Quadratic Video Interpolatoin

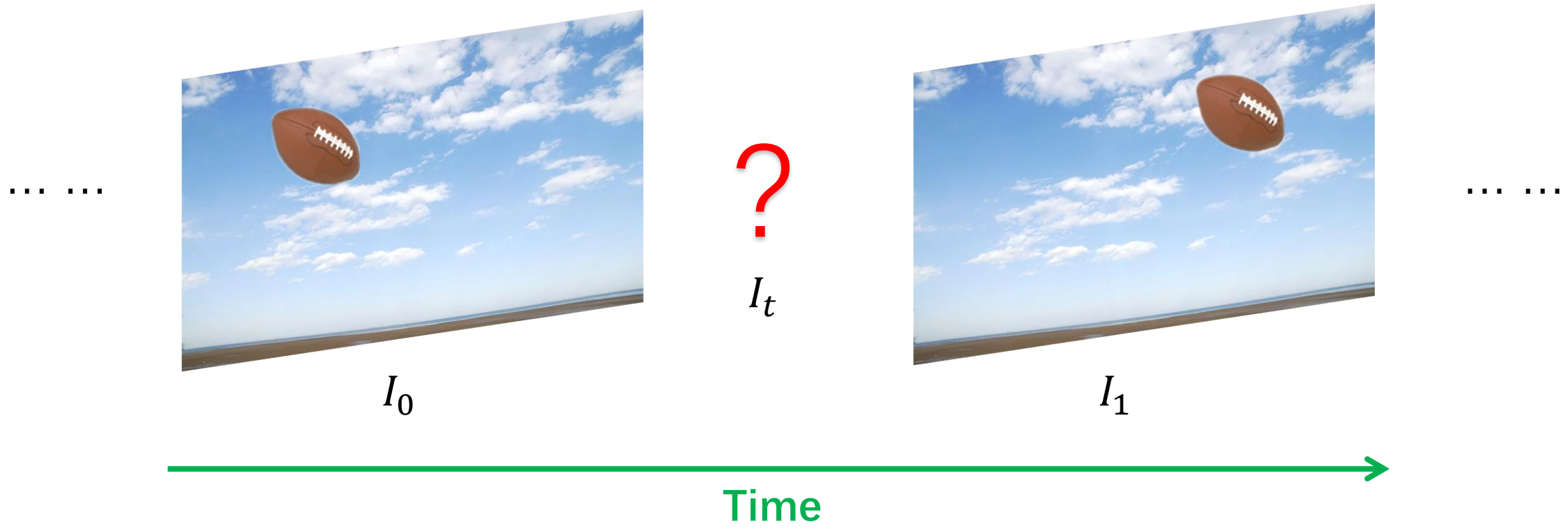
Project page: https://sites.google.com/view/xiangyuxu/qvi_nips19

Xiangyu Xu[1*] Siyao Li[2*] Wenxiu Sun[2] Qian Yin[3] Ming-Hsuan Yang[4]

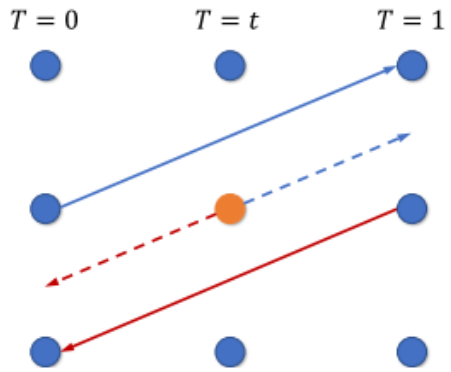
[1]Carnegie Mellon University [2]SenseTime [3]Beijing Normal University [4]University of California, Merced

Problem statement

Temporal upsampling: how to interpolate between existing frames?

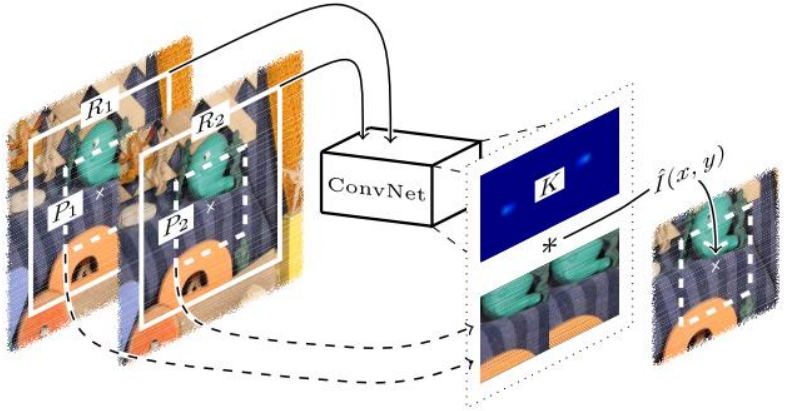


Previous methods



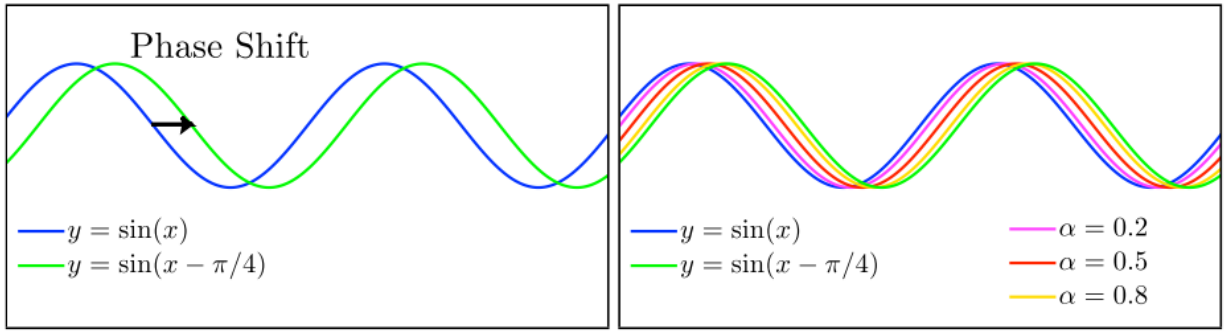
Flow-based

Liu et al, Video frame synthesis using deep voxel flow, ICCV17
Jiang et al, Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation, CVPR18



Kernel-based

Niklaus et al, Video Frame Interpolation via Adaptive Convolution, CVPR17
Niklaus et al, Video Frame Interpolation via Adaptive Separable Convolution, ICCV17

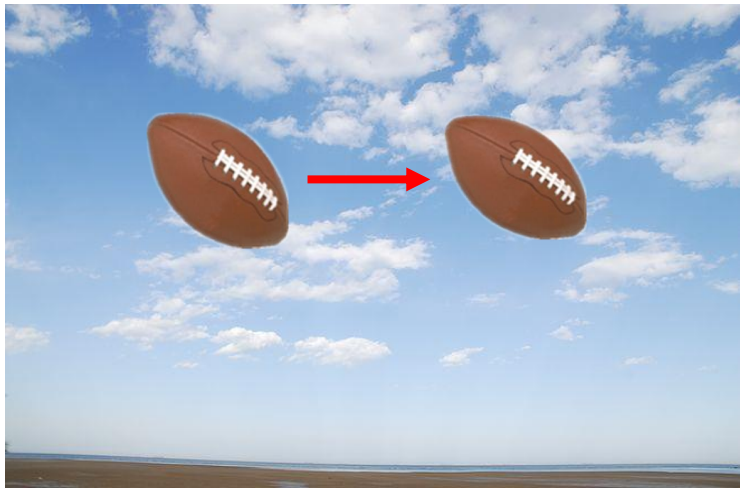


Phase-based

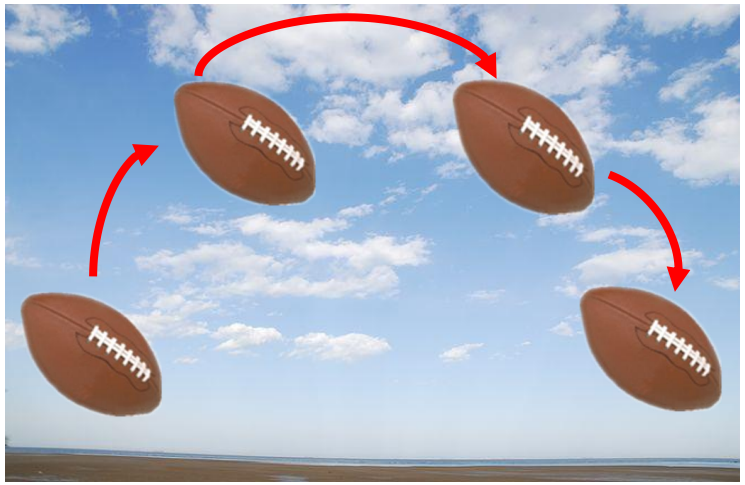
Meyer et al, Phase-Based Frame Interpolation for Video, CVPR15
Meyer et al, PhaseNet for Video Frame Interpolation, CVPR18

Existing methods usually assume **linear motion**. However, the motion in real scenarios can be more complex and **nonlinear**!

Motivation



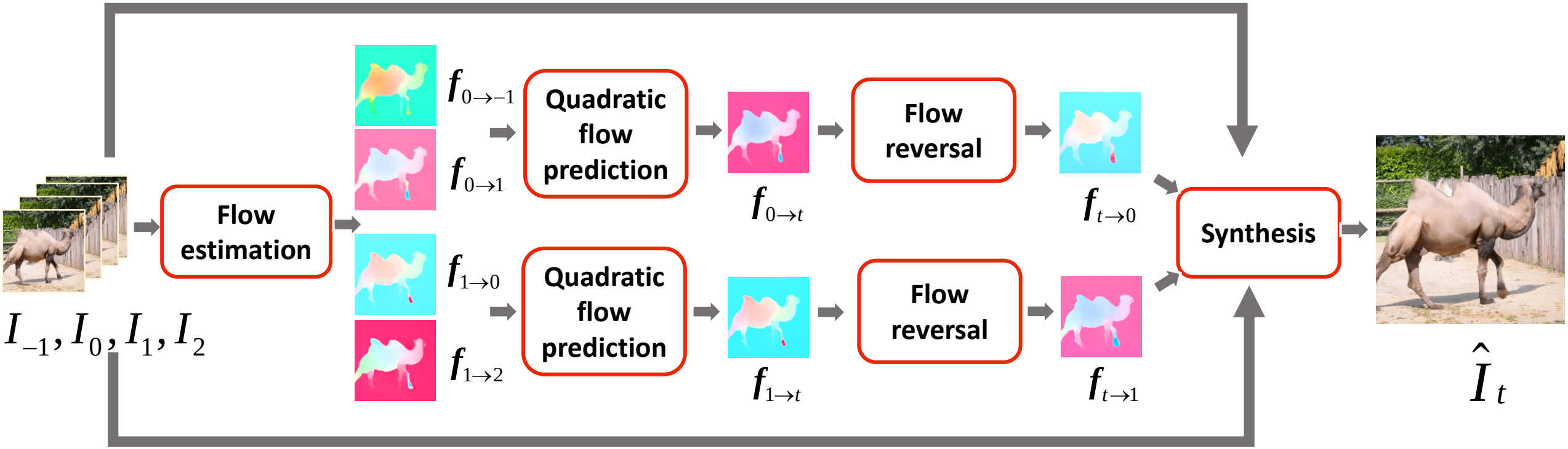
result of the state-of-the-art method
Jiang et al, CVPR18



result of our quadratic model

Applying higher-order model to exploit the acceleration information from more neighboring frames.

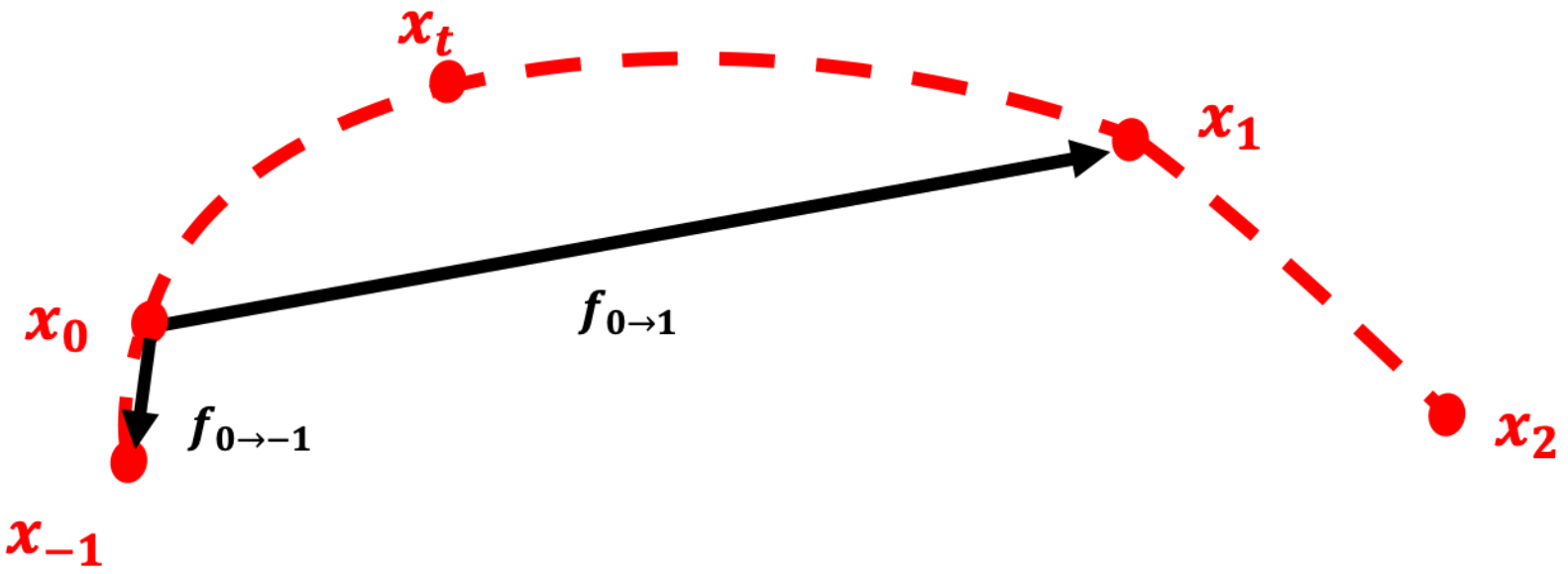
Algorithm: overview



1. Quadratic flow prediction
2. Flow reversal layer
3. Synthesis

Algorithm: 1. quadratic flow prediction

Equation of motion $f_{0\to t} = \int_0^t (v_0 + \int_0^\kappa a_\tau d\tau) d\kappa,$

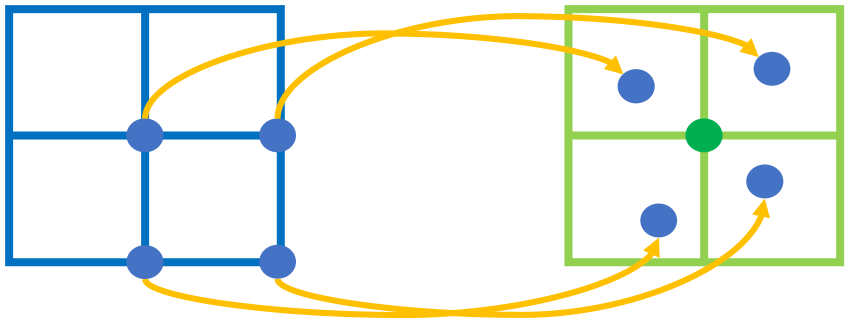


With $a_\tau = C$, we have

$$f_{0\to t} = (f_{0\to 1} + f_{0\to -1})/2 \times t^2 + (f_{0\to 1} - f_{0\to -1})/2 \times t$$

Algorithm: 2. flow reversal layer

While we have $f_{0\to t}$ from the last step, we need $f_{t\to 0}$ for frame warping.



$f_{0\to t}$ $f_{t\to 0}$

$$f_{t\to 0}(u)=\frac{\sum_{x+f_{0\to t}(x)\in N(u)} w(\| x+f_{0\to t}(x)-u \|_2)(-f_{0\to t}(x))}{\sum_{x+f_{0\to t}(x)\in N(u)} w(\| x+f_{0\to t}(x)-u \|_2)}$$

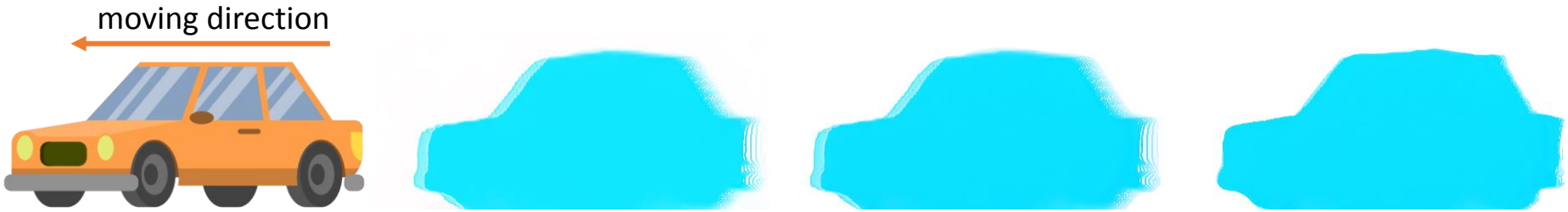
Algorithm: 3. synthesis

Adaptive flow filtering (learnable “median filter”)

The proposed flow filtering method:

$$f'_{t\to 0}(u) = f_{t\to 0}(u + \delta(u)) + r(u),$$

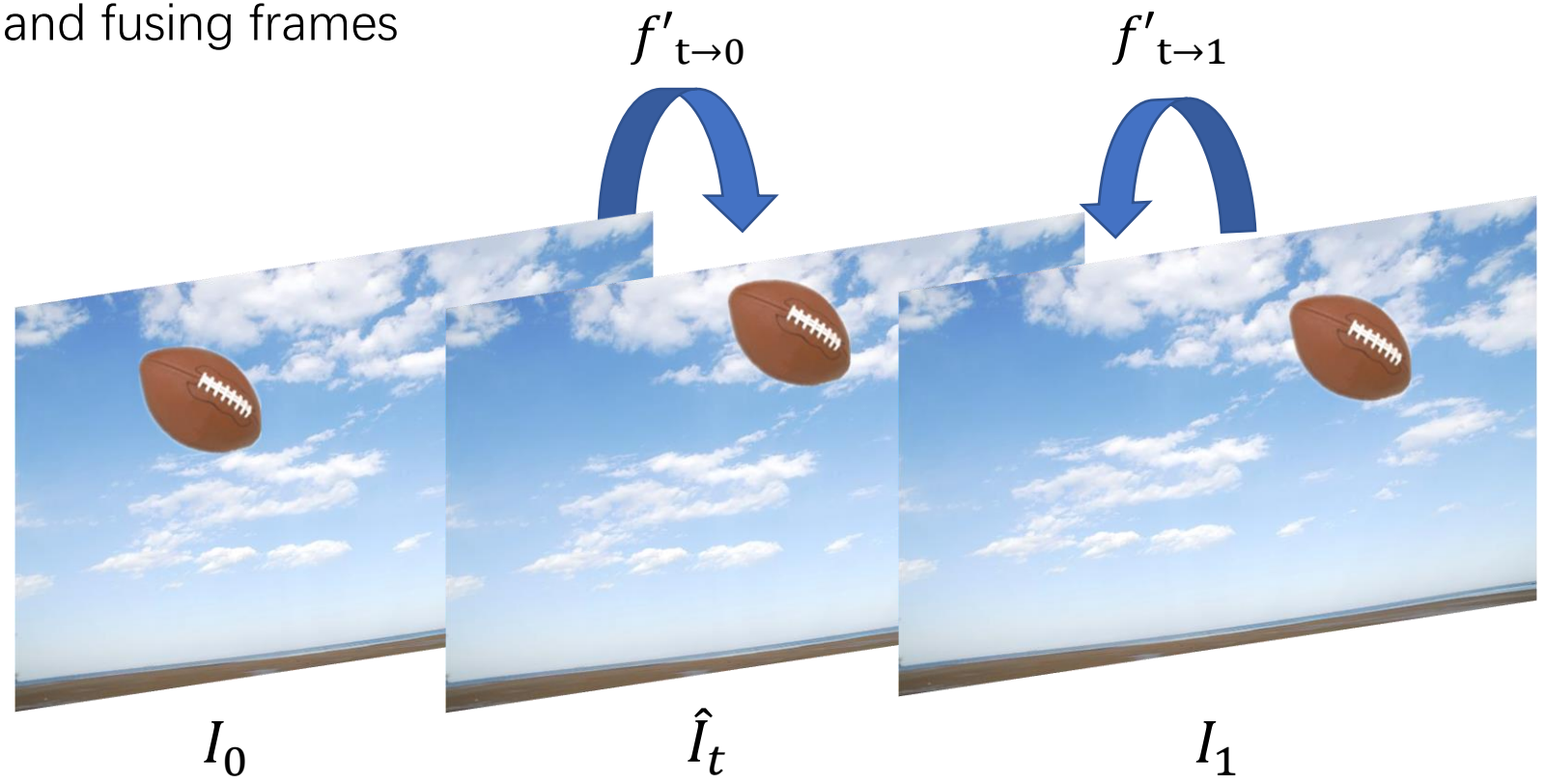
where δ denotes the learned offset, and r is the learned residual map.



I_0 $f_{t\to 0}$ Refined $f_{t\to 0}$ by convolution Refined $f_{t\to 0}$ by our method

Algorithm: 3. synthesis

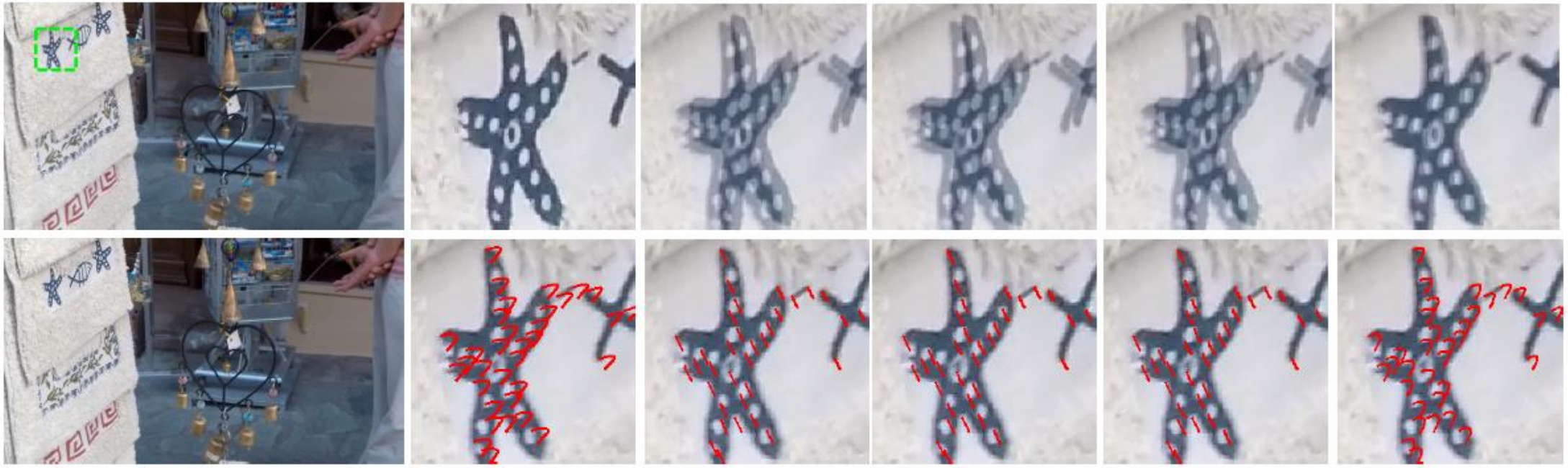
Warping and fusing frames



$$\hat{I}_t(u) = mI_0^t + (1-m)I_1^t,$$

where m is the learned mask for fusing the warped frames.

Results



frame 1 & frame 2 (a) GT (b) SepConv (c) SuperSloMo (d) Ours w/o qua. (e) Ours

	GOPRO						Adobe240					
	whole			center			whole			center		
Method	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE
Phase	23.95	0.700	17.89	22.05	0.620	22.08	25.60	0.735	16.93	23.65	0.647	20.65
DVF	21.94	0.776	21.30	20.55	0.720	25.14	28.23	0.896	11.76	26.90	0.871	13.30
SepConv	29.52	0.922	9.26	27.69	0.895	11.38	32.19	0.954	7.71	30.87	0.940	8.91
SuperSloMo	29.00	0.918	9.51	27.33	0.892	11.50	31.30	0.949	8.18	30.17	0.935	9.22
Ours w/o qua.	29.57	0.923	9.02	27.86	0.898	10.93	31.64	0.952	7.93	30.48	0.939	8.96
Ours	**31.27**	**0.948**	**7.23**	**29.62**	**0.929**	**8.73**	**32.95**	**0.966**	**6.84**	**32.09**	**0.959**	**7.47**

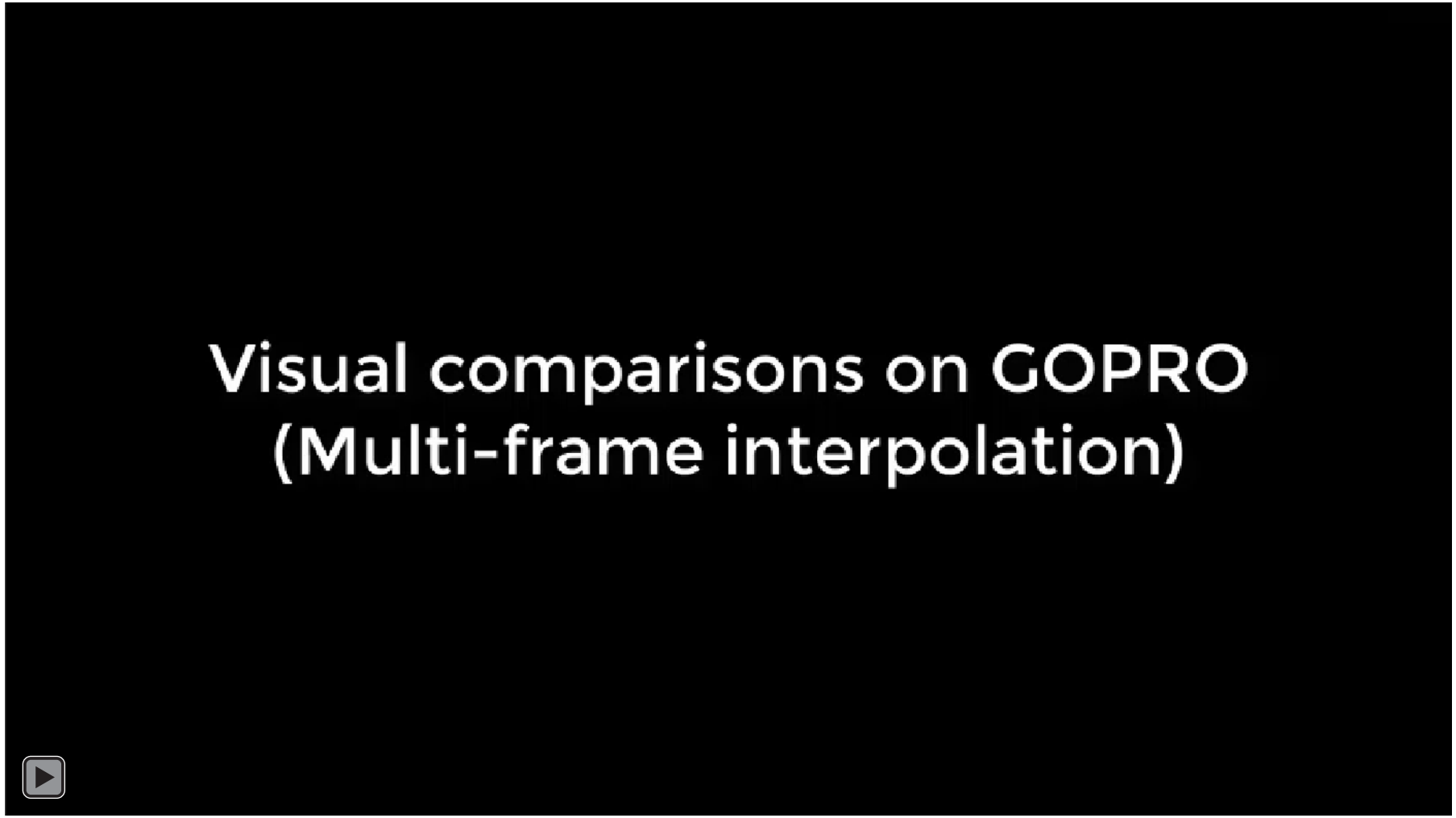
Results

The proposed method ranks the **1st** in the ICCV 2019 video interpolation challenge.

Method	PSNR	SSIM
Ours	**25.47**	**0.7383**
NoFlow	25.05	0.7231
Team Eraser	24.58	0.7052
DAIN	24.56	0.7062
Team Eraser	24.55	0.7045
BOE_IOT_AIBE_IMP	24.41	0.6998
KAIST-VICLAB	23.93	0.6869
ZSFI	21.83	0.6094
baseline (overlay)	20.39	0.6625

Results

Video examples:



Linear model

Quadratic model

Project

Poster: East Exhibition Hall B+C **#97**

Time: **5:00-7:00 PM**, Dec 12