Depth-Aware Video Frame Interpolation
Supplementary Material

Wenbo Bao[1] Wei-Sheng Lai[3] Chao Ma[2] Xiaoyun Zhang[1*] Zhiyong Gao[1] Ming-Hsuan Yang[3,4]
[1] Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
[2] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3] University of California, Merced [4] Google
https://sites.google.com/view/wenbobao/dain

1. Overview

In this supplementary document, we present additional results to complement the paper. First, we provide the algorithmic details and network configuration of the proposed model. Second, we conduct additional analysis on the adaptive warping layer and depth-aware flow projection layer and evaluate the performance of the depth estimation network. Finally, we present more experimental results on interpolating arbitrary intermediate frames, qualitative comparisons with the state-of-the-art video frame interpolation methods on the Middlebury and Vimeo90K datasets, as well as discussions of the limitations on the HD video dataset. More video results are provided on our project website.

2. Algorithm Details and Network Configurations

We provide the derivation of the back-propagation in the proposed depth-aware flow projection layer, the algorithmic details of the adaptive warping layer, and the configuration of our kernel estimation and frame synthesis networks.

2.1. Back-Propagation in Depth-Aware Flow Projection

Given the input flows ($\mathbf{F}_{0\to1}$ and $\mathbf{F}_{1\to0}$) and the depth maps (D_0 and D_1), our depth-aware flow projection layer generates the intermediate flows $\mathbf{F}_{t\to0}$ and $\mathbf{F}_{t\to1}$. The proposed model jointly optimizes the flow estimation and depth estimation networks to achieve a better performance for video frame interpolation. The gradient of $\mathbf{F}_{t\to0}$ with respect to the input optical flow $\mathbf{F}_{0\to1}$ is calculated by:

$$\frac{\partial \mathbf{F}_{t\to0}(\mathbf{x})}{\partial \mathbf{F}_{0\to1}(\mathbf{y})} = \begin{cases} -t \cdot \dfrac{w_0(\mathbf{y})}{\sum\limits_{\mathbf{y}'\in\mathcal{S}(\mathbf{x})} w_0(\mathbf{y}')}, \text{for } \mathbf{y} \in \mathcal{S}(\mathbf{x}); \\ 0, \text{for } \mathbf{y} \notin \mathcal{S}(\mathbf{x}). \end{cases} \tag{1}$$

The gradient of $\mathbf{F}_{t\to0}$ with respect to the depth D_0 is calculated by:

$$\frac{\partial \mathbf{F}_{t\to0}(\mathbf{x})}{\partial D_0(\mathbf{y})} = \frac{\partial \mathbf{F}_{t\to0}(\mathbf{x})}{\partial w_0(\mathbf{y})} \cdot \frac{\partial w_0(\mathbf{x})}{\partial D_0(\mathbf{y})}, \tag{2}$$

where

$$\frac{\partial \mathbf{F}_{t\to0}(\mathbf{x})}{\partial w_0(\mathbf{y})} = \begin{cases} -t \cdot \dfrac{\mathbf{F}_{0\to1}(\mathbf{y}) \cdot \sum\limits_{\mathbf{y}'\in\mathcal{S}(\mathbf{x})} w_0(\mathbf{y}') - \sum\limits_{\mathbf{y}'\in\mathcal{S}(\mathbf{x})} \mathrm{w}_0(\mathbf{y}) \cdot \mathbf{F}_{0\to1}(\mathbf{y}')}{\Big(\sum\limits_{\mathbf{y}'\in\mathcal{S}(\mathbf{x})} w_0(\mathbf{y}')\Big)^2}, \text{for } \mathbf{y} \in \mathcal{S}(\mathbf{x}); \\ 0, \text{for } \mathbf{y} \notin \mathcal{S}(\mathbf{x}). \end{cases} \tag{3}$$

and

$$\frac{\partial w_0(\mathbf{y})}{\partial D_0(\mathbf{y})} = -1 \cdot \big(D_0(\mathbf{y})\big)^{-2}. \tag{4}$$

2.2. Adaptive Warping Layer

The adaptive warping layer [2] warps images or features based on the estimated optical flow and local interpolation kernels. Let $\mathbf{I}(\mathbf{x}) : \mathbb{Z}^2 \rightarrow \mathbb{R}^3$ denote the RGB image where $\mathbf{x} \in [1, H] \times [1, W]$, $\mathbf{f}(\mathbf{x}) \coloneqq (u(\mathbf{x}), v(\mathbf{x}))$ represent the optical flow field and $\mathbf{k}^l(\mathbf{x}) = [k^l_{\mathbf{r}}(\mathbf{x})]_{H \times W}$ ($\mathbf{r} \in [-R+1, R]^2$) indicate the interpolation kernel where $R = 2$ is the kernel size. The adaptive warping layer synthesizes an output image by:

$$\hat{\mathbf{I}}(\mathbf{x}) = \sum_{\mathbf{r} \in [-R+1, R]^2} k_{\mathbf{r}}(\mathbf{x}) \mathbf{I}(\mathbf{x} + \lfloor \mathbf{f}(\mathbf{x}) \rfloor + \mathbf{r}), \tag{5}$$

where the weight $k_{\mathbf{r}} = k^l_{\mathbf{r}} k^d_{\mathbf{r}}$ is determined by both the interpolation kernel $k^l_{\mathbf{r}}$ and bilinear coefficient $k^d_{\mathbf{r}}$. The bilinear coefficient is defined by:

$$k^d_{\mathbf{r}} = \begin{cases} [1 - \theta(u)][1 - \theta(v)], & \mathbf{r}_u \leq 0, \mathbf{r}_v \leq 0, \\ \theta(u)[1 - \theta(v)], & \mathbf{r}_u > 0, \mathbf{r}_v \leq 0, \\ [1 - \theta(u)]\theta(v), & \mathbf{r}_u \leq 0, \mathbf{r}_v > 0, \\ \theta(u)\theta(v), & \mathbf{r}_u > 0, \mathbf{r}_v > 0, \end{cases} \tag{6}$$

where $\theta(u) = u - \lfloor u \rfloor$ denotes the fractional part of a float point number, and the subscript u, v of the 2-D vector $\mathbf{r}$ represent the horizontal and vertical components, respectively. The bilinear coefficient allows the layer to back-propagate the gradients to the optical flow estimation network. The interpolation kernels $k^l_{\mathbf{r}}$ have the same spatial resolution as the input image with a channel size of $(2 \times R)^2 = 16$, as listed in the last row of Table 1.

2.3. Network Architectures

Our kernel estimation network generates two separable 1D interpolation kernels for each pixel. We use a U-Net architecture and provide the configuration details in Table 1. In our frame synthesis network, we use 3 residual blocks to predict the residual between the blended warped frames and the ground-truth frame. Table 2 provides the configuration details of the frame synthesis network.

Table 1. **Detailed configuration of the kernel estimation network**.

	Input	Output	Kernel size	#input channels	#output channels	Stride	Activation	Output size
in	——	RGBs	——	——	6	——	——	$H \times W$
encoder	RGBs	enc_conv1	3× 3	6	16	1	ReLU	$H \times W$
	enc_conv1	enc_conv2	3× 3	16	32	1	ReLU	$H \times W$
	enc_conv2	enc_pool1	2× 2	32	32	2	——	$H/2 \times W/2$
	enc_pool1	enc_conv3	3× 3	32	64	1	ReLU	$H/2 \times W/2$
	enc_conv3	enc_pool2	2× 2	64	64	2	——	$H/4 \times W/4$
	enc_pool2	enc_conv4	3× 3	64	128	1	ReLU	$H/4 \times W/4$
	enc_conv4	enc_pool3	2× 2	128	128	2	——	$H/8 \times W/8$
	enc_pool3	enc_conv5	3× 3	128	256	1	ReLU	$H/8 \times W/8$
	enc_conv5	enc_pool4	2× 2	256	256	2	——	$H/16 \times W/16$
	enc_pool4	enc_conv6	3× 3	256	512	1	ReLU	$H/16 \times W/16$
	enc_conv6	enc_pool5	2× 2	512	512	2	——	$H/32 \times W/32$
decoder	enc_pool5	dec_conv6	3× 3	512	512	1	ReLU	$H/32 \times W/32$
	dec_conv6	dec_up5	4× 4	512	512	1/2	——	$H/16 \times W/16$
	enc_conv6+dec_up5	dec_conv5	3× 3	512	256	1	ReLU	$H/16 \times W/16$
	dec_conv5	dec_up4	4× 4	256	256	1/2	——	$H/8 \times W/8$
	enc_conv5+dec_up4	dec_conv4	3× 3	256	128	1	ReLU	$H/8 \times W/8$
	dec_conv4	dec_up3	4× 4	128	128	1/2	——	$H/4 \times W/4$
	enc_conv4+dec_up3	dec_conv3	3× 3	128	64	1	ReLU	$H/4 \times W/4$
	dec_conv3	dec_up2	4× 4	64	64	1/2	——	$H/2 \times W/2$
	enc_conv3+dec_up2	dec_conv2	3× 3	64	32	1	ReLU	$H/2 \times W/2$
	dec_conv2	dec_up1	4× 4	32	32	1/2	——	$H \times W$
	enc_conv2+dec_up1	dec_conv1	3× 3	32	16	1	ReLU	$H \times W$
out	dec_conv1	out_conv1	3× 3	16	16	1	ReLU	$H \times W$
	out_conv1	kernel_horizontal	3× 3	16	16	1	——	$H \times W$
	dec_conv1	out_conv2	3× 3	16	16	1	ReLU	$H \times W$
	out_conv2	kernel_vertical	3× 3	16	16	1	——	$H \times W$

Table 2. **Detailed configuration of the frame synthesis network**.

	Input	Output	Kernel size	#input channels	#output channels	Stride	Activation	Output size
in	——	features	——	——	428	——	——	$H \times W$
	features	in_conv	7×7	428	128	1	ReLU	$H \times W$
resblocks	in_conv	res1_conv1	3×3	128	128	1	ReLU	$H \times W$
	res1_conv1	res1_conv2	3×3	128	128	1	——	$H \times W$
	in_conv+res1_conv2	resblock1	——	128	128	1	ReLU	$H \times W$
	resblock1	res2_conv1	3×3	128	128	1	ReLU	$H \times W$
	res2_conv1	res2_conv2	3×3	128	128	1	——	$H \times W$
	resblock1+res2_conv2	resblock2	——	128	128	1	ReLU	$H \times W$
	resblock2	res3_conv1	3×3	128	128	1	ReLU	$H \times W$
	res3_conv1	res3_conv2	3×3	128	128	1	——	$H \times W$
	resblock2+res3_conv2	resblock3	——	128	128	1	ReLU	$H \times W$
out	resblock3	out_conv	3×3	128	3	1	——	$H \times W$

3. Additional Analysis

We conduct an additional evaluation to compare the proposed depth-aware flow projection layer and adaptive warping layer with their alternatives. We also evaluate the accuracy of the depth estimation network.

3.1. Depth-aware flow projection layer

In our depth-aware flow projection layer, we use the inverse of depth value as the weight for aggregating flow vectors, which is a *soft* blending of flows. A straightforward baseline is to project the flow with the smallest depth value, which can be referred to as a *hard* selection scheme. We show a quantitative comparison between these two schemes in Table 3, where the soft blending scheme obtains better performance on all the datasets. As shown in Figure 1, the hard selection scheme obtains broken skateboard in both the black and blue close-ups. We note that the soft blending can better account for the uncertainty of depth estimation. On the other hand, our flow projection layer allows the network to back-propagate gradients to the depth estimation module for fine-tuning, which leads to performance gain as shown in the main paper.

Hard selection | Soft blending (ours) | Ground-truth

Figure 1. **Effect of the depth-aware flow projection.**

Table 3. **Analysis on the depth-aware flow projection.** M.B. is short for the OTHER set of the Middlebury dataset. The proposed model Soft blending scheme shows a substantial improvement against the Hard selection scheme using flow with smallest depth.

Method	UCF101 [10]		Vimeo90K [11]		M.B. [1]
	PSNR	SSIM	PSNR	SSIM	IE
Hard selection	34.89	0.9680	34.35	0.9740	2.11
Soft blending (ours)	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**

3.2. Adaptive warping layer

To understand the effect of the adaptive warping layer, we remove the kernel estimation network from the proposed model and replace the adaptive warping layer with a bilinear warping layer. The results are presented in Table 4. The adaptive warping consistently provides significant performance improvement over the bilinear warping layer among the UCF101 [10], Vimeo90K [11], and Middlebury [1] datasets. The results in Figure 2 demonstrate that the adaptive warping layer generates clearer textures than bilinear warping.

Bilinear warping | Adaptive warping (ours) | Ground-truth

Figure 2. **Effect of the adaptive warping layer.**

Table 4. **Comparison between bilinear and adaptive warping layers.** M.B. is short for the OTHER set of the Middlebury dataset.

Method	UCF101 [10]		Vimeo90K [11]		M.B. [1]
	PSNR	SSIM	PSNR	SSIM	IE
Bilinear warping	34.73	0.9672	33.81	0.9680	2.58
Adaptive warping (ours)	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**

4. Depth Estimation

Our model learns the *relative* depth order instead of absolute depth values. Therefore, we use the SDR (SfM Disagreement Rate) [4] to measure the preservation of depth order. The $SDR^{=}$ / $SDR^{\neq}$ is the disagreement rate for pairs of pixels with similar / different depth orders. We compare the depth maps from our depth estimation network and the MegaDepth [4] on the dataset of [4] and show the results in Table 5. We observe that our method improves the $SDR^{\neq}$ substantially as our depth-aware flow projection layer is more effective on the motion boundaries where objects have different depth orders.

Table 5. **Evaluation of depth estimation.**

Method	$SDR^{=}\%$	$SDR^{\neq}\%$	SDR%
MegaDepth [4]	**33.4**	26.0	**29.2**
Ours	59.0	**18.9**	36.4

5. Experimental Results

5.1. Arbitrary Frame Interpolation

In Figure 3, we demonstrate that our method can generate arbitrary intermediate frames to create $10\times$ slow-motion videos. An Adobe PDF Reader is recommended to view the videos.

Figure 3. **Videos with $10\times$ slow-motion of inputs.** Please view in Adobe PDF Reader to play the videos.

5.2. Qualitative Comparisons

We provide more visual comparisons with state-of-the-art methods on the Middlebury and Vimeo90K datasets.

5.2.1 Middlebury Dataset

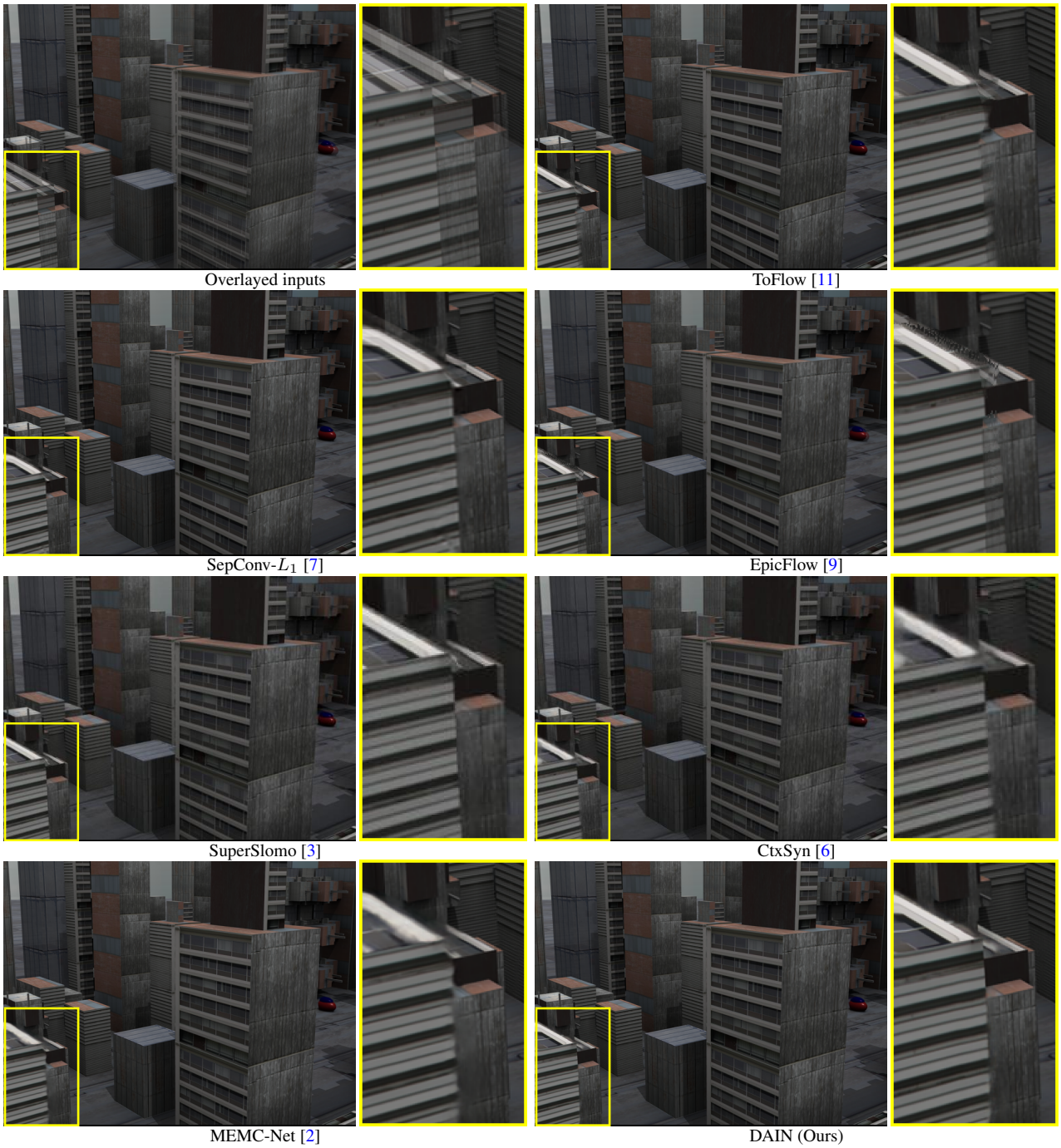


Figure 4. **Visual comparisons on Middlebury [1] EVALUATION set.** Our method reconstructs the clearer roof and sharper edges than the state-of-the-art algorithms.

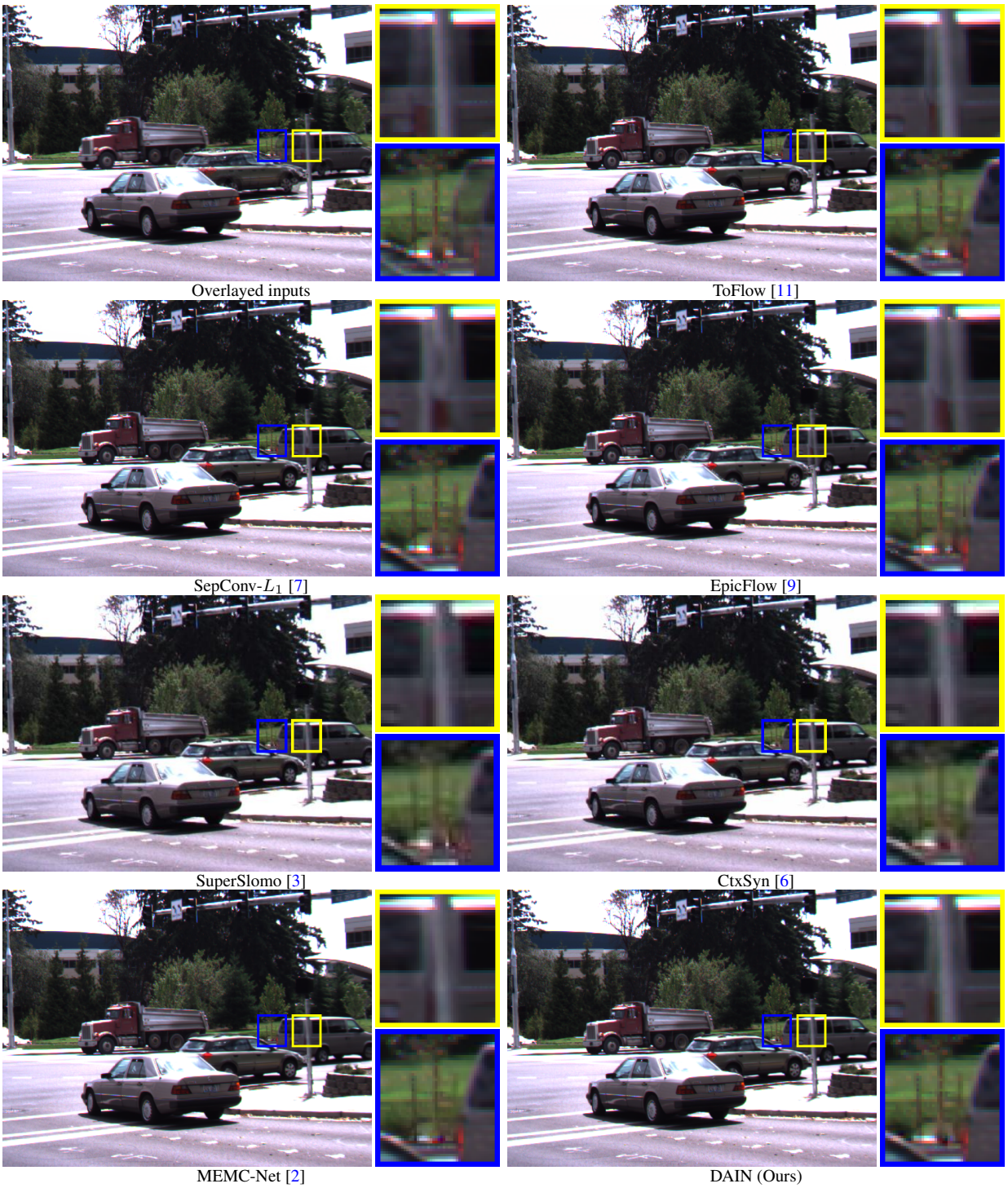


Figure 5. **Visual comparisons on the Middlebury [1] EVALUATION set.** Our method reconstructs a straight and complete lamppost, while the state-of-the-art approaches cannot reconstruct the lamppost well. In addition, our model generates clearer texture behind the moving car.

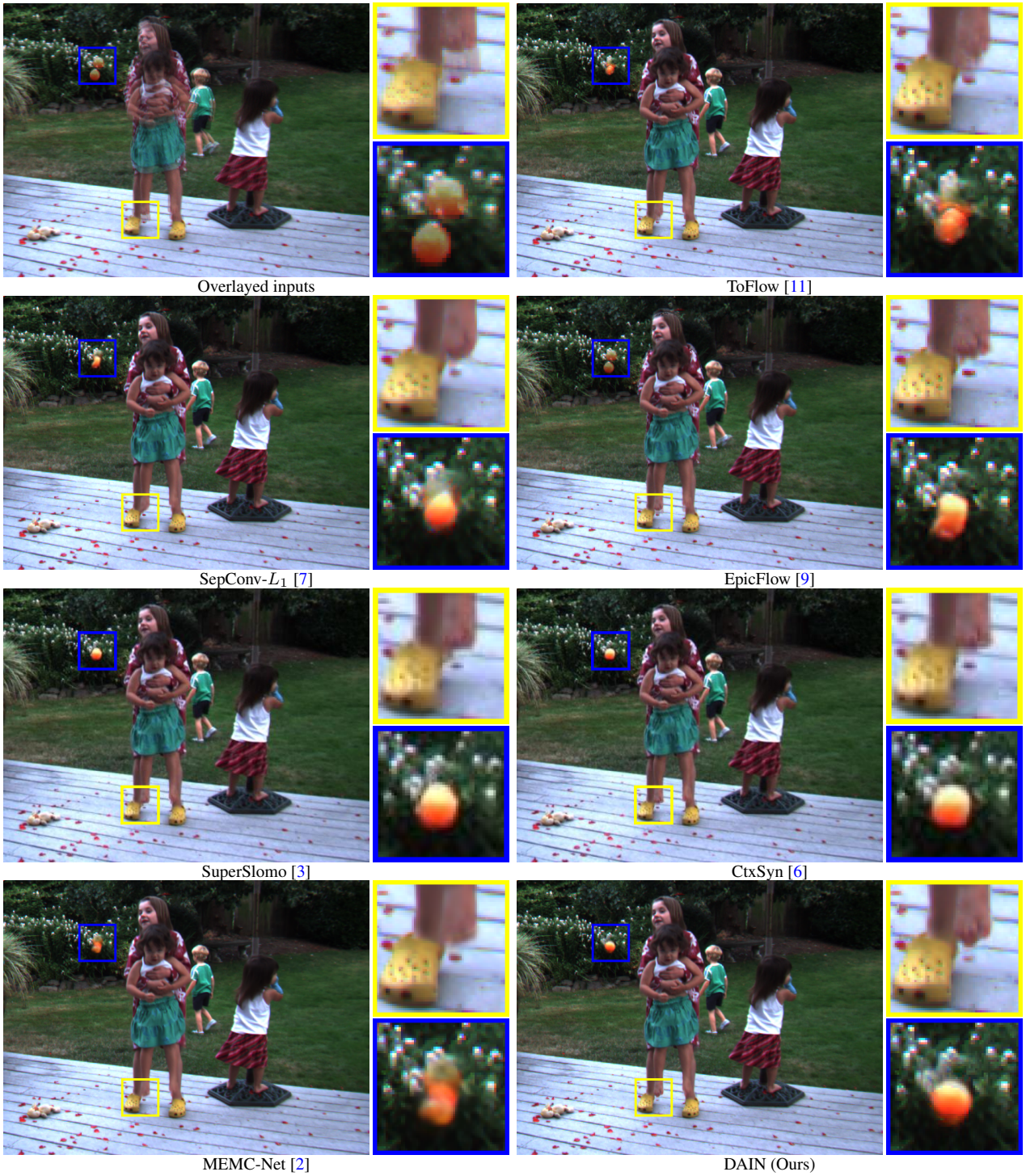


Figure 6. **Visual comparisons on the Middlebury [1] EVALUATION set.** The proposed method reconstructs the falling ball with a clear shape and generates fewer artifacts on the foot.

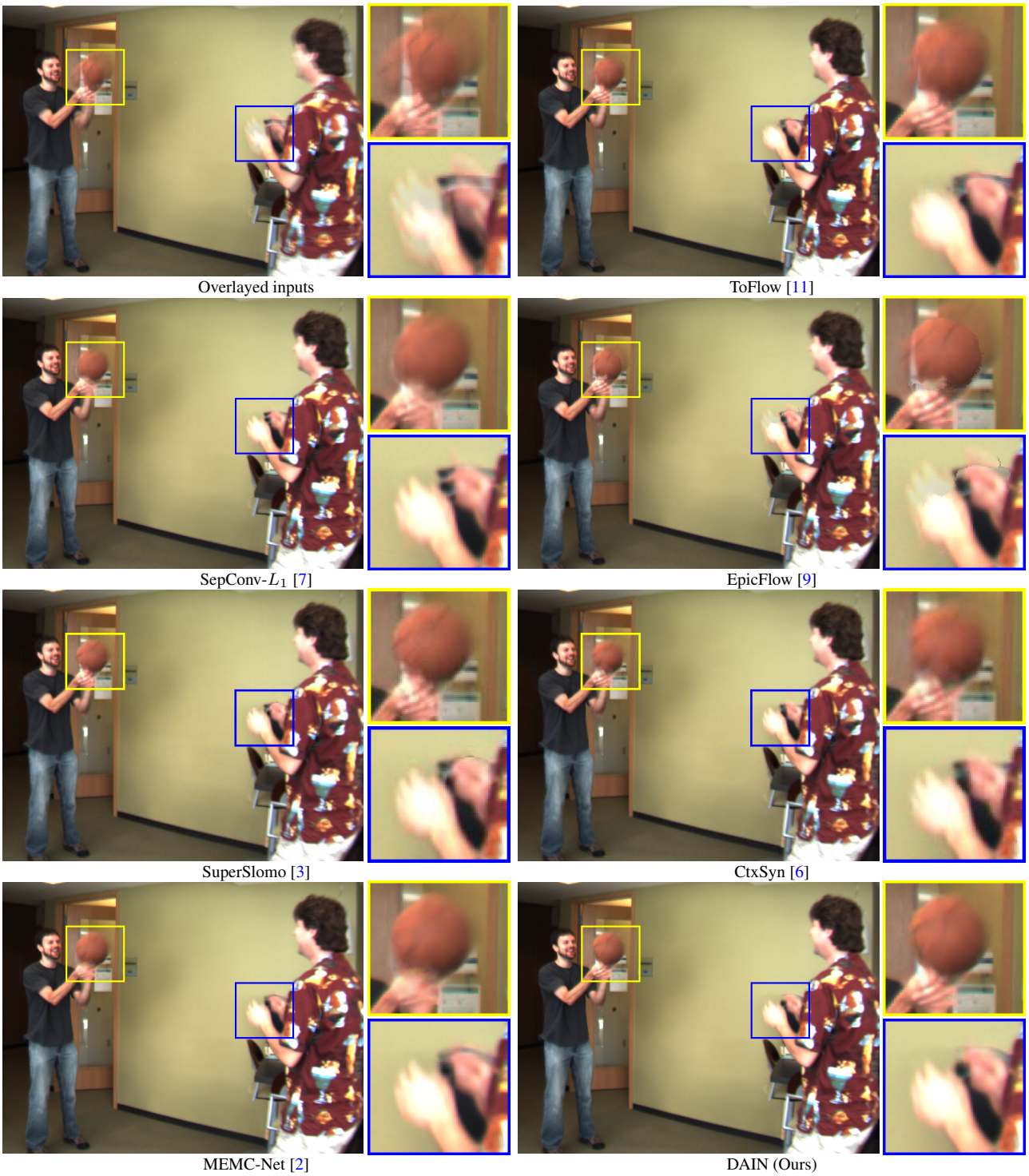


Figure 7. **Visual comparisons on the Middlebury [1] EVALUATION set.** Our method preserves the fine textures of the basketball well and does not produce blockiness or ghost effect.

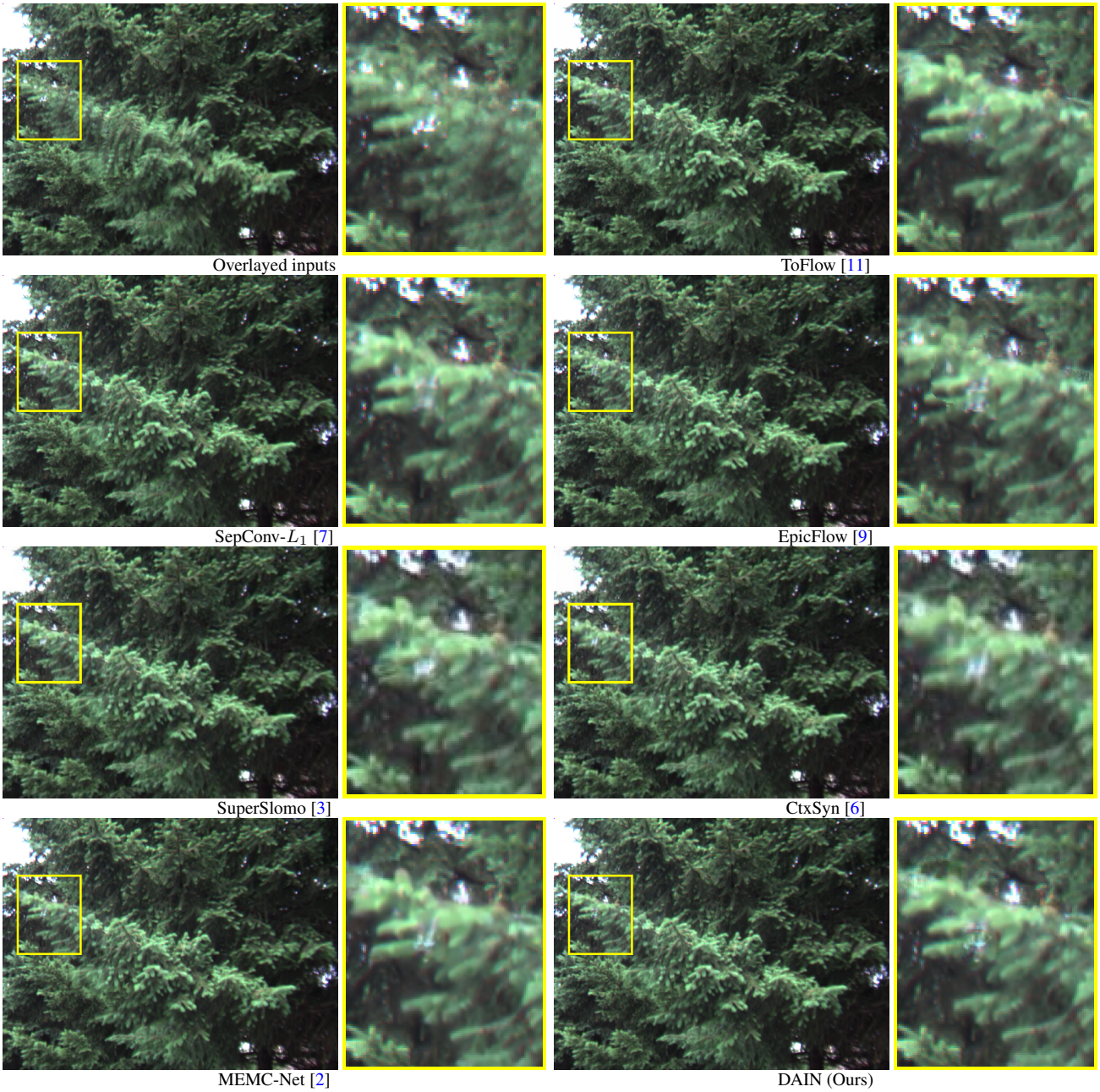
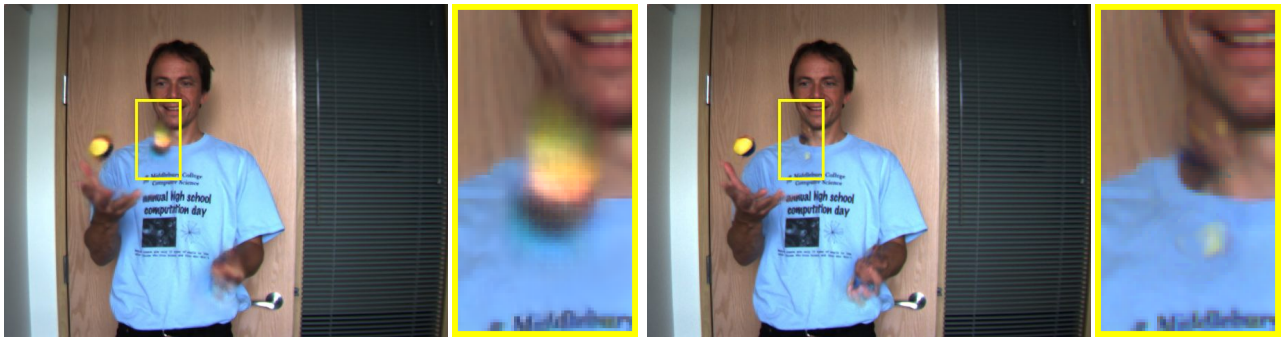
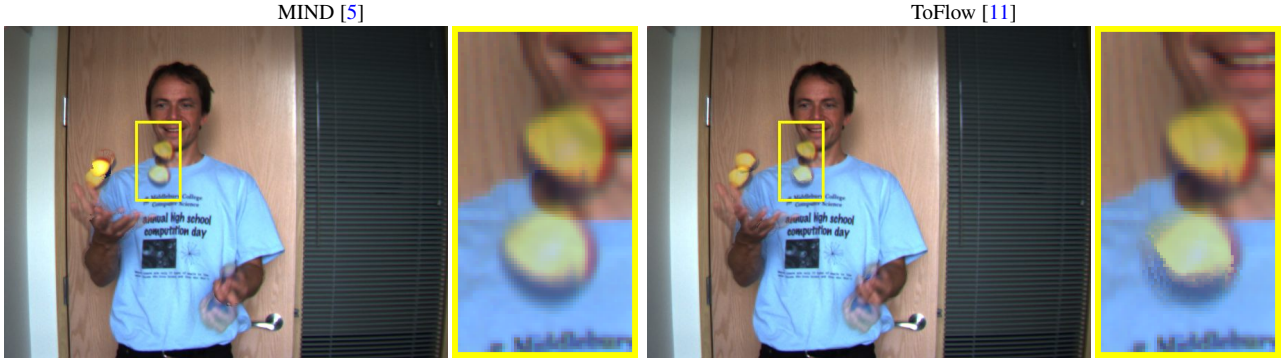


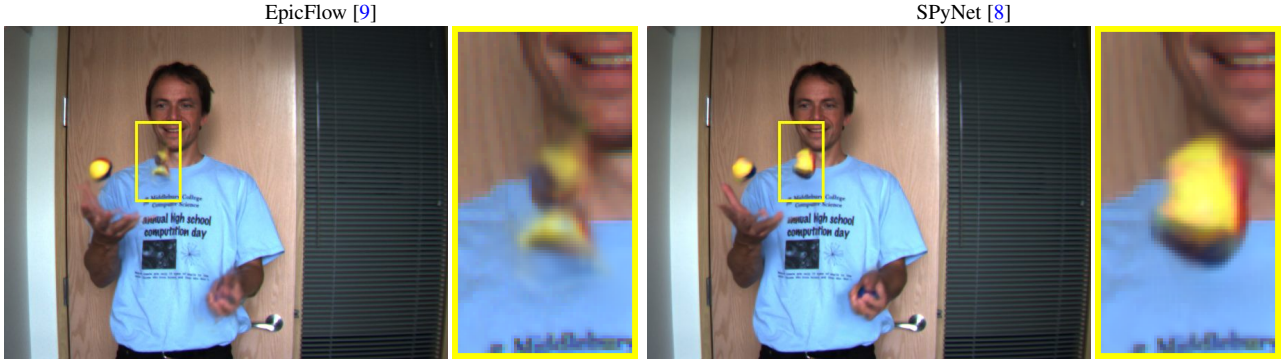
Figure 8. **Visual comparisons on the Middlebury [1] EVALUATION set.** Our method generates favorable results in the highly textured region.



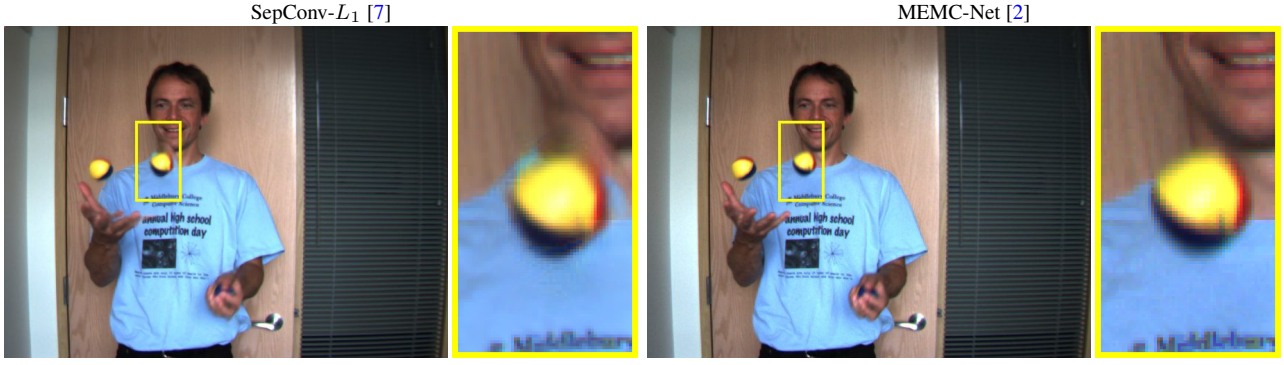
MIND [5] ToFlow [11]



EpicFlow [9] SPyNet [8]



SepConv-L_1 [7] MEMC-Net [2]



DAIN (Ours) Ground-truth

Figure 9. **Visual comparisons on the Middlebury [1] OTHER set.** Our method preserves the shapes of the balls well.


MIND [5]

ToFlow [11]


EpicFlow [9]

SPyNet [8]


SepConv-L_1 [7]

MEMC-Net [2]


DAIN (Ours)

Ground-truth

Figure 10. **Visual comparisons on the Middlebury [1] OTHER set.** The fine structure around the shadow of the lid constructed by our method is more consistent with the ground truth than by the state-of-the-art approaches.

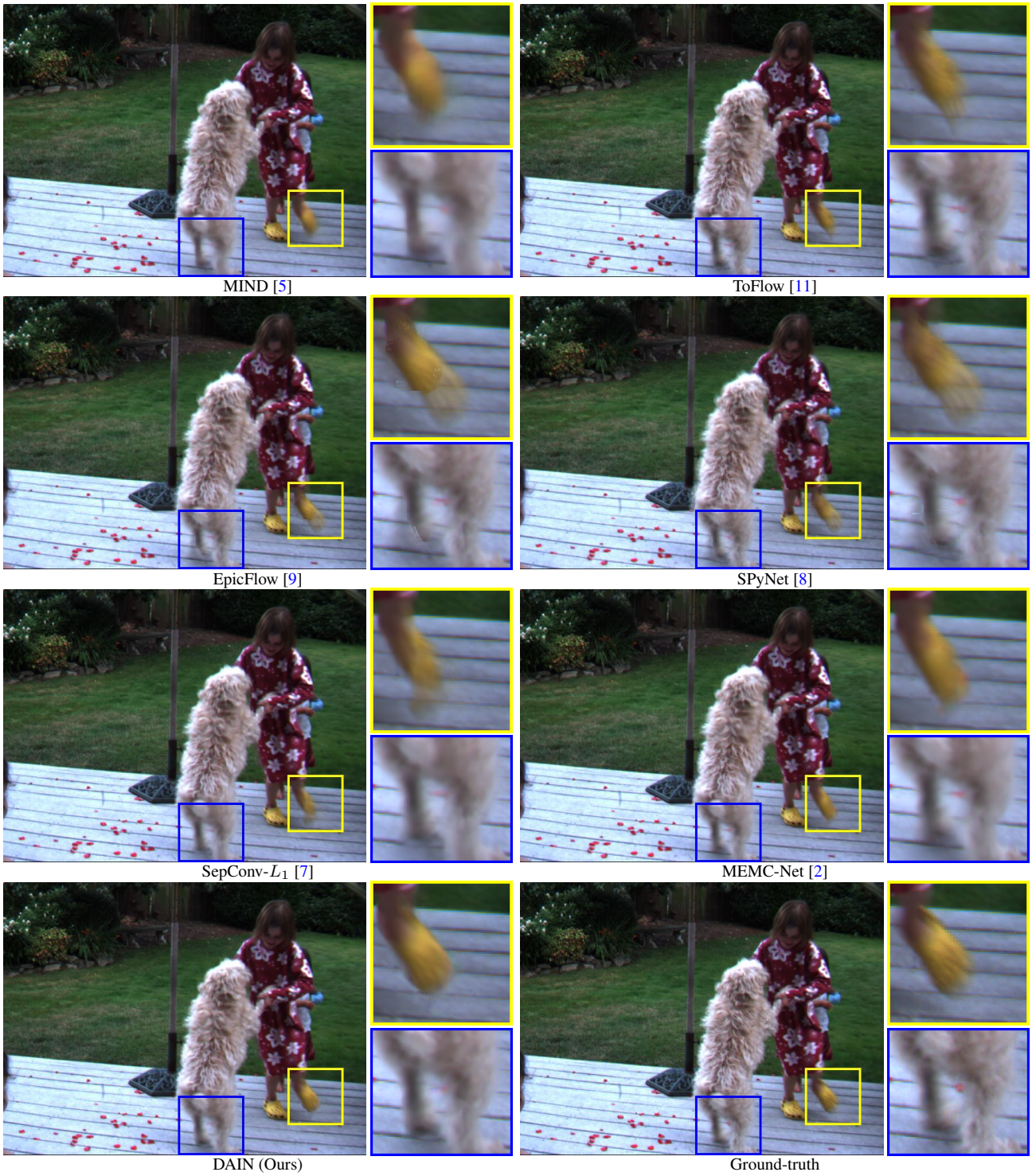
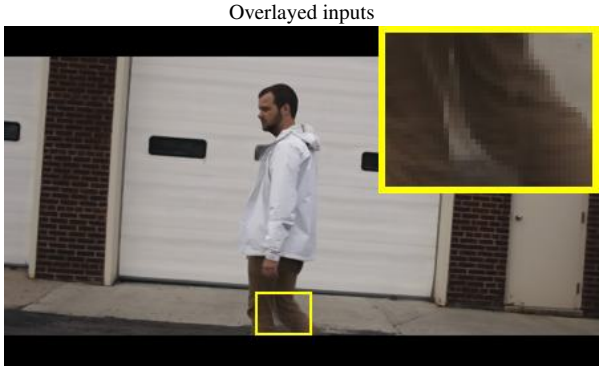


Figure 11. **Visual comparisons on the Middlebury [1] OTHER set.** Our method reconstructs the fine details of the fur and preserves the shape of the shoe well.

5.2.2 Vimeo90K dataset

Overlayed inputs

MIND [5]



ToFlow [11]

SepConv-L_f [7]

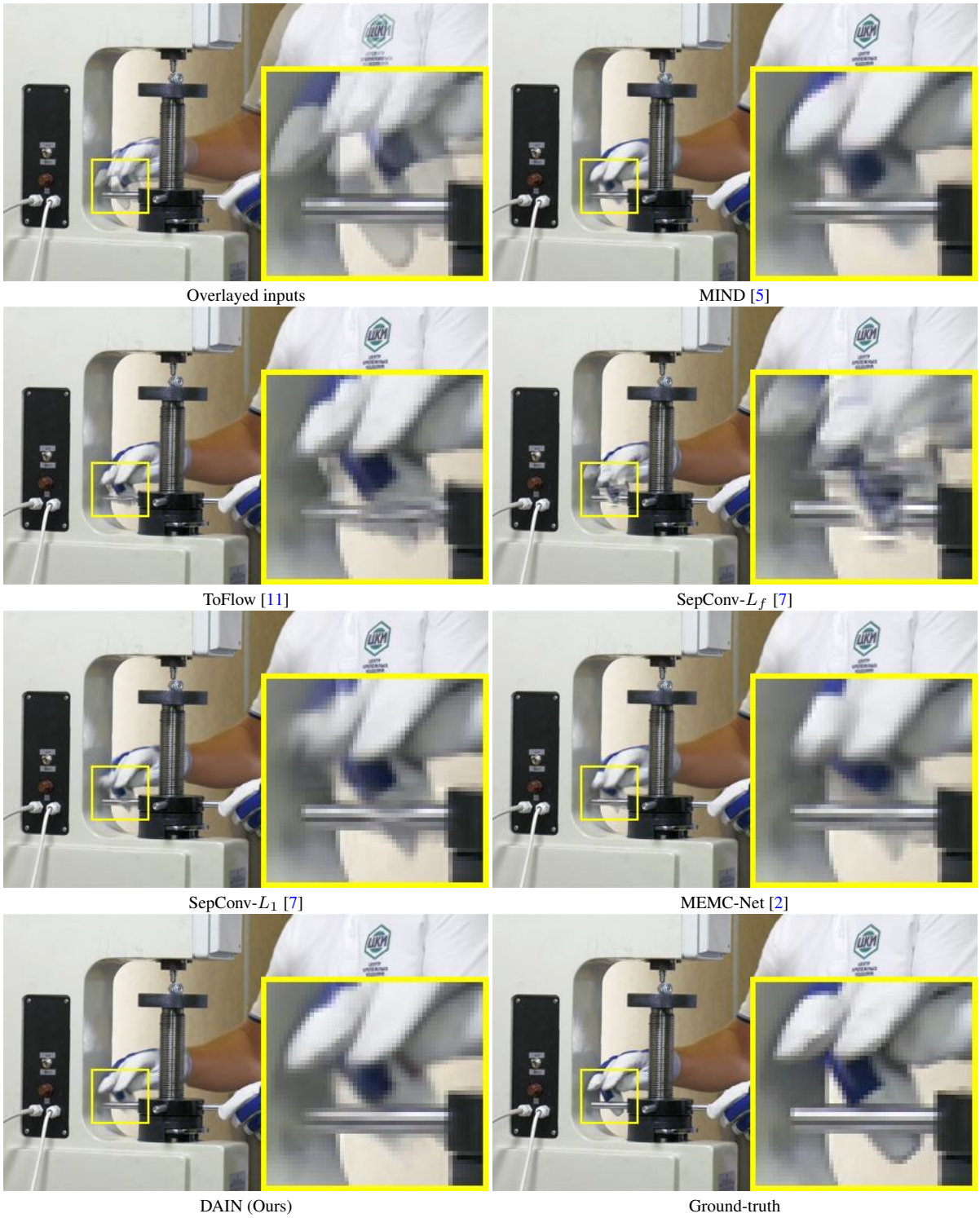
SepConv-L_1 [7]

MEMC-Net [2]

DAIN (Ours)

Ground-truth

Figure 12. **Visual comparisons on the Vimeo90K [11] test set.** Our method reconstructs the legs well.



Overlayed inputs | MIND [5]

ToFlow [11] | SepConv-L_f [7]

SepConv-L_1 [7] | MEMC-Net [2]

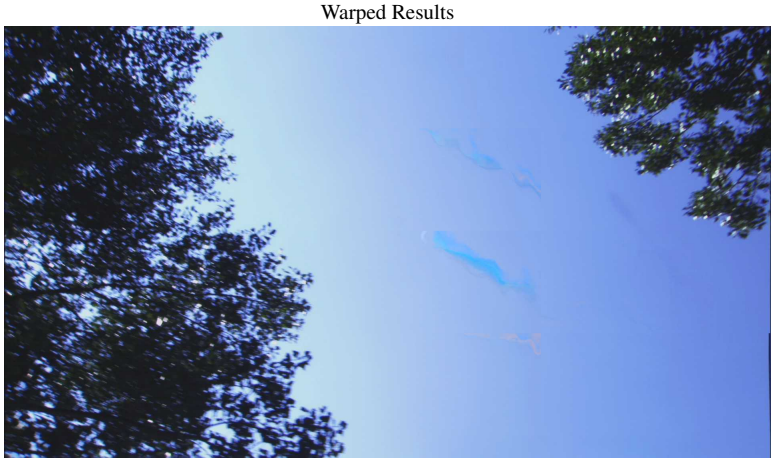
DAIN (Ours) | Ground-truth

Figure 13. **Visual comparisons on the Vimeo90K [11] test set.** Our method maintains the structures of both the fingers in gloves and the steel bar of the devices well.

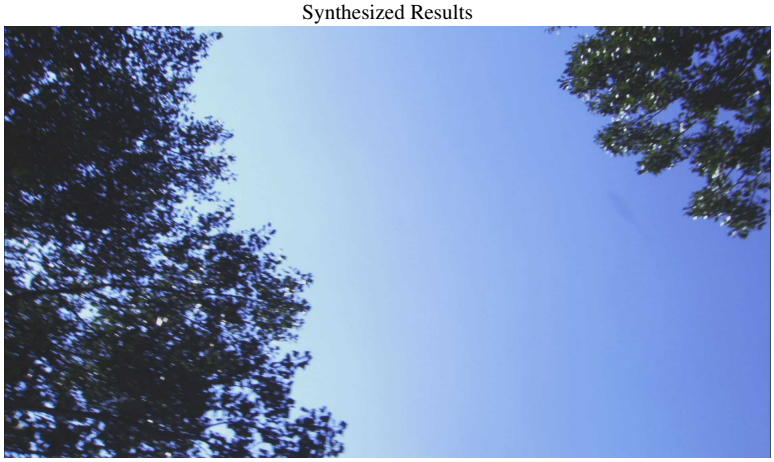
5.3. HD video results

The HD video results are available in our website. Although we show in the main paper that our DAIN model achieves better PSNR and SSIM values against the MEMC-Net [2] algorithm on the HD dataset, we observe that there are some annoying artifacts in the *Bluesky* and *Sunflower* videos. Specifically, we discover that these artifacts are introduced by the frame synthesis network. In Figure 14, we present the 4-th frame results of the *Bluesky* video. The three images are generated by the adaptive warping layer, the frame synthesis network and the corresponding ground-truth frame respectively. The artifacts appeared in the flat sky area of the synthesized result suggest us that a more robust network should be proposed to deal with high-resolution images.

Warped Results



Synthesized Results



Ground-Truth

Figure 14. **Limitations of the proposed method on HD dataset.**

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 3, 4, 6, 7, 8, 9, 10, 11, 12, 13

[2] 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. *arXiv*, 2018. 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

[3] 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 *CVPR*, 2018. 6, 7, 8, 9, 10

[4] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 4

[5] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *ECCV*, 2016. 11, 12, 13, 14, 15

[6] S. Niklaus and F. Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 6, 7, 8, 9, 10

[7] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

[8] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 11, 12, 13

[9] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 6, 7, 8, 9, 10, 11, 12, 13

[10] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 3, 4

[11] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *arXiv*, 2017. 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15