$. The center channel C_{40} explicitly captures the zero-displacement anchor $(d_x, d_y) = (0, 0)$, signifying static or structurally invariant regions between the warped representations.

For example, computing the index for channel C_{40}: substituting $(d_x, d_y) = (0, 0)$ and $d = 4$ into the indexing formula yields

$$k = (0 + 4) \times (2 \times 4 + 1) + (0 + 4) = 4 \times 9 + 4 = 40, \tag{4.5.6}$$

confirming that the zero-displacement vector is mapped to the central channel C_{40}, consistent with Fig. 4.7.

These 81 correspondence channels are subsequently processed through a 1×1 convolutional layer ($\text{Conv}_{\text{corr}}$) to compress the spatial correspondence maps into condensed

representations $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$, allowing the network to internalize non-linear motion vectors from the structural activation profiles across the tensor depth. Finally, to compute the final displacement fields, the compressed correlation maps, attention-refined warped features, and projected coarser-level priors ($\mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{flow},0}, \mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{flow},1}, \mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{occ}}$) are concatenated into a unified motion-context tensor:

$$\mathbf{X}_{\text{concat}} = \left[\hat{\mathbf{C}}^{\mathcal{P}_\mathcal{S}}_0 \parallel \hat{\mathbf{C}}^{\mathcal{P}_\mathcal{S}}_1 \parallel \hat{F}^{l,m,2}_{0,\text{warp}} \parallel \hat{F}^{l,m,2}_{1,\text{warp}} \parallel \mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{flow},0} \parallel \mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{flow},1} \parallel \mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{occ}}\right] \tag{4.5.7}$$

where $\mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{flow},k} = \text{Conv}_{\text{flow}}(\tilde{\mathbf{V}}^l_{t\to k})$ and $\mathbf{H}^{\mathcal{P}_\mathcal{S}}_{\text{occ}} = \text{Conv}_{\text{occ}}(O^{l+1,\mathcal{P}_\mathcal{S}})$.

The tensor $\mathbf{X}_{\text{concat}}$ is decoded by a stacked convolutional block (FlowHead) with residual connections to output the low-resolution flow $\widetilde{\mathbf{V}}^l$ and the bidirectional residual correction $\mathbf{V}^l_{\text{res}}$, which updates the motion state via: $\mathbf{V}^{l,\mathcal{P}_\mathcal{S}}_{t\to 0,1} = \tilde{\mathbf{V}}^l_{t\to 0,1} + \mathbf{V}^l_{\text{res}}$

The internal tensors, mapping operators, and structural embedding variables instantiated throughout the execution of MD-Net (Algorithm 4.4 and Algorithm 4.5) are formally defined as follows:

- $\tilde{\mathbf{V}}^l_{t\to 0}, \tilde{\mathbf{V}}^l_{t\to 1}$ represent the temporally aligned coarse prior flow fields scaled relative to timestamp t via the velocity ratio r.
- $\mathcal{W}(F, \mathbf{V})$ denotes the backward-warping operator that aligns feature map F using optical flow field $\mathbf{V}$ via bilinear grid sampling.
- $F^{l,m,2}_{0,\text{warp}}, F^{l,m,2}_{1,\text{warp}}$ designate the coarsely warped motion feature maps mapped toward temporal position t prior to attention recalibration.
- $\hat{F}^{l,m,2}_{0,\text{warp}}, \hat{F}^{l,m,2}_{1,\text{warp}}$ identify the attention-refined warped motion features recalibrated via the channel-wise Squeeze-and-Excitation (SE) block.
- $\mathbf{x} = (x, y)$ denotes the 2D spatial pixel coordinate on the feature map grid.
- $\mathbf{d} = (d_x, d_y) \in [-d, d]^2$ represents the 2D spatial displacement vector constrained within a search radius $d = 4$ (generating 81 correlation channels).
- $\mathbf{C}^{\mathcal{P}_\mathcal{S}}_0, \mathbf{C}^{\mathcal{P}_\mathcal{S}}_1$ specify the localized 81-channel spatial cost volumes constructed via inner-product matching between student warped features.
- $\mathbf{C}^{\mathcal{P}_\mathcal{T}}_k$ represents the corresponding single-direction localized cost volume generated within branch $k \in \{0, 1\}$ of the teacher process ($\mathcal{P}_\mathcal{T}$).

- $\text{Conv}_{\text{corr}}(\cdot), \text{Conv}_{\text{flow}}(\cdot), \text{Conv}_{\text{occ}}(\cdot)$ denote 1×1 convolutional projection layers that project correlation maps, coarse flows, and occlusion masks into unified feature channels, respectively.
- $\hat{\mathbf{C}}_0, \hat{\mathbf{C}}_1$ identify the compressed spatial correlation embeddings generated via $\text{Conv}_{\text{corr}}$.
- $\mathbf{H}_{\text{flow},k}, \mathbf{H}_{\text{occ}}$ represent the projected feature representations of coarse optical flows and prior occlusion masks, respectively.
- $\mathbf{X}_{\text{concat}}$ denotes the composite multi-source feature tensor formed by concatenating correlation embeddings, warped features, projected flows, and occlusion priors along the channel dimension.
- FlowHead$(\cdot)$ designates the stacked convolutional predictor network responsible for estimating low-resolution flow fields and residual refinements.
- $\widetilde{\mathbf{V}}_{t\to 0,1}^{l,\mathcal{P}_\mathcal{S}}$ defines the low-resolution bidirectional flow tensor estimated by MD-Net for student processing.
- $\mathbf{V}_{\text{res}}^{l}$ specifies the localized residual flow correction tensor estimated to refine coarse trajectory projections.
- $\Delta\mathbf{V}_{t\to k}^{l,\mathcal{P}_\mathcal{T}}$ represents the residual flow vector predicted by teacher branch k, utilized for calculating distillation guidance parameters (r, ϕ).

4.5.1 Channel-wise Feature Recalibration via SE Attention

Unlike the baseline BiMFN from BiM-VFI [18], which feeds warped motion features $F_{0,\text{warp}}^{l,m,2}$ and $F_{1,\text{warp}}^{l,m,2}$ directly into correspondence matching, MD-Net inserts a shared Squeeze-and-Excitation (SE) Block [87] prior to this step, recalibrating each feature channel to suppress uninformative responses while enhancing discriminative tracking primitives.

The detailed computation is illustrated in Fig. 4.8. First, a global average pooling operation squeezes the spatial dimensions $(H \times W)$ of an input tensor into a global channel descriptor vector $\mathbf{z} \in \mathbb{R}^{C\times 1\times 1}$. This operation is applied independently, with shared weights, to each warped stream $k \in \{0, 1\}$ (i.e., $k = 0; F_{0,\text{warp}}^{l,m,2}$ and $k = 1; F_{1,\text{warp}}^{l,m,2}$):

$$\mathbf{z} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} F_{k,\text{warp}}^{l,m,2}(i, j) \tag{4.5.8}$$

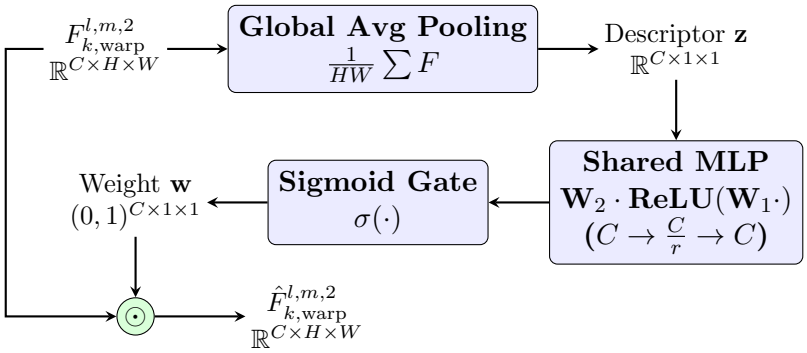


Figure 4.8: Architectural implementation of the Squeeze-and-Excitation (SE) Block. The input warped motion feature $F^{l,m,2}_{k,\text{warp}}$ is spatially squeezed, mapped through a Multi-Layer Perceptron (MLP) with Sigmoid activation to generate channel weights $\mathbf{w}$, and multiplied element-wise to yield the recalibrated feature $\hat{F}^{l,m,2}_{k,\text{warp}}$.

Second, to model non-linear inter-channel dependencies, $\mathbf{z}$ is processed through a bottleneck MLP with a reduction ratio $r = 4$ and gated by a Sigmoid activation $\sigma(\cdot)$, yielding a continuous channel weight vector $\mathbf{w} \in (0,1)^{C\times 1\times 1}$:

$$\mathbf{w} = \sigma\left(\mathbf{W}_2 \cdot \max(0, \mathbf{W}_1 \cdot \mathbf{z})\right) \tag{4.5.9}$$

where $\cdot$ denotes matrix multiplication. $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $\mathbf{W}2 \in \mathbb{R}^{C\times\frac{C}{r}}$ are the learnable down- and up-projection matrices, respectively, with $r = 4$ specifying the channel reduction ratio. The final attention-calibrated feature map $\hat{F}k, \text{warp}^{l,m,2}$ is produced via channel-wise Hadamard product (element-wise multiplication denoted by $\odot$):

$$\hat{F}^{l,m,2}_{k,\text{warp}} = F^{l,m,2}_{k,\text{warp}} \odot \mathbf{w} \tag{4.5.10}$$

Theoretically, this feature recalibration operates as an adaptive signal filter conditioned on spatial information density. Channels exhibiting high structural energy—such as high-frequency edges, distinctive textures, and motion boundaries—receive higher attention weights (**close to 1**) to preserve critical tracking primitives. Conversely, channels dominated by flat lighting or uninformative noise are suppressed with weights **close to 0**. By filtering out these non-discriminative channels prior to localized spatial correlation, the SE module reduces matching ambiguity and sharpens inner-product correlation peaks within the cost volume.

4.6 Guided Flow Upsampling (GFU)

As the second primary technical contribution, the proposed Guided Flow Upsampling (GFU) module projects the low-resolution motion fields estimated by MD-Net back to high-resolution spatial pixel coordinates using a context-guided convex upsampling mechanism. Conceptually grounded in the Content-Aware Upsampling Network (CAUN) of BiM-VFI proposed by Seo et al. [18], GFU adopts a multi-scale flow representation $(\mathbf{V}^l, \mathbf{V}^{l\times 0.5}, \widetilde{\mathbf{V}}^l)$ to preserve fine motion details and sharp spatial boundaries. To prevent edge-blurring and motion bleeding along geometric boundaries, the interpolation is conditioned exclusively on the low-resolution bidirectional flow $\widetilde{\mathbf{V}}^l_{t\to 0,1}$ and the finest-scale contextual features $(F_0^{l,c,0}, F_1^{l,c,0})$ from the C_0 scale of the Context Feature Extractor (**CFE**); the privileged intermediate frame (I_t) is excluded, ensuring identical behavior at training and inference.

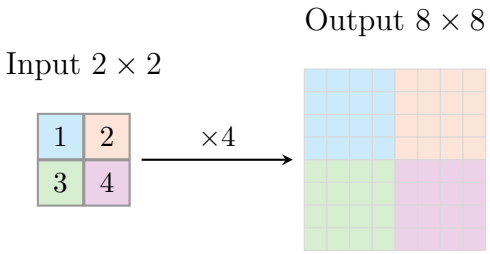


Figure 4.9: Overview of spatial mapping between input pixels and the upsampled output grid (×4 scaling).

Specifically, GFU systematically applies a fixed spatial scaling factor of $s = 4$ across all pyramid levels. As illustrated in Fig. 4.9, each low-resolution input pixel is expanded into a corresponding 4×4 block within the high-resolution output grid, with color coding indicating the source-to-output pixel correspondence. Crucially, to preserve physical displacement consistency on the expanded spatial coordinate grid, all interpolated flow vectors undergo mandatory magnitude rescaling multiplied strictly by the same scaling factor (×4).

The detailed pixel synthesis mechanism is illustrated in Fig. 4.10. For each target sub-pixel position, GFU extracts a 3×3 neighboring source region and applies a unique, position-specific convex mask. The Guidance Network outputs 16 distinct 3×3 normalized interpolation masks per direction, allowing every output pixel to be synthesized via a weighted convex combination over its localized neighborhood.

Algorithm 4.6 details the full routine (with the overall architecture illustrated in Fig. 4.11); when a spatial mismatch exists, context features are first resized to align with the flow grid.

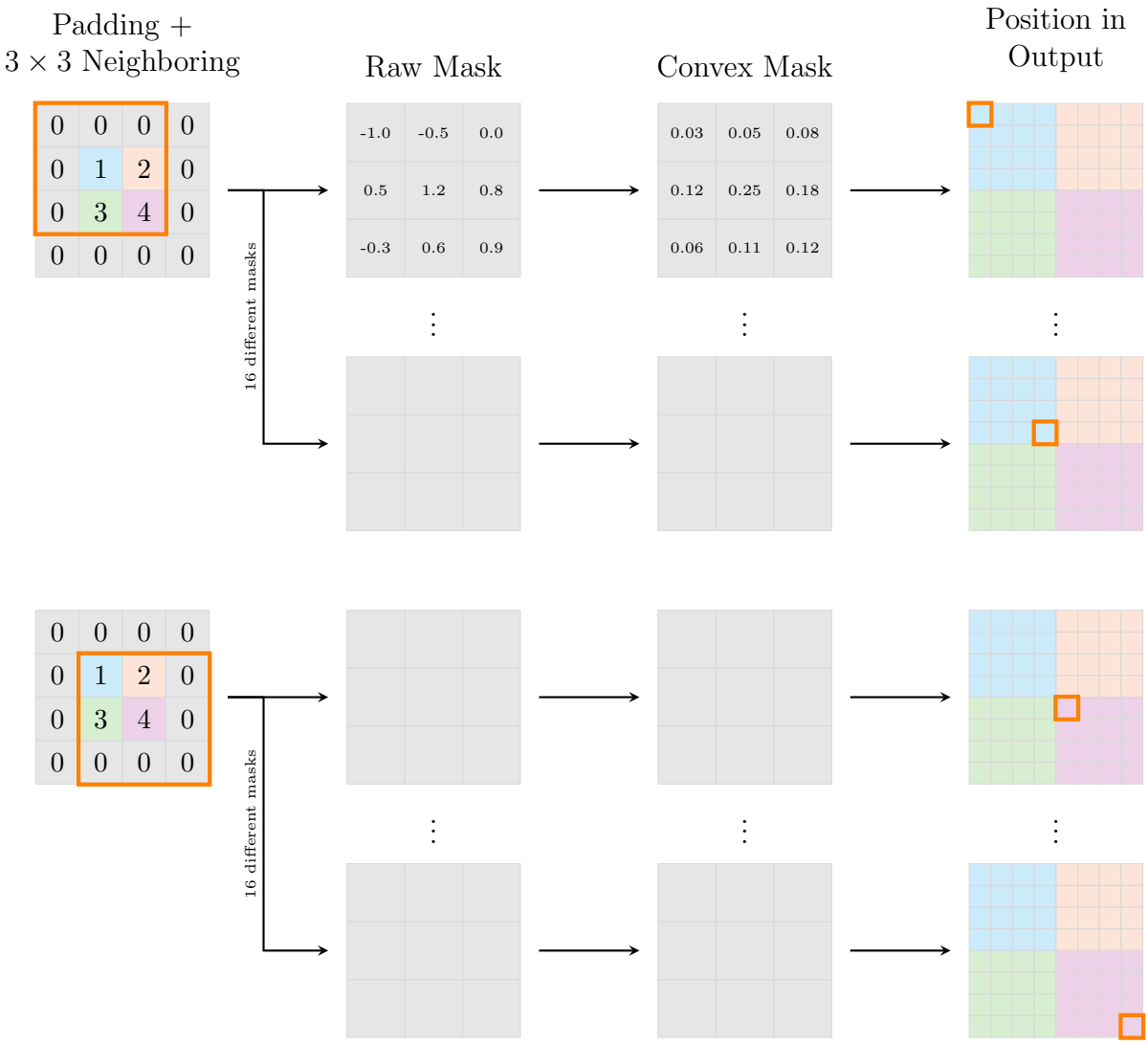


Figure 4.10: Illustration of convex mask generation and local 3×3 neighborhood interpolation for output pixel reconstruction.

Algorithm 4.6 Guided Flow Upsampling (GFU).

Input:
- $\widetilde{\mathbf{V}}^{l}_{t\to 0,1} = [\widetilde{\mathbf{V}}^{l}_{t\to 0}, \widetilde{\mathbf{V}}^{l}_{t\to 1}]$ (Low-resolution bidirectional flow from MD-Net)
- $F_0^{l,c,0}, F_1^{l,c,0}$ (Context features of source images I_0, I_1 extracted at CFE C_0 scale)
- O^{l+1} (Coarse occlusion prior from level $l+1$)

Output:
- $\mathbf{V}^{l,\text{multi}}_{t\to 0,1} = \{\mathbf{V}^{l}_{t\to 0,1}, \mathbf{V}^{l\times 0.5}_{t\to 0,1}, \widetilde{\mathbf{V}}^{l}_{t\to 0,1}\}$ (3-scale flow pyramid)
- O^{l} (Upsampled occlusion prior at level l)

if spatial size of $F_0^{l,c,0}, F_1^{l,c,0} \neq \text{shape}(\widetilde{\mathbf{V}}^{l}_{t\to 0,1})$ **then**
  $F_0^{l,c,0}, F_1^{l,c,0} \leftarrow \text{BilinearInterpolate}\big(F_{0,1}^{l,c,0}, \text{target_size} = \text{shape}(\widetilde{\mathbf{V}}^{l}_{t\to 0,1})\big)$
end if
$\mathbf{M}_{\text{guidance}} \leftarrow \text{GuidanceNet}\big([F_0^{l,c,0} \parallel F_1^{l,c,0} \parallel \widetilde{\mathbf{V}}^{l}_{t\to 0,1}]\big)$ ▷ Outputs $[\text{Mask}_0, \text{Mask}_1]$ tensor
Extract direction-specific masks: $\text{Mask}_0, \text{Mask}_1 \leftarrow \text{Split}(\mathbf{M}_{\text{guidance}})$
$\mathbf{V}^{l}_{t\to 0} \leftarrow \text{ConvexUpsample}(\widetilde{\mathbf{V}}^{l}_{t\to 0}, \text{Mask}_0, \times 4) \times 4$ ▷ Upsample with magnitude rescaling
$\mathbf{V}^{l}_{t\to 1} \leftarrow \text{ConvexUpsample}(\widetilde{\mathbf{V}}^{l}_{t\to 1}, \text{Mask}_1, \times 4) \times 4$
Assemble full-res flow: $\mathbf{V}^{l}_{t\to 0,1} \leftarrow \big[\mathbf{V}^{l}_{t\to 0}, \mathbf{V}^{l}_{t\to 1}\big]$
Construct mid-scale representation: $\mathbf{V}^{l\times 0.5}_{t\to 0,1} \leftarrow \text{BilinearDownsample}(\mathbf{V}^{l}_{t\to 0,1}, 0.5)$
Let coarsest scale pass-through: $\mathbf{V}^{l\times 1}_{t\to 0,1} \leftarrow \widetilde{\mathbf{V}}^{l}_{t\to 0,1}$
Construct 3-scale flow pyramid: $\mathbf{V}^{l,\text{multi}}_{t\to 0,1} \leftarrow \{\mathbf{V}^{l}_{t\to 0,1}, \mathbf{V}^{l\times 0.5}_{t\to 0,1}, \mathbf{V}^{l\times 1}_{t\to 0,1}\}$
$O^{l} \leftarrow \text{BilinearUpsample}(O^{l+1}, \times 4)$ ▷ Upsample coarse occlusion prior
return $\mathbf{V}^{l,\text{multi}}_{t\to 0,1}, O^{l}$

All notations used in Algorithm 4.6 are defined below, and are used without re-definition for the remainder of this section.

- $\widetilde{\mathbf{V}}^{l}_{t\to 0,1} = [\widetilde{\mathbf{V}}^{l}_{t\to 0}, \widetilde{\mathbf{V}}^{l}_{t\to 1}]$: low-resolution bidirectional flow estimated by MD-Net at level l.
- $F_0^{l,c,0}, F_1^{l,c,0} \in \mathbb{R}^{C\times H\times W}$ (base channel $C = 32$): finest-scale context features (CFE C_0 scale) of source images I_0, I_1.
- O^{l+1}, O^{l}: coarse occlusion prior at level $l+1$, and its bilinearly upsampled counterpart at level l.
- GuidanceNet$(\cdot)$: stacked 3×3 convolutions predicting localized convex combination weights from the concatenated context and flow input.
- $\mathbf{M}_{\text{guidance}} \in \mathbb{R}^{288\times H\times W}$: the composite tensor output by GuidanceNet, split into direction-specific masks $\text{Mask}_0, \text{Mask}_1 \in \mathbb{R}^{144\times H\times W}$.

- $\text{Mask}_0, \text{Mask}_1$: normalized 3×3 convex combination weights for directions $t \to 0$ and $t \to 1$, with $144 = 9 \times 4 \times 4$ encoding a 3×3 neighborhood per 4×4 output sub-pixel block.
- ConvexUpsample($\mathbf{V}$, Mask, $\times 4$): the operator synthesizing a $\times 4$ spatially expanded flow map via position-specific mask weights over a 3×3 neighborhood.
- $\mathbf{V}^l_{t\to 0}, \mathbf{V}^l_{t\to 1}$: full-resolution single-direction flow fields after ConvexUpsample and magnitude rescaling by 4.
- $\mathbf{V}^l_{t\to 0,1}$: the assembled full-resolution bidirectional flow, $[\mathbf{V}^l_{t\to 0}, \mathbf{V}^l_{t\to 1}]$.
- $\mathbf{V}^{l\times 0.5}_{t\to 0,1}$: mid-scale flow obtained by bilinearly downsampling $\mathbf{V}^l_{t\to 0,1}$ by 0.5.
- $\mathbf{V}^{l,\text{multi}}_{t\to 0,1}$: the 3-scale flow pyramid $\{\mathbf{V}^l_{t\to 0,1}, \mathbf{V}^{l\times 0.5}_{t\to 0,1}, \widetilde{\mathbf{V}}^l_{t\to 0,1}\}$.

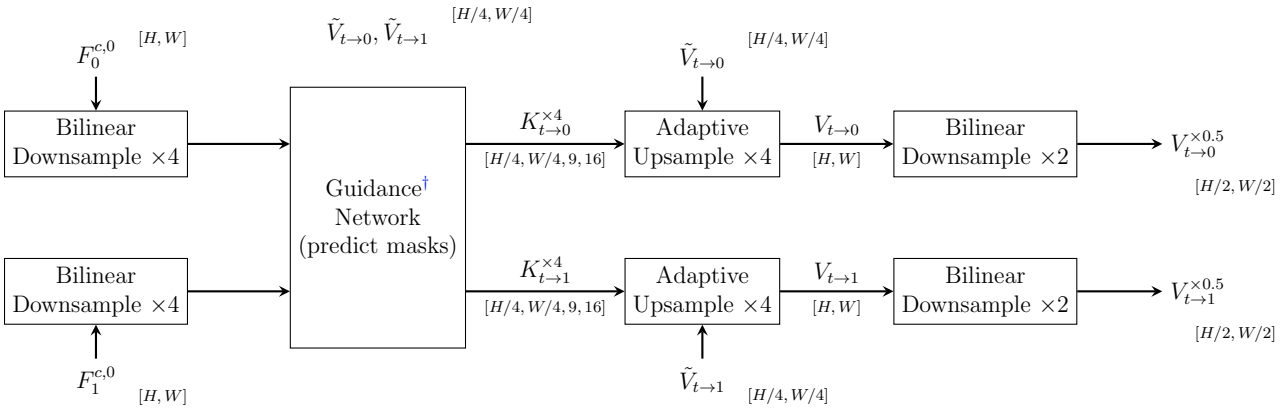


Figure 4.11: Architectural diagram of GFU. (†Introduced in this work.)

The design of the Guidance Net is motivated by the convex upsampling mechanism introduced by Teed and Deng [67] in RAFT, later also adopted by BiM-VFI's CAUN [18]. This mechanism predicts a softmax-normalized weighting mask over a 3×3 neighborhood. Aggregating neighboring flow vectors preserves motion boundaries far better than bilinear upsampling. However, RAFT predicts this mask from a single frame's context features alone. To address this limitation, we design a novel Guided Feature Upsampling (GFU) module. Our GFU extends this concept by incorporating content-aware kernel prediction inspired by CARAFE [91], together with classical guided/joint upsampling [92, 93], where high-resolution guidance signals drive the upsampling of low-resolution representations.

Specifically, context features from both adjacent frames, $F_0^{l,c,0}$ and $F_1^{l,c,0}$, are concatenated and fed to a shared Guidance Net. Consequently, the predicted masks are conditioned on both temporal directions simultaneously—a design tailored for VFI, where inter-frame motion is often non-uniform and cannot be fully characterized from a single frame. Following Seo et al. [18], we adopt a 3-scale flow representation ($\mathbf{V}^l, \mathbf{V}^{l\times 0.5}, \widetilde{\mathbf{V}}^l$) to balance computational efficiency with fine-grained optical flow resolution across stages, which subsequently serves as the multi-scale flow inputs for the Synthesis Network (SN).

A single Guidance Net is shared across both flow directions rather than using two independent networks: this halves the parameter count relative to per-direction networks while letting the model exploit the correlated motion structure between the two directions.

The context features and low-resolution flow are concatenated along the channel axis into a $(2C + 4)$-channel tensor and processed by the Guidance Network:

$$\mathbf{M}_{\text{guidance}} = [\text{Mask}_0 \,\|\, \text{Mask}_1] = \text{GuidanceNet}\left(\left[F_0^{l,c,0} \,\|\, F_1^{l,c,0} \,\|\, \widetilde{\mathbf{V}}^l_{t\to 0,1}\right]\right) \tag{4.6.1}$$

Intuitively, each output pixel is reconstructed as a weighted average of its 9 neighboring low-resolution flow vectors, where the weights are predicted per pixel rather than fixed (unlike bilinear upsampling).

GuidanceNet first predicts an unnormalized logit tensor, the **Raw Mask** $\mathbf{M}_{\text{raw}} \in \mathbb{R}^{2\times 9\times 4\times 4\times H\times W}$ (2 directions, 9 neighbors, 4×4 sub-pixel positions). A spatial Softmax is then applied across the 3×3 neighborhood dimension ($K = 9$) for each target sub-pixel:

$$w_k = \frac{\exp(m_k)}{\sum_{j=1}^{9} \exp(m_j)}, \quad \text{for } k \in \{1, 2, \ldots, 9\} \tag{4.6.2}$$

where m_k is the k-th scalar activation of the Raw Mask and w_k the normalized weight in the final **Convex Mask**. This guarantees:

1. **Strict Positivity:** $w_k > 0$ for all k.
2. **Affine Hull Invariance (Partition of Unity):** $\sum_{k=1}^{9} w_k = 1.0$, preventing artificial brightness shifts, magnitude inflation, or grid artifacts during warping.

The full-resolution flow is reconstructed as a weighted convex combination over each pixel's 3×3 neighborhood, with magnitude rescaled by the upsampling factor to preserve

physical displacement:

$$\mathbf{V}^l_{t\to 0} = \text{ConvexUpsample}\left(\widetilde{\mathbf{V}}^l_{t\to 0}, \text{Mask}_0,\ \times 4\right) \times 4 \tag{4.6.3}$$

$$\mathbf{V}^l_{t\to 1} = \text{ConvexUpsample}\left(\widetilde{\mathbf{V}}^l_{t\to 1}, \text{Mask}_1,\ \times 4\right) \times 4 \tag{4.6.4}$$

These, together with $\mathbf{V}^{l\times 0.5}_{t\to 0,1}$ and $\widetilde{\mathbf{V}}^l_{t\to 0,1}$, form the 3-scale flow pyramid $\mathbf{V}^{l,\text{multi}}_{t\to 0,1}$; O^{l+1} is bilinearly upsampled to O^l in parallel.

Overall, the key novelty of GFU lies in how the guidance masks are predicted. Building on the convex upsampling mechanism introduced in RAFT and adopted by CAUN in BiM-VFI, GFU uses context features from both input frames and explicitly incorporates the low-resolution bilateral flows. Unlike RAFT, which predicts masks from a recurrent hidden state derived from a single frame's context encoder, GFU exploits information from both frames. Unlike CAUN, GFU predicts the masks without first warping the context features, avoiding dependence on features aligned using the flow being upsampled. Furthermore, a single lightweight Guidance Net serves both flow directions, while the intermediate-resolution flow is derived by downsampling the full-resolution result rather than using a separate prediction head.

4.7 Synthesis Network (SN)

The Synthesis Network (SN) is adopted directly from BiM-VFI [18], without architectural modification, to reconstruct the intermediate frame with photorealistic boundaries. Rather than warping low-level pixels independently, the SN executes a joint pixel-and-feature warping strategy, dynamically aggregating context-specific feature pyramids ($\mathbf{F}^{l,c}_0, \mathbf{F}^{l,c}_1$) and aligning them via the multi-scale flow pyramid ($\mathbf{V}^{l,\text{multi}}_{t\to 0,1}$) provided by the GFU module. The complete feed-forward sequence is structured in Algorithm 4.7.

The network initializes structural boundaries by warping the raw source keyframes I^l_0, I^l_1 into temporal alignment using the finest-scale flow pair $\mathbf{V}^l_{t\to 0,1}$, producing the warped images $\hat{W}_0, \hat{W}_1$. These are concatenated, denoted by $\|$, with the occlusion prior O^l to form a 7-channel tensor (3 channels per warped image, plus 1 occlusion channel) and passed through the first encoder block, ConvDown_1, to produce the first-level encoder feature s_0:

$$s_0 = \text{ConvDown}_1\left(\left[\hat{W}_0 \parallel \hat{W}_1 \parallel O^l\right]\right) \tag{4.7.1}$$

Algorithm 4.7 Synthesis Network (SN).

Input:
- I_0^l, I_1^l (Source keyframes downsampled to pyramid level l)
- $\mathbf{F}_0^{l,c} = \{F_0^{l,c,0}, F_0^{l,c,1}, F_0^{l,c,2}\}$ (3-scale context feature pyramid of I_0^l)
- $\mathbf{F}_1^{l,c} = \{F_1^{l,c,0}, F_1^{l,c,1}, F_1^{l,c,2}\}$ (3-scale context feature pyramid of I_1^l)
- $\mathbf{V}_{t\to0,1}^{l,\text{multi}} = \{\mathbf{V}_{t\to0,1}^{l}, \mathbf{V}_{t\to0,1}^{l\times0.5}, \widetilde{\mathbf{V}}_{t\to0,1}^{l}\}$ (3-scale flow pyramid from GFU)
- O^l (Occlusion prior from level l)

Output:
- $\hat{I}^l$ (Interpolated intermediate frame at level l)
- $\hat{O}^l$ (Refined occlusion mask)

Warp raw source images at finest scale:
$\hat{W}_0 \leftarrow \text{Backwarp}(I_0^l, \mathbf{V}_{t\to0}^l)$; $\hat{W}_1 \leftarrow \text{Backwarp}(I_1^l, \mathbf{V}_{t\to1}^l)$ ▷ RGB warping

Warp context features at finest scale (C_0):
$\hat{F}_0^{l,c,0} \leftarrow \text{Backwarp}(F_0^{l,c,0}, \mathbf{V}_{t\to0}^l)$; $\hat{F}_1^{l,c,0} \leftarrow \text{Backwarp}(F_1^{l,c,0}, \mathbf{V}_{t\to1}^l)$

$s_0 \leftarrow \text{ConvDown}_1\big([\hat{W}_0 \parallel \hat{W}_1 \parallel O^l]\big)$ ▷ Process warped RGB and occlusion (6+1 channels)

$s_1 \leftarrow \text{ConvDown}_2\big([s_0 \parallel \hat{F}_0^{l,c,0} \parallel \hat{F}_1^{l,c,0}]\big)$ ▷ Inject warped finest-scale context

Warp context features at mid scale (C_1):
$\hat{F}_0^{l,c,1} \leftarrow \text{Backwarp}(F_0^{l,c,1}, \mathbf{V}_{t\to0}^{l\times0.5})$; $\hat{F}_1^{l,c,1} \leftarrow \text{Backwarp}(F_1^{l,c,1}, \mathbf{V}_{t\to1}^{l\times0.5})$

$s_2 \leftarrow \text{ConvDown}_3\big([s_1 \parallel \hat{F}_0^{l,c,1} \parallel \hat{F}_1^{l,c,1}]\big)$ ▷ Inject warped mid-scale context

Warp context features at coarsest scale (C_2):
$\hat{F}_0^{l,c,2} \leftarrow \text{Backwarp}(F_0^{l,c,2}, \widetilde{\mathbf{V}}_{t\to0}^l)$; $\hat{F}_1^{l,c,2} \leftarrow \text{Backwarp}(F_1^{l,c,2}, \widetilde{\mathbf{V}}_{t\to1}^l)$

$x \leftarrow \text{ConvUp}_1\big([s_2 \parallel \hat{F}_0^{l,c,2} \parallel \hat{F}_1^{l,c,2}]\big)$ ▷ Inject warped coarsest-scale context

Decode multi-scale representations through skip connections:

$x \leftarrow \text{ConvUp}_2\big([x \parallel s_1]\big)$; $x \leftarrow \text{ConvUp}_3\big([x \parallel s_0]\big)$

$\Delta I, \Delta O \leftarrow \text{Split}\big(\text{Conv}_{\text{out}}(x)\big)$ ▷ 4-channel output: 3 RGB residual + 1 occ residual

Perform blending and residual refinement:

$\hat{O}^l \leftarrow O^l + \Delta O \times \lambda_{\text{occ}}$ ▷ Apply occlusion mask residual strength

$M \leftarrow \text{Sigmoid}(\hat{O}^l)$ ▷ Synthesize blending mask

$\hat{I}^l \leftarrow M \odot \hat{W}_0 + (1 - M) \odot \hat{W}_1 + \Delta I \times \lambda_{\text{ref}}$ ▷ Apply photometric refinement scaling

return $\hat{I}^l, \hat{O}^l$

The finest-scale context features $F_0^{l,c,0}, F_1^{l,c,0}$ are warped by the same flow pair to give $\hat{F}_0^{l,c,0}, \hat{F}_1^{l,c,0}$, and fused with s_0 by the second encoder block ConvDown_2 to produce s_1:

$$s_1 = \text{ConvDown}_2\left(\left[s_0 \parallel \hat{F}_0^{l,c,0} \parallel \hat{F}_1^{l,c,0}\right]\right) \tag{4.7.2}$$

Multi-scale structural contexts across subsequent depths (C_1 and C_2) are symmetrically warped via their respective flow scales and injected into the remaining hierarchical encoder blocks to extract deep multi-scale geometric priors:

$$s_2 = \text{ConvDown}_3\left(\left[s_1 \parallel \hat{F}_0^{l,c,1} \parallel \hat{F}_1^{l,c,1}\right]\right) \tag{4.7.3}$$

At the coarsest depth C_2, the warped features $\hat{F}_0^{l,c,2}, \hat{F}_1^{l,c,2}$ are fused with s_2 by the first decoder block ConvUp_1, producing the decoder feature x that initiates the upsampling stage:

$$x = \text{ConvUp}_1\left(\left[s_2 \parallel \hat{F}_0^{l,c,2} \parallel \hat{F}_1^{l,c,2}\right]\right) \tag{4.7.4}$$

The three flow scales $\mathbf{V}_{t\to0,1}^{l}$, $\mathbf{V}_{t\to0,1}^{l\times0.5}$, and $\widetilde{\mathbf{V}}_{t\to0,1}^{l}$ correspond to the three-level pyramid $\mathbf{V}_{t\to0,1}^{l,\text{multi}}$ produced by the GFU module (Algorithm 4.7), applied to the context pyramid from finest to coarsest. Intuitively, the encoder builds up its understanding of the scene progressively: it first estimates rough visibility from the two warped RGB frames and the occlusion prior (Eq. 4.7.1), then refines this at each subsequent block by fusing in context features warped at a finer resolution, from C_0 through C_1 to C_2 (Eqs. 4.7.2–4.7.4). This lets the network combine fine-scale detail such as sharp boundaries and texture with coarse-scale structural patterns, rather than relying on a single resolution alone.

Spatial resolution is progressively recovered through decoder upsampling coupled with dense skip connections from previous encoder layers ($x' = \text{ConvUp}_2([x \parallel s_1])$ and $x'' = \text{ConvUp}_3([x' \parallel s_0])$). The final projection layer (Conv_{out}) outputs a 3-channel RGB residual ΔI and a 1-channel occlusion residual ΔO. The final occlusion tracking mask and interpolated intermediate frame are synthesized via gated blending layers, where $\odot$ denotes element-wise multiplication:

$$\hat{O}^l = O^l + \Delta O \times \lambda_{\text{occ}} \tag{4.7.5}$$

$$M = \text{Sigmoid}(\hat{O}^l) \tag{4.7.6}$$

$$\hat{I}^l = M \odot \hat{W}_0 + (1 - M) \odot \hat{W}_1 + \Delta I \times \lambda_{\text{ref}} \tag{4.7.7}$$

where λ_{occ} and λ_{ref} represent scalar hyper-parameters dictating the contribution strength of occlusion mask updates and localized RGB residual refinements, respectively, thereby mitigating warping-induced ghosting and structural artifacts.

4.8 Loss Functions

The optimization framework of the architecture is formulated using two primary supervisory domains: Photometric loss, which enforces visual reconstruction fidelity, and Flow-centric loss, which governs motion geometry and optical flow consistency. To assist in understanding the entire training target at a glance, a comprehensive overview of all utilized loss components is summarized in Table 4.3.

With the exception of the Perceptual Loss, which is introduced in this work to improve fine-detail reconstruction, all loss components below follow the formulation of BiM-VFI [18].

Table 4.3: Summary of Photometric and Flow-Centric Loss Functions

Domain	**Loss Function**	**Process**	**Brief Description**
$\mathcal{L}_{\text{pho}}$	$\mathcal{L}_{\text{char}}$	$\mathcal{P}_{\mathcal{S}}$	Measures pixel-level reconstruction fidelity.
$\mathcal{L}_{\text{pho}}$	$\mathcal{L}_{\text{ternary}}$	$\mathcal{P}_{\mathcal{S}}$	Captures structural/textural similarity via Census Transform.
$\mathcal{L}_{\text{pho}}$	$\mathcal{L}_{\text{photo_tea}}$	$\mathcal{P}_{\mathcal{T}}$	Establishes a high-fidelity frame synthesis anchor.
$\mathcal{L}_{\text{pho}}$	$\mathcal{L}^{\dagger}_{\text{perc}}$	$\mathcal{P}_{\mathcal{S}}$	Enhances fine textures via pre-trained VGG16 features.
$\mathcal{L}_{\text{flo}}$	$\mathcal{L}_{\text{flow_distill}}$	$\mathcal{P}_{\mathcal{S}} \leftarrow \mathcal{P}_{\mathcal{T}}$	Distills motion geometry from teacher to student residuals.
$\mathcal{L}_{\text{flo}}$	$\mathcal{L}_{\text{flow_reg}}$	$\mathcal{P}_{\mathcal{T}}$	Constrains the unbounded growth of raw teacher flow vectors.
$\mathcal{L}_{\text{flo}}$	$\mathcal{L}_{\text{flow_smooth}}$	$\mathcal{P}_{\mathcal{T}}$	Enforces spatial smoothness while preserving sharp image edges.

† Introduced in this work; all other losses follow BiM-VFI [18].

Photometric Loss supervises the pixel-level fidelity, texture sharpness, and localized structural accuracy of the synthesized intermediate frames. This domain consists of the following components:

- **Charbonnier Loss:** A robust pixel-level reconstruction loss utilized by the student process to penalize spatial mismatches against out-of-distribution artifact sensitivities. It calculates the continuous variations between the predicted intermediate frame

$\hat{I}$ and the ground truth I_t:

$$\mathcal{L}_{\text{char}} = \frac{1}{HW} \sum_{x,y} \sqrt{(\hat{I}(x,y) - I_t(x,y))^2 + \epsilon^2} \tag{4.8.1}$$

where H and W represent the frame dimensions, and $\epsilon = 10^{-3}$ is a numerical stability constant.

- **Ternary Loss:** Complements pixel-level alignment by mapping textural similarity over a local structural landscape. It computes a soft Hamming distance on a 7×7 Census Transform (T), offering high immunity against uniform luminance variations or illumination shifts:

$$\mathcal{L}_{\text{ternary}} = \frac{1}{HW} \sum_{x,y} H\big(T(\text{gray}(\hat{I}(x,y))),\, T(\text{gray}(I_t(x,y)))\big) \cdot M(x,y) \tag{4.8.2}$$

  where M acts as a valid boundary mask to strip out peripheral padding artifacts, and the soft Hamming metric $H(t_1, t_2)$ is evaluated via $\frac{(t_1-t_2)^2}{0.1+(t_1-t_2)^2}$.

- **Photo Teacher Loss:** Guides the teacher network to establish an idealized frame synthesis baseline. Because the teacher module receives direct direction guidance parameters derived explicitly from the ground-truth target, this constraint forms a reliable anchor by calculating the average of the Charbonnier and Ternary errors on the teacher's reconstructed output $\hat{I}^{\text{tea}}$:

$$\mathcal{L}_{\text{photo_tea}} = \frac{\mathcal{L}_{\text{char}}(\hat{I}^{\text{tea}}, I_t) + \mathcal{L}_{\text{ternary}}(\hat{I}^{\text{tea}}, I_t)}{2} \tag{4.8.3}$$

- **Perceptual Loss:**

  Implemented exclusively during the late-stage network refinement phase to capture high-frequency visual patterns. To mitigate over-smoothing, $\mathcal{L}_{\text{perc}}$ is added in late-stage refinement using the VGG16 formulation (Section 2.4.2) as:

$$\mathcal{L}_{\text{perc}} = \sum_{k \in \mathcal{K}} \|\text{VGG16}_k(\hat{I}) - \text{VGG16}_k(I_t)\|_1 \tag{4.8.4}$$

  where the layer set is selected from deep feature spaces $\mathcal{K} = \{\text{relu1_2}, \text{relu2_2}, \text{relu3_3}\}$.

Flow-Centric Loss guarantees the geometric alignment, structural consistency, and physical correctness of the predicted optical flow patterns. This domain consists of the following components:

- **Flow Distillation Loss:** Directs knowledge distillation from the privileged teacher pipeline ($\mathcal{P}_\mathcal{T}$) to the student network ($\mathcal{P}_\mathcal{S}$). The student learns to replicate spatial flow fields while a dynamic clipping mask M suppresses optimization updates over extreme outlier vectors whose magnitudes exceed 32 pixels:

$$\mathcal{L}_{\text{flow_distill}} = \frac{1}{HW} \sum_{x,y} \sqrt{(\mathbf{V}^{\mathcal{P}_\mathcal{S}}(x,y) - \hat{\mathbf{V}}^{\mathcal{P}_\mathcal{T}}(x,y))^2 + \epsilon^2} \cdot M(x,y) \tag{4.8.5}$$

  where the teacher's optical flow gradient is completely frozen without backpropagating gradients to ensure optimization stability.

- **Flow Regularization Loss:** Enforces a strict L_1 magnitude penalty directly upon raw, unwarped bidirectional teacher flows ($\mathbf{V}^{\mathcal{P}_\mathcal{T}}_{t\to 0}$ and $\mathbf{V}^{\mathcal{P}_\mathcal{T}}_{t\to 1}$). This self-regularization bounds the flow decomposition scale and prevents boundless numerical escalation:

$$\mathcal{L}_{\text{flow_reg}} = \frac{1}{HW} \sum_{x,y} \left(|\mathbf{V}^{\mathcal{P}_\mathcal{T}}_{t\to 0}(x,y)| + |\mathbf{V}^{\mathcal{P}_\mathcal{T}}_{t\to 1}(x,y)| \right) \tag{4.8.6}$$

- **Edge-Aware Smoothness Loss:** Enforces local spatial smoothness across the upsampled flow fields to prevent erratic pixel displacements. It applies an exponential gating function driven by the first-order derivatives of the ground-truth image (∂I_t), allowing sharp motion transitions directly over object contours while heavily penalizing flow discontinuities inside flat areas:

$$\mathcal{L}_{\text{flow_smooth}} = \frac{1}{HW} \sum_{x,y} \left[\rho(\partial_x \mathbf{V}^{\mathcal{P}_\mathcal{T}}) \cdot e^{-(\alpha \overline{|\partial_x I_t|})^2} + \rho(\partial_y \mathbf{V}^{\mathcal{P}_\mathcal{T}}) \cdot e^{-(\alpha \overline{|\partial_y I_t|})^2} \right] \tag{4.8.7}$$

  where $\rho(x) = \sqrt{x^2 + \epsilon^2}$ represents the standard Charbonnier penalty function, $\overline{|\cdot|}$ denotes the average value computed across all color channels, and $\alpha = 150$ dictates a steep, localized boundary gating sensitivity boundary.

Chapter 5

Experiments

5.1 Experiment Setup

5.1.1 Training Strategy and Implementation Details

The Vimeo-90K septuplet split [94] is used for both training and evaluation. The official split file lists 64,612 training sequences and 7,824 test sequences. In practice, however, the data loader sets aside the last 5% of the training list, about 3,231 sequences, as a validation set, checked every 10 epochs. This leaves about 61,381 sequences for actual training. The official 7,824-sequence test split is not used during training at all; it is kept separate for the final benchmark results reported in this chapter.

To enable frame interpolation at arbitrary timestamps ($t \in (0, 1)$) rather than fixing $t = 0.5$, we adopt dynamic temporal sampling over the Vimeo-90K septuplet dataset, following prior arbitrary-timestep VFI work [18, 37, 38]. For each 7-frame sequence (`im1`–`im7`), three frames are randomly sampled without requiring them to be consecutive, then sorted chronologically. The earliest and latest frames become I_0 and I_1, and the middle one becomes I_t, with $t = \frac{\text{index}(I_t)-\text{index}(I_0)}{\text{index}(I_1)-\text{index}(I_0)}$ computed from their relative spacing instead of always being the midpoint. This exposes the network to diverse temporal gaps and non-uniform velocity profiles during training.

Each training sample is randomly cropped to a 256×256 patch before being passed through the network. During training, this patch size corresponds to a fixed active pyramid depth of $L = 3$. This patch size was chosen as a middle ground between two competing constraints: it must be small enough to fit a reasonable batch size (8 samples per step,

with 8 parallel data-loading workers) in GPU memory, while still large enough for the model's three-level pyramid to resolve meaningful motion structure at its coarsest stage. With $L = 3$ and a 256×256 input, the coarsest pyramid level operates at 64×64 (a 1/4 downsample). The Motion Feature Encoder further reduces this by 1/4, so the bidirectional motion field is estimated on a 16×16 feature map. At this resolution, only broad, low-frequency motion trends are visible; fine local detail is left to the finer levels above.

To reduce overfitting to the viewpoints and color statistics in Vimeo-90K, each training triplet is augmented using randomized transformations. The same transformations are applied to I_0, I_t, and I_1 to preserve inter-frame correspondence. Each triplet has an independent 50% chance of horizontal flipping, vertical flipping, and RGB channel reordering using one of six fixed permutations. There is also a 50% chance of swapping I_0 and I_1, with the interpolation time remapped from t to $1 - t$. Finally, the triplet is rotated by 90°, 180°, 270°, or not rotated, with equal probability.

Optimization and Hyperparameters. The network is trained end-to-end using the AdamW optimizer with a weight decay of 1×10^{-5} and gradient norm clipping capped at 1.0. The training is conducted with a total batch size of 8 for 400 epochs. The learning rate is scheduled using a cosine annealing policy with a 1% warm-up phase, reaching a peak value of 3.75×10^{-5}. The unified objective function integrates Charbonnier, Census Ternary, teacher photometric, flow distillation, edge-aware flow smoothness, and VGG-16 perceptual losses (weighted at 0.04), evaluating feature distances over early layers of a frozen ImageNet backbone.

Hardware Infrastructure and Reproducibility. All experiments share a fixed random seed (1234) for reproducibility. Executed on the high-performance computing infrastructure of the Digital Research Alliance of Canada, a full training run takes approximately 5–6 days using 4 NVIDIA A100 GPUs. Inference latency benchmarking was executed on a workstation equipped with a single NVIDIA GeForce RTX 5060 Ti GPU (16 GB VRAM).

5.1.2 Dataset

To train and evaluate the proposed framework under conditions consistent with prior work in the field, this study adopts the Vimeo90K dataset [94] for training and two independent benchmarks for validation: the Vimeo90K triplet test split and SNU-FILM [95]. Table 5.1 summarizes the composition of each split.

Table 5.1: Summary of the Datasets Used in This Thesis.

Dataset	**# Samples**	**Purpose**
Vimeo90K Septuplet	91,701 sequences	Training with diverse temporal offsets and motion patterns.
Vimeo90K Triplet	3,782 triplets	Standard in-domain benchmark for direct comparison with prior work.
SNU-FILM	1,240 triplets	Out-of-domain evaluation under increasingly challenging motion.

Vimeo90K Septuplet (Training)

The model is trained on the Vimeo90K *Septuplet* split [94], which consists of 91,701 seven-frame sequences at a resolution of 448×256, collected from diverse real-world video content. The availability of seven consecutive frames per clip enables sampling of triplets at multiple temporal offsets around the center frame, exposing the network to a broader range of motion magnitudes and non-uniform motion patterns during training than the fixed-triplet alternative. The official training and test partitions provided by the dataset are adopted throughout this work.

Vimeo90K Triplet (Benchmark)

The standard test split of Vimeo90K [94], comprising 3,782 fixed triplets at a resolution of 448×256, is held out entirely from training and used to assess interpolation fidelity under moderate, near-uniform motion. Vimeo90K Triplet is adopted because it serves as the de facto benchmark for VFI evaluation, with virtually all recent state-of-the-art methods reporting results on the same test split. Evaluating on this benchmark therefore enables a direct, transparent, and fair comparison against existing methods without requiring every baseline to be retrained, which is beyond the scope of this thesis. Furthermore, since the proposed model is trained on the Vimeo90K Septuplet dataset, evaluating on the Triplet split provides an in-domain assessment of generalization under the same visual distribution while maintaining a strict separation between training and testing samples.

SNU-FILM (Benchmark)

SNU-FILM [95] consists of 1,240 triplets spanning resolutions from 640×368 to 1280×720, partitioned into four difficulty subsets—*Easy*, *Medium*, *Hard*, and *Extreme*—corresponding to increasing temporal distance between input frames. Unlike Vimeo90K, SNU-FILM is collected from an entirely different source, providing an out-of-domain evaluation that measures robustness under distribution shift. Its progressively harder subsets expose the model to increasingly large, non-uniform motion. This makes SNU-FILM well suited for validating the framework's core objective of reducing time-to-location (TTL) ambiguity under complex motion. In addition, the wider range of image resolutions (up to 1280×720) complements Vimeo90K by assessing the model across more diverse spatial scales rather than a single fixed resolution.

5.2 Evaluation Metrics

We evaluate the proposed method from four perspectives: (1) Fidelity metrics (PSNR/SSIM), (2) Perceptual quality (LPIPS/STLPIPS/NIQE), (3) Qualitative visual results, and (4) Computational efficiency.

Baseline Selection. To benchmark the proposed framework comprehensively, we select representative VFI methods that cover a wide spectrum of architectural designs and optimization goals:

- **BiM-VFI** [18]: Serves as our primary baseline architecture. Since our framework builds upon and extends its bidirectional motion field formulation, comparing against BiM-VFI directly demonstrates the quantitative and qualitative improvements achieved by our proposed modules.
- **AMT-G** [50]: Represents all-pairs motion trajectory modeling. In our observations, AMT-G provides the strongest visual results among competing SOTA methods by preserving object shapes and motion integrity well, though slight visual blurriness remains its main limitation.
- **UPR-Net** [49]: Represents unified pyramid-based architectures. Like our approach, it relies on a coarse-to-fine image pyramid structure, achieving high pixel-level fidelity (PSNR/SSIM) across standard benchmarks.

- **RIFE** [37]: A widely adopted real-time flow estimation pipeline that prioritizes inference speed and computational efficiency.
- **XVFI** [48]: A specialized model designed for extreme video frame interpolation, capable of handling large spatial displacements and ultra-high-resolution (4K) motion dynamics.

5.2.1 Qualitative Results (Visual Comparisons)

We highlight representative test cases of uniform and non-uniform motion in Fig. 5.1, with detailed qualitative comparisons across all sequences presented in Fig. 5.2–5.7. For all evaluated cases, each model synthesizes all seven intermediate frames (Frames 1–7) between the input keyframes, demonstrating 8× multi-frame interpolation. Our method produces sharper and more structurally consistent results under large motion compared to existing approaches, while maintaining a favorable balance between visual quality and efficiency.

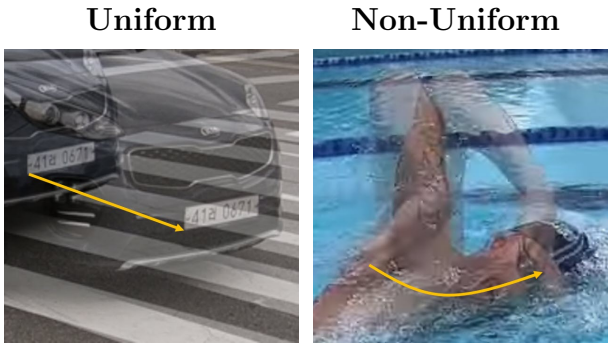


Figure 5.1: Example Test Cases for Uniform and Non-Uniform Motion

- **Fig. 5.2 (Black Car):** Focuses on high-frequency alphanumeric details (license plate characters), which present severe difficulty for models like UPR-Net and RIFE to reconstruct without blurring or structural distortion.
- **Fig. 5.3 (Watermelon):** Evaluates localized spatial tracking and boundary preservation under small-to-medium object motion and fast hand interactions.
- **Fig. 5.4 (Swimmer):** Evaluates model performance under severe large displacement combined with non-uniform curvilinear motion. As seen in the overlay input, the

swimmer's arm exhibits rapid angular motion around the elbow joint across a wide spatial gap, where our method effectively mitigates motion ambiguity to reconstruct sharp articulated limb boundaries.

- **Fig. 5.5 (Soccer):** Focuses on complex multi-entity interactions and severe dynamic occlusions rather than displacement magnitude alone. This sequence tests the network's resilience against ambiguity caused by overlapping legs, distinct sock colors, and ball interactions in close proximity.
- **Fig. 5.6 (Kite Lines):** Tests structural continuity over delicate, ultra-thin linear structures (kite control lines) to prevent line tearing or breakage artifacts under large motion gaps.
- **Fig. 5.7 (Water Splash):** Focuses on preserving intricate, high-frequency visual textures of splashing water and scattering droplets without over-smoothing.

Across all challenging scenarios, our method produces sharper and more structurally consistent results under large motion compared to existing approaches, while maintaining a favorable balance between visual quality and efficiency.

Qualitative Performance Analysis Across Localized Features. A closer examination of the visual results reveals specifically where and why our proposed framework outperforms prior state-of-the-art methods:

- **Sharp Structural Continuity and Fine Text (Where: License Plates and Thin Lines):** As highlighted in Fig. 5.2 and Fig. 5.6, baseline methods suffer from severe blurring or line tearing along thin objects. Our method successfully reconstructs distinct license plate characters and unbroken kite lines. This improvement stems from the **Guided Flow Upsampling (GFU)** module, which conditions spatial flow upsampling on high-resolution context features ($F^{l,c,0}$) via learned convex masks, preventing edge-bleeding and preserving fine spatial continuity during $\times 4$ expansion.
- **Ghosting Mitigation Under Fast Body Motion (Where: Human Limbs and Dynamic Objects):** In complex sports scenarios like soccer (Fig. 5.5), competing architectures like UPR-Net and XVFI exhibit noticeable ghosting and double-edge artifacts around fast-moving limbs. In contrast, our model predicts clean, well-aligned

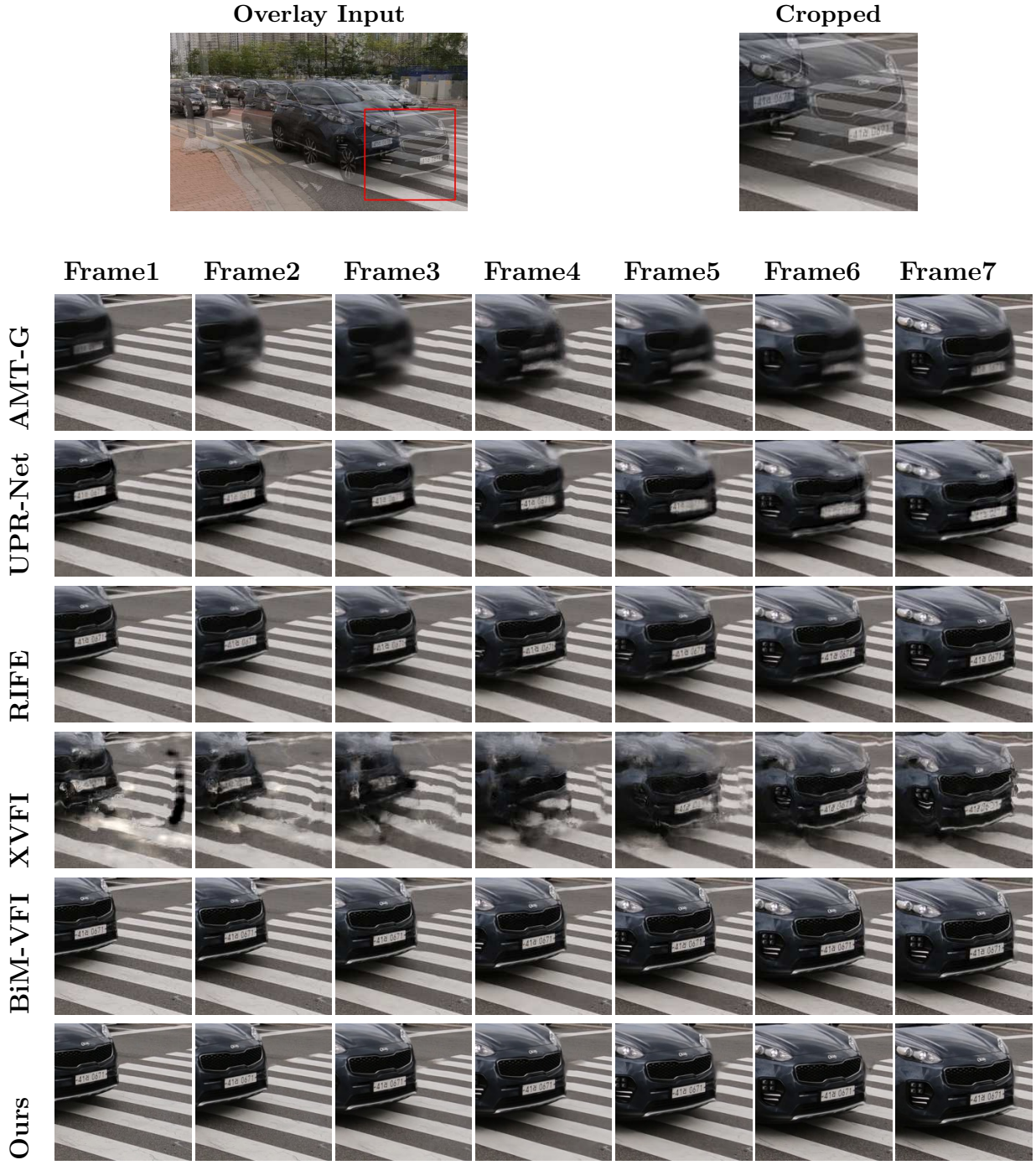


Figure 5.2: Black car case study with 7 interpolated intermediate frames.

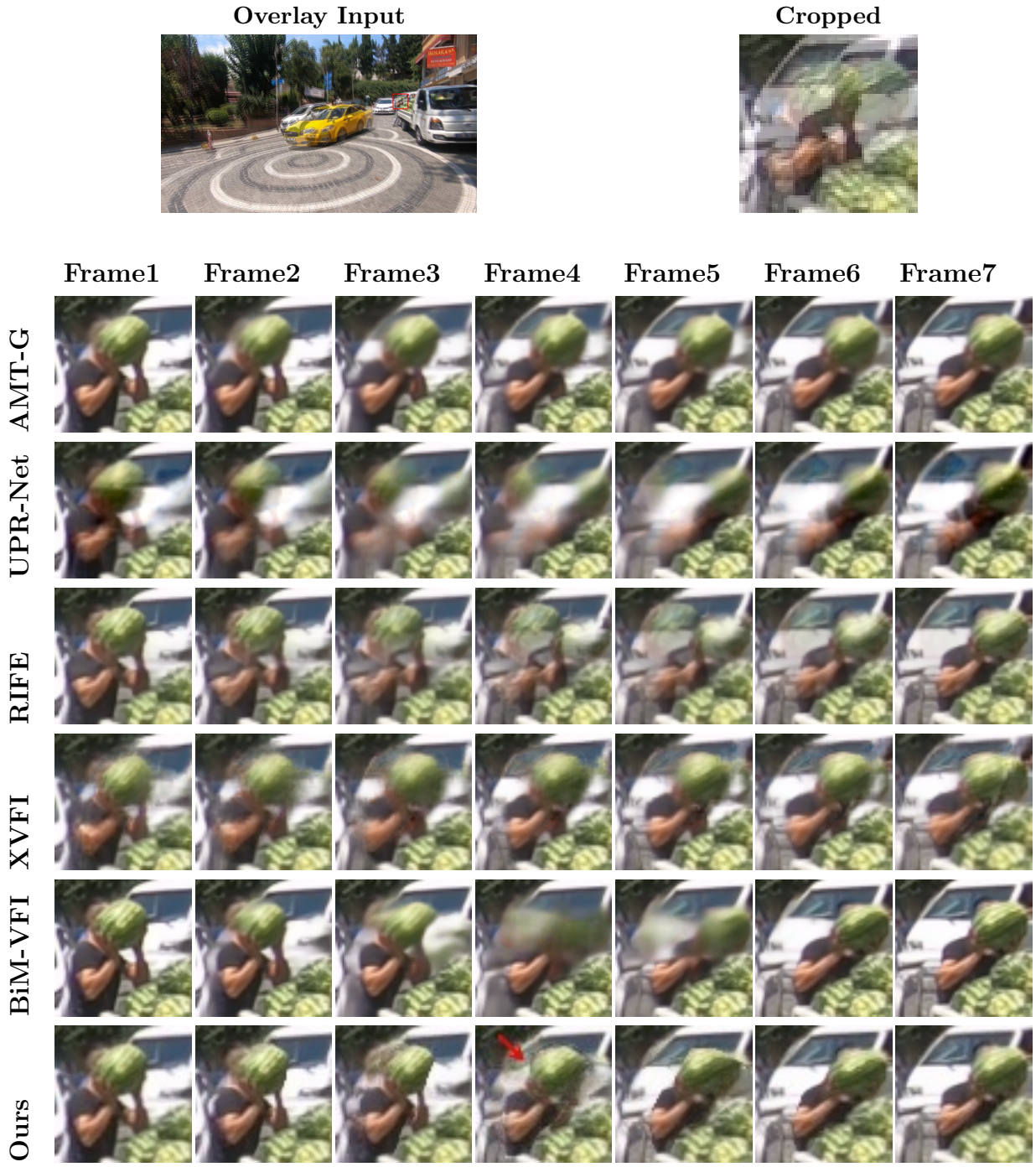


Figure 5.3: Watermelon case study with 7 interpolated intermediate frames.

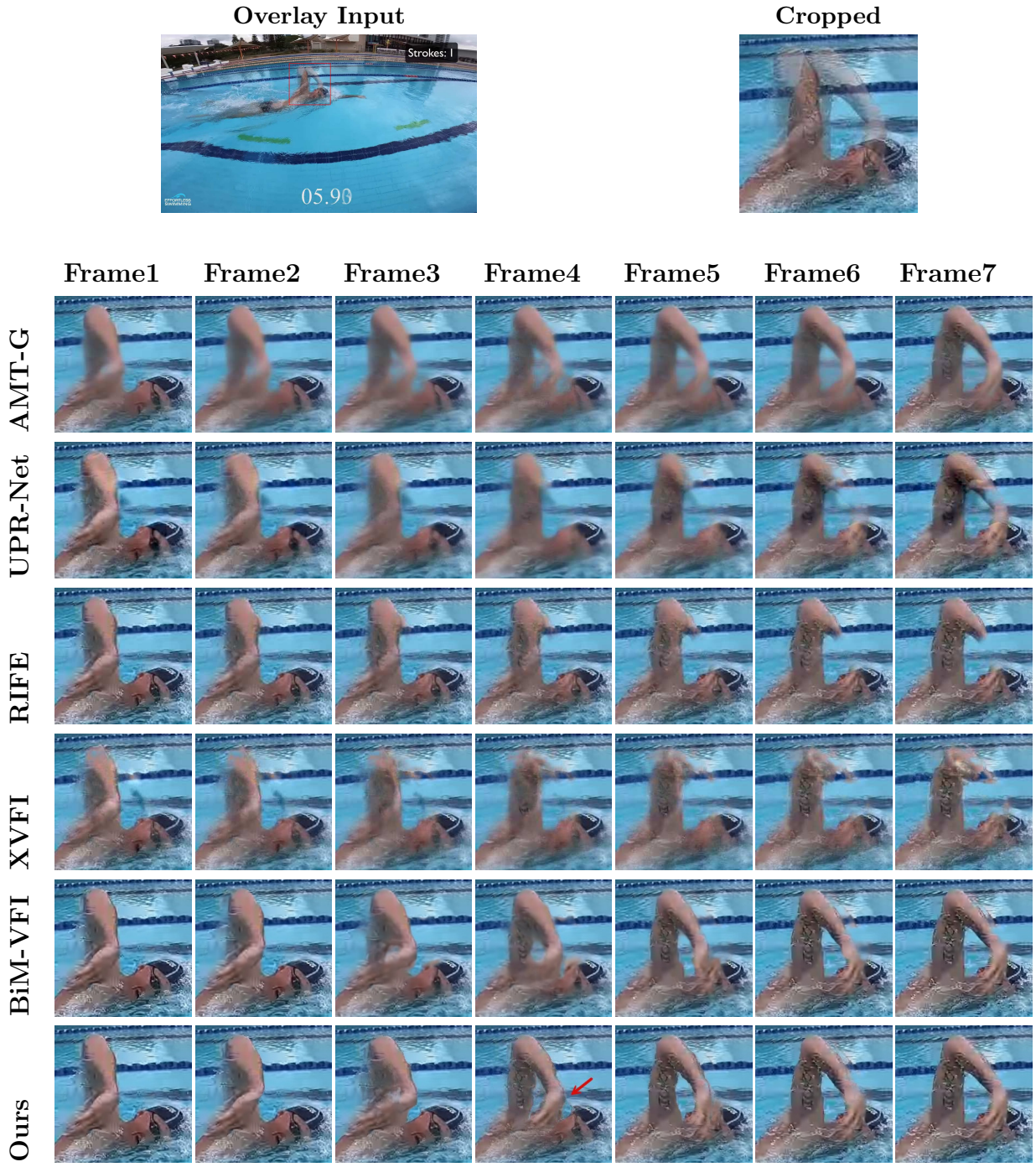


Figure 5.4: Swimmer case study with 7 intermediate frames under large motion.



Figure 5.5: Soccer case study with 7 intermediate frames.

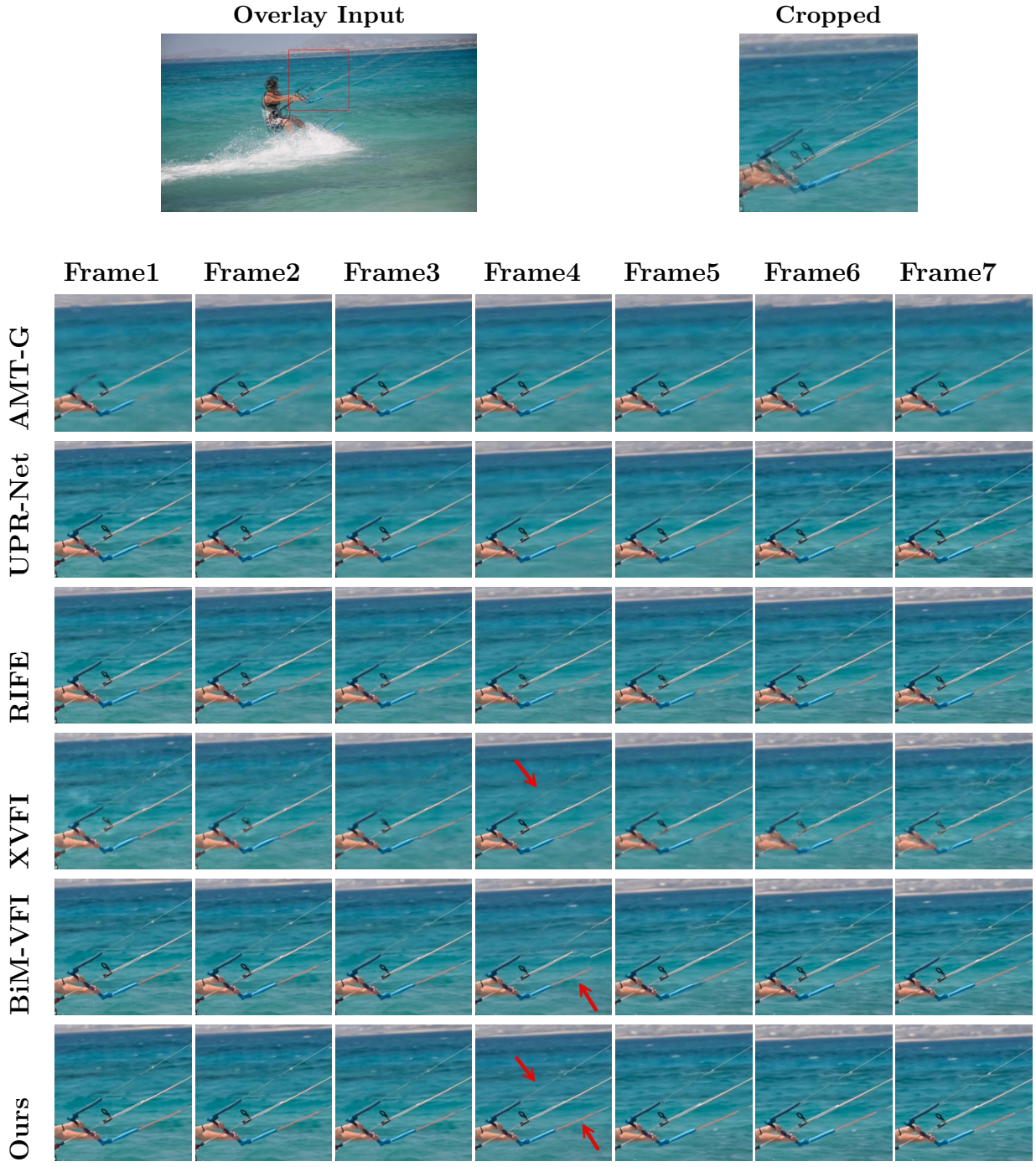


Figure 5.6: Kite lines case study with 7 intermediate frames focusing on thin linear structures under complex motion.

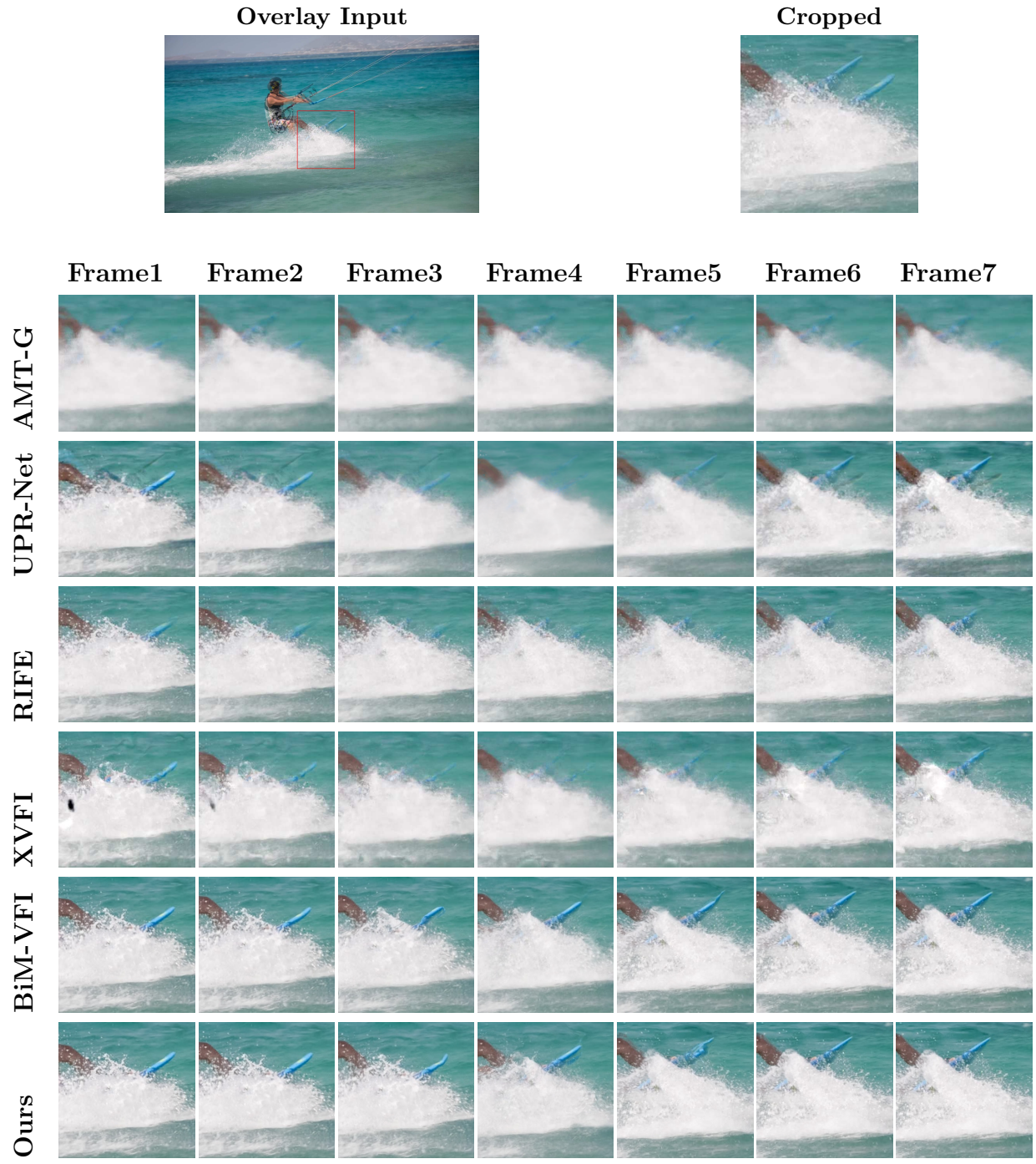


Figure 5.7: Water splash case study with 7 intermediate frames under intricate motion and scattering droplets.

boundaries. This advantage is enabled by **MD-Net**, which integrates dynamic trajectory guidance (r, ϕ) with SE channel recalibration to effectively constrain the optical flow search space for large spatial displacements.

- **High-Frequency Anatomical Details and Texture Realism (Where: Swimmer and Water Dynamics):** In challenging fluid and human dynamics scenarios (Fig. 5.4 and Fig. 5.7), conventional L_1/photometric losses tend to produce over-smoothed structures and severed extremities. By incorporating late-stage **Perceptual Loss ($\mathcal{L}_{\text{perc}}$)**, our synthesis network preserves high-frequency visual details—clearly resolving fine anatomical features such as distinct individual fingers during fast arm strokes (highlighted by the red arrow in Frame 4 of Fig. 5.4) alongside realistic water splashing patterns without distorting underlying motion fields.

It should be noted that the real-world test sequences in Figs. 5.2–5.7 lack ground-truth reference frames, thereby precluding direct frame-level quantitative evaluation. To complement these qualitative assessments with objective verification, Fig. 5.8 illustrates a visual and quantitative comparison on a representative benchmark case at temporal point $t = 0.5$ where ground truth is available.

While our proposed framework demonstrates competitive performance when averaged across the entire benchmark dataset (as reported in Sections 5.2.2 and 5.2.3), individual frame-level metrics can exhibit variance depending on local structural characteristics. Consequently, a single synthesized frame may not uniformly surpass all baseline methods across every evaluation metric simultaneously; nevertheless, our model consistently preserves fine structural details and perceptual naturalness, maintaining an optimal trade-off between fidelity and perceptual quality.

5.2.2 Perceptual Quality

As discussed in Section 2.5, PSNR and SSIM are computed on a pixel-wise or local-window basis and therefore systematically reward blurred predictions over sharp-but-slightly-misaligned ones, failing to reflect the perceptual quality attributes that matter most for interpolated content, such as blurring artifacts, texture degradation, and structural realism [16, 31, 96]. Both metrics also require a pixel-aligned ground-truth reference frame, which limits their applicability to benchmark datasets and prevents their use on real-world video where no reference is available.

To close this gap, we adopt three complementary perceptual metrics, each targeting a distinct limitation that PSNR and SSIM cannot capture. **LPIPS** [30] compares deep fea-

Overlay Input



Ground Truth



AMT-G



PSNR: 16.9323, SSIM: 0.4003
LPIPS: 0.2075, STLPIPS: 0.1662, NIQE: 2.7693

UPR-Net



PSNR: **17.6919**, SSIM: **0.4339**
LPIPS: 0.1724, STLPIPS: 0.1533, NIQE: 2.9628

RIFE



PSNR: 17.5610, SSIM: 0.4227
LPIPS: 0.1493, STLPIPS: 0.1059, NIQE: 3.3687

XVFI



PSNR: 15.8871, SSIM: 0.3400
LPIPS: 0.4181, STLPIPS: 0.3852, NIQE: 3.1575

BiM-VFI



PSNR: 17.0262, SSIM: 0.3905
LPIPS: 0.1291, STLPIPS: **0.0690**, NIQE: 2.8046

Ours



PSNR: 17.2575, SSIM: 0.3985
LPIPS: **0.1263**, STLPIPS: 0.0696, NIQE: **2.4367**

Figure 5.8: Visual comparison and quantitative evaluation metrics of interpolated frames at $t = 0.5$ across six models.

ture representations extracted from a pretrained network rather than raw pixel intensities, aligning more closely with deep feature-based similarity metrics and penalizing perceptually implausible textures that pixel-wise metrics overlook. **STLPIPS** [33] addresses a specific weakness of LPIPS for the interpolation setting: its oversensitivity to small *spatial* shifts between the reference and the prediction. This is directly relevant to VFI because a correctly interpolated frame can be sub-pixel misaligned with the ground truth simply due to genuine motion ambiguity rather than an actual reconstruction error; LPIPS would over-penalize such cases, whereas STLPIPS remains tolerant to benign shifts and therefore more accurately isolates true perceptual degradation. **NIQE** [34], unlike the two metrics above, is a no-reference metric that requires no ground-truth frame at all, instead measuring how far the output deviates from the statistical regularities of natural images. It is included both to assess perceptual naturalness in a reference-free setting and to remain consistent with our primary baseline, BiM-VFI [18], which reports NIQE on the same Vimeo90K and SNU-FILM benchmarks, ensuring a fair, like-for-like comparison.

Together, LPIPS, STLPIPS, and NIQE cover three complementary aspects of visual quality that PSNR and SSIM cannot: reference-based deep-feature fidelity, robustness to benign spatial misalignment inherent to interpolation, and reference-free naturalness. Unlike pixel-wise metrics, none of the three is fooled by the blur-averaging behavior that a purely pixel-wise training objective tends to produce.

Table 5.2: Comparison of perceptual metrics (LPIPS[30]/STLPIPS[33]/NIQE[34]) on Vimeo90K[94] and SNU-Film [95] datasets
Bold: Best performance, underlined: Second best performance, dashed underline: Third best performance

Methods	Vimeo90K triplet	SNU-Film			
		Easy	Medium	Hard	Extreme
AMT-G [50]	0.019/0.012/5.327	0.022/0.008/4.822	0.035/0.015/4.924	0.060/0.028/4.993	0.112/0.068/4.993
M2M-PWC [75]	0.025/0.017/5.346	0.021/0.009/4.765	0.036/0.016/4.824	0.063/0.033/4.773	0.212/0.057/6.082
UPRNet[49]	0.022 /0.015/5.367	0.018/0.008/4.703	0.034/0.015/4.853	0.062/0.030/4.975	0.110/0.067/5.052
RIFE [37]	0.022/0.014/5.240	0.018/0.007/4.709	0.032/0.014/4.813	0.066/0.037/4.872	0.138/0.099/4.935
XVFI [48]	0.028/0.019/5.236	0.027/0.015/4.829	0.040/0.024/4.847	0.068/0.043/4.780	0.120/0.083/**4.618**
IFRNet [72]	0.019/0.013/5.267	0.020/0.008/4.820	0.032/0.013/4.889	0.056/0.027/4.890	0.113/0.073/4.856
EMA-VFI[38]	0.020/0.013/5.350	0.019/0.008/4.704	0.033/0.015/4.847	0.059/0.030/4.979	0.113/0.073/5.087
BiM-VFI[18]	0.020/0.012/5.283	0.017/0.006/4.678	0.029/0.011/4.773	0.052/0.022/4.863	0.097/0.052/4.942
Ours	**0.016/0.010/5.095**	**0.012/0.005/4.537**	**0.023/0.010/4.613**	**0.046/0.021/4.670**	**0.090/0.051**/4.719

Tables 5.2 report the quantitative results of these perceptual metrics on the Vimeo90K and SNU-Film datasets. Overall, these metrics provide a comprehensive evaluation framework that jointly captures spatial fidelity, temporal stability, and perceptual naturalness,

which are all essential aspects of high-quality video frame interpolation.

Experimental results show that our model achieves consistently superior performance across LPIPS, STLPIPS, and NIQE. In particular, it attains the lowest LPIPS scores across all datasets, indicating improved visual fidelity under complex motion. The STLPIPS results follow the same trend even after accounting for tolerance to benign sub-pixel misalignment, confirming that the LPIPS gains reflect genuine perceptual improvement rather than an artifact of shift-sensitivity. NIQE further confirms improved perceptual naturalness in a no-reference setting, consistent with the reference-based results. These results suggest that although our method is not explicitly optimized for pixel-wise reconstruction, it achieves better perceptual metric scores that remain robust to inherent interpolation ambiguity compared to existing approaches.

5.2.3 Fidelity metrics

Table 5.3 reports quantitative results on Vimeo90K and SNU-Film datasets in terms of fidelity metrics (PSNR/SSIM). These metrics measure pixel-wise reconstruction accuracy and are widely used for evaluating video frame interpolation methods. Overall, our method achieves competitive performance compared to recent approaches while maintaining a compact model size.

Table 5.3: Comparison of fidelity metrics (PSNR/SSIM [31]) on Vimeo90K and SNU-Film datasets
Bold: Best performance, underlined: Second best performance, dashed underline: Third best performance

Methods	Vimeo90K triplet	SNU-Film			
		Easy	Medium	Hard	Extreme
AMT-G [50]	36.53/ **0.982**	39.88/**0.991**	36.12/**0.981**	30.78/**0.981**	25.43/0.865
M2M-PWC [75]	35.49/0.978	39.66/**0.991**	35.74/0.980	30.32/0.980	25.07/0.863
UPRNet[49]	36.42/**0.982**	**40.44**/**0.991**	**36.29**/0.980	30.86/0.938	25.63/0.864
RIFE [37]	35.61/0.978	40.02/0.990	35.72/0.979	30.08/0.933	24.84/0.853
XVFI [48]	33.99/0.968	38.37/0.987	34.42/0.973	29.52/0.928	24.88/0.854
IFRNet [72]	36.16/0.980	40.10/**0.991**	36.12/0.978	30.63/0.936	25.27/0.861
EMA-VFI[38]	**36.64**/**0.982**	39.98/**0.991**	36.09/0.980	**30.94**/0.939	**25.69**/**0.866**
BiM-VFI[18]	35.01/0.977	40.09/0.990	35.89/0.979	30.54/0.935	25.33/0.860
Ours	34.87/0.976	39.74/0.990	35.56/0.978	30.32/0.933	25.29/0.858

Although our model does not achieve the highest PSNR and SSIM scores, its performance remains comparable across all evaluation settings. This behavior is expected, as it is primarily designed to improve perceptual quality rather than optimize pixel-level reconstruction fidelity. In video frame interpolation, higher PSNR and SSIM values do not necessarily correspond to better visual quality, particularly in regions containing large motion, occlusions, and fine textures where multiple plausible interpolations may exist.

Therefore, PSNR and SSIM are reported to provide a comprehensive evaluation from a reconstruction-fidelity perspective, while perceptual metrics and visual comparisons offer complementary insights into the visual realism of the generated frames. As shown in the following sections, our model achieves substantially better perceptual quality despite slightly lower fidelity scores, indicating a favorable trade-off between pixel-wise accuracy and perceptual realism.

Evaluation in video frame interpolation inherently involves trade-offs, as no single metric provides a one-size-fits-all assessment. Rather than relying on a solitary indicator, we report a multi-faceted evaluation strategy—spanning pixel-wise accuracy (PSNR), structural integrity (SSIM), full-reference perceptual quality (LPIPS, STLPIPS), and no-reference naturalness (NIQE)—to analyze model performance from complementary perspectives.

Tables 5.2 and 5.3 should be read together. The proposed method achieves the best LPIPS and STLPIPS on all five test sets and the best NIQE on four. However, its fidelity scores are slightly below those of the strongest baselines. Relative to BiM-VFI, PSNR is lower by 0.04–0.35 dB and SSIM by at most 0.002. This pattern is consistent with the perception–distortion trade-off described by Blau and Michaeli [97]. Under ambiguous motion, minimizing pixel-wise error can favor an average of several plausible intermediate frames. This often appears as blur or ghosting. Such an output scores well on PSNR, but it depicts a state that never occurs in the real scene, since a moving object is never observed at two positions at once. A sharper output that commits to one plausible position is penalized more heavily by PSNR when that position is offset by a fraction of a pixel, even though it is closer to the true frame in appearance.

It is important to note that the perceptual improvement reported here does not mean that the output is merely smoother or more visually pleasing. LPIPS and STLPIPS are full-reference metrics computed against the ground-truth intermediate frame. Therefore, lower values indicate greater perceptual similarity to the reference frame rather than simply favoring an arbitrary natural-looking image. The no-reference metric NIQE is reported only as a supplementary indicator, and none of the conclusions in this chapter rely on it alone. The near-identical SSIM further indicates that scene structure and object placement are preserved, since SSIM is sensitive to structural distortion. Taken together, the

results suggest that the proposed method trades a small amount of pixel-wise accuracy for improved perceptual similarity to the ground truth. This does not imply that the generated content departs from the underlying scene. Part of the PSNR gap is also attributable to the reduced model capacity targeted by this thesis rather than to the training objective. Nevertheless, interpolated frames remain estimates of unobserved instants rather than actual observations; the method is intended for visual applications such as frame-rate up-conversion and slow-motion playback, and this limitation is discussed further in Section 6.3

5.2.4 Computational complexity

To ensure reproducible inference evaluation, all models are tested under consistent hardware and implementation conditions using the same NVIDIA GeForce RTX 5060 Ti, PyTorch 2.11, and CUDA 12.9 environment. We follow standard benchmarking practices in deep learning systems. Warm-up iterations are used to mitigate compilation and GPU initialization overhead [98, 99], and inference latency is reported as average frame time over multiple runs for stability [100]. We remove initial unstable executions and outliers to reduce variance [101], and use a fixed input resolution for fair comparison [102]. Frames Per Second (FPS) and latency are standard metrics in benchmarking execution efficiency in vision frameworks [103].

	AMT-G	UPR-Net	RIFE	XVFI	BiM-VFI	Ours
Parameter (M)	30.6	6.6	9.8	5.7	6.9	6.3
FLOPS (G)	144.76	23.49	9.38	27.44	78.70	67.85

Table 5.4: Comparison of total parameter counts (in Millions) and computational complexity (FLOPs in Giga-operations) across evaluated models. All FLOPs are measured under a uniform input resolution of 256×256 pixels.

Fig. 5.9 and Table 5.4 illustrate the trade-off between peak GPU memory consumption (6.43 GB), total parameters (6.3M), computational complexity (67.85G FLOPs), and per-frame inference latency evaluated at 1920×1080 resolution. Our method achieves a favorable balance, executing significantly faster than heavy baseline architectures like AMT-G ($41,016$ ms) and UPR-Net Large (626.49 ms at 12.85 GB peak memory), as well

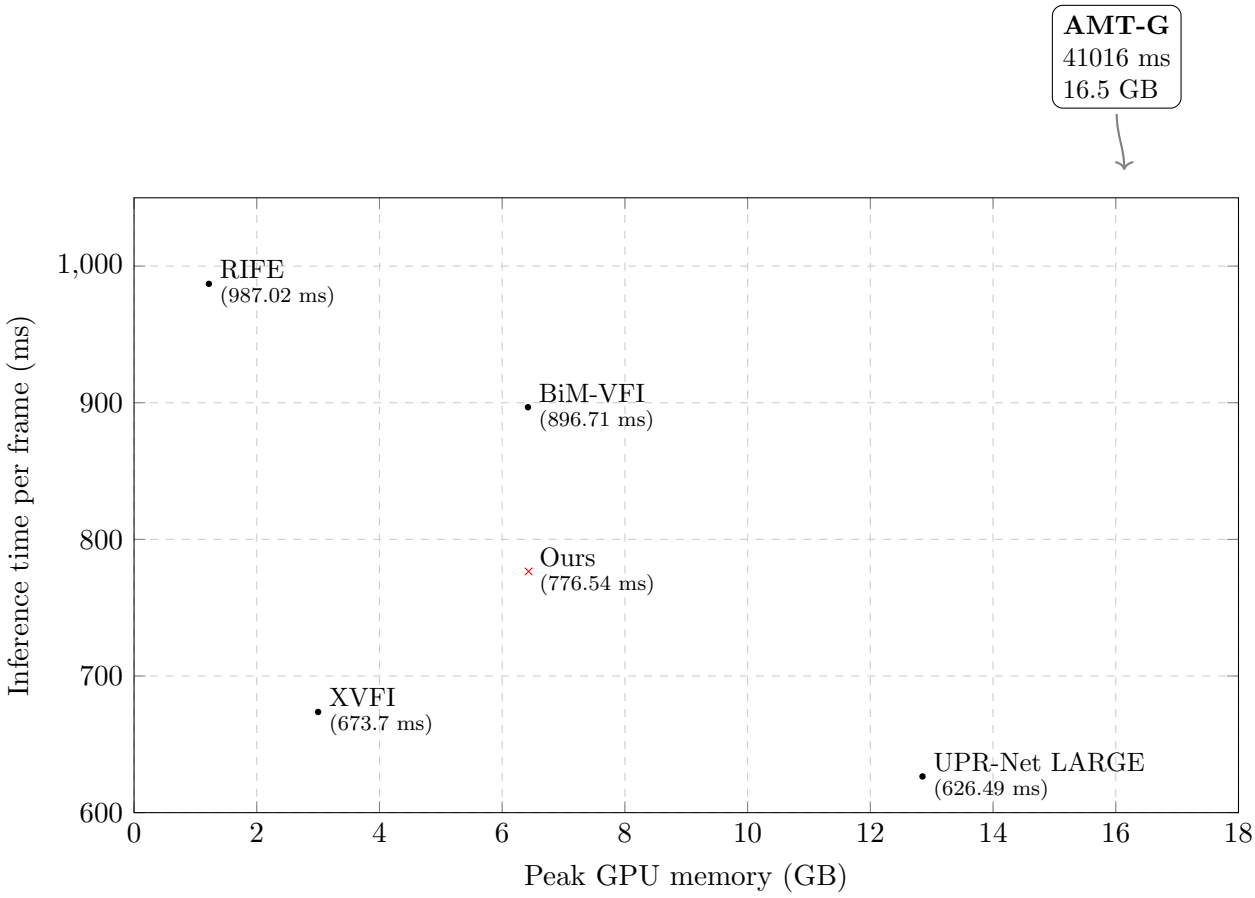


Figure 5.9: Comparison of peak GPU memory and per-frame inference time. Results are reported on an NVIDIA GeForce RTX 5060 Ti GPU with 1920×1080 input resolution.

as RIFE (987.02 ms). While UPR-Net Large is faster in raw inference time (626.49 ms), it requires nearly double the peak memory (12.85 GB) compared to our method (6.43 GB), reflecting a different efficiency trade-off. Notably, compared to our direct baseline BiM-VFI (896.71 ms with 6.42 GB peak memory and 78.70G FLOPs), our proposed framework achieves a lower inference time of 776.54 ms while maintaining a nearly identical memory footprint (6.43 GB), demonstrating improved operational efficiency alongside superior visual quality.

Table 5.5 demonstrates that GFU significantly improves parameter efficiency, reducing the parameter count of the Flow Upsampling module from 0.606 M to 0.029 M (a 95% reduction) while keeping the Feature Extraction, Motion Estimation, and Synthesis Net-

	Feature Extraction	Motion Estimation	Flow Upsampling	Synthesis Network	Total Parameters
BiM-VFI	1.154	3.414	0.606	1.700	6.878
GFU only	1.154	3.414	0.029	1.700	6.310
			↓95%		↓8.26%
MD-Net only	1.154	3.427	0.606	1.700	6.891
		↑0.38%			↑0.19%
Ours (Full Model)	1.154	3.427	0.029	1.700	6.314
		↑0.38%	↓95%		↓8.20%

Table 5.5: Number of parameters (in millions, M) for each module. Percentages (↑/↓) indicate the relative increase or decrease in parameters compared to the baseline model (BiM-VFI).

	Feature Extraction	Motion Estimation	Flow Upsampling	Synthesis Network	Total FLOPS(G)
BiM-VFI	24.04	18.81	11.00	24.64	78.70
Ours	24.04	18.84	0.16	1.700	67.85
		↑0.16%	↓98.55%		↓13.79%

Table 5.6: Per-module FLOPs comparison between the BiM-VFI baseline and GEM-VFI at 256×256 resolution.

work intact, verified in Section 5.3.3. While GFU primarily targets model compactness rather than reconstruction quality, superior visual fidelity is achieved through $\mathcal{L}_{\text{perc}}$, which provides effective supervision for fine temporal details. Crucially, $\mathcal{L}_{\text{perc}}$ is utilized exclusively during the training phase. Because it serves as a loss function rather than a network component, it introduces no additional inference parameters and thus does not alter the overall parameter counts reported in Table 5.5. This reduction in parameter count is also reflected in runtime efficiency: as shown in Fig. 5.9, our full model achieves lower inference latency and comparable GPU memory usage relative to BiM-VFI.

FLOPs are measured using ptflops [104] during inference with a single frame pair (batch size 1, RGB, $t = 0.5$) at 256×256 resolution. The count is accumulated across supported neural network modules, treating one multiply-accumulate (MAC) as one FLOP, following the BiM-VFI [18] evaluation protocol. Non-module tensor operations, including cost volume construction, warping, and convex upsampling, are excluded.

5.3 Ablation test

To ensure a fair comparison, all ablation variants are evaluated using identical training configurations, hyperparameters, and optimization schedules. Only the targeted architectural component is modified, allowing its individual contribution to overall model performance to be systematically assessed.

5.3.1 Effect of Motion Detection Network (MD-Net)

To evaluate the effectiveness of the proposed Motion Detection Network (MD-Net), we compare our full model against a baseline variant where MD-Net is removed.

Quantitative Analysis As shown in Table 5.7, incorporating MD-Net leads to consistent performance gains across all motion categories. The model with MD-Net achieves lower perceptual error scores (LPIPS/STLPIPS/NIQE) compared to the baseline, with the most notable improvements observed in fast-motion sequences. This demonstrates that MD-Net effectively enhances motion feature estimation and perceptual visual quality for complex displacement trajectories.

Table 5.7: Quantitative ablation study on MD-Net in terms of perceptual metrics (LPIPS / STLPIPS / NIQE). **Bold**: Best performance.

MD-Net	Vimeo90K triplet
without	0.021/0.013/5.315
with	**0.016/0.010/5.095**

	SNU-Film			
MD-Net	Easy	Medium	Hard	Extreme
without	0.017/0.006/4.654	0.030/0.011/4.763	0.054/0.024/4.858	0.100/0.055/4.946
with	**0.013/0.005/4.537**	**0.023/0.010/4.613**	**0.050/0.022/4.751**	**0.094/0.052/4.823**

Qualitative Analysis Figure 5.10 illustrates the visual comparison between the two variants. Without MD-Net, the baseline exhibits noticeable motion blur and ghosting artifacts around dynamic object boundaries. In contrast, integrating MD-Net resolves localized motion vectors accurately, yielding sharp and structurally preserved intermediate frames.



Figure 5.10: Visual comparison for the ablation study on MD-Net. Incorporating MD-Net effectively removes ghosting and motion blur around fast-moving regions, producing sharper intermediate frames.

5.3.2 Effect of Perceptual Loss ($\mathcal{L}_{\text{perc}}$)

To evaluate the contribution of the perceptual loss component, we conduct an ablation study comparing model performance with and without $\mathcal{L}_{\text{perc}}$. Quantitative metrics on the Vimeo90K triplet dataset and the SNU-Film benchmark across four difficulty levels are summarized in Table 5.8. The results demonstrate that integrating $\mathcal{L}_{\text{perc}}$ consistently improves perceptual quality scores across all evaluation subsets, achieving lower LPIPS, STLPIPS, and NIQE values compared to the variant trained without perceptual supervision.

Table 5.8: Quantitative ablation study on perceptual loss ($\mathcal{L}_{\text{perc}}$) in terms of perceptual metrics (LPIPS / STLPIPS / NIQE). **Bold**: Best performance.

$\mathcal{L}_{\text{perc}}$	Vimeo90K triplet
without	0.020/0.012/5.278
with	**0.016/0.010/5.095**

	SNU-Film			
$\mathcal{L}_{\text{perc}}$	Easy	Medium	Hard	Extreme
without	0.016/0.006/4.640	0.028/0.011/4.738	0.052/0.022/4.826	0.097/0.053/4.899
with	**0.013/0.005/4.537**	**0.023/0.010/4.613**	**0.050/0.022/4.751**	**0.094/0.052/4.823**

Qualitative comparisons further validate these numerical gains, as illustrated in Figure 5.11. In complex motion scenarios involving splashing water, the baseline model omitting $\mathcal{L}_{\text{perc}}$ yields noticeable motion blur and structural distortion around the swimmer's arm. Conversely, the inclusion of $\mathcal{L}_{\text{perc}}$ effectively enhances spatial sharpness and successfully reconstructs distinct fine details, such as the clear structure of the hand and fingers (indicated by the red arrow).

These visual improvements highlight a key design synergy. Although GFU substantially reduces the parameter count compared with the baseline CAUN, its lightweight architecture may involve a trade-off in reconstruction fidelity. To mitigate this limitation, perceptual loss, $\mathcal{L}_{\text{perc}}$, is incorporated during training to encourage the reconstruction of sharper details and more realistic textures. Since perceptual loss is computed only during training, it introduces no additional computational overhead at inference time. This allows GFU to benefit from improved perceptual quality while preserving its lightweight design and inference efficiency.

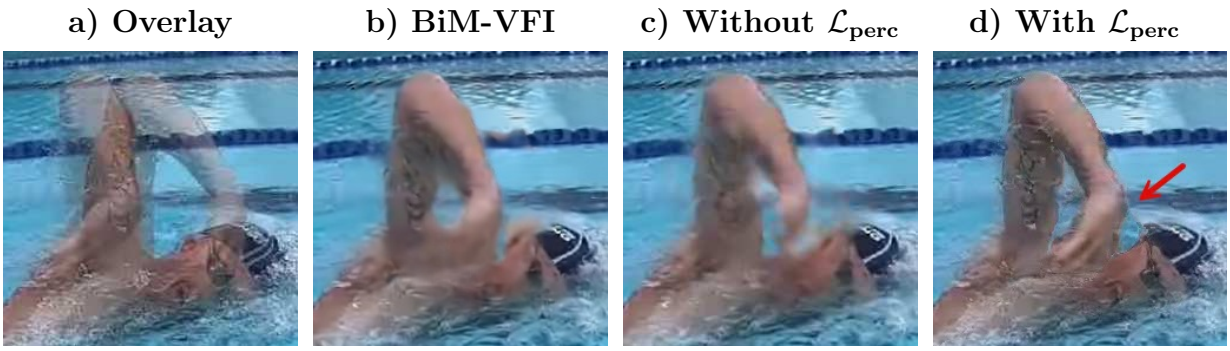


Figure 5.11: Qualitative ablation study on perceptual loss ($\mathcal{L}_{\text{perc}}$).

5.3.3 Parameter Efficiency Analysis of GFU

To verify the theoretical foundation of the 95% parameter reduction achieved by our proposed Guided Flow Upsampling (**GFU**) module compared to the baseline CAUN upsampling architecture, we conduct an analytical complexity evaluation. The parameter count for any 2D convolutional layer (Conv2D) with bias and PReLU activation is strictly defined by the closed-form equation:

$$\text{Params} = (C_{\text{out}} \times C_{\text{in}} \times K_H \times K_W + C_{\text{out}}) + N_{\text{PReLU}} \tag{5.3.1}$$

where C_{in} and C_{out} denote input/output channel dimensions, $K_H \times K_W$ represents kernel spatial size, and N_{PReLU} accounts for learnable channel slope parameters.

Table 5.9 presents the layer-by-layer parameter breakdown between the baseline CAUN architecture and our GFU module.

The closed-form calculation yields $\frac{29{,}184}{606{,}180} \approx 4.81\%$, confirming a precise **95.19% reduction** in parameter count without empirical variance. This drastic efficiency stems from three structural design choices:

1. **Single-stream Guidance Network:** Replacing CAUN's multi-scale 3-layer hierarchical encoders (enc_0, enc_1, enc_2, which account for 83.7% of CAUN's total parameters) with a lightweight, single-resolution guidance pipeline.

2. **Elimination of Sub-network Duplication:** Utilizing non-parametric bilinear interpolation for the $\times 2$ scale upsampling rather than maintaining a dedicated, parameter-heavy mask predictor sub-network.

Table 5.9: Analytical parameter comparison between the baseline CAUN upsampling module and the proposed GFU module.

Architecture	Sub-module/Layer	Channel Map ($C_{\text{in}} \rightarrow C_{\text{out}}$)	Kernel	Parameters
CAUN (Baseline)	Hierarchical Encoder 0	$256 \rightarrow 128$	3×3	295,168
	Hierarchical Encoder 1	$160 \rightarrow 128$	3×3	184,576
	Hierarchical Encoder 2	$96 \rightarrow 32$	3×3	27,712
	Predictor $\times 2$	$128 \rightarrow 64 \rightarrow 18$	3×3	84,242
	Predictor $\times 4$	$32 \rightarrow 32 \rightarrow 18$	3×3	14,482
	Total Baseline Parameters			**606,180**
GFU (Ours)	Guidance Network	$68 \rightarrow 32$	3×3	**29,184**
		$32 \rightarrow 16$	3×3	
		$16 \rightarrow 288$	1×1	

3. **Compact Input Channel Width:** Reducing incoming feature dimensions from concatenated multi-scale channels (256/160/96) to a streamlined single-scale motion-feature representation (68 channels).

5.3.4 Coarse-to-Fine vs. Fine-Only Pyramid Processing

We evaluate the importance of the coarse-to-fine pyramid hierarchy in the motion estimator (MD-Net) by comparing the default three-level configuration ($L = 3$) against a fine-only configuration ($L = 1$). Both settings use the identical trained checkpoint. Only the number of pyramid levels executed at inference differs.

As shown in Table 5.10, we use three diagnostic metrics to further analyze these differences. Cosine Similarity (Level 1) measures how well the backwarped source feature matches the target feature. A value closer to 1 indicates better alignment before correlation. Self-Consistency Error measures the mean absolute difference between the forward flow ($\mathbf{V}_{t\rightarrow 0}$) and the backward-mapped flow ($\mathbf{V}_{t\rightarrow 1}$) after warping them to the same reference frame. A lower value indicates that the two flow estimates are more consistent. L1 Error measures the mean absolute pixel-wise difference between the predicted intermediate frame $\hat{I}$ and the ground-truth frame I_t. A lower value indicates more accurate

reconstruction.

The improvement comes from two properties of the coarse-to-fine design. First, the pyramid increases the maximum detectable displacement from ±16 pixels at the finest scale to ±64 pixels at the coarsest scale without increasing the correlation kernel size. Second, motion estimated at coarser levels is propagated to finer levels to pre-align the feature maps before correlation. This increases the Level 1 cosine similarity from 0.136 to 0.864, allowing large motion to be handled as smaller residual displacements.

Table 5.10: Ablation study comparing Coarse-to-Fine ($L = 3$) and Fine-Only ($L = 1$) modes on the full Vimeo90K test set (3,782 triplets).

Evaluation Scope	**Metric**	**Coarse-to-Fine** ($L = 3$)	**Fine-Only** ($L = 1$)
Feature Pre-alignment	Cosine Similarity (Level 1) ↑	**0.864**	0.136
Self-Consistency Proxy	Self-Consistency Error ↓	**0.0183**	0.0249
Vimeo90K Test Set	PSNR (dB) ↑	**34.761**	33.440
(Full: 3,782 triplets)	SSIM ↑	**0.9756**	0.9624
	L1 Error ↓	**0.0091**	0.0114

Chapter 6

Conclusions and Future Work

6.1 Thesis Contributions

Through a review of existing VFI architectures, we identified BiM-VFI [18] as the strongest baseline among non-generative frameworks, based on its Fréchet Inception Distance (FID) advantage over generative approaches such as SNM-VFI on DAVIS. This established the foundation for the remaining three contributions of this thesis.

Building on BiM-VFI, we identified three limitations. First, its motion representation struggles to capture complex, non-rigid motion, such as an athlete's body deforming during motion. Second, its flow upsampling module (CAUN) has a high parameter cost relative to its contribution. Third, its reconstructed frames tend to be over-smoothed and lack fine texture details.

To address these limitations, we proposed GEM-VFI, an enhanced BiM-VFI framework. It improves motion representation, flow upsampling efficiency, and perceptual output quality through three complementary contributions: Motion Detection Network (MD-Net), Guided Flow Upsampling (GFU), and perceptual loss supervision.

MD-Net strengthens motion-aware feature refinement through Squeeze-and-Excitation (SE) channel attention, yielding consistent perceptual gains across all evaluated datasets. GFU replaces the multi-scale warping-dependent upsampling of CAUN with a lightweight guidance-based convex combination, substantially reducing inference cost with only minor quality trade-off. Perceptual loss supervision, applied via a pre-trained VGG16 network, provides several benefits. It encourages the model to recover high-frequency spatial details, including fine textures, sharp edges, and small object boundaries. These details are often

suppressed by pixel-wise losses due to averaging effects. As a result, the combination of these components enables GEM-VFI to produce interpolated frames with improved perceptual sharpness and better detail preservation compared to the BiM-VFI baseline.

These findings suggest that jointly optimizing motion representation, flow upsampling, and perceptual supervision is an effective direction for building efficient VFI systems under resource constraints.

6.2 Discussion: Component transferability to Other Models

The transferability of our proposed contributions to other VFI architectures, depends closely on their underlying structural and operational paradigms:

- **Squeeze-and-Excitation (SE) Block (a component of Motion Detection Network (MD-Net)):** The channel-attention block inside MD-Net is transferable to any architecture that performs feature matching through a correlation or cost-volume layer (e.g., UPR-Net, AMT). Placed immediately before the correlation operator, which is itself non-parametric, it acts as a learnable, input-adaptive similarity metric and is the only point at which the matching process can be reweighted before the cost volume discards per-channel information. Architectures that regress flow directly from images without a matching step (e.g., RIFE) offer no such insertion point; the static part of the per-channel scaling is absorbed by the following convolution, and what remains is generic channel gating with limited expected benefit. For UPR-Net, the benefit is also expected to be smaller than in our model because its correlation input is forward-splatted and therefore degraded spatially (holes and averaged overlaps), which a global per-channel weight cannot correct.
- **MD-Net as a whole:** Transferring the complete module is infeasible, since MD-Net is a full motion estimator rather than an add-on. It requires a dedicated feature extractor and a local cost volume, and inserting it would replace the native estimator of the host model, discarding its defining mechanism: the recurrent feature state of UPR-Net, the all-pairs correlation lookup of AMT, or the direct image-domain regression of RIFE. The resulting model would be a new hybrid rather than an integration.

- **GFU (Upsampling):** GFU can be applied to pipelines that estimate motion at a lower resolution and then use a separate upsampling step before image reconstruction, as in UPR-Net and RIFE, although some adaptation is required. For UPR-Net, the guidance network must be adjusted to match the feature dimensions of each model variant, and a substitute must be provided for the occlusion input, which UPR-Net does not predict. For RIFE, each block uses a different upsampling factor, requiring a more flexible design. In addition, guidance must be extracted from internal features rather than a feature pyramid, and the occlusion mask must be handled separately. In contrast, GFU is not directly applicable to AMT, which integrates flow upsampling into its multi-level refinement process. Therefore, GFU is adaptable to different pipelines, but it is not a drop-in replacement.

- **Perceptual Loss Supervision:** Unlike the structural modules, the perceptual loss imposes no architectural requirement: it operates only on the final RGB prediction and the ground truth, both bounded in $[0, 1]$ in all three models, and is independent of flow representation and warping scheme. It is best applied as a low-weight fine-tuning stage on a checkpoint already trained with pixel-wise losses, so that it sharpens texture without distorting the flow geometry learned beforehand.

These findings indicate that component integration in deep VFI models is constrained by feature matching dynamics, flow grid assumptions, and upsampling structures rather than acting as universal drop-in solutions.

6.3 Limitations

While the proposed method achieves strong perceptual quality across diverse benchmarks, certain limitations remain. Because motion estimation relies on local structural cues, dense and repetitive patterns—such as fine pool drainage grates—can cause spatial ambiguity. When adjacent elements share near-identical geometry, the deformation field struggles to resolve distinct correspondences. Consequently, the model averages plausible positions and produces localized texture blurring to minimize reconstruction error.

As illustrated in Figure 6.1, this spatial ambiguity affects both baseline models and the proposed method when processing dense, adjacent patterns. Across all evaluated methods, tracking errors on the fine pool grate manifest either as severe texture blurring or as artificial position shifts relative to the underlying geometry.

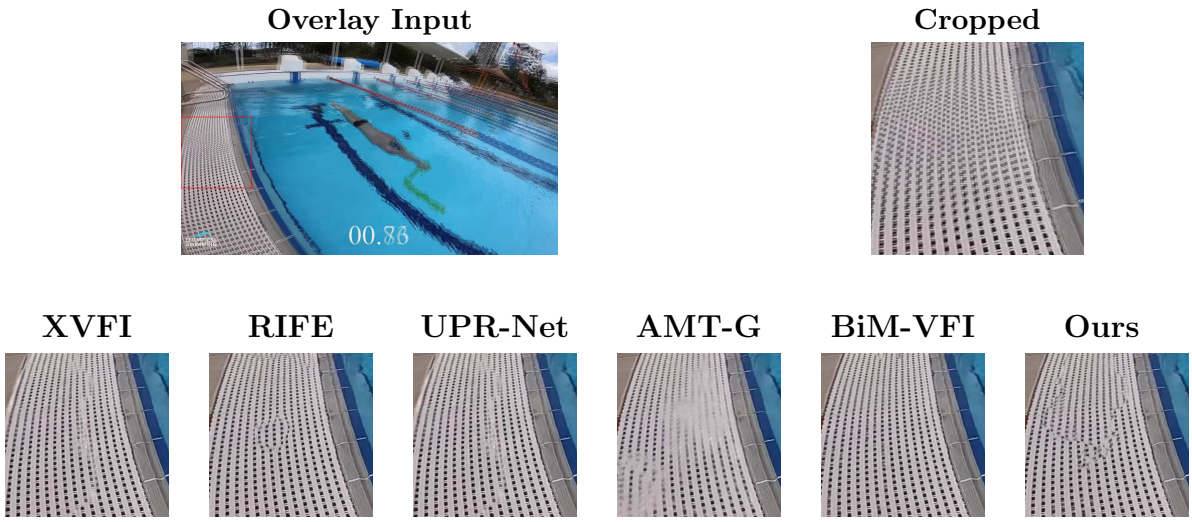


Figure 6.1: Qualitative comparison on a sequence with a dense repetitive pattern. Under ambiguous local motion, all methods exhibit texture blurring or spatial position shifts.

6.4 Future Work

Future work could build upon these findings in two main directions. First, integrating spatially-adaptive upsampling or distortion-aware perceptual supervision could further refine complex motion boundaries and mitigate VFI-specific artifacts. Second, exploring hardware-oriented optimizations such as model quantization and structured pruning will be key to improving the execution efficiency of VFI models for resource-constrained applications.

References

[1] 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 (CVPR)*, pages 9000–9008, 2018.

[2] Xiaoyu Xiang, Yapeng Tian, Yulun Zhang, Yun Fu, Jan P. Allebach, and Chenliang Xu. Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3370–3379, 2020.

[3] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yu Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4463–4471, 2017.

[4] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 670–679, 2017.

[5] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1710, 2018.

[6] Tomer Peleg, Peter Szekely, Daniel Sabo, and Orit Sendik. Im-net for high resolution video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2398–2407, 2019.

[7] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9308–9316, 2019.

[8] 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 (TPAMI)*, 43(3):933–948, 2019.

[9] Xianing Cheng and Zhenzhong Chen. Video frame interpolation via deformable separable convolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10607–10614, 2020.

[10] Junheum Park, Kwanghoon Ko, Chulwoo Lee, and Chang-Su Kim. BMBC: Bilateral Motion Estimation with Bilateral Cost Volume for Video Interpolation. In *European Conference on Computer Vision (ECCV)*, pages 109–125. Springer International Publishing, 2020.

[11] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Donghyeon Pak, Yoonwoo Ban, and Seungjun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5316–5325, 2020.

[12] Shuhang Gui, Chen Wang, Qi Chen, and Dacheng Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14004–14013, 2020.

[13] Zhen Lyu, Ming Li, Jian Jiao, and Chun Chen. Frame interpolation with consecutive brownian bridge diffusion. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, pages 3449–3458, 2024.

[14] Zhen Zhang, Hao Chen, He Zhao, Guang Lu, Yun Fu, Huan Xu, and Zhen Wu. Eden: Enhanced diffusion for high-quality large-motion video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[15] Cheng Liu, Hao Yang, Jian Fu, and Xue Qian. TTVFI: Learning trajectory-aware transformer for video frame interpolation. *IEEE Transactions on Image Processing*, 32:4728–4741, 2023.

[16] Dahyeon Kye, Changhyun Roh, Sukhun Ko, Chanho Eom, and Jihyong Oh. AceVFI: A comprehensive survey of advances in video frame interpolation. *arXiv preprint arXiv:2506.01061*, 2025.

[17] Jiaben Chen and Huaizu Jiang. Sportsslomo: A new benchmark and baselines for human-centric video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6475—-6486, 2024.

[18] Wonyong Seo, Jihyong Oh, and Munchurl Kim. BiM-VFI: Bidirectional motion field-guided frame interpolation for video with non-uniform motions. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 7244–7253, 2025.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL https://www.deeplearningbook.org/.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[24] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[25] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[28] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[29] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[30] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[31] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

[32] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[33] Akash Ghildyal and Feng Liu. Shift-tolerant perceptual similarity metric. In *Proceedings of the European Conference on Computer Vision*, pages 91–107. Springer, 2022.

[34] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2012.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[37] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision*, pages 624–642. Springer, 2022.

[38] Kun Zhou, Wenbo Li, Xiaoguang Han, and Jiangbo Lu. Exploring motion ambiguity and alignment for high-quality video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22169–22179, 2023.

[39] Tian Shu, Fuqiang Bai, Xinxin Chen, Yuantong Zhang, Jiaxin Chen, Wenzhuo Ma, Daiqin Yang, and Zhenzhong Chen. Motion-guided multi-scale fusion for high-fps video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2518–2525, 2026.

[40] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, pages 63–79, 2018.

[41] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1905–1914, 2021.

[42] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, pages 7324–7334, 2019.

[43] Duolikun Danier, Fan Zhang, and David Bull. Flolpips: A bespoke video quality metric for frame interpolation. In *2022 Picture Coding Symposium (PCS)*, pages 283–287, 2022.

[44] Daniel L Ruderman and William Bialek. The statistics of natural images. *Physical Review Letters*, 73(6):814–817, 1994.

[45] Z. Zhong, Y. Zhang, W. Wang, X. Sun, Y. Qiao, G. Krishnan, S. Ma, and J. Wang. Velocity Disambiguation for Video Frame Interpolation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(7):7938–7953, 2026. doi: 10.1109/TPAMI.2026.3667437.

[46] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[47] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10663–10671, 2020.

[48] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: Extreme Video Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14489–14498, 2021.

[49] Xin Jin, Longhai Wu, Jie Chen, Youxin Chen, Jayoon Koo, and Cheul hee Hahm. A unified pyramid recurrent network for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1578–1587, 2023.

[50] Zhen Li, Zuo-Liang Zhu, Ling-Hao Han, Qibin Hou, Chun-Le Guo, and Ming-Ming Cheng. AMT: All-Pairs Multi-Field Transforms for Efficient Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9801–9810, 2023.

[51] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. CDFI: Compression-Driven Network Design for Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8001–8011, June 2021.

[52] Li Lu, Rong Wu, Hong Lin, Jiwen Lu, and Jiaya Jia. Video frame interpolation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3532–3542, 2022.

[53] Duolikun Danier, Fan Zhang, and David Bull. LDMVFI: Video Frame Interpolation with Latent Diffusion Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1472–1480, 2024.

[54] Vikram Voleti, Alex Linares-López, and Christopher Pal. Masked conditional video diffusion for prediction, generation, and interpolation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[55] Jisoo Jeong, Hong Cai, Jamie Menjay Lin, Hanno Ackermann, Hyeonjun Sim, Yinhao Zhu, Yunxiao Shi, and Fatih Porikli. SNM-VFI: Symmetric Nonlinear Motion-Guided Generative Video Frame Interpolation. *arXiv preprint arXiv:2608.13460*, 2026.

[56] Xuan Peng, Hao Li, Yifan Huang, Zhi Zheng, Yi Wang, Xi Chen, others, and Hui Xiong. Towards holistic modeling for video frame interpolation with auto-regressive diffusion transformers. *arXiv preprint arXiv:2601.14959*, 2026.

[57] Zhenyu Lyu and Chao Chen. TLB-VFI: Temporal-aware Latent Brownian Bridge Diffusion for Video Frame Interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16260–16269, 2025.

[58] Berthold K P Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[59] Simon Baker, Daniel Scharstein, JP Lewis, et al. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92:1–31, 2011.

[60] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4463–4471, 2017.

[61] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019.

[62] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3703–3712, 2019.

[63] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655. IEEE, 2017.

[64] Rui Xu, Xiaoxiao Li, Baigui Zhang, et al. Deep flow-guided video inpainting. In *CVPR*, 2019.

[65] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4161–4170, 2017.

[66] Peter J. Burt and Edward H. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4):532–540, 1983. doi: 10.1109/TCOM.1983.1095851.

[67] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*, pages 402–419. Springer, 2020.

[68] Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *Proceedings of the European Conference on Computer Vision*, pages 36–54. Springer, 2024.

[69] Azin Jahedi, Lukas Mehl, Marc Rivinius, and Andrés Bruhn. Multi-Scale RAFT: Combining Hierarchical Concepts for Learning-Based Optical Flow Estimation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1236–1240. IEEE, 2022. doi: 10.1109/ICIP46576.2022.9897992.

[70] A. Jahedi, M. Luz, M. Rivinius, L. Mehl, and A. Bruhn. MS-RAFT+: High Resolution Multi-Scale RAFT. *International Journal of Computer Vision*, 132(5):1835–1856, 2024.

[71] Stefan Roth Kiefhaber, Simon and Simone Schaub-Meyer. Removing cost volumes from optical flow estimators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 79–89, 2025.

[72] Lingtong Kong, Boyuan Jiang, Donghao Luo, Xiaoming Huang Wenqing Chu, Ying Tai, Chengjie Wang, and Jie Yang. Ifrnet: Intermediate feature refine network for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1969–1978, 2022.

[73] Jisoo Jeong, Hong Cai, Risheek Garrepalli, Jamie Menjay Lin, Munawar Hayat, and Fatih Porikli. OCAI: Improving Optical Flow Estimation by Occlusion and Consistency Aware Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19352–19362, June 2024.

[74] Chenyang Wu, Jiayi Fu, Chun-Le Guo, Shuhao Han, and Chongyi Li. Vtinker: Guided flow upsampling and texture mapping for high-resolution video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 10638–10645, 2026.

[75] Ping Hu, Simon Niklaus, Stan Sclaroff, and Kate Saenko. Many-to-many splatting for efficient video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3553–3562, 2022.

[76] Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Siow Mong Goh, and Hongyuan Zhu. Craft: Cross-attentional flow transformer for robust optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17602–17611, 2022.

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

[78] Alexander Gielisse, Nergis Tömen, and Jan van Gemert. Local Attention Transformers for High-Detail Optical Flow Upsampling. *arXiv preprint arXiv:2412.06439*, 2024.

[79] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[80] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[81] Lin Wang and Kuk-Jin Yoon. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3048–3068, 2022. doi: 10.1109/TPAMI.2021.3055564.

[82] Yifan Liu, Jiahui Cao, Bo Li, Chen Yuan, and Han Hu. Knowledge distillation via instance relationship graph. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[83] Crispian Morris, Duolikun Danier, Fan Zhang, Nantheera Anantrasirichai, and David R. Bull. ST-MFNet Mini: Knowledge Distillation-Driven Frame Interpolation. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 1045–1049. IEEE, 2023.

[84] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *International Conference on Learning Representations*, 2015. URL https://mlanthology.org/iclr/2015/romero2015iclr-fitnets/.

[85] Daniel Danier, Fan Zhang, and David Bull. ST-MFNet: A Spatio-Temporal Multi-Flow Network for Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3521–3531, 2022.

[86] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash

equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, pages 6626–6637, 2017.

[87] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[88] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-Time Intermediate Flow Estimation for Video Frame Interpolation. In *European Conference on Computer Vision (ECCV)*, pages 624–642, 2022.

[89] Zhaoyang Jia, Yan Lu, and Houqiang Li. Neighbor Correspondence Matching for Flow-based Video Frame Synthesis. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 5389–5397. ACM, 2022.

[90] Shili Zhou, Weimin Tan, and Bo Yan. A Motion Distillation Framework for Video Frame Interpolation. *IEEE Transactions on Multimedia*, 26:3728–3740, 2024. doi: 10.1109/TMM.2023.3314971.

[91] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. CARAFE: Content-aware reassembly of features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3007–3016, 2019.

[92] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (TOG)*, 26(3):96:1–96:6, 2007.

[93] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2010.

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

[95] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 10663–10671, 2020.

[96] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[97] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[98] Google JAX Team. Jax performance and compilation behavior. https://jax.readthedocs.io, 2024.

[99] M. Pochelu. On the impact of jit warmup and compilation overhead in deep learning benchmarks. *arXiv preprint arXiv:2210.04323*, 2022.

[100] A. et al. Koubaa. Reliable evaluation of real-time deep learning systems: Why average fps matters. *arXiv preprint arXiv:2409.16808*, 2024.

[101] Y. et al. Zhou. Autoscratch: Reducing variance in gpu benchmarking for deep learning. In *Proceedings of MLSys*, 2023.

[102] J. et al. Huang. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.

[103] P. L. Asselin, V. Coulombe, W. Guimont-Martin, and W. Larrivée-Hardy. Replication study and benchmarking of real-time object detection models. *arXiv preprint arXiv:2405.06911*, 2024.

[104] Vladislav Sovrasov. ptflops: a flops counting tool for neural networks in pytorch framework. https://github.com/sovrasov/flops-counter.pytorch, 2018–2024. Version 0.7.5.