Depth-Aware Video Frame Interpolation

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

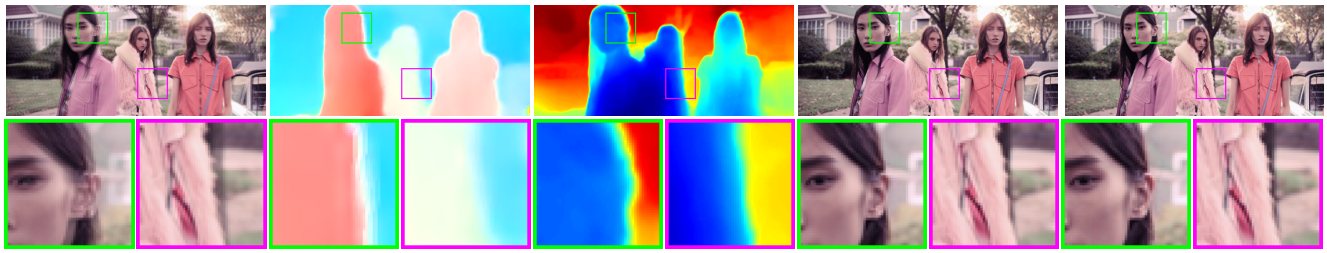

Figure 1. **Example of video frame interpolation.** We propose a depth-aware video frame interpolation approach to exploit the depth cue for detecting occlusion. Our method estimates optical flow with clear motion boundaries and thus generates high-quality frames.

Abstract

Video frame interpolation aims to synthesize non-existent frames in-between the original frames. While significant advances have been made from the recent deep convolutional neural networks, the quality of interpolation is often reduced due to large object motion or occlusion. In this work, we propose a video frame interpolation method which explicitly detects the occlusion by exploring the depth information. Specifically, we develop a depth-aware flow projection layer to synthesize intermediate flows that preferably sample closer objects than farther ones. In addition, we learn hierarchical features to gather contextual information from neighboring pixels. The proposed model then warps the input frames, depth maps, and contextual features based on the optical flow and local interpolation kernels for synthesizing the output frame. Our model is compact, efficient, and fully differentiable. Quantitative and qualitative results demonstrate that the proposed model performs favorably against state-of-the-art frame interpolation methods on a wide variety of datasets. The source code and pre-trained model are available at https://github.com/baowenbo/DAIN.

1. Introduction

Video frame interpolation has attracted considerable attention in the computer vision community as it can be applied to numerous applications such as slow motion generation [14], novel view synthesis [10], frame rate up-conversion [3, 4], and frame recovery in video streaming [38]. The videos with a high frame rate can avoid common artifacts, such as temporal jittering and motion blurriness, and therefore are visually more appealing to the viewers. However, with the advances of recent deep convolutional neural networks (CNNs) on video frame interpolation [14, 21, 23, 25, 39], it is still challenging to generate high-quality frames due to large motion and occlusions.

To handle large motion, several approaches use a coarse-to-fine strategy [21] or adopt advanced flow estimation architecture [23], e.g., PWC-Net [34], to estimate more accurate optical flow. On the other hand, a straightforward approach to handle occlusion is to estimate an occlusion mask for adaptively blending the pixels [2, 14, 39]. Some recent methods [24, 25] learn spatially-varying interpolation kernels to adaptively synthesize pixels from a large neighborhood. Recently, the contextual features from a pre-trained classification network have been shown effective for frame synthesis [23] as the contextual features are extracted from a large receptive field. However, all the existing methods rely on a large amount of training data and the model capacity to *implicitly* infer the occlusion, which may not be effective to handle a wide variety of scenes in the wild.

In this work, we propose to *explicitly* detect the occlusion by exploiting the depth information for video frame interpolation. The proposed algorithm is based on a simple observation that closer objects should be preferably synthesized in the intermediate frame. Specifically, we first estimate the bi-directional optical flow and depth maps from the two input frames. To warp the input frames, we adopt a

*Corresponding author

flow projection layer [2] to generate intermediate flows. As multiple flow vectors may encounter at the same position, we calculate the contribution of each flow vector based on the depth value for aggregation. In contrast to a simple average of flows, the proposed depth-aware flow projection layer generates flows with clearer motion boundaries due to the effect of depth.

Based on our depth-aware flow projection layer, we propose a Depth-Aware video frame INterpolation (DAIN) model that effectively exploits the optical flow, local interpolation kernels, depth maps, and contextual features to synthesize high-quality video frames. Instead of relying on a pre-trained recognition network, e.g., ResNet [13], we learn *hierarchical* features to extract effective context information from a large neighborhood. We use the adaptive warping layer [2] to warp the input frames, contextual features, and depth maps based on the estimated flows and local interpolation kernels. Finally, we generate the output frame with residual learning. As shown in Figure 1, our model is able to generate frames with clear object shapes and sharp edges. Furthermore, the proposed method can generate arbitrary in-between frames for creating slow-motion videos. Extensive experiments on multiple benchmarks, including the Middlebury [1], UCF101 [33], Vimeo90K [39], and HD [2] datasets, demonstrate that the proposed DAIN performs favorably against existing video frame interpolation methods.

We make the following contributions in this work:

- We explicitly detect the occlusion within a depth-aware flow projection layer to preferably synthesize closer objects than farther ones.
- We propose a depth-aware video frame interpolation method that tightly integrates optical flow, local interpolation kernels, depth maps, and learnable hierarchical features for high-quality frame synthesis.
- We demonstrate that the proposed model is more effective, efficient, and compact than the state-of-the-art approaches.

2. Related Work

Video frame interpolation is a long-standing topic and has been extensively studied in the literature [3, 7, 16, 26, 36]. In this section, we focus our discussions on recent learning-based algorithms. In addition, we discuss the related topic on depth estimation.

Video frame interpolation. As a pioneer of CNN-based methods, Long et al. [22] train a generic CNN to directly synthesize the in-between frame. Their results, however, suffer from severe blurriness as a generic CNN is not able to capture the multi-modal distribution of natural images and videos. Then, Liu et al. [21] propose the deep voxel flow, a 3D optical flow across space and time, to warp input frames based on a trilinear sampling. While the frames synthesized from flow suffer less blurriness, the flow estimation is still challenging for scenes with large motion. Inaccurate flow may result in severe distortion and visual artifacts.

Instead of relying on optical flow, the AdaConv [24] and SepConv [25] methods estimate spatially-adaptive interpolation kernels to synthesize pixels from a large neighborhood. However, these kernel-based approaches typically require high memory footprint and entail heavy computational load. Recently, Bao et al. [2] integrate the flow-based and kernel-based approaches into an end-to-end network to inherit the benefit from both sides. The input frames are first warped by the optical flow and then sampled via the learned interpolation kernels within an adaptive warping layer.

Existing methods *implicitly* handle the occlusion by estimating occlusion masks [2, 14, 39], extracting contextual features [2, 23], or learning large local interpolation kernels [24, 25]. In contrast, we *explicitly* detect the occlusion by utilizing the depth information in the flow projection layer. Moreover, we incorporate the depth map with the learned hierarchical features as the contextual information to synthesize the output frame.

Depth estimation. Depth is one of the key visual information to understand the 3D geometry of a scene and has been exploited in several recognition tasks, e.g., image segmentation [41] and object detection [35]. Conventional methods [12, 15, 27] require stereo images as input to estimate the disparity. Recently, several learning-based approaches [8, 9, 11, 18, 20, 31, 32, 37] aim to estimate the depth from a single image. In this work, we use the model of Chen et al. [6], which is an hourglass network trained on the MegaDepth dataset [19], for predicting the depth maps from the input frames. We show that the initialization of depth network is crucial to infer the occlusion. We then jointly fine-tune the depth network with other sub-modules for frame interpolation. Therefore, our model learns a *relative* depth for warping and interpolation.

We note that several approaches jointly estimate optical flow and depth by exploiting the cross-task constraints and consistency [40, 42, 43]. While the proposed model also jointly estimates optical flow and depth, our flow and depth are optimized for frame interpolation, which may not resemble the real values of the pixel motion and scene depth.

3. Depth-Aware Video Frame Interpolation

In this section, we first provide an overview of our frame interpolation algorithm. We then introduce the proposed depth-aware flow projection layer, which is the key component to handle occlusion for flow aggregation. Finally, we describe the design of all the sub-modules and provide the implementation details of the proposed model.

3.1. Algorithm Overview

Given two input frames $\mathbf{I}_0(\mathbf{x})$ and $\mathbf{I}_1(\mathbf{x})$, where $\mathbf{x} \in [1, H] \times [1, W]$ indicates the 2D spatial coordinate of the image plane, and H and W are the height and width of the image, our goal is to synthesize an intermediate frame $\hat{\mathbf{I}}_t$ at time $t \in [0, 1]$. The proposed method requires optical flows to warp the input frames for synthesizing the intermediate frame. We first estimate the bi-directional optical flows, denoted by $\mathbf{F}_{0\to1}$ and $\mathbf{F}_{1\to0}$, respectively. To synthesize the intermediate frame $\hat{\mathbf{I}}_t$, there are two common strategies. First, one could apply the forward warping [23] to warp $\mathbf{I}_0$ based on $\mathbf{F}_{0\to1}$ and warp $\mathbf{I}_1$ based on $\mathbf{F}_{1\to0}$. However, the forward warping may lead to holes on the warped image. The second strategy is to approximate the intermediate flows, i.e., $\mathbf{F}_{t\to0}$ and $\mathbf{F}_{t\to1}$, and then apply the backward warping to sample the input frames. To approximate the intermediate flows, one can borrow the flow vectors from the same grid coordinate in $\mathbf{F}_{0\to1}$ and $\mathbf{F}_{1\to0}$ [14], or aggregate the flow vectors that pass through the same position [2]. In this work, we adopt the flow projection layer in Bao et al. [2] to aggregate the flow vectors while considering the depth order to detect the occlusion.

After obtaining the intermediate flows, we warp the input frames, contextual features, and depth maps within an adaptive warping layer [2] based on the optical flows and interpolation kernels. Finally, we adopt a frame synthesis network to generate the interpolated frame.

3.2. Depth-Aware Flow Projection

The flow projection layer approximates the intermediate flow at a given position $\mathbf{x}$ by "reversing" the flow vectors passing through $\mathbf{x}$ at time t. If the flow $\mathbf{F}_{0\to1}(\mathbf{y})$ passes through $\mathbf{x}$ at time t, one can approximate $\mathbf{F}_{t\to0}(\mathbf{x})$ by $-t\ \mathbf{F}_{0\to1}(\mathbf{y})$. Similarly, we approximate $\mathbf{F}_{t\to1}(\mathbf{x})$ by $-(1-t)\ \mathbf{F}_{1\to0}(\mathbf{y})$. However, as illustrated in the 1D space-time example of Figure 2, multiple flow vectors could be projected to the same position at time t. Instead of aggregating the flows by a simple average [2], we propose to consider the depth ordering for aggregation. Specifically, we assume that D_0 is the depth map of $\mathbf{I}_0$ and $\mathcal{S}(\mathbf{x}) = \{\mathbf{y} : \text{round}(\mathbf{y} + t\ \mathbf{F}_{0\to1}(\mathbf{y})) = \mathbf{x}, \forall\ \mathbf{y} \in [1, H] \times [1, W]\}$ indicates the set of pixels that pass through the position $\mathbf{x}$ at time t. The projected flow $\mathbf{F}_{t\to0}$ is defined by:

$$\mathbf{F}_{t\to0}(\mathbf{x}) = -t \cdot \frac{\sum\limits_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} w_0(\mathbf{y}) \cdot \mathbf{F}_{0\to1}(\mathbf{y})}{\sum\limits_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} w_0(\mathbf{y})}, \tag{1}$$

where the weight w_0 is the reciprocal of depth:

$$w_0(\mathbf{y}) = \frac{1}{D_0(\mathbf{y})}. \tag{2}$$

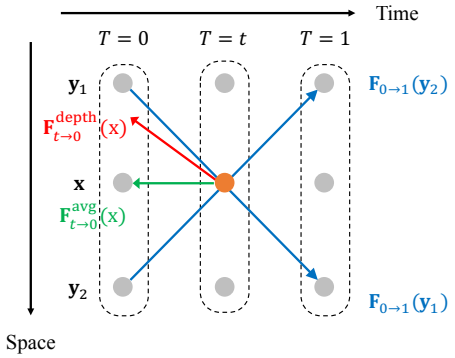

Figure 2. **Proposed depth-aware flow projection.** The existing flow projection method [2] obtains an average flow vector which may not point to the correct object or pixel. In contrast, we rewrite the flows according to the depth values and generate the flow vector pointing to the closer pixel.

Similarly, the projected flow $\mathbf{F}_{t\to1}$ can be obtained from the flow $\mathbf{F}_{1\to0}$ and depth map D_1. By this way, the projected flows tend to sample the closer objects and reduce the contribution of occluded pixels which have larger depth values. As shown in Figure 2, the flow projection used in [2] generates an average flow vector (the **green** arrow), which may not point to the correct pixel for sampling. In contrast, the projected flow from our depth-aware flow projection layer (the **red** arrow) points to the pixel with a smaller depth value.

On the other hand, there might exist positions where none of the flow vectors pass through, leading to holes in the intermediate flow. To fill in the holes, we use the outside-in strategy [1]: the flow in the hole position is computed by averaging the available flows from its neighbors:

$$\mathbf{F}_{t\to0}(\mathbf{x}) = \frac{1}{|\mathcal{N}(\mathbf{x})|} \sum_{\mathbf{x}'\in\mathcal{N}(\mathbf{x})} \mathbf{F}_{t\to0}(\mathbf{x}'), \tag{3}$$

where $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}' : |\mathcal{S}(\mathbf{x}')| > 0\}$ is the 4-neighbors of $\mathbf{x}$. From (1) and (3), we obtain dense intermediate flow fields $\mathbf{F}_{t\to0}$ and $\mathbf{F}_{t\to1}$ for warping the input frames.

The proposed depth-aware flow projection layer is fully differentiable so that both the flow and depth estimation networks can be jointly optimized during the training. We provide the details of back-propagation in depth-aware flow projection in the supplementary materials.

3.3. Video Frame Interpolation

The proposed model consists of the following sub-modules: the flow estimation, depth estimation, context extraction, kernel estimation, and frame synthesis networks. We use the proposed depth-aware flow projection layer to obtain intermediate flows and then warp the input frames, depth maps, and contextual features within the adaptive warping layer. Finally, the frame synthesis network generates the output frame with residual learning. We show the

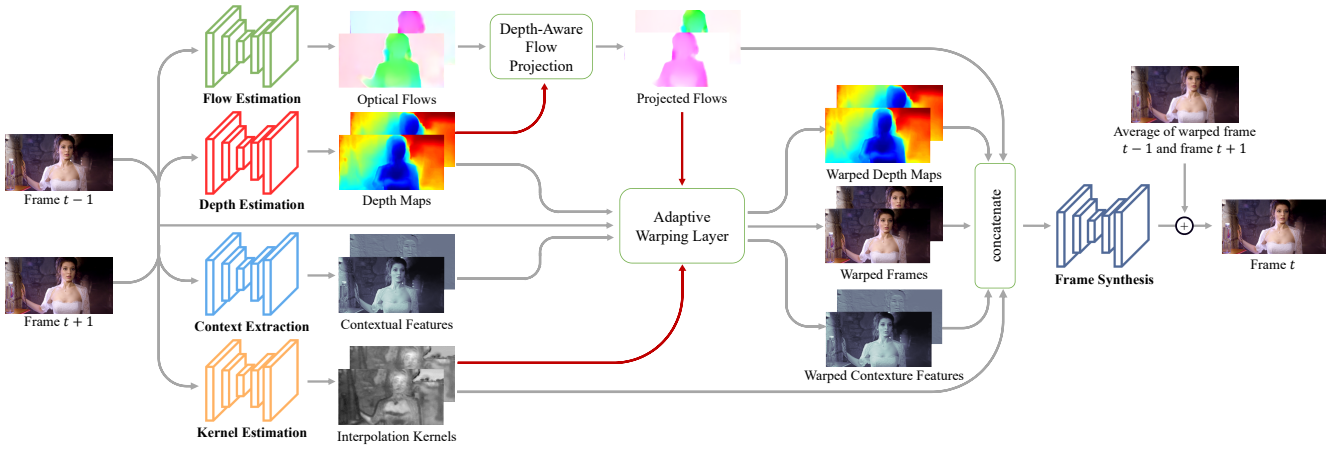

Figure 3. **Architecture of the proposed depth-aware video frame interpolation model.** Given two input frames, we first estimate the optical flows and depth maps and use the proposed depth-aware flow projection layer to generate intermediate flows. We then adopt the adaptive warping layer to warp the input frames, depth maps, and contextual features based on the flows and spatially varying interpolation kernels. Finally, we apply a frame synthesis network to generate the output frame.

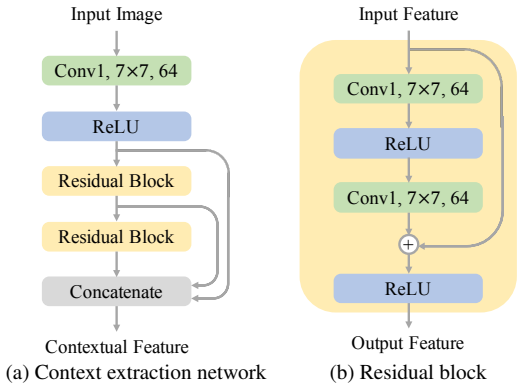

Figure 4. **Structure of the context extraction network.** Instead of using the weights of a pre-trained classification network [23], we train our context extraction network from scratch and learn hierarchical features for video frame interpolation.

overall network architecture in Figure 3. Below we describe the details of each sub-network.

Flow estimation. We adopt the state-of-the-art flow model, PWC-Net [34], as our flow estimation network. As learning optical flow without ground-truth supervision is extremely difficult, we initialize our flow estimation network from the pre-trained PWC-Net.

Depth estimation. We use the hourglass architecture [6] as our depth estimation network. To obtain meaningful depth information for the flow projection, we initialize the depth estimation network from the pre-trained model of Li et al. [19].

Context extraction. In [2] and [23], the contextual information is extracted by a pre-trained ResNet [13], i.e., the feature maps of the first convolutional layer. However, the features from the ResNet are for the image classification task, which may not be effective for video frame interpolation. Therefore, we propose to *learn* the contextual features. Specifically, we construct a context extraction network with one 7×7 convolutional layer and two residual blocks, as shown in Figure 4(a). The residual block consists of two 3×3 convolutional and two ReLU activation layers (Figure 4(b)). We do not use any normalization layer, e.g., batch normalization. We then concatenate the features from the first convolutional layer and the two residual blocks, resulting in a *hierarchical* feature. Our context extraction network is trained from scratch and, therefore, learns effective contextual features for video frame interpolation.

Kernel estimation and adaptive warping layer. The local interpolation kernels have been shown to be effective for synthesizing a pixel from a large local neighborhood [24, 25]. Bao et al. [2] further integrate the interpolation kernels and optical flow within an adaptive warping layer. The adaptive warping layer synthesizes a new pixel by sampling the input image within a local window, where the center of the window is specified by optical flow. Here we use a U-Net architecture [30] to estimate 4×4 local kernels for each pixel. With the interpolation kernels and intermediate flows generated from the depth-aware flow projection layer, we adopt the adaptive warping layer [2] to warp the input frames, depth maps, and contextual features. More details of the adaptive warping layer and the configuration of the kernel estimation network are provided in the supplementary materials.

Frame synthesis. To generate the final output frame, we construct a frame synthesis network, which consists of 3 residual blocks. We concatenate the warped input frames, warped depth maps, warped contextual features, projected flows, and interpolation kernels as the input to the frame synthesis network. In addition, we linearly blend the two

warped frames and enforce the network to predict the residuals between the ground-truth frame and the blended frame. We note that the warped frames are already aligned by the optical flow. Therefore, the frame synthesis network focuses on enhancing the details to make the output frame look sharper. We provide the detailed configurations of the frame synthesis network in the supplementary material.

3.4. Implementation Details

Loss Function. We denote the synthesized frame by $\hat{\mathbf{I}}_t$ and the ground-truth frame by $\mathbf{I}_t^{GT}$. We train the proposed model by optimizing the following loss function:

$$\mathcal{L} = \sum_{\mathbf{x}} \rho \left(\hat{\mathbf{I}}_t(\mathbf{x}) - \mathbf{I}_t^{GT}(\mathbf{x}) \right), \tag{4}$$

where $\rho(x) = \sqrt{x^2 + \epsilon^2}$ is the Charbonnier penalty function [5]. We set the constant ϵ to $1e-6$.

Training Dataset. We use the Vimeo90K dataset [39] to train our model. The Vimeo90K dataset has 51,312 triplets for training, where each triplet contains 3 consecutive video frames with a resolution of 256×448 pixels. We train our network to predict the middle frame (i.e., $t = 0.5$) of each triplet. At the test time, our model is able to generate arbitrary intermediate frames for any $t \in [0, 1]$. We augment the training data by horizontal and vertical flipping as well as reversing the temporal order of the triplet.

Training Strategy. We use the AdaMax [17] to optimize the proposed network. We set the β_1 and β_2 to 0.9 and 0.999 and use a batch size of 2. The initial learning rates of the kernel estimation, context extraction, and frame synthesis networks are set to $1e-4$. As both the flow estimation and depth estimation networks are initialized from pre-trained models, we use smaller learning rates of $1e-6$ and $1e-7$, respectively. We jointly train the entire model for 30 epochs and then reduce the learning rate of each network by a factor of 0.2 and fine-tune the entire model for another 10 epochs. We train our model on an NVIDIA Titan X (Pascal) GPU card, which takes about 5 days to converge.

4. Experimental Results

In this section, we first introduce the datasets for evaluation. We then conduct ablation study to analyze the contribution of the proposed depth-aware flow projection and hierarchical contextual features. Then, we compare the proposed model with state-of-the-art frame interpolation algorithms. Finally, we discuss the limitation and future work of our method.

4.1. Evaluation Datasets and Metrics

We evaluate the proposed algorithm on multiple video datasets with different image resolutions.

Table 1. **Analysis on Depth-Aware (DA) flow projection.** M.B. is short for the OTHER set of the Middlebury dataset. The proposed model (DA-*Opti*) shows a substantial improvement against the other variations.

Method	UCF101 [33]		Vimeo90K [39]		M.B. [1]	HD [2]	
	PSNR	SSIM	PSNR	SSIM	IE	PSNR	SSIM
DA-*None*	34.91	0.9679	34.47	0.9746	2.10	31.46	0.9174
DA-*Scra*	34.85	0.9677	34.30	0.9735	2.13	31.42	0.9164
DA-*Pret*	34.91	0.9680	34.52	0.9747	2.07	31.52	0.9178
DA-*Opti*	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**	**31.70**	**0.9193**

Middlebury. The Middlebury benchmark [1] is widely used to evaluate video frame interpolation methods. There are two subsets. The OTHER set provides the ground-truth middle frames, while the EVALUATION set hides the ground-truth and can be evaluated by uploading the results to the benchmark website. The image resolution in this dataset is around 640×480 pixels.

Vimeo90K. There are 3,782 triplets in the test set of the Vimeo90K dataset [39]. The image resolution in this dataset is 448×256 pixels.

UCF101. The UCF101 dataset [33] contains videos with a large variety of human actions. There are 379 triplets with a resolution of 256×256 pixels.

HD. Bao et al. [2] collect 11 high-resolution videos for evaluation. The HD dataset consists of four 1920×1080p, three 1280×720p and four 1280×544p videos. The motion in this dataset is typically larger than other datasets.

Metrics. We compute the average Interpolation Error (IE) and Normalized Interpolation Error (NIE) on the Middlebury dataset. Lower IEs or NIEs indicate better performance. We evaluate the PSNR and SSIM on the UCF101, Vimeo90K, and the HD datasets for comparisons.

4.2. Model Analysis

We analyze the contribution of the two key components in the proposed model: the depth-aware flow projection layer and learned hierarchical contextual features.

Depth-aware flow projection. To analyze the effectiveness of our depth-aware flow projection layer, we train the following variations (DA is short for Depth-Aware):

- DA-*None*: We remove the depth estimation network and use a simple average [2] to aggregate the flows in the flow projection layer.
- DA-*Scra*: We initialize the depth estimation network from scratch and optimize it with the whole model.
- DA-*Pret*: We initialize the depth estimation network from the pre-trained model of [19] but freeze the parameters.
- DA-*Opti*: We initialize the depth estimation network from the pre-trained model of [19] and jointly optimize it with the entire model.

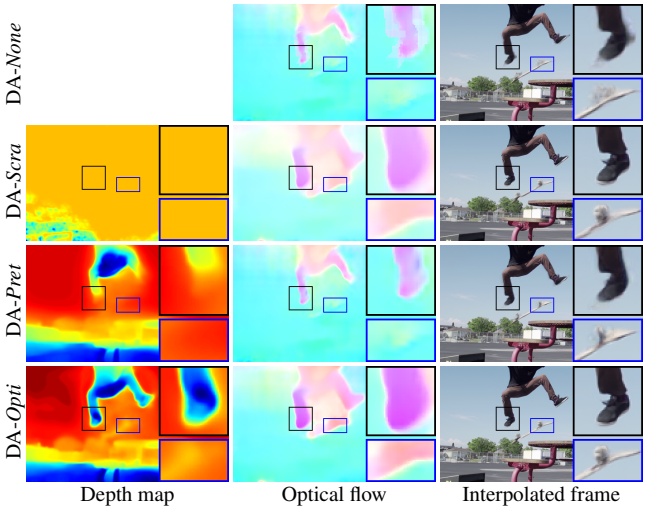

Figure 5. **Effect of the depth-aware flow projection.** The DA-*Scra* model cannot learn any meaningful depth information. The DA-*Pret* model initializes the depth estimation network from a pre-trained model and generates clear motion boundaries for frame interpolation. The DA-*Opti* model further optimizes the depth maps and generates sharper edges and shapes.

We show the quantitative results of the above models in Table 1 and provide a visualization of the depth, flow, and interpolated frames in Figure 5. First, the DA-*Scra* model performs worse than the DA-*None* model. As shown in the second row of Figure 5, the DA-*Scra* model cannot learn any meaningful depth information from the random initialization. When initializing from the pre-trained depth model, the DA-*Pret* model shows a substantial performance improvement and generates flow with clear motion boundaries. After jointly optimizing the whole network, the DA-*Opti* model further improves the depth maps, e.g., the man's legs, and generates sharper edges for the shoes and skateboard in the interpolated frame. The analysis demonstrates that the proposed model effectively utilizes the depth information to generate high-quality results.

Learned hierarchical context. In the proposed model, we use contextual features as one of the inputs to the frame synthesis network. We analyze the contribution of the different contextual features, including the pre-trained *conv1* features (PCF), the learned *conv1* features (LCF), and the learned hierarchical features (LHF). In addition, we also consider the depth maps (D) as the additional contextual features.

We show the quantitative results in Table 2 and compare the interpolated images in Figure 6. Without using any contextual information, the model does not perform well and generates blurred results. By introducing the contextual features, e.g., the pre-trained *conv1* features or depth maps, the performance is greatly improved. We further demonstrate that the *learned* contextual features, especially the learned hierarchical features, lead to a substantial improvement on

Table 2. **Analysis on contextual features.** We compare the contextual features from different sources: the pre-trained *conv1* features (PCF), learned *conv1* features (LCF), learned hierarchical features (LHF), and the depth maps (D).

Context	UCF101 [33]		Vimeo [39]		M.B. [1]	HD [2]	
	PSNR	SSIM	PSNR	SSIM	IE	PSNR	SSIM
None	34.84	0.9679	34.38	0.9738	2.21	31.35	0.9178
PCF	34.90	0.9681	34.41	0.9740	2.16	31.43	0.9160
D	34.90	0.9682	34.44	0.9740	2.14	31.62	0.9183
PCF + D	34.97	0.9682	34.49	0.9746	2.13	**31.73**	**0.9194**
LCF + D	34.87	0.9680	34.54	0.9749	2.08	31.56	0.9185
LHF + D	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**	31.70	0.9193

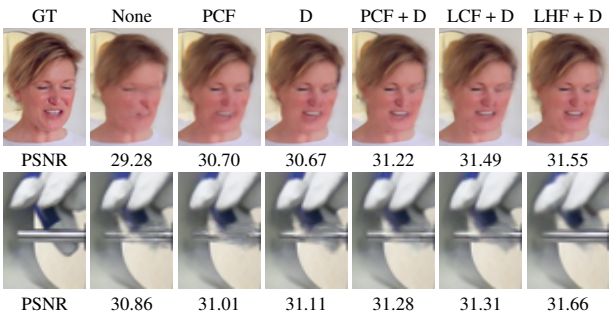

Figure 6. **Effect of contextual features.** The proposed model uses the learned hierarchical features (LHF) and depth maps (D) for frame synthesis, which generates clearer and sharper content.

the Vimeo90K and the Middlebury datasets. The model using both the depth maps and learned hierarchical features also generates sharper and clearer content.

4.3. Comparisons with State-of-the-arts

We evaluate the proposed DAIN against the following CNN-based frame interpolation algorithms: MIND [22], DVF [21], SepConv [25], CtxSyn [23], ToFlow [39], Super SloMo [14] and MEMC-Net [2]. In addition, we use the algorithm of Baker et al. [1] to generate interpolation results for two optical flow estimation algorithms, EpicFlow [29] and SPyNet [28], for comparisons.

In Table 3, we show the comparisons on the EVALUATION set of the Middlebury benchmark [1], which are also available on the Middlebury website. The proposed model performs favorably against all the compared methods. At the time of submission, our method ranks 1st in terms of NIE and 3rd in terms of IE among all published algorithms on the Middlebury website. We show a visual comparison in Figure 7, where the EpicFlow [29], ToFlow [39], SepConv [25] and MEMC-Net [2] methods produce ghosting artifacts on the balls or foot. In contrast, the proposed method reconstructs a clear shape of the ball. Compared to the CtxSyn [23] and Super SloMo [14] methods, our approach generates more details on the slippers and foot.

In Table 4, we provide quantitative performances on the UCF101 [33], Vimeo90K [39], HD [2], and Middle-

Table 3. **Quantitative comparisons on the Middlebury EVALUATION set.** The numbers in **red** and blue represent the best and second best performance. The proposed DAIN method performs favorably against other approaches in terms of IE and NIE.

Method	Mequon		Schefflera		Urban		Teddy		Backyard		Basketball		Dumptruck		Evergreen		Average	
	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE
EpicFlow [29]	3.17	0.62	3.79	0.70	4.28	1.06	6.37	1.09	11.2	1.18	6.23	1.10	8.11	1.00	8.76	1.04	6.49	0.97
SepConv-L_1 [25]	2.52	0.54	3.56	0.67	4.17	1.07	5.41	1.03	10.2	0.99	5.47	0.96	6.88	0.68	6.63	0.70	5.61	0.83
ToFlow [39]	2.54	0.55	3.70	0.72	3.43	0.92	5.05	0.96	9.84	0.97	5.34	0.98	6.88	0.72	7.14	0.90	5.49	0.84
Super SloMo [14]	2.51	0.59	3.66	0.72	**2.91**	0.74	5.05	0.98	9.56	0.94	5.37	0.96	6.69	0.60	6.73	0.69	5.31	0.78
CtxSyn [23]	**2.24**	**0.50**	**2.96**	**0.55**	4.32	1.42	**4.21**	0.87	9.59	0.95	5.22	0.94	7.02	0.68	6.66	0.67	5.28	0.82
MEMC-Net [2]	2.47	0.60	3.49	0.65	4.63	1.42	4.94	0.88	8.91	0.93	**4.70**	0.86	6.46	0.66	6.35	**0.64**	5.24	0.83
DAIN (Ours)	2.38	0.58	3.28	0.60	3.32	**0.69**	4.65	**0.86**	**7.88**	**0.87**	4.73	**0.85**	**6.36**	**0.59**	**6.25**	0.66	**4.86**	**0.71**

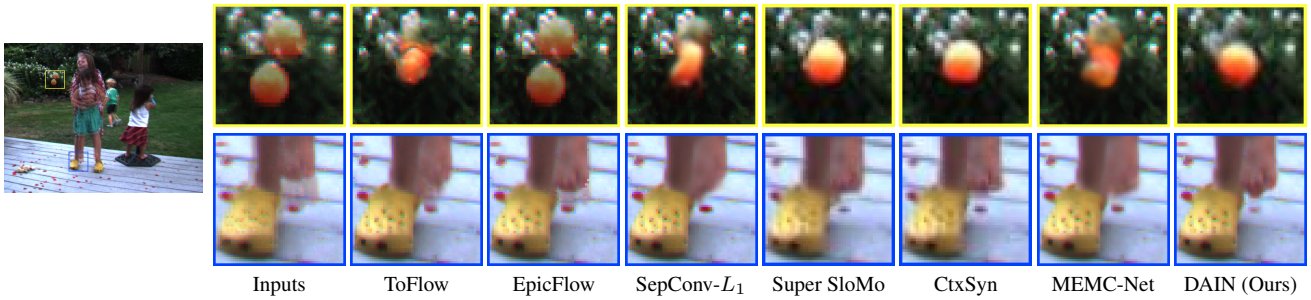

Figure 7. **Visual comparisons on the Middlebury EVALUATION set.** The proposed method reconstructs a clear shape of the ball and restores more details on the slippers and foot.

Table 4. **Quantitative comparisons on the UCF101, Vimeo90K, HD, and Middlebury OTHER datasets.** The numbers in **red** and blue indicate the best and second best performance. We also compare the model parameters and runtime of each method.

Method	#Parameters (million)	Runtime (seconds)	UCF101 [33]		Vimeo90K [39]		Middlebury [1]	HD [2]	
			PSNR	SSIM	PSNR	SSIM	IE	PSNR	SSIM
SPyNet [28]	1.20	0.11	33.67	0.9633	31.95	0.9601	2.49	—	—
EpicFlow [29]	—	8.80	33.71	0.9635	32.02	0.9622	2.47	—	—
MIND [22]	7.60	0.01	33.93	0.9661	33.50	0.9429	3.35	—	—
DVF [21]	1.60	0.47	34.12	0.9631	31.54	0.9462	7.75	—	—
ToFlow [39]	1.07	0.43	34.58	0.9667	33.73	0.9682	2.51	29.37	0.8772
SepConv-L_f [25]	21.6	0.20	34.69	0.9655	33.45	0.9674	2.44	30.61	0.8978
SepConv-L_1 [25]	21.6	0.20	34.78	0.9669	33.79	0.9702	2.27	30.87	0.9077
MEMC-Net [2]	70.3	0.12	34.96	0.9682	34.29	0.9739	2.12	31.39	0.9163
DAIN (Ours)	24.0	0.13	**34.99**	**0.9683**	**34.71**	**0.9756**	**2.04**	**31.64**	**0.9205**

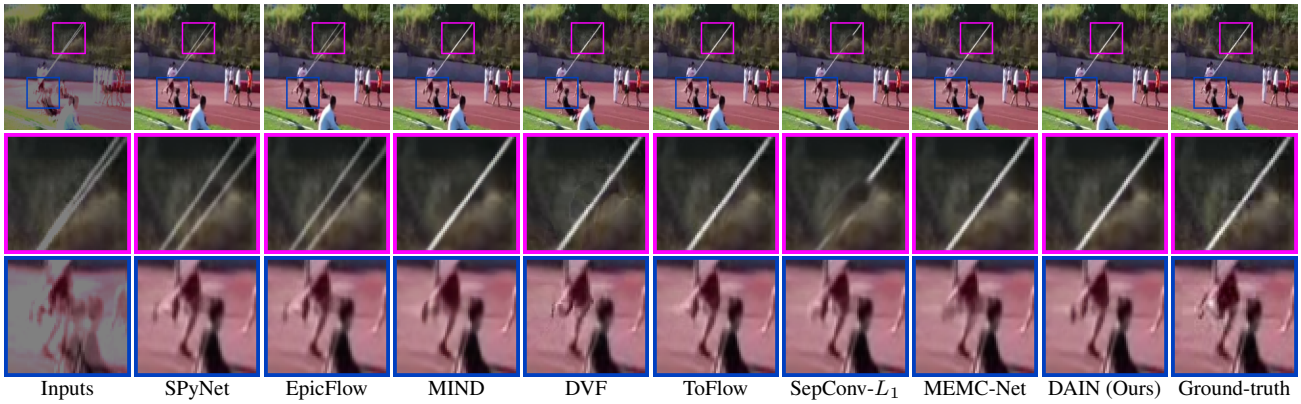

Figure 8. **Visual comparisons on the UCF101 dataset [33].** The proposed method aligns the content (e.g., the pole) well and restores more details on the man's leg.

bury [1] OTHER set. Our approach performs favorably against existing methods for all the datasets, especially on the Vimeo90K [39] dataset with a 0.42dB gain over MEMC-Net [2] in terms of PSNR.

Table 5. **Comparisons with MEMC-Net [2] on parameter and runtime**. We list the parameters (million) and runtime (seconds) of each sub-module in the MEMC-Net and the proposed model.

Sub-module	MEMC-Net [2]		DAIN (Ours)	
	#Parameters	Runtime	#Parameters	Runtime
Depth	—	—	5.35	0.043
Flow	38.6	0.024	9.37	0.074
Context	0.01	0.002	0.16	0.002
Kernel	14.2	0.008	5.51	0.004
Mask	14.2	0.008	—	—
Synthesis	3.30	0.080	3.63	0.002
Total	70.3	0.122	24.0	0.125

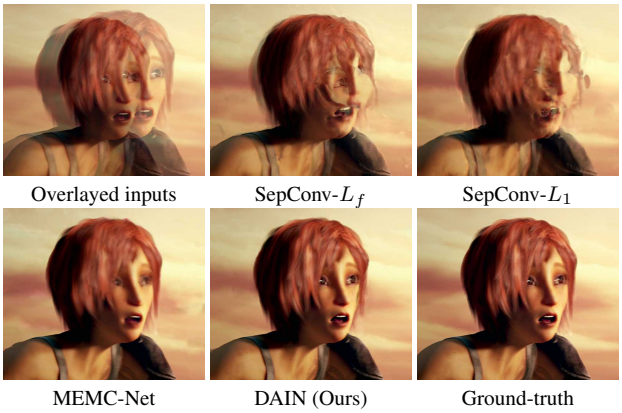

Figure 9. **Visual comparisons on the HD dataset [2].** The SepConv [25] method cannot align the content as the motion is larger than the size of interpolation kernels, e.g., 51×51. The proposed DAIN reveals more details on the hair and eyes than the state-of-the-art MEMC-Net [2].

In Figure 8, the SPyNet [28], EpicFlow [29] and SepConv [25] methods cannot align the pole well and thus produce ghosting or broken results. The MIND [22], DVF [21], ToFlow [39] and MEMC-Net [2] methods generate blurred results on the man's leg. In contrast, the proposed method aligns the pole well and generates clearer results. In Figure 9, we show an example from the HD dataset. The SepConv [25] method cannot align the content at all as the motion is larger than the size of the interpolation kernels (e.g., 51×51). Compared to the MEMC-Net [2], our method restores clearer details on the hair and face (e.g., eyes and mouth). Overall, the proposed DAIN generates more visually pleasing results with fewer artifacts than existing frame interpolation methods. In our supplementary materials, we demonstrate that our method can generate arbitrary intermediate frames to create $10\times$ slow-motion videos. More image and video results are available in our project website.

We also list the number of model parameters and execution time (test on a 640×480 image) of each method in Table 4. The proposed model uses a similar amount of parameters as the SepConv [25] but runs faster. Compared to the MEMC-Net [2], we use 69% fewer parameters (see the detailed comparison of the sub-modules in Table 5) and achieve better performance.

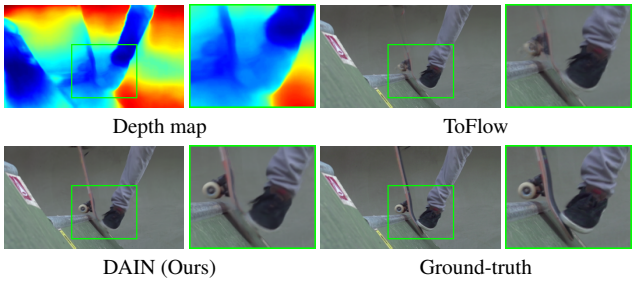

Figure 10. **Limitations of the proposed method.** When the depth maps are not estimated well, our method tends to generate blurred results and less clear boundaries.

4.4. Discussions and limitations

The proposed method relies on the depth maps to detect the occlusion for flow aggregation. However, in some challenging cases, the depth maps are not estimated well and lead to ambiguous object boundaries, as shown in the highlight region of Figure 10. Our method generates blurred results with unclear boundaries (e.g., between the shoe and skateboard). However, compared to the ToFlow [39], our method still reconstructs the skateboard well. While our current model estimates depth from a single image, it would be beneficial to obtain more accurate depth maps by jointly estimating the depth from the two input frames or modeling the consistency between optical flow and depth [43].

5. Conclusion

In this work, we propose a novel depth-aware video frame interpolation algorithm, which explicitly detects the occlusion using the depth information. We propose a depth-aware flow projection layer that encourages sampling of closer objects than farther ones. Furthermore, we exploit the learned hierarchical features and depth maps as the contextual information to synthesize the intermediate frame. The proposed model is compact and efficient. Extensive quantitative and qualitative evaluations demonstrate that the proposed method performs favorably against existing frame interpolation algorithms on diverse datasets. The state-of-the-art achievement from the proposed method sheds light for future research on exploiting the depth cue for video frame interpolation.

Acknowledgment. This work was supported in part by National Key Research and Development Program of China (2016YFB1001003), NSFC (61771306), Natural Science Foundation of Shanghai (18ZR1418100), Chinese National Key S&T Special Program (2013ZX01033001-002-002), Shanghai Key Laboratory of Digital Media Processing and Transmissions (STCSM 18DZ2270700 and 18DZ1112300). It was also supported in part by NSF Career Grant (1149783) and gifts from Adobe, Verisk, and NEC.

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. 2, 3, 5, 6, 7

[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. 1, 2, 3, 4, 5, 6, 7, 8

[3] W. Bao, X. Zhang, L. Chen, L. Ding, and Z. Gao. High-Order Model and Dynamic Filtering for Frame Rate Up-Conversion. *TIP*, 2018. 1, 2

[4] R. Castagno, P. Haavisto, and G. Ramponi. A method for motion adaptive frame rate up-conversion. *TCSVT*, 1996. 1

[5] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*, 1994. 5

[6] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. In *NIPS*, 2016. 2, 4

[7] D. Choi, W. Song, H. Choi, and T. Kim. Map-based motion refinement algorithm for block-based motion-compensated frame interpolation. *TCSVT*, 2016. 2

[8] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *CVPR*, 2015. 2

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 2

[10] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016. 1

[11] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 2

[12] H. Ha, S. Im, J. Park, H.-G. Jeon, and I. So Kweon. High-quality depth from uncalibrated small motion clip. In *CVPR*, 2016. 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4

[14] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In *CVPR*, 2018. 1, 2, 3, 6, 7

[15] K. Karsch, C. Liu, and S. B. Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *TPAMI*, 2014. 2

[16] U. S. Kim and M. H. Sunwoo. New frame rate up-conversion algorithms with low computational complexity. *TCSVT*, 2014. 2

[17] D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. In *ICLR*, 2015. 5

[18] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *CVPR*, 2017. 2

[19] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 2, 4, 5

[20] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI*, 2016. 2

[21] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 1, 2, 6, 7, 8

[22] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *ECCV*, 2016. 2, 6, 7, 8

[23] S. Niklaus and F. Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 1, 2, 3, 4, 6, 7

[24] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *CVPR*, 2017. 1, 2, 4

[25] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 1, 2, 4, 6, 7, 8

[26] M. T. Orchard and G. J. Sullivan. Overlapped block motion compensation: An estimation-theoretic approach. *TIP*, 1994. 2

[27] A. Rajagopalan, S. Chaudhuri, and U. Mudenagudi. Depth estimation and image restoration using defocused stereo pairs. *TPAMI*, 2004. 2

[28] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 6, 7, 8

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

[30] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4

[31] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *CVPR*, 2016. 2

[32] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2006. 2

[33] 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. 2, 5, 6, 7

[34] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 1, 4

[35] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *ECCV*, 2010. 2

[36] C. Wang, L. Zhang, Y. He, and Y.-P. Tan. Frame rate up-conversion using trilateral filtering. *TCSVT*, 2010. 2

[37] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. Towards unified depth and semantic prediction from a single image. In *CVPR*, 2015. 2

[38] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen. Modeling and optimization of high frame rate video transmission over wireless networks. *TWC*, 2016. 1

[39] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *arXiv*, 2017. 1, 2, 5, 6, 7, 8

[40] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 2

[41] C. Zhang, L. Wang, and R. Yang. Semantic segmentation of urban scenes using dense depth maps. In *ECCV*, 2010. 2

[42] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2

[43] Y. Zou, Z. Luo, and J.-B. Huang. DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency. In *ECCV*, 2018. 2, 8