MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask

Shengyu Zhao* Yilun Sheng* Yue Dong Eric I-Chao Chang Yan Xu†

Abstract

Feature warping is a core technique in optical flow estimation; however, the ambiguity caused by occluded areas during warping is a major problem that remains unsolved. In this paper, we propose an asymmetric occlusion-aware feature matching module, which can learn a rough occlusion mask that filters useless (occluded) areas immediately after feature warping without any explicit supervision. The proposed module can be easily integrated into end-to-end network architectures and enjoys performance gains while introducing negligible computational cost. The learned occlusion mask can be further fed into a subsequent network cascade with dual feature pyramids with which we achieve state-of-the-art performance. At the time of submission, our method, called MaskFlownet, surpasses all published optical flow methods on the MPI Sintel, KITTI 2012 and 2015 benchmarks. Code is available at https://github.com/microsoft/MaskFlownet.

1. Introduction

Optical flow estimation is a core problem in computer vision and a fundamental building block in many real-world applications [2, 22, 29]. Recent development towards fast, accurate optical flow estimation has witnessed great progress of learning-based methods using a principled network design — feature pyramid, warping, and cost volume — proposed by PWC-Net [32] and LiteFlowNet [11], and used in many follow-up works [12, 13, 18, 24, 31]. Feature warping effectively resolves the long-range matching problem between the extracted feature maps for the subsequent cost volume computation. However, we observe that a major problem of the warping operation is that it introduces unreliable information in the presence of occlusions. As shown in Fig. 1, the warped image as well as the warped feature map can be even “doubled” at the occluded areas (also called the ghosting effect). It remains unclear that

![Figure 1. Motivation of the learnable occlusion mask. (a) Image warping induces ambiguity in the occluded areas (see the doubled hands). (b) The same problem exists in the feature warping process. Such areas can be masked without any explicit supervision.](https://github.com/microsoft/MaskFlownet)
whether the source image would be mismatched to such areas yet raises a natural question: are they really distinguishable without being supervised of occlusions?

We answer this question positively by showing that the network can indeed learn to mask such areas without any explicit supervision. Rather than enforcing the network to distinguish useful parts from those confusing information, we propose to apply a multiplicative learnable occlusion mask immediately on the warped features (see Fig. 1). We can see that there is a clear distinction between the black and white areas in the learned occlusion mask, indicating that there exists solid gradient propagation. The masked image (features) has much cleaner semantics, which could potentially facilitate the subsequent cost volume processing.

The masking process interprets how those areas can be distinguished in a clear way. While previous works commonly believe that the feature symmetricity is crucial for the cost volume processing, we in contrast demonstrate that the network further benefits from a simple asymmetric design despite the explicit masking. The combined asymmetric occlusion-aware feature matching module (AsymOFMM) can be easily integrated into end-to-end network architectures and achieves significant performance gains.

We demonstrate how the proposed AsymOFMM would contribute to the overall performance using a two-stage architecture named MaskFlownet (see Fig. 2). MaskFlownet is trained on standard optical flow datasets (not using the occlusion ground truth), and predicts the optical flow together with a rough occlusion mask in a single forward pass. At the time of submission, MaskFlownet surpasses all published optical flow methods on the MPI Sintel (on both clean and final pass), KITTI 2012 and 2015 benchmarks while using only two-frame inputs with no additional assumption.

2. Related Work

Optical Flow Estimation. Conventional approaches formulate optical flow estimation as an energy minimization problem based on brightness constancy and spatial smoothness since [10] with many follow-up improvements [3, 21, 36]. Estimating optical flow in a coarse-to-fine manner achieves better performance since it better solves large displacements [4, 37]. Later works propose to use CNN extractors for feature matching [1, 39]. However, their high accuracy is at the cost of huge computation, rendering those kinds of methods impractical in real-time settings.

An important breakthrough of deep learning techniques in optical flow estimation is made by FlowNet [8], which proposes to train end-to-end CNNs on a synthetic dataset and first achieves a promising performance. Although they only investigate two types of simple CNNs (FlowNetS and FlowNetC), the correlation layer in FlowNetC turns out to be a key component in the modern architectures. Flownet2 [14] explores a better training schedule and makes significant improvements by stacking multiple CNNs which are stage-wise trained after fixing the previous ones.

SpyNet [27] explores a light-weight network architecture using feature pyramid and warping, but the learned features are not correlated so it can only achieve a comparable performance to FlowNetS. PWC-Net [32] and Lite-FlowNet [11] present a compact design using feature pyramid, warping, and cost volume, and achieve remarkable performance over most conventional methods while preserving high efficiency, and they further make some slight improvements in the later versions [12, 31]. VCN [40] recently exploits the high-dimensional invariance during cost volume processing and achieves state-of-the-art performance.
Note that this paper focuses on the feature matching process prior to the correlation layer, which is independent of the improvement made by VCN. To our knowledge, none of those works realizes that an asymmetric design of the feature matching process can achieve better performance.

**Occlusions and Optical Flow.** Occlusions and optical flow are closely related. Optical flow methods such as FlowNet and FlowNet2 can be easily extended to joint optical flow and occlusion estimation with slight modifications as proposed in [15, 19]. IRR-PWC[13] presents an iterative residual refinement approach with joint occlusion estimation using bilateral refinement. However, all those methods can only explicitly learn from the ground-truth occlusions, which require additional efforts on the training labels that limit their applicability [13, 19].

Unsupervised or self-supervised learning of optical flow is another promising direction. Handling occlusions is clearly a vital aspect in such setting, since the brightness error does not make sense at occluded pixels. Some initial works show the feasibility of the unsupervised learning guided by the photometric loss [17, 28]. Later works realize that the occluded pixels should be excluded from the loss computation [20, 35]. Occlusions also facilitate multi-frame estimation [16, 18, 24]. However, the only occlusion estimation approach used by those kinds of methods is the forward-backward consistency [33] that requires bidirectional flow estimation, which limits its flexibility and could lead to noisy predictions. We would like to remark that our promising approach can jointly estimate occlusions without any explicit supervision in a single forward pass, which we expect can be helpful to future unsupervised or self-supervised learning methods.

**Occlusion-Aware Techniques in Other Applications.** Occlusions commonly exist in object detection and might affect the performance of standard approaches in some scenarios, e.g., crowded pedestrian detection [41]. Recent works propose to explicitly learn a spatial attention mask that highlights the foreground area for occluded pedestrian detection [26, 42], which requires additional supervising information. Occlusions in face recognition is also a major problem which can be addressed by the guided mask learning [30]. Our work is also related to the attention mechanism in computer vision [34], which addresses a different problem of capturing pixel-wise long-range dependencies. None of those works realizes a global attention mask can be learned to filter occluded areas with no explicit supervision.

3. Occlusion-Aware Feature Matching

Given an image pair $I_1$ (the source image) and $I_2$ (the target image), the task is to estimate the flow displacement $\phi$, representing the correspondence between $I_1(x)$ and $I_2(x + \phi(x))$. Image warping is the process of constructing

\[ (\phi \circ I_2)(x) \triangleq I_2(x + \phi(x)) \]  

(1)

using the estimated displacement $\phi$, and ideally we have $I_1(x) \approx (\phi \circ I_2)(x)$ at all non-occluded pixels $x$. This operation is differentiable w.r.t. both inputs because non-integral points can be dealt with bilinear interpolation. Feature warping is similarly defined by replacing $I_2$ in Eq. (1) with the extracted feature map.

Feature warping followed by a correlation layer is the common practice to compute the cost volume for non-local feature matching in recent works [11, 12, 13, 18, 24, 31, 32]. The feature extractor of an image is a pyramid of convolutional layers, which are proposed to be symmetric for $I_1$ and $I_2$ in the sense that they share the same convolution kernels. The cost volume at pyramid level $l$ can be formulated as

\[ c(\mathcal{F}^l(I_1), \phi \circ \mathcal{F}^l(I_2)), \]  

(2)

where $\mathcal{F}^l$ denotes the shared feature extractor for level $l$, and $\phi$ denotes the flow displacement predicted by the previous level. $c$ represents the correlation layer that computes the element-wise dot product between the two feature maps within a maximum displacement. Fig. 4(a) illustrates this process, which we call a feature matching module (FMM).

We observe that a major consequence caused by the warping operation is the ambiguity in the presence of occlusions. Fig. 3 illustrates a simplified example, where the foreground object has a large movement while the background stays still. During warping, a copy of the foreground object still stays at the occluded area after warping.

![Figure 3. A simplified case of occlusions.](image-url)
it provides extra information at the masked areas.

OFMM learns to mask the occluded areas simply because it realizes they are useless comparing to the trade-off term, even if there is no explicit supervision to the occlusions at all. Although the OFMM itself might not contribute significantly to the performance, it can learn a rough occlusion mask at negligible cost, which can be further fed into the new occlusion-aware feature pyramid (see §4).

Asymmetric Occlusion-Aware Feature Matching Module (AsymOFMM). We suggest that an asymmetric design of the feature extraction layers consistently gains the performance. Intuitively, the warping operation induces ambiguity to the occluded areas and breaks the symmetry of the feature matching process, so an asymmetric design might be helpful to conquer this divergence.

Based on the OFMM, we introduce an extra convolutional layer prior to the warping operation, which is asymmetrically drawn on the feature extraction process of only \( I_2 \). In practice, we replace the extra convolutional layer and the warping operation by a deformable convolution. In the general setting of deformable convolutional networks [7], different locations in each of the convolution kernels are associated with different offsets, but here we introduce a specialized setting where each convolution kernel is warped in parallel according to the corresponding flow displacement at center. The deformable convolution slightly differs from the initial design since it reverts the order of convolution and (bilinear) interpolation, which is proved to be better in the experiments. As illustrated in Fig. 4(c), the resulting cost volume can be formulated as

\[
c(\mathcal{F}(I_1), D^l(\mathcal{F}(I_2), \phi) \otimes \theta \oplus \mu),
\]

where \( D^l(\cdot, \phi) \) denotes the deformable convolution layer at level \( l \) using the displacement \( \phi \).

4. MaskFlownet

The overall architecture of the proposed MaskFlownet is illustrated in Fig. 2. MaskFlownet consists of two cascaded subnetworks. The first stage, named MaskFlownet-S, generally inherits the network architecture from PWC-Net [32], but replaces the feature matching modules (FMMs) by the proposed AsymOFMMs.

MaskFlownet-S first generates a 6-level shared feature pyramid as PWC-Net, and then makes predictions from level 6 to 2 in a coarse-to-fine manner. The final predictions at level 2 are 4-time upsampled to level 0. Fig. 5 illustrates the network details at each level \( l \) (modifications needed at level 6 and level 2). The previous level is responsible for providing \( \phi^{l+1}, \theta^{l+1}, \) and \( \mu^{l+1} \), which are then upsampled and fed into the AsymOFMM. \( \phi^{l+1}, \theta^{l+1} \) are upsampled using bilinear interpolation; \( \mu^{l+1} \) is upsampled.
using a deconvolutional layer (of 16 channels), followed by a convolutional layer to match the channels of the pyramid feature extractor. All convolutional layers have a kernel size of 3; all deconvolutional layers have a kernel size of 4. The feature matching module at level 6 is simply a correlation layer since there is no initial flow for warping. The maximum displacement of the correlation layers is kept to be 4. The resulting cost volume is concatenated with the upsampling displacement, and the upsampling features, and then passed through 5 densely connected convolutional layers as PWC-Net. The final layer predicts the flow displacement \( \phi^l \) with residual link from the previous flow estimation, the occlusion mask \( \theta^l \) after the sigmoid activation, and the features \( \mu^l \) passing to the next level. Level 2 only predicts the flow displacement, ended with the context network (as PWC-Net) that produces the final flow prediction.

Occlusion-Aware Feature Pyramid. The learned occlusion mask, concatenated with the warped image, is fed into the occlusion-aware feature pyramid for the subsequent flow refinement. The occlusion mask is subtracted by 0.5 before concatenation; a zero mask is concatenated with \( I_1 \) for symmetricity. The occlusion-aware pyramid extracts features for the concatenated images (both with 4 channels) with shared convolutional layers as usual. We suggest that the occlusion mask facilitates the feature representation of the warped image, given the vast existence of occluded areas during warping.

Cascaded Flow Inference with Dual Pyramids. We propose to cascade the network by utilizing dual feature pyramids. The occlusion-aware feature pyramid provides abundant information about the warped image from the previous flow estimation for refinement, but it cannot feedback to the new coarse-to-fine flow predictions. Hence, we suggest that the network can still gain complementary information from the original feature pyramid.

The network architecture of this stage is similar to the former stage except some modifications and the incorporation of the new occlusion-aware feature pyramid (see Fig. 2). The original feature pyramid is directly placed into this stage using the same parameters. The maximum displacement of all correlation layers in this stage is set to 2 since we expect it to perform mainly refinements. Correlation layers are used as the feature matching modules for the occlusion-aware feature pyramid since there is no need for feature warping. At each level, the resulting cost volumes from dual feature pyramids are concatenated together with the other terms including an extra flow predicted from the previous stage at the current level. As suggested in FlowNet2 [14], we fix all the parameters in the first stage (MaskFlowNet-S) when training the whole MaskFlowNet.

5. Experiments

5.1. Implementation

We implement a Python-trainable code framework using MXNet [6]. Mostly, we follow the training configurations as suggested in PWC-Net+ [31, 32] and FlowNet2 [14]. More details can be found in the supplementary material.

Training Schedule. MaskFlowNet-S is first trained on FlyingChairs [8] and then tuned on FlyingThings3D [19] following the same schedule as PWC-Net [32]. When fine-tuning on Sintel, we use the same batch configuration (2 from Sintel, 1 from KITTI 2015, and 1 from HD1K) and a longer training schedule (1000k iterations) referring to the cyclic learning rate proposed by PWC-Net+ [31]. When training the whole MaskFlowNet, we fix all the parameters in MaskFlowNet-S as suggested in FlowNet2 [14] and follow again the same schedule except that it is shorter on FlyingChairs (800k iterations). For submission to KITTI, we fine-tune our model on the combination of KITTI 2012 and 2015 datasets based on the tuned checkpoint on Sintel, while the input images are resized to 1280 \( \times \) 576 (before augmentation and cropping) since the decreased aspect ratio better balances the vertical and horizontal displacement.

Data Augmentation. We implement geometric and chromatic augmentations referring to the implementation of FlowNet [8] and IRR-PWC [13]. We suppress the degree of augmentations when fine-tuning on KITTI as suggested. For sparse ground-truth flow in KITTI, the augmented flow is weighted averaged based on the interpolated valid mask.

Training Loss. We follow the multi-scale end-point error (EPE) loss when training on FlyingChairs and FlyingThings3D, and its robust version on Sintel and KITTI, using the same parameters as suggested in PWC-Net+ [31]. Weight decay is disabled since we find it of little help.
<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Sintel clean</th>
<th>Sintel final</th>
<th>KITTI 2012</th>
<th>KITTI 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AEPE train</td>
<td>AEPE test</td>
<td>AEPE train</td>
<td>AEPE test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AEPE train</td>
<td>AEPE test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fl-all</td>
<td>Fl-all</td>
</tr>
<tr>
<td>FlowNetS [8]</td>
<td>0.01</td>
<td>3.66</td>
<td>6.16</td>
<td>4.76</td>
<td>7.22</td>
</tr>
<tr>
<td>FlowNetC [8]</td>
<td>0.05</td>
<td>3.57</td>
<td>6.08</td>
<td>5.25</td>
<td>7.88</td>
</tr>
<tr>
<td>FlowNet2 [14]</td>
<td>0.12</td>
<td>2.02</td>
<td>3.96</td>
<td>3.14</td>
<td>6.02</td>
</tr>
<tr>
<td>SpyNet [27]</td>
<td>0.16</td>
<td>4.12</td>
<td>6.64</td>
<td>5.57</td>
<td>8.36</td>
</tr>
<tr>
<td>MR-Flow [38]</td>
<td>480</td>
<td>-</td>
<td>2.53</td>
<td>-</td>
<td>5.38</td>
</tr>
<tr>
<td>LiteFlowNet [11]</td>
<td>0.09</td>
<td>2.48</td>
<td>4.54</td>
<td>4.04</td>
<td>5.38</td>
</tr>
<tr>
<td>LiteFlowNet2 [12]</td>
<td>0.04</td>
<td>2.24</td>
<td>3.45</td>
<td>3.78</td>
<td>4.90</td>
</tr>
<tr>
<td>PWC-Net [32]</td>
<td>0.03</td>
<td>2.55</td>
<td>3.86</td>
<td>3.93</td>
<td>5.13</td>
</tr>
<tr>
<td>PWC-Net+ [31]</td>
<td>0.03</td>
<td>-</td>
<td>3.45</td>
<td>-</td>
<td>4.60</td>
</tr>
<tr>
<td>SelFlow [18]</td>
<td>0.09</td>
<td>-</td>
<td>3.74</td>
<td>-</td>
<td>4.26</td>
</tr>
<tr>
<td>VCN [40]</td>
<td>0.03</td>
<td>2.21</td>
<td>2.81</td>
<td>3.62</td>
<td>4.40</td>
</tr>
<tr>
<td>MaskFlownet-S</td>
<td>0.03</td>
<td>2.33</td>
<td>2.77</td>
<td>3.72</td>
<td>4.38</td>
</tr>
<tr>
<td>MaskFlownet</td>
<td>0.06</td>
<td>2.25</td>
<td>2.52</td>
<td>3.61</td>
<td>4.17</td>
</tr>
</tbody>
</table>

Table 1. Results of different methods on the MPI Sintel, KITTI 2012 and 2015 benchmarks. Values listed in the train columns only consider those models which are not trained on the corresponding training set and thus comparable. AEPE: average end-point error over all valid pixels. Fl-all: percentage of optical flow outliers over all valid pixels. Running times are referred to [32]; our time is measured on an NVIDIA TITAN Xp GPU, which is comparable to the NVIDIA TITAN X used by [32].

Figure 6. Qualitative comparison among PWC-Net [32], MaskFlownet-S, and MaskFlownet. Highlights for each row: (a) weakening the checkerboard effect; (b) separating background from two fighting figures; (c) preserving the weakly connected head of the moving figure; (d) maintaining a fluent flow for the background at left-bottom; (e) preserving boundary details of the flying creature and buildings.

### 5.2. Main Results

MaskFlownet outperforms all published optical flow methods on the MPI Sintel [5], KITTI 2012 [9] and KITTI 2015 [23] benchmarks as presented in Table 1, while the end-to-end MaskFlownet-S achieves a satisfactory result as well. Values listed in the training sets only consider the models that have never seen them and hence comparable; note that training on the synthetic datasets is of limited generalizability. Fig. 6 visualizes the predicted flows as a comparison. All samples are chosen from the Sintel training set (final pass). We can observe that MaskFlownet in general better separates moving objects from the background,
Figure 7. Learnable occlusion mask. MaskFlownet can jointly learn a rough occlusion mask without any explicit supervision.

Figure 8. Asymmetricity in the learned feature maps. The source features and the target features are level-2 feature maps prior to the correlation layer in the AsymOFMM. This figure presents the input image overlay, image 1 features (source features), the warped image 2 features after masking (masked features) and after trade-off (target features). Comparing the source features with the target features, we can see that the AsymOFMM enables the network to learn very different feature representations.

Figure 9. The trade-off term facilitates the learning of occlusions. Without the trade-off term, the learnable occlusion mask fails to achieve a clear estimation; if there is an additive shortcut that skips over warping, only motion boundaries are learned. With the trade-off term, large occlusions are successfully learned.

5.3. Ablation Study

Feature Matching Module. Table 2 presents the results when replacing the AsymOFMM in MaskFlownet-S with OFMM or FMM. We split about 20% sequences for validation when fine-tuning on the Sintel training set. While
Table 2. Feature matching module.

<table>
<thead>
<tr>
<th>Module</th>
<th>Chairs test</th>
<th>Sintel (train)</th>
<th>Things3D train</th>
<th>Sintel (val)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMM</td>
<td>1.61</td>
<td>3.25</td>
<td>4.59</td>
<td>2.55</td>
</tr>
<tr>
<td>OFMM</td>
<td>1.62</td>
<td>3.20</td>
<td>4.50</td>
<td>2.52</td>
</tr>
<tr>
<td>AsymOFMM</td>
<td>1.56</td>
<td>2.88</td>
<td>4.25</td>
<td>2.33</td>
</tr>
</tbody>
</table>

Table 3. Asymmetry and deformable convolution.

<table>
<thead>
<tr>
<th>Module</th>
<th>Chairs test</th>
<th>Sintel (train)</th>
<th>Things3D train</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFMM</td>
<td>1.62</td>
<td>3.20</td>
<td>4.50</td>
</tr>
<tr>
<td>+ sym-conv</td>
<td>1.61</td>
<td>3.33</td>
<td>4.64</td>
</tr>
<tr>
<td>+ asym-conv</td>
<td>1.52</td>
<td>2.96</td>
<td>4.29</td>
</tr>
<tr>
<td>+ deform-conv</td>
<td>1.56</td>
<td>2.88</td>
<td>4.25</td>
</tr>
</tbody>
</table>

Table 4. Learnable occlusion mask and the trade-off term.

<table>
<thead>
<tr>
<th>Network</th>
<th>Tuned on Sintel val</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaskFlownet-S</td>
<td>2.70 4.07</td>
</tr>
<tr>
<td>+ single pyramid w/o mask</td>
<td>2.53 3.90</td>
</tr>
<tr>
<td>+ single pyramid w/ mask</td>
<td>2.55 3.88</td>
</tr>
<tr>
<td>+ dual pyramids w/o mask</td>
<td>2.52 3.85</td>
</tr>
<tr>
<td>+ dual pyramids w/ mask</td>
<td>2.52 3.83</td>
</tr>
</tbody>
</table>

Table 5. Network cascading with dual pyramids.

| OFMM achieves relative gains compared with the original FMM, the proposed AsymOFMM significantly outperforms the symmetric variants.

Asymmetry. We demonstrate the effectiveness of the asymmetry by comparing AsymOFMM (i.e., “OFMM + deform-conv”, MaskFlownet-S) with the raw OFMM, “OFMM + sym-conv” (adding an extra convolutional layer at each feature pyramid level), and “OFMM + asym-conv” (adding an asymmetric convolutional layer prior to the warping operation at each level). As shown in Table 3, increasing the depth of convolutional layers has a limited impact in the symmetric setting, while a simple asymmetric design achieves consistently better performance. It also indicates that our deformable convolution can be a better choice over the “asym-conv” version. Although it is commonly believed that matched patterns should be embedded into similar feature vectors, we suggest that the network can really benefit from learning very different feature representations as visualized in Fig. 8.

Learnable Occlusion Mask. Table 4 presents the results if the mask or the trade-off term is disabled. Interestingly, only the two factors combined lead to performance gains. A possible explanation is that the occlusion mask helps if and only if it is learned properly, where the trade-off term plays a vital role (see Fig. 9).

Network Cascading. Table 5 indicates that MaskFlownet consistently benefits from dual feature pyramids over a single new pyramid, while the concatenated occlusion mask gains the performance on the Sintel final pass. We hypothesize that the occlusion-aware feature pyramid mainly contributes to the harder final pass since the occluded areas can be more easily mismatched, but it might be overfitted on the easier clean pass. We demonstrate how the learned occlusion mask could affect the extracted feature map in Fig. 10. The occluded areas are smoothed during feature extraction and hence become more distinguishable.

6. Conclusion

We propose the AsymOFMM, which incorporates a learnable occlusion mask that filters occluded areas immediately after feature warping without any explicit supervision. AsymOFMM can be easily integrated into an end-to-end network while introducing negligible computational cost. We further propose a two-stage network — MaskFlownet — which exploits dual pyramids and achieves superior performance on all modern optical flow benchmarks. Our approach opens a promisingly new perspective on dealing with occlusions for both supervised and unsupervised optical flow estimation, and we also expect it as an initiative and effective component in many other applications.
References


