2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Richer Aggregated Features for Optical Flow Estimation with Edge-aware Refinement*

Xianshun Wang[1,2], Dongchen Zhu[1], Jiafei Song[1,3], Yanqing Liu[1],
Jiamao Li[1,2] and Xiaolin Zhang[1,2,3]

Abstract— **Recent CNN-based optical flow approaches have a separated structure of feature extraction and flow estimation. The core task of optical flow is finding the corresponding points while rich representation is just the key part of such matching problems. However, the prior work usually pays more attention to the design of flow decoder than the feature extraction. In this paper, we present a novel optical flow estimation network to enrich the feature representation of each pyramid level, with a hierarchical dilated architecture and a bottom-up aggregation scheme. In addition, inspired by edge guided classical methods, we bring the edge-aware idea into our approach and propose an edge-aware refinement (EAR) subnetwork to handle motion boundaries. Using the same decoding structure as PWC-Net, our network outperforms it by a large margin and leads all its derivatives both on KITTI-2012 and KITTI-2015. Further performance analysis proves the effectiveness of proposed ideas.**

I. INTRODUCTION

Optical flow plays a fundamental role in many computer vision applications, especially on motion related problems, e.g., object tracking, , visual navigation and obstacle avoidance. Variational-based methods [1], [2], [25] optimize the energy function that integrates a data term and a smoothness term. Interpolation-based methods [4], [13], [3] get the dense optical flow by finding a local mapping relationship between frames with sparse one. Recently, deep-learning-based methods gradually take a commanding lead against classical methods on challenging benchmarks. It was noteworthy that the encoder-decoder (a.k.a feature extraction and flow estimation) architecture is widely used in such networks.

Dosovitskiy *et al*. [5] firstly introduce convolution neural network (CNN) based on a naive encoder-decoder architecture for optical flow estimation by proposing two models, i.e., FlowNetC and FlowNetS. Although the performance is still below the classical methods, this work proves the feasibility of such architecture considering the speed and accuracy simultaneously. The successor of FlowNet, i.e., FlowNet2.0 borrows the idea of iterative methods [26], [8] as a iterative decoding structure. It outperforms FlowNet by a large margin and performs on par with the highly-tuned traditional methods. SpyNet [7] further learns from the multi-scale schemes and designs a coarse-to-fine decoding structure. It achieves a good tradeoff between the accuracy and the model size.

Based on the same idea, LiteFlowNet and PWC-Net [9], [10] further combine the feature pyramid network. Instead of warping the RGB images, they directly warp the feature maps and separate the process of feature extraction and flow estimation. More recently, SelFlow [34], IRR [11], MFF [27] and ContinueFlow [28] combine multi-frame and(or) bidirectional information to refine existing PWC-Net-like structures and achieve leading results on public benchmarks. However these improvements have come at a great sacrifice in speed, limiting their application in real robotic applications.

Throughout the development of CNN-based optical flow estimation, all these methods pay great attention to one aspect, namely how to decode the optical flow from the features extracted by plain stacked convolution layers. However, another very important factor is ignored, i.e., how to get better feature maps for these decoders? The feature maps are the cornerstone for matching problems, thus more works need to be done on this aspect. Besides, due to the lack of constraints on the motion boundaries, the flow field estimated by these CNN-based methods appears severely blurred at object boundaries. Therefore, how to refine the predicted results at the edges is also an urgent problem to be solved.

We argue that by enriching the extracted features in the encoder and paying proper attention to the motion boundaries, the PWC-Net-like model can be greatly improved while maintaining the speed advantage. In this work, we propose a novel optical flow estimation network, dubbed RichFlow, to exploit more informative features to improve flow estimation. To begin with, we innovatively combine the contextual information with the popular cascade structure leveraging our designed hierarchical dilated (HD) architecture. Then, we find that previous coarse-to-fine schemes[9], [10] prefer to estimate the optical flow in each layer only with the features extracted from the corresponding (and upper) layer(s). Long *et al*., nevertheless, point out that the features of different depths contain different level semantic and texture information. The deeper the layer is, the more semantic and less visually specific it represents [12]. As thus, estimating the flow field using features from a single (two) pyramid level(s) will lose lots of important information. Aiming at this problem, we design an iterative information aggregation architecture (I^2A) which makes it feasible to

*This work was supported by the National Natural Science Foundation of China (Grant No. 61873255), the Shanghai Municipal Science and Technology Major Project (Grant No.2018SHZDZX01, ZHANGJIANG LAB), the National Natural Science Foundation of China (Grant No. 61671014) and the Youth Innovation Promotion Association, Chinese Academy of Sciences (Grant No. 2018270).

[1]Bionic Vision System Laboratory, State Key Laboratory of Transducer Technology, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences. jmli@mail.sim.ac.cn

[2]University of Chinese Academy of Sciences, Beijing, China.

[3]ShanghaiTech University, Shanghai, China.

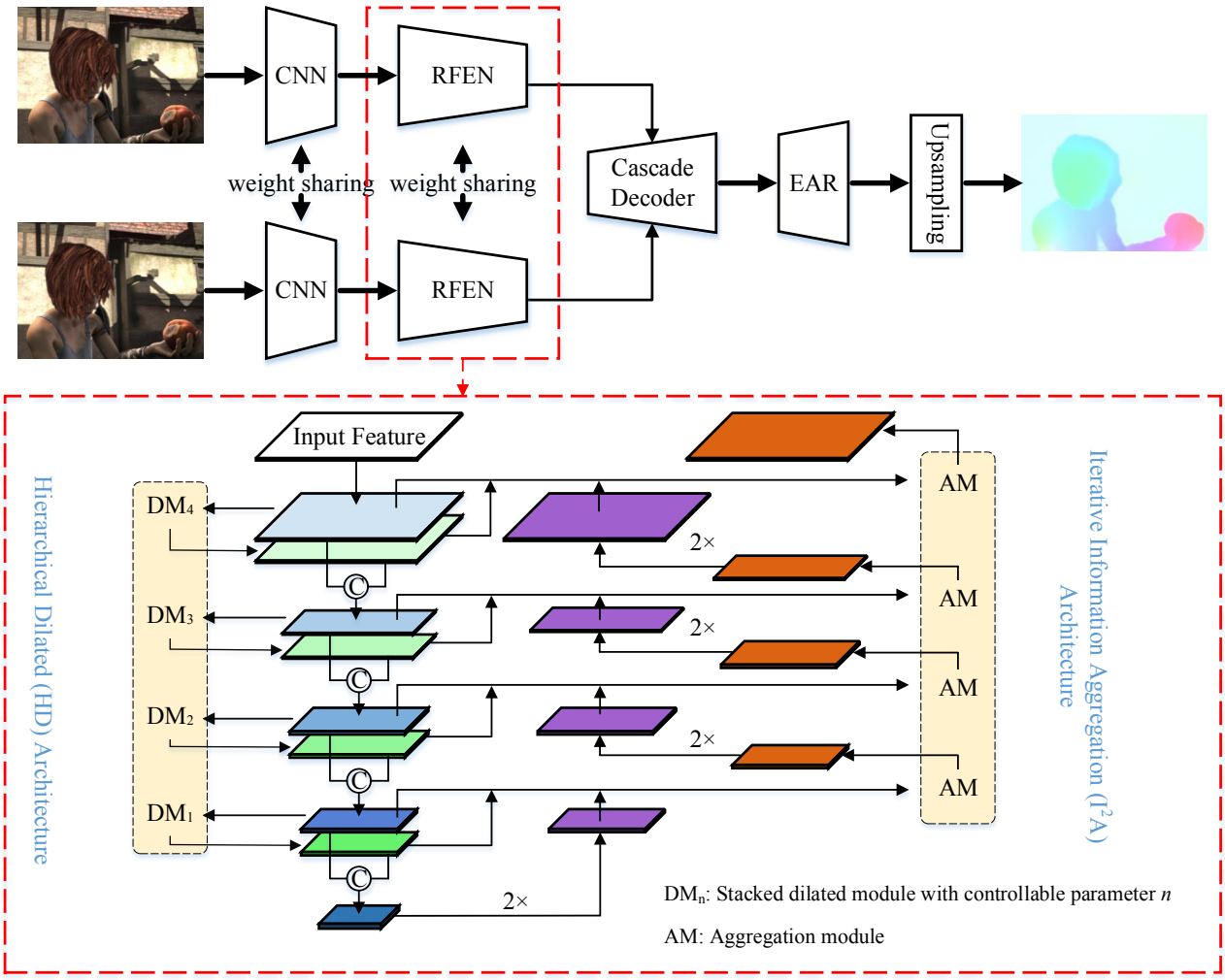

Fig. 1. Top: Pipeline of our proposed method. To begin with, the two consecutive frames are sent to a simple CNN for preprocessing. Secondly, its results will be fed into our proposed richer feature extraction network (RFEN) to enrich feature representations at each pyramid level. The extracted informative features will then be passed into the cascade decoder for flow estimation. Eventually, the finest prediction of the decoder will be refined using our proposed edge-aware refinement (EAR) model. Bottom: Details of our RFEN. The RFEN mainly consists of two components. The first one is the hierarchical dilated (HD) architecture. It contains several DM_n which are responsible for collecting contextual information (green toned) from the stem features (blue toned) at each level. The second one is the iterative information aggregation ($\mathrm{I}^2\mathrm{A}$) architecture. It takes use of a bottom-up scheme to progressively enrich different levels of semantic and visually specific information. Its results are colored in orange.

collect different level semantic and texture information progressively. In addition, we borrow the idea from the classical variational schemes a step further and design an edge-aware refinement (EAR) module to refine the motion boundaries.

To summarize, we make the following contributions:

1) We propose a richer feature extraction network that aggregates different levels of visually specific information, semantic information and contextual information to enrich features at each level.

2) We innovatively bring the edge constraint into CNN-based optical flow estimation method and propose an edge-aware refinement subnetwork. It significantly improves the performance of our model at motion boundaries.

3) We show the efficiency of proposed model by still achieving SOTA accuracy compared with other PWC-NET (Encoder-Decoder architecture) based networks.

II. APPROACH

Figure 1 shows the pipeline of our proposed flow estimation framework. Our network uses a coarse-to-fine cascading structure, and separate the process of feature extraction and flow estimation. It mainly consists of three components, which are richer features extraction network(RFEN), optical flow decoder and edge-aware refinement (EAR). Our work mainly focuses on the first and third parts. In this section, we will emphatically introduce these two parts and briefly summarize the second one.

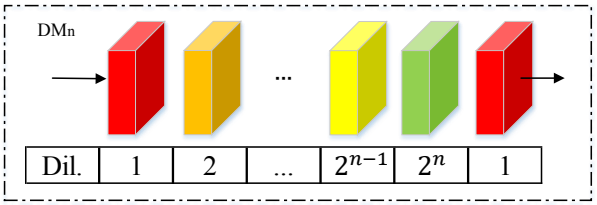

Fig. 2. Stacked dilated module with controllable parameter n.

A. Richer features extraction network

Our richer features extraction network (RFEN) is based on a feature pyramid encoder. We denote the extracted feature at level l as a stem feature, represented as $\mathbf{F}_s^l$. The following hierarchical dilated (HD) architecture and iterative information aggregation ($\mathrm{I}^2\mathrm{A}$) architecture are all extended from these stem features.

Hierarchical dilated architecture. PWC-Net and many other works [16], [17], [14] show the importance of contextual information in the matching problems. Nevertheless, most of these methods only collect contextual information at the finest layer. Considering that the cascading flow inference scheme needs to predict (residual) flow at every pyramid level, we propose a hierarchical dilated architecture with several dilated convolution layers after each stem feature to enrich contextual information.

Instead of using SPP [15] to compress features into different scales, our work adapts stacked dilated convolution layers to collect contextual information, since we argue that the spatial resolution tends to be lost by pooling or downsampling. Inspired by the context module [10], [16], we design a series of DM_n module with controllable parameter n, as shown in **Figure 2**. It consists of a set of 3×3 stacked dilated convolutions with exponentially decreasing dilation factors. By adjusting n, DM_n module can adapt to different size of feature map. Eq. (1) gives a closed expression:

$$\mathbf{F}_c^l = \mathcal{D}^1\left(\mathcal{D}^{2^n}\left(\mathcal{D}^{2^{n-1}}\left(...\mathcal{D}^1\left(\mathbf{F}_s^l\right)\right)\right)...\right) \tag{1}$$

Here $\mathcal{D}^n(*)$ denotes a dilated convolution with a dilation factor of n. $\mathbf{F}_c^l$ represents the extracted contextual feature at level l. The DM_n module is applied hierarchically to every stem feature to aggregate contextual information as much as possible. In order to ensure the integrity of the information disseminated in the pyramid network, the stem feature map and the extracted contextual feature map are concatenated together and passed to the next pyramid level.

Iterative information aggregation. Textual information is the cornerstone of the matching problem. Many researchers have utilized it to design feature descriptors [18], [19], [20]. However, localized textual information often fails or even lies. A famous example is the aperture problem. Instead, the human vision system can tackle such problems easily most of the time, since humans not only pay attention to the texture of each patch but also which object it belongs to.

The design of our aggregation architecture is motivated by the above mechanism. The ambiguous areas require more information from different levels to identify their matching targets. Therefore, the proposed architecture have the bottom-up iterative aggregation settings, as shown in the bottom of **Figure 1**. At the lth level, the preceding aggregated feature map $\mathbf{F}_a^{l+1}$ is upsampled leveraging a 3×3 dilated convolution with dilation factor equal to 2. To bring different information together, the contextual map, the stem feature map and the upsampled feature map are then fed into the aggregation module. Note that, all the upsampled features are coming from the preceding aggregated feature map instead of the stem feature map except for the bottom layer. With the alternate use of upsampling and aggregation, the aggregated feature maps enrich more and more information, including semantic information and visually specific information.

To integrate information and reduce the dimensions of the aggregated features, in the aggregation module (AM), we firstly concatenate the three input feature maps and then feed them into an 1×1 convolution. Besides, a residual connection is added between the stem and aggregated features. The whole module can be defined as follow:

$$\mathbf{F}_a^l = \mathcal{G}\left(\mathcal{C}\left(\mathbf{F}_s^l, \mathbf{F}_c^l, \mathbf{F}_u^l\right), \{\mathbf{W}_i\}\right) + \mathcal{G}\left(\mathbf{F}_s^l, \{\mathbf{W}_j\}\right) \tag{2}$$

Here $\mathbf{F}_a^l$ is aggregated feature at level l. $\mathbf{F}_u^l$ is the upsampled aggregated feature from level $l+1$. The functions $\mathcal{C}(*)$ and $\mathcal{G}(*, \mathbf{W}_*)$ represent the concatenate operation and the 1×1 convolution mapping with parameter $\mathbf{W}_*$ respectively.

B. Cascade decoder

PWC-Net has a very powerful decoding structure. It has become the baseline of many recent works [27], [11], [28], [34]. Therefore our work uses the same decoding structure as PWC-Net. The main components are described below.

Cost volume layer. The cost volume (CV) has proven to be very effective in matching problems [29], [30], [5]. Typical choices for CV can be by simple feature concatenation [14], [31], [5] or by calculation of metrics [10], [9], [7]. Here we follow the same idea as PWC-Net and define the matching cost as the correlation between two consecutive feature maps:

$$CV(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N}\left\langle \mathbf{F}_1(\mathbf{x}_1), \mathbf{F}_2(\mathbf{x}_2) \right\rangle \tag{3}$$

N represents the dimension of the column vector $\mathbf{F}$. $\|\mathbf{x}_1 - \mathbf{x}_2\|_\infty \leq 4$ is totally enough for our settings.

Cascade and Warping. Unlike the stereo matching problem, the search space of optical flow is two-dimensional, which results in a huge candidate sets for each reference pixel. The popular cascade inference scheme comparing with warping technique is used to cope with this problem. Before estimating the optical flow at level l, the feature map of second image is wrapped using the upsampled flow from $l+1$th level:

$$\mathbf{F}_2^w(\mathbf{x}) = \mathbf{F}_2(\mathbf{x} + \hat{\mathbf{w}}_{l+1}^u) \tag{4}$$

Essentially, we just need to estimate the flow filed between $\mathbf{F}_1$ and the wrapped feature map $\mathbf{F}_2^w$, i.e, the residual flow

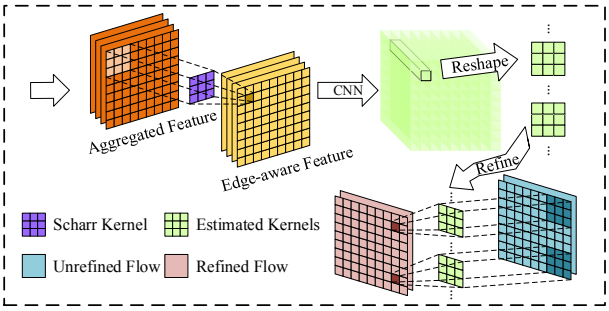

Fig. 3. Main process of EAR module. There are two different scharr kernels on two directions, only one of them is illustrated here for simplicity.

of previous predicted result. This cascade scheme greatly reduces the search space of each reference pixel.

Optical flow estimator. The flow estimator consists of several stacked convolution layers. Its input consists of four parts, which are the aggregation feature of first image, the cost volume, the upsampled aggregation feature and flow of previous layer. The flow estimator at each level is independent. Additionally, a dense connection and a residual connection are used to enhance the estimator at the top level.

III. EDGE-AWARE REFINEMENT

A crucial part of the CNN-based methods is the estimation of motion boundaries, because they correspond to the discontinues of the optical flow filed. However, most of deep learning approaches do not pay sufficient attention to it. Inspired by these traditional methods [4], [2], [25], the edge map contains lots of information related to motion boundaries. Here we utilize edge information to describe the motion details and propose an edge-aware refinement subnetwork, i.e., EAR.

In order to get better performance on boundaries, we extract our desired edges leveraging a scharr-convolution on the aggregated feature map instead of the image. The parameters of scharr-convolution kernel are fixed and initialized from scharr filter [32]. This convolution is represented as follow:

$$g(c) = s * f(c) \tag{5}$$

Here $f(c)$ denotes the cth channel of $\mathbf{F}$ and $g(c)$ represents its corresponding output. The convolution results of all channels make up the edge-aware feature $\mathbf{G}$. s is the scharr kernel. This operation can be easily implemented using a group convolution and its parameters remain fixed throughout the training process.

Then the estimated $\mathbf{G}$ is passed into a multi-layer convolution structure to estimate the kernel parameters for each pixel location. These parameters will eventually be used for the final refinement. In the final process, each estimated kernel is convolved with the corresponding flow patch and get refined flow value. **Figure 3** shows above process in detail.

IV. EXPERIMENTS

In this section we evaluate our RichFlow with several state-of-the-art methods on public benchmarks including Sintel [21], KITTI-2012 [22] and KITTI-2015 [23]. Sintel is a synthetic benchmark derived from an open source 3D animated short film. KITTI-2012 and KITTI-2015 are created from real driving scenes. Compared to Sintel, they are closer to the real application environment.

A. Main results

Pre-training. In this stage, we will only train the parameters before EAR module. It is pretrained on the two popular datasets, i.e., FlyingChairs [5] and FlyingThings [24]. We first train our model on FlyingChairs using the S_{long} training schedule as FlowNet 2.0 [6], the multiscale L2 loss function is applied here:

$$L(\mathbf{W}) = \sum_{l=l_0}^{L} \beta_l \sum_{\mathbf{x}} \left| \hat{\mathbf{w}}^l(\mathbf{x}) - \mathbf{w}_g^l(\mathbf{x}) \right|_2 \tag{6}$$

Here $\hat{\mathbf{w}}^l$ and $\mathbf{w}_g^l$ represent the estimated and ground truth flow field at level l respectively. $\mathbf{W}$ denotes all the learnable parameters of our model. We randomly crop the input image pairs into 448×320 and set the batch size to 8.

The trained model is then fine-tuned on FlyingThings using S_{fine} training schedule introduced by Ilg *et al.* [6]. In this phase, the robust loss function proposed in PWC-Net is applied:

$$L(\mathbf{W}) = \sum_{l=l_0}^{L} \beta_l \sum_{\mathbf{x}} (\left| \hat{\mathbf{w}}^l(\mathbf{x}) - \mathbf{w}_g^l(\mathbf{x}) \right| + p)^q \tag{7}$$

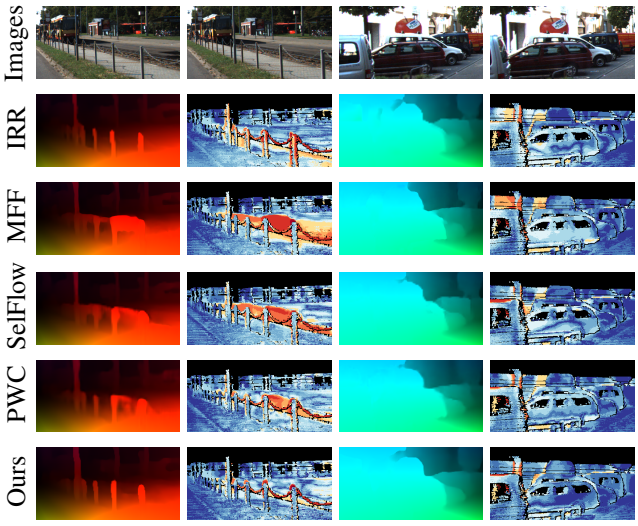

Fig. 4. Comparison results on the test set of KITTI. Correct estimates colored in blue and wrong estimates colored in red. The 1st row shows the input images. Except for the 1st row, the 1st and 3rd column are the predicted flow of each method, the 2nd and 4th column are the corresponding error maps.

KITTI. We fine-tune our model on the mixture of KITTI-2012 and KITTI-2015, since they have an approximate

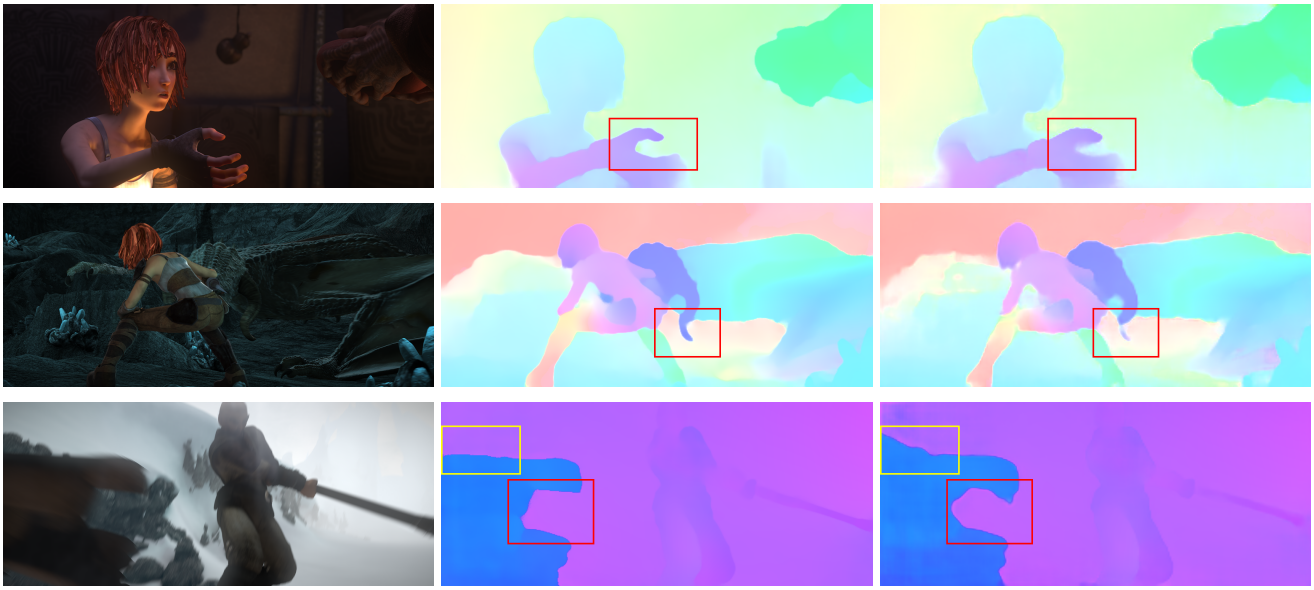

Fig. 5. Comparison results on test set of Sintel Final. Notice how the motion boundaries (1st and 3rd row), fine structures (2nd row) and motion blur (3rd row) are handled properly by our model.

Methods	KITTI-2012				KITTI-2015		
	Out-Noc	Out-All	Avg-Noc	Avg-All	Fl-fg	Fl-bg	Fl-all
MFF [27]‡	4.19%	7.87%	0.9	1.7	7.25%	7.15%	7.17%
SelFlow [34]‡	3.32%	6.19%	0.9	1.5	7.61%	12.48%	8.42%
ContinueFlow [28]‡	-	-	-	-	8.54%	17.48%	10.03%
IRR-PWC [11]‡	3.21%	6.70%	0.9	1.6	7.52%	7.68%	7.65%
PWC-Net+ [10]	3.36%	6.72%	0.8	1.4	7.88%	7.69%	7.72%
RichFlow	**2.58%**	**5.62%**	**0.7**	**1.3**	**6.98%**	**6.62%**	**6.68%**

TABLE I

COMPARISON RESULTS ON KITTI-2012 AND KITTI-2015. OUR MODEL LEADS PWC-NET AND ALL ITS DERIVATIVES COMPREHENSIVELY. ‡REPRESENTS MULTI-FRAME AND/OR BIDIRECTIONAL SOLUTIONS.

distribution. The input image pairs are cropped into 896×320 patches randomly. We use a similar data augmentation scheme as PWC-Net. The batch size is 4. To prevent overfitting, we continue to use the robust loss function in Eq. (7) and set $p = 0.01$, $q = 0.2$. A stage-wise training scheme is used to accelerate convergence and reduce the memory usage. To begin with, we remove the EAR module and directly fine-tune the rest of our network by supervising the results of the cascade decoder at each level. At this stage, these parameters is trained for 450k iterations. Then all there parameters remain fixed, we train the EAR module separately for another 300k iterations.

We first show qualitative comparisons in **Figure 4**. Then we refer to **Table I** for quantitative results. Our model outperforms PWC-Net by a large margin and leads all its derivatives [27], [11], [28], [34] including multi-frame and bidirectional solutions on KITTI-2012 and KITTI-2015. Among these methods, RichFlow is the only one with F1-All below 7% in KITTI-2015 and Out-Noc below 3% in KITTI-2012. Compared with PWC-Net, we reduce Out-All by 16.4%, Out-Noc by 23.2% relatively on KITTI-2012 and F1-All by 13.5%, F1-fg by 11.4%, F1-bg by 13.9% relatively on KITTI-2015. Note that, the larger improvement on non-occluded regions proves the effectiveness of our RFEN.

MPI-Sintel. When fine-tuning on Sintel, the batch size is set to 4. We still use the robust loss function in Eq. (7) with $p = 0.01$ and $q = 0.04$. The input image pairs are cropped into 768×384 during training. We use a similar stage-wise fine-tune schedule as KITTI but with 300k iterations at first stage. Since Sintel contains two different subsets for training and testing. We adapt two different fine-tune schemes as previous methods. The first one is fine-tuned on the clean and final passes together throughout the fine-tuning process. The second one uses only the final pass for the last 150k iterations to pay more attention to the final subset. We add different suffixes to distinguish between these two models (-ft and -ft-fnl).

Figure 5 shows some examples of qualitative comparisons. The comparison results are shown in **Table II**. Com-

Methods	EPE All		EPE Matched		Runtime
	Clean	Final	Clean	Final	
MFF[27]‡	**3.42**	4.57	**1.38**	2.22	0.27s
SelFlow[34]‡	3.75	**4.26**	1.45	**2.04**	0.18s
ContinueFlow[28]‡	3.34	4.53	1.75	2.72	-
IRR-PWC[11]‡	3.84	4.58	1.47	2.15	0.33s
PWC-Net+[36]	3.45	4.60	1.41	2.25	0.06s
PWC-Net-ft[10]	3.86	5.13	-	-	0.06s
PWC-Net-fnl[10]	4.39	5.04	1.72	2.45	0.06s
RichFlow-ft	**3.54**	5.08	**1.34**	2.38	0.13s
RichFlow-ft-fnl	4.32	**4.63**	1.61	**2.15**	0.13s

TABLE II

AVERAGE END-POINT ERROR (EPE) ON TWO SUBSETS OF SINTEL TEST. ‡REPRESENTS MULTI-FRAME AND/OR BIDIRECTIONAL SOLUTIONS. THE RUNNING TIMES ARE ACQUIRED ON THE SAME COMPUTER WITH AN NVIDIA MAXWELL TITAN X GPU, IO TIME IS EXCLUDED.

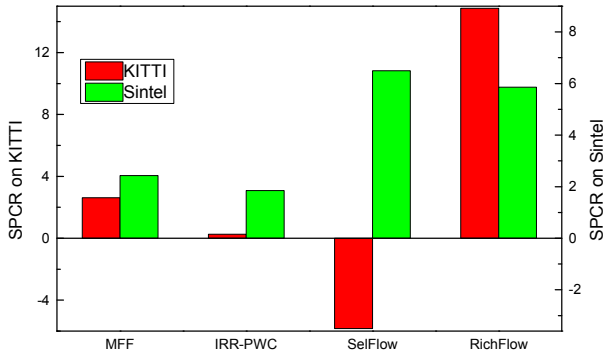

Fig. 6. Speed-to-performance conversion ratio on KITTI and Sintel. Running times are acquired on the same computer with an NVIDIA Maxwell Titan X GPU.

pared with our baseline, i.e., PWC-Net [1], we reduce EPE relatively by 8.3% (Clean) and 8.1% (Final) with slightly increased running time. Due to the larger occlusion in Sintel, our approach is slightly worse than these multi-frame and/or bidirectional schemes. Nevertheless, compared to these methods, our method achieves the lowest EPE on matched regions of clean pass and second lowest EPE on matched regions of final pass which proves our advantage. What's more, our two-frame solution has a clear speed advantage.

B. Performance analysis

Gain with less speed sacrifice. We define a speed-to-performance conversion ratio leveraging PWC-Net as the baseline to evaluate the efficiency of each method:

$$SPCR(M) = \frac{A_{PWC} - A_M}{T_{PWC} - T_M} \tag{8}$$

Here A_* represents the accuracy metric and T_* is the corresponding running time. **Figure 6** counts the SPCR of some state-of-the-arts on KITTI and Sintel. Note that, our EAR and RFEN accounted for 33.8% and 33.1% of the total runtime respectively. And the EAR is more focused on boundaries, so it may weaken our advantage to some extent. Even so, our model can still achieve larger improvement at a smaller speed sacrifice.

[1]PWC-Net+ is proposed in TPAMI'19, its model on Sintel leverage a mixture dataset to achieve a better result. However, this technique not always work which is invalid on KITTI. Hence we skip the comparison with this result. The result of PWC-Net+ on KITTI have the same architecture and training schedule as PWC-Net but with an I/O bug fixed, thus we compare with PWC-Net+ in Table I but cited as PWC-Net.

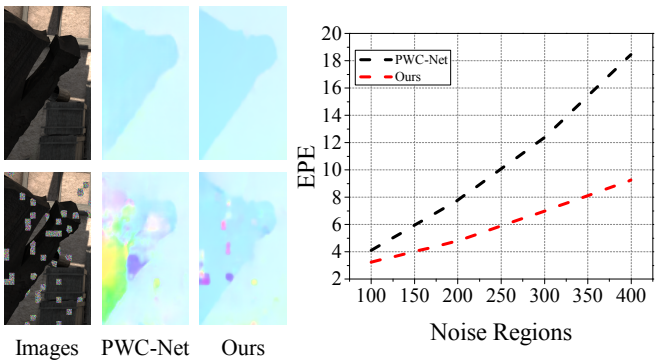

Fig. 7. Comparison between FEN and RFEN with different levels of input noise regions. Notice how our methods is more tolerant compared with PWC-Net when local texture is destroyed.

Comparison of feature extraction network. In order to evaluate the performance of our feature extraction network (RFEN), we compare RFEN with the feature extraction network (hereinafter referred as FEN) of PWC-Net. We choose PWC-Net because it's a widely used baseline in recent works and has a separated structure of feature extraction and flow estimation. We combine RFEN and FEN with the same optical flow decoder respectively and then test the performance of these two combinations. For a fair comparison, the EAR is removed and we use the same training schedule, data augmentation and parameter initialization methods during experiments.

We train the two combinations on FlyingChairs [5] and FlyingThings [24] in turn and evaluate the models on Sintel and KITTI respectively at each training stage. All the comparison results are shown in **Table III**. We can see that our RFEN outperforms FEN in all cases which proves that our RFEN has better adaptability in each dataset.

In order to demonstrate the importance of information aggregation, we have evaluated the performance of above two combinations with different levels of noised input. We split the input image into around 2000 superpixels [33] and randomly perturb several regions with random noise. Then we evaluate the performance of two combinations on Sintel using same noised image pairs as their input. **Figure 7** shows the clear robustness of our RFEN over FEN. Since our RFEN aggregates both semantic and visually specific information, it is less sensitive to the loss of local texture information.

Improvement on motion boundaries by EAR. In this part, we analyze the impact on motion boundaries of our EAR module in detail. We apply a similar two-stage training scheme mentioned in section IV-A on FlyingChairs. Then we evaluate the performance changes after EAR refinement on its test subset. Since our EAR adopts similar vector-

Training Data	FEN	Sintel Clean AEPE	Sintel Final AEPE	KITTI-2012 AEPE	KITTI-2012 F1-all	KITTI-2015 AEPE	KITTI-2015 F1-all
Chairs	PWC-Net	3.33	4.59	5.14	28.67%	13.20	41.79%
	Ours	**3.17**	**4.50**	**4.21**	**22.47%**	**12.38**	**38.01%**
Chairs + Things	PWC-Net	2.55	3.93	4.14	21.38%	10.35	33.67%
	Ours	**2.30**	**3.78**	**3.27**	**14.83%**	**9.46**	**27.21%**

TABLE III
PERFORMANCE ANALYSIS OF OUR RFEN COMPARED WITH PWC-NET.

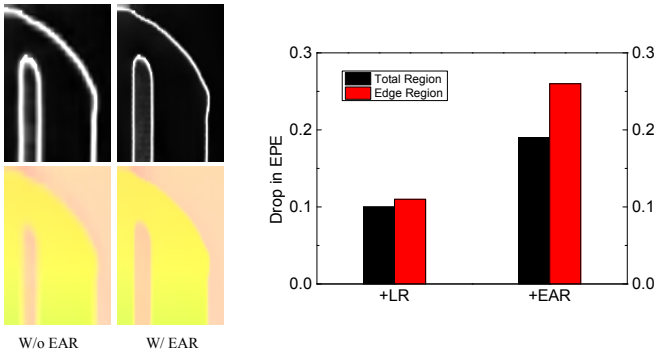

Fig. 8. Left: Predicted results and their corresponding error maps. White corresponds to erroneous prediction. Right: Performance changes after EAR and LR refinement.

RFEN	EAR	KITTI-2012 Train	KITTI-2012 Test	KITTI-2015 Train	KITTI-2015 Test
		2.57%	3.36%	5.86%	7.72%
	✓	2.43%	3.27%	5.53%	7.28%
✓		1.90%	2.62%	4.94%	6.98%
✓	✓	**1.86%**	**2.58%**	**4.45%**	**6.68%**

TABLE IV
ABLATION STUDY ON KITTI-2012 AND KITTI-2015. WE USE THE SAME EVALUATION INDICATORS AS THE BENCHMARKS. OUT-NOC IS USED FOR KITTI-2012 AND F1-ALL FOR KITTI-2015.

Stem	Contextual	Upsampled	Sintel Clean	Sintel Final
✓			3.42	4.64
✓	✓		3.33	4.55
✓	✓	✓	**3.26**	**4.41**

TABLE V
ABLATION STUDY ON THE AGGREGATION MODULE.

value refinement scheme as the LiteFlowNet regularization model (LR), we also reported the performance changes after LR is applied. In order to analyze the performance change at motion boundaries, we make the same assumption of EpicFlow [4] to approximate the motion boundaries as image edges [35]. Both the overall performance improvement and the improvement at edge regions are illustrated on the right of **Figure 8**. It shows that our EAR can significantly improve the performance of optical flow estimation. Compared with LR, EAR not only performs better overall, but also more sensitive at edge regions. An example of visible improvement after EAR refinement is shown on the left of **Figure 8**.

Ablation on RFEN and EAR. In order to prove these two components do not suppress each other, we design an ablation study on the KITTI benchmarks. All the models are pre-trained on FlyingChairs and FlyingThings and fine-tuned on the mixture dataset of KITTI-2012 and KITTI-2015 using the same training schedule and data augmentation schemes. All the models are evaluated both on the training and test sets which eliminates the effects of overfitting. The results in **Table IV** show whether the RFEN is applied, our EAR can bring significantly improvements, and vice versa.

Ablation on aggregation module. We aggregate three different features (stem feature, contextual feature and upsampled feature) in the AM. To analyze the impact of these features, we evaluate the performance of different variants with some of them removed. All of these variants are trained on FlyingChairs with S_{short} schedule [6], and then the EPE on Sintel is reported in **Table V**. We remove the EAR subnetwork in all models to rule out its impact. Adding the proposed contextual feature, upsampled feature in turn improves the performance by 0.09, 0.16 in clean pass and 0.09, 0.23 in final pass. It demonstrates the effectiveness of our introduced features.

V. CONCLUSIONS

In this paper, we propose a richer feature extraction network which uses a hierarchical dilated architecture together with a bottom-up aggregation scheme to enrich the feature representation at each pyramid level. Inspired by the edge-aware idea, an EAR subnetwork is designed to pay attention to the motion details. Experiments show that our feature extraction network has a better performance and is less sensitive to the loss of local texture information. Our EAR can bring significantly improvement at motion boundaries while maintaining the performance at other regions. What's more, our work performs better with less speed sacrifice, which is more important in the deployment of robots.

This work confirms the benefit of richer features at non-occluded regions. We are also interested in improving the performance at occluded areas while keeping faster running speed.

ACKNOWLEDGMENT

Project supported by Shanghai Municipal Science and Technology Major Project (Grant No. 2018SHZDZX01,

ZHANGJIANG LAB). We would like to thank Minghong Chen for helpful suggestions.

REFERENCES

[1] B. K. P. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, 1981, p. 185–203.

[2] H. Zimmer, A. Bruhn, and J. Weickert, "Optic flow in harmony," *International Journal of Computer Vision*, vol. 93, no. 3, pp. 368–388, 2011.

[3] Y. Hu, Y. Li, and R. Song, "Robust interpolation of correspondences for large displacement optical flow," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Computer Vision and Pattern Recognition*, 2015, pp. 1164–1172.

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[6] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[7] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[8] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1385–1392.

[9] T.-W. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 8981–8989.

[10] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[11] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5754–5763.

[12] J. L. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" in *Advances in Neural Information Processing Systems*, 2014, pp. 1601–1609.

[13] X. Wang, D. Zhu, Y. Liu, X. Ye, J. Li, and X. Zhang, "Semflow: Semantic-driven interpolation for large displacement optical flow," *IEEE Access*, vol. 7, pp. 51 589–51 597, 2019.

[14] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[16] X. Du, M. El-Khamy, and J. Lee, "Amnet: Deep atrous multiscale stereo disparity estimation networks," *arXiv preprint arXiv:1904.09099*, 2019.

[17] X. Song, X. Zhao, H. Hu, and L. Fang, "Edgestereo: A context integrated residual pyramid network for stereo matching," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 20–35.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: speeded up robust features," *european conference on computer vision*, vol. 3951, pp. 404–417, 2006.

[20] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[21] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European conference on computer vision*. Springer, 2012, pp. 611–625.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[23] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[24] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[25] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*. Springer, 2004, pp. 25–36.

[26] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2010.

[27] Z. Ren, O. Gallo, D. Sun, M.-H. Yang, E. Sudderth, and J. Kautz, "A fusion approach for multi-frame optical flow estimation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 2077–2086.

[28] M. Neoral, J. Šochman, and J. Matas, "Continual occlusion and optical flow estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 159–174.

[29] J. Zbontar, Y. LeCun *et al.*, "Stereo matching by training a convolutional neural network to compare image patches." *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

[30] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[31] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," 2017.

[32] B. Jähne, H. Haussecker, and P. Geissler, *Handbook of computer vision and applications*. Citeseer, 1999, vol. 2.

[33] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[34] P. Liu, M. R. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," in *CVPR*, 2019.

[35] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1841–1848.

[36] D. Sun, X. Yang, M. Liu, and J. Kautz, "Models matter, so does training: An empirical study of cnns for optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.