Range-nullspace Video Frame Interpolation with Focalized Motion Estimation

Zhiyang Yu[1,*], Yu Zhang[2,✉], Dongqing Zou[2,3], Xijun Chen[1], Jimmy S. Ren[2,3], Shunqing Ren[1,✉]
[1] Harbin Institute of Technology, Harbin, China
[2] SenseTime Research and Tetras.AI, Beijing, China
[3] Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, China

Abstract

Continuous-time video frame interpolation is a fundamental technique in computer vision for its flexibility in synthesizing motion trajectories and novel video frames at arbitrary intermediate time steps. Yet, how to infer accurate intermediate motion and synthesize high-quality video frames are two critical challenges. In this paper, we present a novel VFI framework with improved treatment for these challenges. To address the former, we propose focalized trajectory fitting, which performs confidence-aware motion trajectory estimation by learning to pay focus to reliable optical flow candidates while suppressing the outliers. The second is range-nullspace synthesis, a novel frame renderer cast as solving an ill-posed problem addressed by learning decoupled components in orthogonal subspaces. The proposed framework sets new records on 7 of 10 public VFI benchmarks.

1. Introduction

Continuous-time Video Frame Interpolation (VFI) aims at upsampling the temporal resolution of low frame-rate videos steplessly by synthesizing the missing frames at arbitrary time steps. It is a fundamental technology for various downstream video applications such as streaming [34], stabilization [5], and compression [35].

A key challenge of this task is to find a continuous mapping from arbitrary time step to the latent scene motion to correctly render the target frame, observing the low frame-rate video. Typically, it was realized via two stages: motion trajectory fitting and frame synthesis, *e.g.* [1, 3, 4, 10, 13, 17, 21, 29, 36, 38, 39]. In the former, a parametric trajectory model is fitted from the optical flows extracted from input frames, which can be resampled at any time step to get intermediate motion. For representing such a motion model,

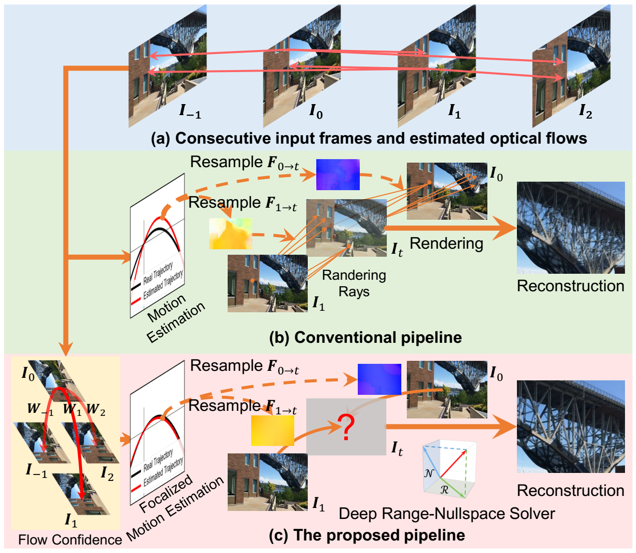

I_{-1}
I_0
I_1
I_2
(a) Consecutive input frames and estimated optical flows
Resample $F_{0\to t}$
Resample $F_{1\to t}$
Motion Estimation
I_t
Rendering
Randering Rays
Reconstruction
(b) Conventional pipeline
W_{-1} W_1 W_2
Flow Confidence
Focalized Motion Estimation
Deep Range-Nullspace Solver
(c) The proposed pipeline

Figure 1. Motivation of the proposed method. Given input frames and extracted optical flows (a), common VFI pipeline (b) is to fit a continuous motion trajectory model, resample flows at the target time step, based on which rendering rays are predicted to synthesize the intermediate frame by reorganizing pixels from the input frames. Our pipeline (c) proposes improved treatment in two aspects: 1) our focalized motion estimation assigns dynamic weights to the extracted optical flows to suppress the outlier flows and improve fitting accuracy, and the novel range-nullspace solver treats intermediate frame synthesis as an inverse problem instead of direct rendering. Please see text for more details.

the Taylor polynomial is often adopted with different orders [4,13,36]. However, fitting trajectory models are prone to optical flow errors, which are inevitable due to occlusion.

Besides motion fitting, the frame synthesis step faces the challenges of correctly inferring scene geometry. Particularly, the intermediate flows resampled from the continuous motion model only depict partial correspondences between the intermediate frame with input frames, while existing methods predict complete rendering rays in a data-driven manner to facilitate full rendering, as shown in Fig. 1 (b).

[*] This work is done during Zhiyang's internship at SenseTime.
✉ Correspondence should be addressed to Yu Zhang (zhangyulb@gmail.com) and Shunqing Ren (renshunqing@hit.edu.cn).

Though powerful architectures [15] and rendering mechanisms [10] were proposed, it requires the network to fully encode the scene geometry to predict correct rays, which may raise the difficulty of architecture design and the burden of learning.

To overcome these issues, we propose a novel framework with improved motion fitting and frame synthesis components. As shown in Fig. 1 (c), the first is focalized trajectory fitting, which extracts a set of optical flow candidates from the input video and assigns learned confidence weights to them. Confidence-aware, differentiable trajectory fitting is then followed, focalizing only the confident flows and producing improved parametric motion models. Such models, when resampled at the target time step, can produce accurate flows even at the occlusion boundary. For the latter, we follow the fact that intermediate frame synthesis is an ill-posed task and propose a deep solver based on the range nullspace theory [2]. Our solver decomposes the latent intermediate frame into several orthogonal image components and adopts different networks to learn each of them. Such physics-guided design encourages learning decoupled subtasks for each network, relieving the burden of encoding scene geometry and benefiting the use of lightweight models. In particular, our framework sets new records on 7 out of 10 public VFI benchmarks while being parameter-efficient.

We summarize the contributions of this paper as follows. 1) A novel lightweight VFI framework is proposed, which refreshes the records of 7 out of 10 public VFI benchmarks. 2) The idea of focalized trajectory fitting, which improves parametric motion estimation in VFI and generates better resampling quality of intermediate flows. 3) A new perspective that treats intermediate frame synthesis as an ill-posed problem, solved with a deep range nullspace solver that decouples frame synthesis into several orthogonal tasks.

2. Related works

Video frame interpolation requires estimating the motion of intermediate frames before synthesizing them. Such motion could be directly predicted as image correspondences in a data-driven manner [11, 15], or evaluated from parameterized motion trajectories fitted with optical flows extracted from input frames. Typically the latter one is more flexible to do frame interpolation at any time step. For representing the motion model, linear motion is mostly assumed [1, 3, 10, 13, 21, 29], while quadratic [17, 36, 38, 39] and cubic [4] models were proposed and show improved performance with higher orders of freedom. Frame synthesis was then carried out by warping the information from input frames, guided by the estimated motion. This involves synthesis techniques of backward warping [13, 17, 29, 36] or forward splatting [10, 21], depending on the direction of intermediate flows. High-quality synthesis poses the demand of reasoning occlusion, which was implemented explicitly with scene depth [1] or implicitly via powerful post-processing architectures [11, 17]. VFI can also be implemented in a highly data-driven manner [14], abstracting out all the above steps with neural networks.

There was also an important track of VFI that only synthesizes the middle frame of a pair of input frames. Without the need of representing motion continuously and interpolating arbitrary time steps, sophisticated components can be designed to optimally fit this task. For example, intermediate frame synthesis was solved with bilateral cost volumes [23], asymmetric bilateral volumes [24], expanded flows [8], and deformable convolutions [7]. There was also kernel-based method [22], with enhanced variants [3,16,27] that introduce more freedom into standard kernels with learned offsets and weights. Recent advances of deep learning architectures, like transformer [28] and channel attention modules [6], are also beneficial.

Range-nullspace learning provides a solver for inverse problems where the forward degeneration is a known linear operator [26]. It finds applications in solving super-resolution [2], denoising [33] and compressed sensing [19]. Treating video frame interpolation as an inverse problem was first performed by [39], which developed iterative solvers based on learned gradients under the Bayesian inference framework. In this work, we adapt the range-nullspace theory to VFI and propose a non-iterative solver that learns orthogonally decomposed image components to relieve the entanglement of learned representations.

3. The Proposed Framework

3.1. Overview

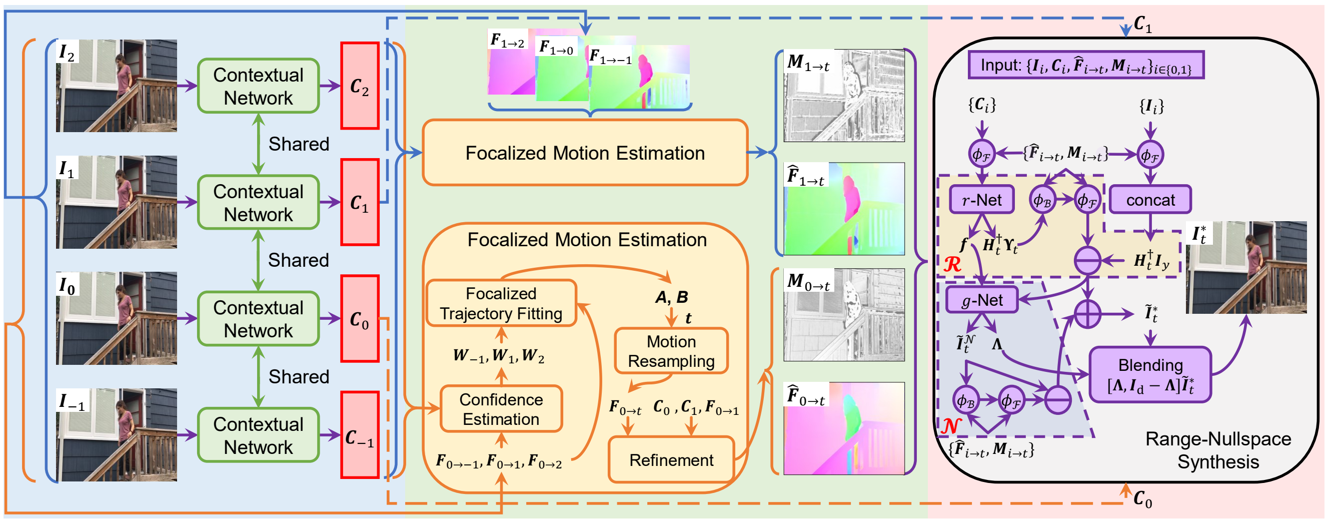
Following the routine of previous works [17, 36, 38, 39], our framework takes 4 adjacent input video frames $\mathbf{I}_{-1}, \mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2 \in [-1, 1]^{H\times W\times 3}$ and generates intermediate frame $\mathbf{I}_t$ for any given time $t \in (0, 1)$. As shown in Fig. 2, our pipeline starts by extracting appearance features for each input frame with a contextual network $\mathcal{C}$:

$$\mathbf{C}_i = \mathcal{C}(\mathbf{I}_i), i \in \{-1, 0, 1, 2\}. \tag{1}$$

In the meantime, optical flows starting from $\mathbf{I}_0$, $\mathbf{I}_1$ to other neighbours are computed. This yields 6 optical flow maps $\mathbf{F}_{i\to j} \in \mathbb{R}^{H\times W\times 2}$, where $i \in \{0, 1\}$ and $j \neq i$.

The Focalized Motion Estimation (FME) in Fig. 2 consumes optical flows and contextual features. It consists of three steps: flow confidence estimation, focalized trajectory fitting, motion resampling and refinement, and outputs motion specifications at a given time t. Bi-directional motions are inferred, regarding both time 0 and 1:

$$\begin{aligned}\hat{\mathbf{F}}_{0\to t}, \mathbf{M}_{0\to t} &= \mathtt{FME}(t, \{\mathbf{F}_{0\to j}\}_{j\neq 0}, \{\mathbf{C}_i\}_{\forall i}),\\ \hat{\mathbf{F}}_{1\to t}, \mathbf{M}_{1\to t} &= \mathtt{FME}(1-t, \{\mathbf{F}_{1\to j}\}_{j\neq 1}, \{\mathbf{C}_i\}_{\forall i}),\end{aligned} \tag{2}$$



I_2
I_1
I_0
I_{-1}
Contextual Network
Shared
C_2
C_1
C_0
C_{-1}
$F_{1\to2}$ $F_{1\to0}$ $F_{1\to-1}$
Focalized Motion Estimation
Focalized Trajectory Fitting
A, B
t
W_{-1}, W_1, W_2
Motion Resampling
Confidence Estimation
$F_{0\to t}$ $C_0, C_1, F_{0\to1}$
$F_{0\to-1}, F_{0\to1}, F_{0\to2}$
Refinement
$M_{1\to t}$
$\hat{F}_{1\to t}$
$M_{0\to t}$
$\hat{F}_{0\to t}$
Input: $\{I_i, C_i, \hat{F}_{i\to t}, M_{i\to t}\}_{i\in\{0,1\}}$
$\{C_i\}$
$\{I_i\}$
$\phi_{\mathcal{F}}$
$\{\hat{F}_{i\to t}, M_{i\to t}\}$
r-Net
$\phi_{\mathcal{B}}$
concat
f $H_t^\dagger Y_t$
$\mathcal{R}$
$H_t^\dagger I_y$
I_t^*
g-Net
$\tilde{I}_t^*$
$\tilde{I}_t^N$ Λ
Blending $[\Lambda, I_d - \Lambda]\tilde{I}_t^*$
$\mathcal{N}$
$\{\hat{F}_{i\to t}, M_{i\to t}\}$
Range-Nullspace Synthesis
C_1
C_0

Figure 2. Pipeline of our proposed method including contextual feature extraction, flow estimation, focalized motion estimation (FME), and range-nullsapce synthesis (RNS). Detailed descriptions of FME (middle) can be found in Sect. 3.2, and RNS (right) is elaborated in Sect. 3.3. For the details of mathematical symbols, please refer to the text.

Here, $\hat{\mathbf{F}}_{i\to t}$ and $\mathbf{M}_{i\to t}$, $i \in \{0,1\}$ are the resampled flows and occlusion relation masks to be explained in Sect. 3.2. Finally, we feed these motion specifications at time t and contextual features of input frames $\mathbf{I}_0$, $\mathbf{I}_1$ into the Range-Nullspace Synthesis (RNS) module,

$$\mathbf{I}_t^* = \texttt{RNS}\left(\{\hat{\mathbf{F}}_{i\to t}, \mathbf{M}_{i\to t}, \mathbf{C}_i\}_{i\in\{0,1\}}\right). \tag{3}$$

In the following, we will elaborate technical details of the proposed FTE and RNS modules.

3.2. Focalized Motion Estimation (FME)

For the sake of conciseness, only the direction $0 \to t$ in Eq. (2) is considered here, as $1 \to t$ follows the similar process. Given contextual features $\mathbf{C}_i$ and candidate flows $\{\mathbf{F}_{0\to j}\}_{j\neq0}$, the confidence estimation step dynamically weights optical flow candidates. To this end, the contextual features $\mathbf{C}_i$ are half split along the channel dimension, yielding $[\mathbf{C}_i^a, \mathbf{C}_i^b]$, and the confidence estimation only works with the right half $\mathbf{C}_i^b$ to save calculation and leave the representation ability of remaining features untouched. The following warping residual is computed:

$$\mathbf{R}_i = \mathbf{C}_0^b - \phi_{\mathcal{B}}(\mathbf{C}_i^b, \mathbf{F}_{0\to i}),\ i \in \{-1,1,2\}, \tag{4}$$

where $\phi_{\mathcal{B}}$ represents the bilinear warping function [12] that projects features $\mathbf{C}_i^b$ to time 0 with backward warping. Intuitively, the residuals $\mathbf{R}_i$ conveys rich information of appearance warping errors induced by flows. We map such residuals to confidence weights via a network $\mathcal{W}$:

$$\mathbf{W}_{-1}, \mathbf{W}_1, \mathbf{W}_2 = \mathcal{W}(\mathbf{R}_{-1}, \mathbf{R}_1, \mathbf{R}_2), \tag{5}$$

where $\mathbf{W}_i \in (0,1)^{H\times W\times 1}$ and $\sum_{i\in\{-1,1,2\}} \mathbf{W}_i = 1$. Higher value of $\mathbf{W}_i$ indicates higher confidence of the flow values in $\mathbf{F}_{0\to i}$, which deserves more focus for fitting parametric motion trajectory models.

Focalized trajectory fitting. We suppose that, in a short time interval around each pixel of $\mathbf{I}_0$, the motion trajectory is approximated with 2nd-order Taylor polynomial:

$$f(\tau; \mathbf{A}, \mathbf{B}) = \mathbf{A}\tau^2 + \mathbf{B}\tau,\ \mathbf{A}, \mathbf{B} \in \mathbb{R}^{H\times W\times 2}, \tag{6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are model coefficients to fit, and $\tau \in [-1, 2]$ is the temporal distance *w.r.t.* time 0. Solving $\mathbf{A}$ and $\mathbf{B}$ can be formulated with the following weighted least squares:

$$\min_{\mathbf{A},\mathbf{B}} \sum_{i\in\{-1,1,2\}} \|\sqrt{\mathbf{W}_i} \odot (\mathbf{F}_{0\to i} - f(i; \mathbf{A}, \mathbf{B}))\|_2^2, \tag{7}$$

in which $\odot$ denotes element-wise product (with broadcasting along channel dimension). The optimal solution of this weighted least squares has the analytical form

$$\hat{\mathbf{\Theta}}_{x,y,z} = (\mathbf{X}\mathbf{W}_{x,y}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{W}_{x,y}\mathbf{F}_{x,y,z}, \tag{8}$$

where x, y and z index the vertical, horizontal and channel dimensions of tensors, $\hat{\mathbf{\Theta}}_{x,y,z} = \left(\hat{\mathbf{A}}(x,y,z), \hat{\mathbf{B}}(x,y,z)\right)^{\mathrm{T}}$ is the solved coefficients. Other matrices are:

$$\begin{gathered}\mathbf{W}_{x,y} = \begin{bmatrix} \mathbf{W}_{-1}(x,y,0) & & \\ & \mathbf{W}_1(x,y,0) & \\ & & \mathbf{W}_2(x,y,0)(x,y) \end{bmatrix}, \\ \mathbf{X} = \begin{bmatrix} 1 & 1 & 4 \\ -1 & 1 & 2 \end{bmatrix}, \mathbf{F}_{x,y,z} = \begin{bmatrix} \mathbf{F}_{0\to-1}(x,y,z) \\ \mathbf{F}_{0\to1}(x,y,z) \\ \mathbf{F}_{0\to2}(x,y,z) \end{bmatrix}.\end{gathered} \tag{9}$$

All the calculations of Eq. (8) are fast to compute, including the 2×2 matrix inverse which is closed-form. This

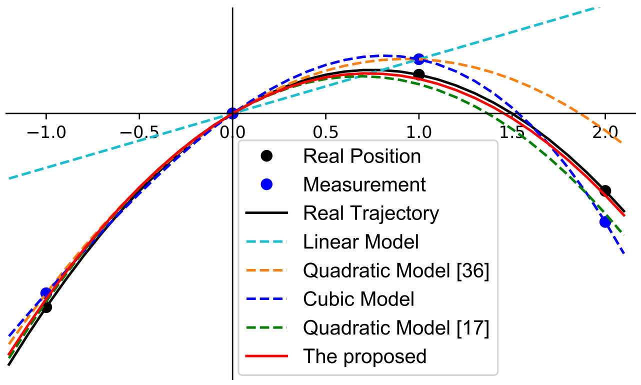


Figure 3. A toy 1-D example comparing different trajectory fitting methods. Linear model [13] is not adequate to approximate curved trajectory. Due to noisy measurements, quadratic models [17, 36] do not approximate the real curve well with fixed optical flow weighting. The cubic model [4] performs even worse due to the outlier at $t = 2$, which deteriorates high-order polynomial regression. The proposed FTF achieves best approximation by dynamically weighting the contributions of measurements. In Sect. 4.4, we show benefits of this idea on real VFI samples.

renders the solver an efficient layer when integrated into the whole architecture.

Motion resampling and refinement. After model fitting, we can resample the intermediate flows $\mathbf{F}_{0\to t}$ by evaluating $f(t; \hat{\mathbf{A}}, \hat{\mathbf{B}})$. Similar with previous works [10, 21], we also learn to refine the resampled flows to get benefits from end-to-end training, and infer the visibility masks indicating occlusion relations at time t. This is achieved with a refinement network $\mathcal{M}$:

$$\Delta\mathbf{F}_{0\to t},\ \mathbf{M}_{0\to t} = \mathcal{M}(\mathbf{F}_{0\to t}, \mathbf{C}_0, \bar{\mathbf{C}}_{1\to 0}), \tag{10}$$

where $\bar{\mathbf{C}}_{1\to 0} = \phi_{\mathcal{B}}(\mathbf{C}_1, \mathbf{F}_{0\to 1})$ is the warped features from time 1 to 0. The refined intermediate flows are $\hat{\mathbf{F}}_{0\to t} = \mathbf{F}_{0\to t} + \Delta\mathbf{F}_{0\to t}$, with a soft mask $\mathbf{M}_{0\to t} \in (0,1)^{H\times W\times 1}$ indicating the occlusion order of pixels used for soft splatting [21], which are to be used in Sect. 3.3.

3.3. Range-nullspace Synthesis (RNS)

The rest pipeline aims to synthesize the target frame at time t. Arguably, rendering cannot be fully performed with the resampled flows $\hat{\mathbf{F}}_{0\to t}$ and $\hat{\mathbf{F}}_{1\to t}$, which only depict partial correspondences from $\mathbf{I}_0$ or $\mathbf{I}_1$ to $\mathbf{I}_t$, and do not account for occluded pixels in $\mathbf{I}_0$ and $\mathbf{I}_1$. Instead of predicting full correspondences, we treat frame synthesis as an inverse problem and solve it with a theory developed for it.

Our starting point is that given the unknown $\mathbf{I}_t$ and forward flows $\hat{\mathbf{F}}_{0\to t}$ and $\hat{\mathbf{F}}_{1\to t}$, we can relate these terms with the observations $\mathbf{I}_0$ and $\mathbf{I}_1$ as follows:

$$\begin{bmatrix}\mathbf{I}_0\\ \mathbf{I}_1\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{0\to t}\\ \mathbf{H}_{1\to t}\end{bmatrix}\mathbf{I}_t + \begin{bmatrix}\boldsymbol{\Upsilon}_{0\to t}\\ \boldsymbol{\Upsilon}_{1\to t}\end{bmatrix}. \tag{11}$$

With slight abuse of notation, $\mathbf{I}_0$, $\mathbf{I}_1$ and $\mathbf{I}_t$ now denote with vectorized and raster scanned images in $[-1,1]^N$, where $N = H\times W\times C, C = 3$ is the total number of pixels. In this case, flow-based warping with $\hat{\mathbf{F}}_{0\to t}$ and $\hat{\mathbf{F}}_{1\to t}$ can be reorganized as matrix multiplication with $\mathbf{H}_{0\to t}, \mathbf{H}_{1\to t} \in \mathbb{R}^{N\times N}$. The additive residuals $\boldsymbol{\Upsilon}_t$ represent image components that cannot be modelled by warping, *e.g.* occluded image content and lost high-frequencies caused by warping operator.

Taking $\mathbf{I}_t$ in (11) as the latent to restore and $\mathbf{I}_y$ as the measurements, we can treat (11) as an inverse problem with linear forward operator $\mathbf{H}_t$. Solving general linear inverse problems is not trivial, especially when the distributions of the residuals $\boldsymbol{\Upsilon}_t$ and the latent $\mathbf{I}_t$ are complex. Replacing the former with simple analytical distributions can lead to classical Bayesian solvers, which are usually iterative even with learned gradients [39]. In this paper, we instead embrace a non-iterative and fast alternative, based on the deep range-nullspace solvers [2].

Range nullspace learning. Range-nullspace theory states that if a forward operator $\mathbf{H}$ has right inverse $\mathbf{H}^+$ that satisfies $\mathbf{H}\mathbf{H}^+ = \mathbf{I}_d$ where $\mathbf{I}_d$ is the identity matrix, it is possible to define two orthogonal projections $\mathcal{R}$ and $\mathcal{N}$:

$$\mathcal{R}(\boldsymbol{x}) \triangleq \mathbf{H}^+\mathbf{H}\boldsymbol{x},\ \ \mathcal{N}(\boldsymbol{x}) \triangleq (\mathbf{I}_d - \mathbf{H}^+\mathbf{H})\boldsymbol{x}. \tag{12}$$

It is easy to verify that $\boldsymbol{x} = \mathcal{R}(\boldsymbol{x}) + \mathcal{N}(\boldsymbol{x})$ for any $\boldsymbol{x}$, and $\mathcal{R}(\boldsymbol{x})^{\mathrm{T}}\mathcal{N}(\boldsymbol{y}) = 0$ for any $\boldsymbol{x}$ and $\boldsymbol{y}$. $\mathcal{R}(\boldsymbol{x})$ and $\mathcal{N}(\boldsymbol{x})$ are called *range*- and *null*-space projections, respectively. Particularly, $\mathcal{N}(\boldsymbol{x})$ satisfies

$$\mathcal{N}(\boldsymbol{x}) \in \{\boldsymbol{u} \in \mathbb{R}^{N\times 3} | \mathbf{H}\boldsymbol{u} = \mathbf{0}\}. \tag{13}$$

Regarding our problem (11), unfortunately, the forward operator $[\mathbf{H}_{0\to t}^{\mathrm{T}}, \mathbf{H}_{1\to t}^{\mathrm{T}}]^{\mathrm{T}}$ has no right inverse (we refer interested readers to the supplementary material for derivations). However, it has the following equivalent form:

$$\underbrace{\begin{bmatrix}\mathbf{I}_0\\ \mathbf{I}_1\end{bmatrix}}_{\mathbf{I}_y} = \underbrace{\begin{bmatrix}\mathbf{H}_{0\to t} & \mathbf{0}\\ \mathbf{0} & \mathbf{H}_{1\to t}\end{bmatrix}}_{\mathbf{H}_t}\tilde{\mathbf{I}}_t + \underbrace{\begin{bmatrix}\boldsymbol{\Upsilon}_{0\to t}\\ \boldsymbol{\Upsilon}_{1\to t}\end{bmatrix}}_{\boldsymbol{\Upsilon}_t},\ \text{s.t. } \mathbf{C}\tilde{\mathbf{I}}_t = \mathbf{0}, \tag{14}$$

where $\tilde{\mathbf{I}}_t$ is divided into the top and bottom half, *i.e.* $\tilde{\mathbf{I}}_t = [\tilde{\mathbf{I}}_{t,1}^{\mathrm{T}}, \tilde{\mathbf{I}}_{t,2}^{\mathrm{T}}]^{\mathrm{T}}$, and $\mathbf{C} = [\mathbf{I}_d, -\mathbf{I}_d]$. The constraint simply confines that $\mathbf{I}_{t,1}^{\mathrm{T}} = \mathbf{I}_{t,2}^{\mathrm{T}}$. Our strategy is to first solve $\tilde{\mathbf{I}}_t$ without the constraint. In this case, $\mathbf{H}_t$ in Eq. (14) can be right inversed. To meet the constraint, we find a projection of the result into the constraint set $\{\boldsymbol{u} \in \mathbb{R}^{2N} | \mathbf{C}\boldsymbol{u} = \mathbf{0}\}$.

The appealing property of nullspace projection (13) can be used to construct the following solver:

$$\tilde{\mathbf{I}}_t^* = \mathbf{H}_t^+\mathbf{I}_y - \mathcal{R}(\mathbf{H}_t^+\boldsymbol{\Upsilon}_t) + \mathcal{N}(g(\cdot)), \tag{15}$$

where $g(\cdot)$ is arbitrary vector-to-vector function. The interesting fact of (15) is, by this construction, $\tilde{\mathbf{I}}_t^*$ always satisfies

$\mathbf{I}_y = \mathbf{H}_t \tilde{\mathbf{I}}_t^* + \Upsilon_t$ for any $g(\cdot)$. Different $g(\cdot)$s lead to different reconstructions, revealing the ill-poseness of inverse problems. To get results that fall onto natural images, we implement $g(\cdot)$ with a neural network trained to minimize the empirical errors on a dataset.

Solving (15) is left to know the residuals $\boldsymbol{\Upsilon}_t$. Though following complex, structured distributions, $\boldsymbol{\Upsilon}_t$ is mostly sparse and universally bounded ($|\boldsymbol{\Upsilon}_t| \le 2$), which can be approximated well with neural networks. Inspired by [2], we use another network $r(\cdot)$ to directly approximate $\mathbf{H}_t^+ \boldsymbol{\Upsilon}_t$, which is also bounded. When ideal convergence is reached after training, $|r(\cdot) - \mathbf{H}_t^+ \boldsymbol{\Upsilon}_t| \to \mathbf{0}$, then the data consistency $\mathbf{I}_y = \mathbf{H}_t \tilde{\mathbf{I}}_t^* + \Upsilon_t$ can be met by (15).

After yielding the result $\tilde{\mathbf{I}}_t$, projection onto $\mathbf{C}\tilde{\mathbf{I}}_t = \mathbf{0}$ is performed by minimizing

$$\boldsymbol{y}^* = \arg \min_{\boldsymbol{y} \in \mathbb{R}^{2N}} \left(\mathcal{N}_{\mathcal{C}}(\boldsymbol{y}) - \tilde{\mathbf{I}}_t\right)^{\mathrm{T}} \boldsymbol{\Sigma} \left(\mathcal{N}_{\mathcal{C}}(\boldsymbol{y}) - \tilde{\mathbf{I}}_t\right), \quad (16)$$

where $\mathcal{N}_{\mathcal{C}}(\boldsymbol{y}) = (\mathbf{I}_d - \mathbf{C}^{\pm}\mathbf{C})\boldsymbol{y}$ is the nullspace projection of $\mathbf{C}$, $\mathbf{C}^{\pm}$ is the pesudo inverse, and $\boldsymbol{\Sigma} \in [0,1]^{2N \times 2N}$ defines the distance metric. The projected result is $\mathbf{I}_t^* = \mathcal{N}_{\mathcal{C}}(\mathbf{y}^*)$. If $\boldsymbol{\Sigma} = \mathbf{I}_d$, it is easy to show $\mathbf{I}_t^*$ is simply the average of the top and bottom half of $\tilde{\mathbf{I}}_t^*$, $\mathbf{I}_t^* = (\tilde{\mathbf{I}}_{t,1}^* + \tilde{\mathbf{I}}_{t,2}^*)/2$ (see supplementary material for details). Instead, we benefit from end-to-end training and predict a dynamic blending mask as a diagonal matrix $\boldsymbol{\Lambda} \in [0,1]^{N \times N}$, and obtain the result by $\mathbf{I}_t^* = \boldsymbol{\Lambda}\tilde{\mathbf{I}}_{t,1}^* + (\mathbf{I}_d - \boldsymbol{\Lambda})\tilde{\mathbf{I}}_{t,2}^*$. It corresponds to learned distance matrix in (16), where $\boldsymbol{\Sigma} = \mathtt{diag}\left([\boldsymbol{\Lambda}, \mathbf{I}_d - \boldsymbol{\Lambda}]\right)$.

The decomposition learning (15) amortizes the difficulty of frame synthesis by several decoupled components. The term $\mathbf{H}_t^+ \mathbf{I}_y$, where $\mathbf{H}_t^+ = \mathtt{diag}([\mathbf{H}_{0\to t}^+, \mathbf{H}_{1\to t}^+])$, can be intuitively thought of warping $\mathbf{I}_0$ and $\mathbf{I}_1$ to time t, yielding physics-guided synthesis results. The network $r(\cdot)$ learns to correct the error of physics-based warping, as the $\mathcal{R}$ operator first back-projects the network output to time 0 and 1, and then does the same forward warping. The network $g(\cdot)$ inpaints image content that cannot be addressed by forward warping, which are often occluded layer. In Sect. 4.4, we provide a visual analysis of the learned components.

Implementation. The network $r(\cdot)$ and $g(\cdot)$ have many design choices, not limited to the following one we use:

$$\begin{aligned} \boldsymbol{f}, \mathbf{H}_t^+ \boldsymbol{\Upsilon}_t &= r\left(\left\{\phi_{\mathcal{F}}(\mathbf{C}_i; \hat{\mathbf{F}}_{i\to t}, \mathbf{M}_{i\to t})\right\}_{i\in\{0,1\}}\right), \\ \boldsymbol{\Lambda}, \tilde{\mathbf{I}}_t^{\mathcal{N}} &= g\left(\mathbf{H}_t^+ \mathbf{I}_y - \mathcal{R}(\mathbf{H}_t^+ \boldsymbol{\Upsilon}_t), \mathbf{H}_t^+ \mathbf{I}_y, \boldsymbol{f}\right), \\ \tilde{\mathbf{I}}_t^* &= \mathbf{H}_t^+ \mathbf{I}_y - \mathcal{R}(\mathbf{H}_t^+ \boldsymbol{\Upsilon}_t) + \mathcal{N}(\tilde{\mathbf{I}}_t^{\mathcal{N}}), \\ \mathbf{I}_t^* &= [\boldsymbol{\Lambda}, \mathbf{I}_d - \boldsymbol{\Lambda}]\tilde{\mathbf{I}}_t^*. \end{aligned} \quad (17)$$

Note $r(\cdot)$ and $g(\cdot)$ form casaded structure, where the estimation result of $r(\cdot)$ is the input of $g(\cdot)$. We also pass through intermediate features $\boldsymbol{f}$ in this casade as unobstructed information flow. Matrix multiplications with $\mathbf{H}_t$ and $\mathbf{H}_t^+$ are all implemented via warping operations. Multiplications with $\mathbf{H}_{0\to t}$ or $\mathbf{H}_{1\to t}$ can be done via backward warping, while with their inverse we use softmax splatting [21]:

$$\begin{aligned} \mathbf{H}_{i\to t}\boldsymbol{x} &= \phi_{\mathcal{B}}(\boldsymbol{x}; \hat{\mathbf{F}}_{i\to t}), \\ \mathbf{H}_{i\to t}^+\boldsymbol{x} &\approx \phi_{\mathcal{F}}(\boldsymbol{x}; \hat{\mathbf{F}}_{i\to t}, \mathbf{M}_{i\to t}), \end{aligned} \quad (18)$$

where $\hat{\mathrm{F}}_{i\to t}$ and $\mathbf{M}_{i\to t}$ are the corresponding flows and masks defined in (10). Note this only slightly breaks the right inverse constraint but improves efficiency and flexibility. The overall working flow of steps (17) is referred to Fig. 2. Due to space limit, the detailed network configurations are referred to the supplementary material.

Loss functions. Our loss function is defined as follows

$$L_{total} = \log(\|\mathbf{I}_t^* - \mathbf{I}_t^{gt}\|_2^2 + \epsilon) + \alpha\|\mathbf{H}^+\boldsymbol{\Upsilon}_t - \mathbf{H}^+\boldsymbol{\Upsilon}_t^{gt}\|_1, \quad (19)$$

where the first one is a variant of standard ℓ_2 error. In the supplementary material we show this loss mitigates the diminishing gradient issue of ℓ_2 loss and yields improved performance. We set $\epsilon = 10^{-8}$ to avoid numeric issues. The second loss enables that $r(\cdot)$ approximates the true residual, where $\boldsymbol{\Upsilon}_t^{gt} = \mathbf{I}_y - \mathbf{H}_t\mathbf{I}_t^{gt}$. Empirically, we set $\alpha = 0.02$.

4. Experimental Results

4.1. Datasets

8× VFI. We test our method on three public 8x VFI datasets: GoPro [20], X4K1000FPS [29] and Adobe240 [31]. For GoPro, we follow [14, 39] and train our method on the official train split of GoPro and evaluate it on its test split at a resolution of 1280×720. We extract training samples according to the strategy applied in [14] and [39], yielding 22128 samples, each having 25 frames. In each sample, the 1st, 9th, 17th and 25th frames are used as input, while 10th to 16th frames serve as the supervision. The test set of GoPro is sampled similarly, resulting into 1500 non-overlapped samples. To verify the generalization capability, our model pre-trained on GoPro is also evaluated on a subset of Adobe240 [31] dataset including 630 samples, sampled with the strategy of [39], as well as the official test set of X4K1000FPS [29]. For fair comparisons on X4K1000FPS dataset, we also retrain and evaluate our method following its benchmarking protocol [29] strictly.

Single-frame (2×) VFI. Though our method is not designed for **2×** VFI, we also evaluate on seven 2× VFI datasets for completeness. For training our method, since 4 input frames are required, we use the training split of the *septuble* set of Vimeo-90k [37], which contains 64612 samples at resolution 256×448. In each septuplet, we take the 1st, 3rd, 5th, 7th frames as input and the 4th frame as the supervision. For evaluation, we use 7824 samples from the test split of Vimeo-90k septuplet set, 100 samples

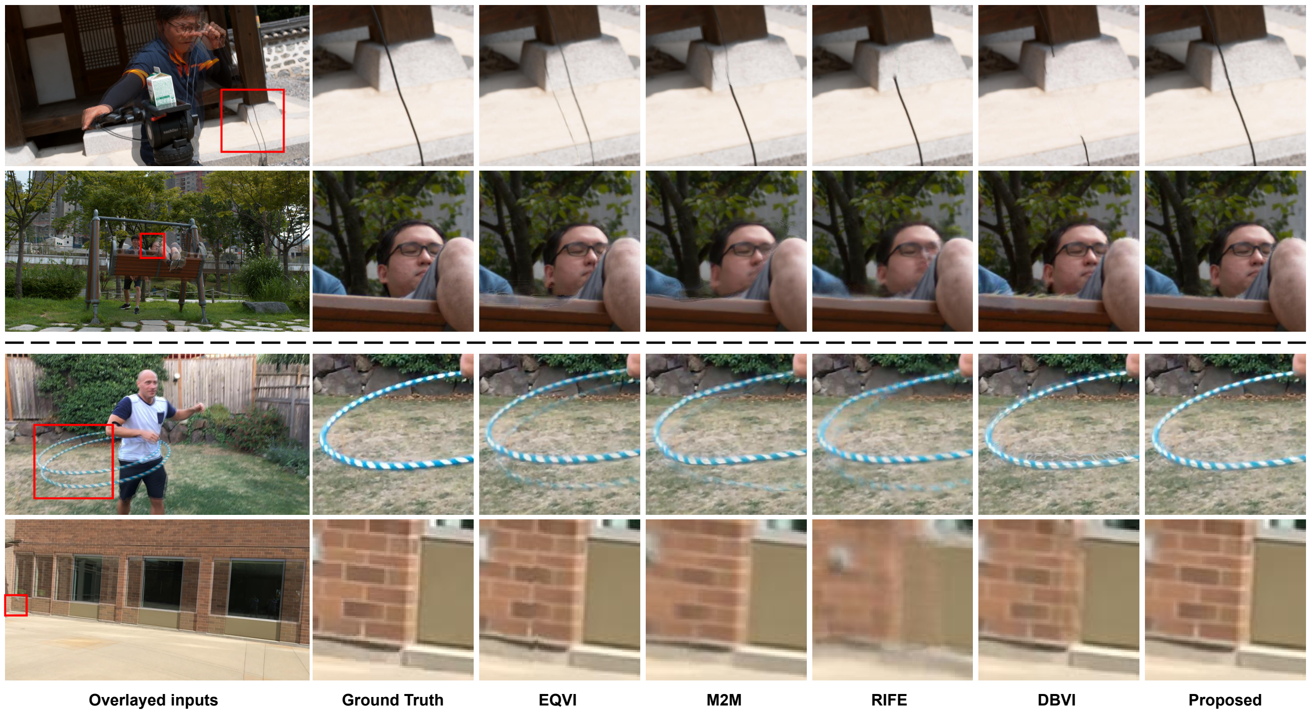


Figure 4. Visual comparisons on X4K1000FPS (top 2 rows), and Adobe240 (bottom 2 rows) datasets, where each row corresponds to a scene. We overlay the nearest 2 input frames to illustrate the input motion. More results can be found in the supplementary material.

from UCF101 [30], 2849 samples from DAVIS [25], and 310 samples from the *easy*, *medium*, *hard* and *extreme* subsets of SNU-FILM [6]. Note that these are the same evaluation protocols also adopted in [14, 28, 39].

4.2. Experimental settings

$8\times$ VFI. On each benchmark, we train the network for 120 epochs with batch size 32 and patch size 512×512, which is optimized by the AdamW [18] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay = 1×10^{-3}. At each iteration, the learning rate is decayed from 5×10^{-4} to 1×10^{-6} gradually via cosine annealing with warm steps = 2000. During the training process, we adopt random cropping, flipping, and color jittering for data augmentation.

Single-frame ($2\times$) VFI. The network is trained for 200 epochs with batch size 48 and patch size 256×256, leaving the remaining settings the same as above.

In all the benchmarks, we use two common metrics, Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) for quantitative evaluation, which is averaged across all the generated frames. Besides that, we use [32] for computing optical flows among input frames.

4.3. Comparisons with State-of-the-Art models

$8\times$ VFI trained on GoPro. We compare our method with 11 competitive VFI methods that supports $8\times$ VFI: SloMo [13], QVI [36], EQVI [17], DAIN [1], XVFI [29], M2M [10], DBVI [39],EDSC [3], FLAVR [14], IFRNet [15] and RIFE [11]. Note our evaluating settings are identical with [39], and for methods that do not follow the unified protocol proposed in [39], including EQVI, M2M, IFRNet and RIFE, we retrain them using the same protocol on GoPro and report the results. The quantitative results on GoPro, Adobe240 and X4K1000FPS are reported in Ta-

Table 1. Quantitative results of $8\times$ VFI in terms of PSNR/SSIM and the number of parameters, evaluated on GoPro, Adobe240 and X4K1000FPS datasets. The best performed model is highlighted in red and the second best is colored in blue.

	GoPro	Adobe240	X4K1000FPS	Param(M)
SloMo [13]	29.71/0.924	29.63/0.927	25.07/0.795	39.61
QVI [36]	30.52/0.941	31.41/0.955	28.06/0.855	29.23
EQVI [17]	30.81/0.942	32.13/0.959	26.96/0.843	28.07
DAIN [1]	29.53/0.920	30.53/0.939	27.28/0.835	24.03
EDSC [3]	29.20/0.916	29.87/0.931	25.30/0.811	8.95
FLAVR [14]	31.10/0.942	30.92/0.938	24.50/0.791	42.06
XVFI [29]	29.80/0.925	29.74/0.930	28.42/0.881	5.61
M2M [10]	30.52/0.933	29.93/0.931	30.04/0.905	7.61
IFRNet [15]	30.00/0.928	29.62/0.925	23.77/0.793	19.69
RIFE$_m$ [11]	29.79/0.925	29.81/0.930	28.70/0.880	10.71
DBVI [39]	31.73/0.947	33.28/0.965	31.10/0.928	15.18
Ours	32.31/0.951	33.32/0.964	31.97/0.932	10.10

Table 2. Quantitative results of 8× VFI on X4K1000FPS dataset. following the benchmark protocol of [29].

	AdaCoF [16]	FeFlow [9]	SloMo	QVI	EQVI	DAIN	FLAVR	XVFI	M2M	IFRNet	RIFE$_m$	DBVI	Ours
PSNR	25.81	25.16	27.77	29.96	28.27	27.52	27.92	30.12	30.84	28.36	30.75	32.89	33.88
SSIM	0.772	0.783	0.849	0.892	0.860	0.821	0.853	0.870	0.916	0.870	0.914	0.939	0.946

Table 3. Quantitative results on 2× interpolation in terms of PSNR/SSIM with number of parameters and flops. All the methods are trained on the training set of Vimeo-90K (septulets).

	Vimeo-90K (septulets)	UCF101	DAVIS	SNU-FILM Easy	SNU-FILM Medium	SNU-FILM Hard	SNU-FILM Extreme	Param(M)
SloMo [13]	34.43/0.969	32.45/0.967	26.10/0.862	36.12/0.984	33.44/0.972	29.17/0.928	24.14/0.843	39.61
QVI [36]	34.98/0.970	32.87/0.966	27.20/0.874	39.53/0.990	36.43/0.983	31.07/0.947	24.96/0.856	29.23
EQVI [17]	35.16/0.973	32.99/0.970	27.51/0.891	37.44/0.978	35.19/0.981	30.72/0.946	25.42/0.868	28.07
XVFI [29]	35.21/0.970	32.68/0.968	26.89/0.868	39.21/0.989	34.96/0.977	29.43/0.928	24.02/0.841	5.61
DAIN [1]	33.57/0.964	31.65/0.963	26.61/0.867	38.53/0.988	34.34/0.974	29.50/0.930	24.54/0.851	24.03
BMBC [23]	34.76/0.965	32.61/0.955	26.42/0.868	39.90/0.991	35.34/0.978	29.34/0.927	23.65/0.837	11.01
ABME [24]	35.67/0.972	32.81/0.969	27.00/0.868	39.59/0.990	35.77/0.977	30.58/0.936	25.42/0.864	18.1
EDSC [3]	34.52/0.967	32.67/0.968	26.28/0.849	40.01/0.990	35.37/0.978	29.59/0.926	24.39/0.843	8.95
GDConv [27]	35.58/0.972	33.11/0.969	27.02/0.870	40.36/0.991	36.14/0.982	30.25/0.940	24.82/0.860	5.14
CAIN [6]	34.69/0.969	32.40/0.966	27.12/0.872	39.33/0.989	35.34/0.977	30.15/0.933	24.88/0.855	42.78
FLAVR [14]	36.30/0.975	33.33/0.971	27.44/0.873	40.44/0.991	36.37/0.981	30.87/0.942	25.18/0.862	42.06
M2M [10]	35.56/0.973	32.70/0.969	27.57/0.887	39.35/0.989	35.23/0.977	29.99/0.934	24.83/0.857	7.60
IFRNet [15]	36.37/0.976	32.87/0.969	27.94/0.890	39.68/0.990	35.57/0.979	30.21/0.938	24.71/0.855	19.69
Softsplat [21]	35.76/0.972	32.89/0.970	27.42/0.878	-	-	-	-	12.46
RIFE$_m$ [11]	35.87/0.974	32.64/0.969	27.75/0.886	39.50/0.990	35.46/0.978	30.17/0.936	24.79/0.854	10.71
VFIT-B [28]	36.96/0.978	33.44/0.971	28.17/0.889	40.57/0.991	36.54/0.982	31.04/0.945	25.50/0.867	29.1
ST-MFNet [8]	36.46/0.976	33.46/0.971	28.30/0.896	40.78/0.992	37.12/0.984	31.61/0.951	25.78/0.874	21.03
DBVI [39]	36.17/0.976	33.01/0.970	28.61/0.905	40.46/0.991	36.95/0.985	31.68/0.953	25.90/0.876	21.69
Ours	36.33/0.975	33.25/0.970	28.84/0.905	40.67/0.991	37.36/0.985	32.21/0.955	26.22/0.877	10.10

ble 1, with the model size in terms of the number of parameters. Our approach achieves the best results across all the datasets, achieving at least 0.58dB improvement on GoPro and 0.87dB on X4K1000FPS. Compared with the state-of-the-art model DBVI, our model has only two-thirds parameters. For those with less parameters than ours (*e.g.* M2M), we achieve nearly 1.19dB improvements at least. **8× VFI on X4K1000FPS dataset, following the protocol of [29].** We following the training and testing protocol of [29] to report fair benchmarking results on X4K1000FPS dataset. For fair comparison, we also retrain several methods (M2M, IFRNet, RIFE, EQVI) on the official training split and compare with the results already available from [29, 39][1]. As summarized in Table 2, our approach achieves 0.99dB improvement than previous leading method DBVI. As shown in Fig. 4, our approach outperforms existing methods especially in several hard cases, like occluded or high frequency area. More visual results can be found in our supplementary material.

[1]The results of AdaCoF [16] and FeFlow [9] are adopted from [29], which were trained and evaluated under the same settings.

Single-frame (2×) VFI. Finally, we report the results on 2× VFI for completeness. Though our method is not dedicatedly designed for single-frame interpolation, we compare with 18 strong existing models and report the results in 3. Note that results of ABME, BMBC and CAIN have been reported in [39], while we extend this benchmarking with ST-MFNet[2], VFIT, GD-Conv, as well as multi-frame VFI models. All the additional models are trained on the train split of Vimeo-90K (setptulets set). Our approach achieves the best results on 4 of 7 datasets. Slightly worse performance is observed on low resolution datasets Vimeo-90K, UCF101 and the *easy* subset of SNU-FILM, whose motion is slight. On more challenging datasets like DAVIS and other subsets of SUN-FILM, the proposed method shows better generalizability.

4.4. Performance Analysis

Ablation analysis of motion trajectory fitting and range-nullspace learning. We have designed a series of

[2]The retrained model of ST-MFNet is kindly provided by the authors.

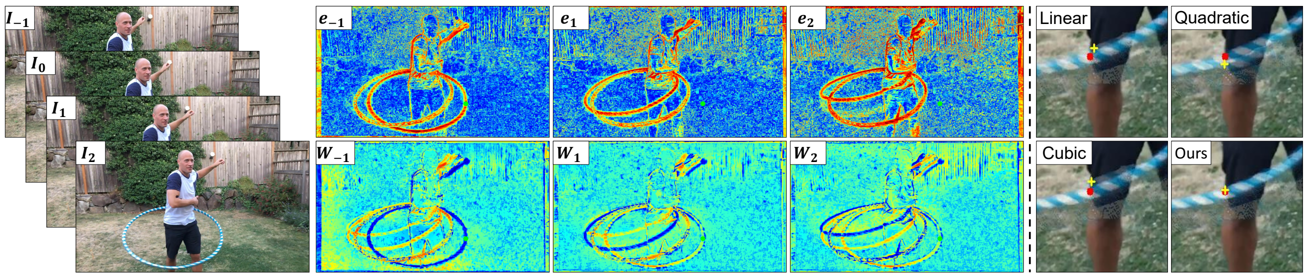


Figure 5. Analyzing the effectiveness of focalized trajectory fitting. Given the input frames (left), we show in the middle optical flow errors as the error maps of warping the frames $\mathbf{I}_{-1}$, $\mathbf{I}_1$ and $\mathbf{I}_2$ to time 0, and the corresponding weight maps of optical flows. Red denotes higher error/weight values, and blue represents lower values. Note a pixel is highlighted in green showing that higher optical flow error leads to lower weight. On the right, we show the warped result of $\mathbf{I}_0$ using the intermediate flows $\mathbf{F}_{0\to t}$ resampled from different trajectory models. Note the warping results are overlayed with groundtruth, thus sharper result means more accurate warping. We also illustrate the position of a pixel at the boundary of hula hoop after warped to time t by the intermediate flows (yellow cross), and the groundtruth position (red point).

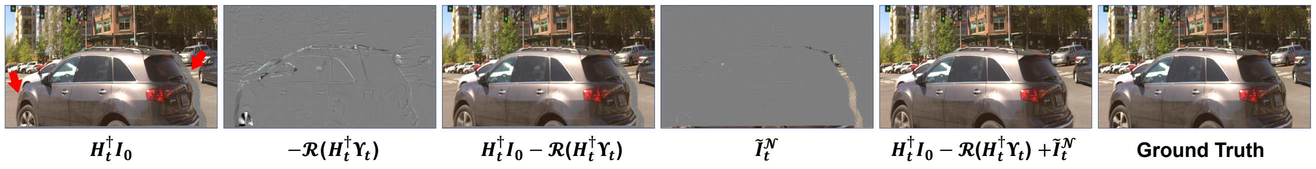

$H_t^\dagger I_0$
$-\mathcal{R}(H_t^\dagger \Upsilon_t)$
$H_t^\dagger I_0 - \mathcal{R}(H_t^\dagger \Upsilon_t)$
$\tilde{I}_t^{\mathcal{N}}$
$H_t^\dagger I_0 - \mathcal{R}(H_t^\dagger \Upsilon_t) + \tilde{I}_t^{\mathcal{N}}$
Ground Truth

Figure 6. Visualization of the learned range-null space image components. From left to right: 1) forward warping result of $\mathbf{I}_0$ by intermediate flows $\hat{\mathbf{F}}_{0\to t}$, 2) range space component, 3) range compensated result, 4) nullspace component, 5) full result, and 6) ground truth.

Table 4. Ablation analysis on the GoPro and Adobe240 datasets.

Trajectory	Renderer	GoPro	Adobe
Linear [10]	$\mathcal{R}+\mathcal{N}$	31.10/0.937	31.13/0.942
Quadratic [36]	$\mathcal{R}+\mathcal{N}$	32.07/0.949	33.20/0.963
Quadratic [17]	$\mathcal{R}+\mathcal{N}$	31.81/0.946	33.02/0.962
Cubic [4]	$\mathcal{R}+\mathcal{N}$	32.01/0.949	33.09/0.962
FTF	$\mathcal{R}$	32.13/0.950	32.99/0.962
FTF	$\mathcal{N}$	32.17/0.950	32.99/0.962
FTF	Unet	32.17/0.951	33.15/0.963
FTF	$\mathcal{R}+\mathcal{N}$	32.31/0.951	33.32/0.964

ablation study experiments in Table 4. We report results on the test set of GoPro and Adobe240 datasets with $8\times$ VFI setting. First, we replace our focalized trajectory fitting with conventional linear, quadratic and cubic trajectory models without flow confidence learning (*i.e.*, estimating the trajectory parameters $\mathbf{\Theta}$ in (8) with conventional models). As shown in the 1-4 rows in Table 4, using linear, quadratic and cubic models without focalized fitting mechanism will degrade the performance at different levels (linear model performs worst, as expected). We also verfiy the contributions of learning range and null space image components. Here, presence of $\mathcal{R}$ or $\mathcal{N}$ represent that the corresponding network outputs are included to constitute final reconstruction. By rows 5 and 6, We observe that removing either component would degrade the performance. In the 7th row, we also implement a baseline using a UNet with similar parameter size to replace the proposed RNS module, which achieves worse performance and demonstrate the effectiveness of RNS architecture.

Visualizing the effectiveness of focalized trajectory fitting. In Fig. 5, we show with one example the optical flow errors, learned weight maps, and quality of flow resampling. As can be seen, low weights are assigned to optical flows with high matching errors. Particularly, our approach yields more accurate correspondence along the occlusion boundary (as shown in Fig. 5 right).

Visualizing the range and nullspace components. In Fig. 6 we visualize the learned range and nullspace image components of one example. As it stands, the rangespace component fixes the errors of warping neighbouring frames (note the front wheel and wheel arches), while the nullspace component addresses image content that cannot be created by warping, *i.e.* inpainting the occluded content.

5. Conclusion

This paper proposes two improved treatments towards video frame interpolation. The first is focalized trajectory fitting, which improves motion estimation by suppressing low-quality optical flows. Another is range nullspace synthesis, that formulates intermediate frame synthesis as an ill-posed problem, solved by learning orthgonally decomposed image components. The proposed model achieves leading results on various public VFI datasets.

References

[1] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3703–3712, 2019. 1, 2, 6, 7

[2] Dongdong Chen and Mike E Davies. Deep decomposition learning for inverse imaging problems. In *Proc. Eur. Conf. Comput. Vis.*, pages 510–526, 2020. 2, 4, 5

[3] Xianhang Cheng and Zhenzhong Chen. Multiple video frame interpolation via enhanced deformable separable convolution. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1–1, 2021. 1, 2, 6, 7

[4] Z. Chi, R. M. Nasiri, Z. Liu, J. Lu, J. Tang, and K. N. Plataniotis. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In *Proc. Eur. Conf. Comput. Vis.*, pages 107–123, 2020. 1, 2, 4, 8

[5] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM Trans. Graph.*, 39(1):1–9, 2020. 1

[6] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *Proc. Assoc. Adv. Artif. Intell.*, volume 34, pages 10663–10671, 2020. 2, 6, 7

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 764–773, 2017. 2

[8] Duolikun Danier, Fan Zhang, and David Bull. ST-MFNet: A spatio-temporal multi-flow network for frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3521–3531, June 2022. 2, 7

[9] Shurui Gui, Chaoyue Wang, Qihua Chen, and Dacheng Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 14004–14013, 2020. 7

[10] Ping Hu, Simon Niklaus, Stan Sclaroff, and Kate Saenko. Many-to-many splatting for efficient video frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3553–3562, 2022. 1, 2, 4, 6, 7, 8

[11] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Proc. Eur. Conf. Comput. Vis.*, 2022. 2, 6, 7

[12] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 2017–2025, 2015. 3

[13] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. G. Learned-Miller, and J. Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 9000–9008, 2018. 1, 2, 4, 6, 7

[14] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv:2012.08512*, 2020. 2, 5, 6, 7

[15] Lingtong Kong, Boyuan Jiang, Donghao Luo, Wenqing Chu, Xiaoming Huang, Ying Tai, Chengjie Wang, and Jie Yang. Ifrnet: Intermediate feature refine network for efficient frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1969–1978, 2022. 2, 6, 7

[16] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 5316–5325, 2020. 2, 7

[17] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *Proc. Eur. Conf. Comput. Vis. Workshops*, pages 41–56, 2020. 1, 2, 4, 6, 7, 8

[18] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv:1711.05101*, 2018. 6

[19] Mahsa Lotfi and Mathukumalli Vidyasagar. Compressed sensing using binary matrices of nearly optimal dimensions. *IEEE Trans. Signal Process*, 68:3008–3021, 2020. 2

[20] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3883–3891, 2017. 5

[21] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 5437–5446, 2020. 1, 2, 4, 5, 7

[22] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2270–2279, 2017. 2

[23] J. Park, K. Ko, C. Lee, and C.-S. Kim. BMBC: bilateral motion estimation with bilateral cost volume for video interpolation. In *Proc. Eur. Conf. Comput. Vis.*, pages 109–125, 2020. 2, 7

[24] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 14539–14548, 2021. 2, 7

[25] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 724–732, 2016. 6

[26] Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. Deep null space learning for inverse problems: convergence analysis and rates. *Inverse Problems*, 35(2):025008, 2019. 2

[27] Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video frame interpolation via generalized deformable convolution. *IEEE Trans. Multimedia*, 24:426–439, 2021. 2, 7

[28] Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. Video frame interpolation transformer. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 17482–17491, 2022. 2, 6, 7

[29] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: Extreme video frame interpolation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 14489–14498, 2021. 1, 2, 5, 6, 7

[30] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 6

[31] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1279–1288, 2017. 5

[32] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. Eur. Conf. Comput. Vis.*, volume 12347, pages 402–419, 2020. 6

[33] Tom Tirer and Raja Giryes. Image restoration by iterative denoising and backward projections. *IEEE Trans. Image Process*, 28(3):1220–1234, 2018. 2

[34] Muhammad Usman, Xiangjian He, Kin-Man Lam, Min Xu, Syed Mohsin Matloob Bokhari, and Jinjun Chen. Frame interpolation for cloud-based mobile video streaming. *IEEE Trans. Multimedia*, 18(5):831–839, 2016. 1

[35] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proc. Eur. Conf. Comput. Vis.*, pages 416–431, 2018. 1

[36] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1645–1654, 2019. 1, 2, 4, 6, 7, 8

[37] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *Int. J. Comput. Vis.*, 127(8):1106–1125, 2019. 5

[38] Zhiyang Yu, Yu Zhang, Deyuan Liu, Dongqing Zou, Xijun Chen, Yebin Liu, and Jimmy Ren. Training weakly supervised video frame interpolation with events. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 14569–14578, 2021. 1, 2

[39] Zhiyang Yu, Yu Zhang, Xujie Xiang, Dongqing Zou, Xijun Chen, and Jimmy S. Ren. Deep bayesian video frame interpolation. In *Proc. Eur. Conf. Comput. Vis.*, pages 144–160, 2022. 1, 2, 4, 5, 6, 7