arXiv:1904.09117v1 [cs.CV] 19 Apr 2019

SelFlow: Self-Supervised Learning of Optical Flow

Pengpeng Liu[†*], Michael Lyu[†], Irwin King[†], Jia Xu[§]
[†] The Chinese University of Hong Kong, [§] Tencent AI Lab

Abstract

We present a self-supervised learning approach for optical flow. Our method distills reliable flow estimations from non-occluded pixels, and uses these predictions as ground truth to learn optical flow for hallucinated occlusions. We further design a simple CNN to utilize temporal information from multiple frames for better flow estimation. These two principles lead to an approach that yields the best performance for unsupervised optical flow learning on the challenging benchmarks including MPI Sintel, KITTI 2012 and 2015. More notably, our self-supervised pre-trained model provides an excellent initialization for supervised fine-tuning. Our fine-tuned models achieve state-of-the-art results on all three datasets. At the time of writing, we achieve EPE=4.26 on the Sintel benchmark, outperforming all submitted methods.

1. Introduction

Optical flow estimation is a core building block for a variety of computer vision systems [30, 8, 39, 4]. Despite decades of development, accurate flow estimation remains an open problem due to one key challenge: occlusion. Traditional approaches minimize an energy function to encourage association of visually similar pixels and regularize incoherent motion to propagate flow estimation from non-occluded pixels to occluded pixels [13, 5, 6, 38]. However, this family of methods is often time-consuming and not applicable for real-time applications.

Recent studies learn to estimate optical flow end-to-end from images using convolutional neural networks (CNNs) [10, 35, 15, 14, 43]. However, training fully supervised CNNs requires a large amount of labeled training data, which is extremely difficult to obtain for optical flow, especially when there are occlusions. Considering the recent performance improvements obtained when employing hundreds of millions of labeled images [40], it is obvious that the size of training data is a key bottleneck for optical flow estimation.

In the absence of large-scale real-world annotations, existing methods turn to pre-train on synthetic labeled datasets [10, 28] and then fine-tune on small annotated datasets [15, 14, 43]. However, there usually exists a large gap between the distribution of synthetic data and natural scenes. In order to train a stable model, we have to carefully follow specific learning schedules across different datasets [15, 14, 43].

One promising direction is to develop unsupervised optical flow learning methods that benefit from unlabeled data. The basic idea is to warp the target image towards the reference image according to the estimated optical flow, then minimize the difference between the reference image and the warped target image using a photometric loss [20, 37]. Such idea works well for non-occluded pixels but turns to provide misleading information for occluded pixels. Recent methods propose to exclude those occluded pixels when computing the photometric loss or employ additional spatial and temporal smoothness terms to regularize flow estimation [29, 46, 18]. Most recently, DDFlow [26] proposes a data distillation approach, which employs random cropping to create occlusions for self-supervision. Unfortunately, these methods fails to generalize well for all natural occlusions. As a result, there is still a large performance gap comparing unsupervised methods with state-of-the-art fully supervised methods.

Is it possible to effectively learn optical flow with occlusions? In this paper, we show that a self-supervised approach can learn to estimate optical flow with any form of occlusions from unlabeled data. Our work is based on distilling reliable flow estimations from non-occluded pixels, and using these predictions to guide the optical flow learning for hallucinated occlusions. Figure 1 illustrates our idea to create synthetic occlusions by perturbing superpixels. We further utilize temporal information from multiple frames to improve flow prediction accuracy within a simple CNN architecture. The resulted learning approach yields the highest accuracy among all unsupervised optical flow learning methods on Sintel and KITTI benchmarks.

Surprisingly, our self-supervised pre-trained model provides an excellent initialization for supervised fine-tuning. At the time of writing, our fine-tuned model achieves the

[*]Work mainly done during an internship at Tencent AI Lab.

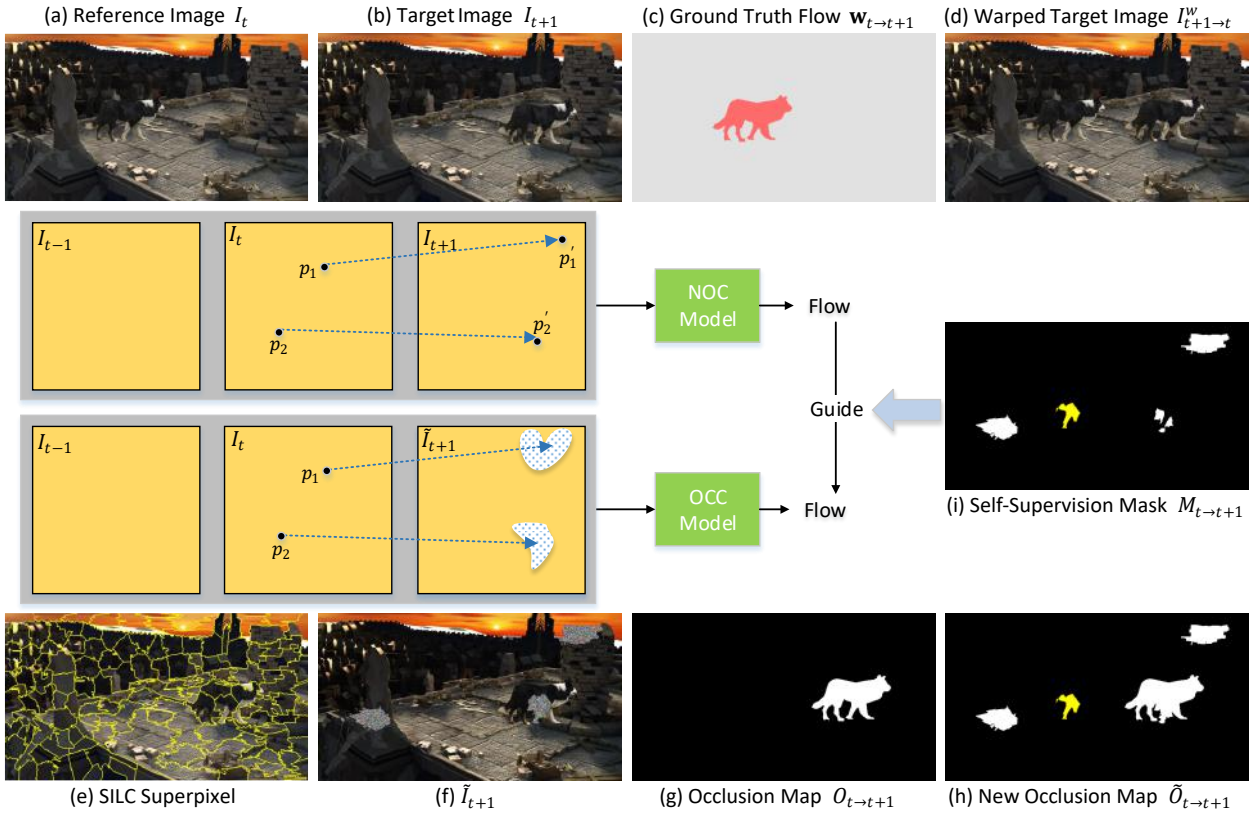

Figure 1. A toy example to illustrate our self-supervised learning idea. We first train our NOC-model with the classical photometric loss (measuring the difference between the reference image (a) and the warped target image(d)), guided by the occlusion map (g). Then we perturbate randomly selected superpixels in the target image (b) to hallucinate occlusions. Finally, we use reliable flow estimations from our NOC-Model to guide the learning of our OCC-Model for those newly occluded pixels (denoted by self-supervision mask (i), where value 1 means the pixel is non-occluded in (g) but occluded in (h)). Note the yellow region is part of the moving dog. Our self-supervised approach learns optical flow for both moving objects and static scenes.

highest reported accuracy (EPE=4.26) on the Sintel benchmark. Our approach also significantly outperforms all published optical flow methods on the KITTI 2012 benchmark, and achieves highly competitive results on the KITTI 2015 benchmark. To the best of our knowledge, it is the first time that a supervised learning method achieves such remarkable accuracies without using any external labeled data.

2. Related Work

Classical Optical Flow Estimation. Classical variational approaches model optical flow estimation as an energy minimization problem based on brightness constancy and spatial smoothness [13]. Such methods are effective for small motion, but tend to fail when displacements are large. Later works integrate feature matching to initialize sparse matching, and then interpolate into dense flow maps in a pyramidal coarse-to-fine manner [6, 47, 38]. Recent works use convolutional neural networks (CNNs) to improve sparse matching by learning an effective feature embedding [49, 2]. However, these methods are often computationally expensive and can not be trained end-to-end. One natural extension to improve robustness and accuracy for flow estimation is to incorporate temporal information over multiple frames. A straightforward way is to add temporal constraints such as constant velocity [19, 22, 41], constant acceleration [45, 3], low-dimensional linear subspace [16], or rigid/non-rigid segmentation [48]. While these formulations are elegant and well-motivated, our method is much simpler and does not rely on any assumption of the data. Instead, our approach directly learns optical flow for a much wider range of challenging cases existing in the data.

Supervised Learning of Optical Flow. One promising direction is to learn optical flow with CNNs. FlowNet [10] is the first end-to-end optical flow learning framework. It takes two consecutive images as input and outputs a dense flow map. The following work FlowNet 2.0 [15] stacks several basic FlowNet models for iterative refinement, and significantly improves the accuracy. SpyNet [35] proposes to warp images at multiple scales to cope with large displacements, resulting in a compact spatial pyramid network.

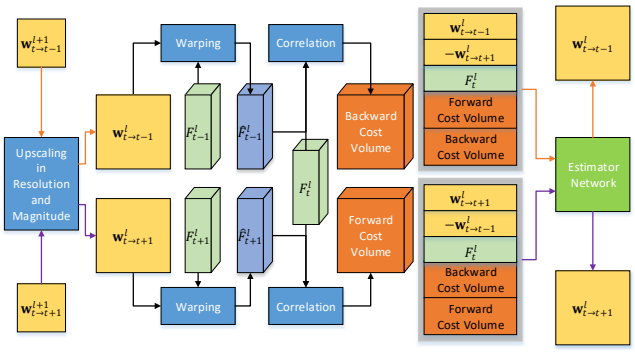

Figure 2. Our network architecture at each level (similar to PWC-Net [43]). $\dot{\mathbf{w}}^l$ denotes the initial coarse flow of level l and $\hat{F}^l$ denotes the warped feature representation. At each level, we swap the initial flow and cost volume as input to estimate both forward and backward flow concurrently. Then these estimations are passed to layer $l-1$ to estimate higher-resolution flow.

Recently, PWC-Net [43] and LiteFlowNet [14] propose to warp features extracted from CNNs and achieve state-of-the-art results with lightweight framework. However, obtaining high accuracy with these CNNs requires pre-training on multiple synthetic datasets and follows specific training schedules [10, 28]. In this paper, we reduce the reliance on pre-training with synthetic data, and propose an effective self-supervised training method with unlabeled data.

Unsupervised Learning of Optical Flow. Another interesting line of work is unsupervised optical flow learning. The basic principles are based on brightness constancy and spatial smoothness [20, 37]. This leads to the most popular photometric loss, which measures the difference between the reference image and the warped image. Unfortunately, this loss does not hold for occluded pixels. Recent studies propose to first obtain an occlusion map and then exclude those occluded pixels when computing the photometric difference [29, 46]. Janai *et al*. [18] introduces to estimate optical flow with a multi-frame formulation and more advanced occlusion reasoning, achieving state-of-the-art unsupervised results. Very recently, DDFlow [26] proposes a data distillation approach to learning the optical flow of occluded pixels, which works particularly well for pixels near image boundaries. Nonetheless, all these unsupervised learning methods only handle specific cases of occluded pixels. They lack the ability to reason about the optical flow of all possible occluded pixels. In this work, we address this issue by a superpixel-based occlusion hallucination technique.

Self-Supervised Learning. Our work is closely related to the family of self-supervised learning methods, where the supervision signal is purely generated from the data itself. It is widely used for learning feature representations from unlabeled data [21]. A pretext task is usually employed, such as image inpainting [34], image colorization [24], solving

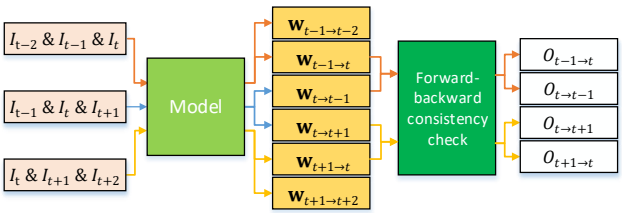

Figure 3. Data flow for self-training with multiple-frame. To estimate occlusion map for three-frame flow learning, we use five images as input. This way, we can conduct a forward-backward consistency check to estimate occlusion maps between I_t and I_{t+1}, between I_t and I_{t-1} respectively.

Jigsaw puzzles [32]. Pathak *et al*. [33] propose to explore low-level motion-based cues to learn feature representations without manual supervision. Doersch *et al*. [9] combine multiple self-supervised learning tasks to train a single visual representation. In this paper, we make use of the domain knowledge of optical flow, and take reliable predictions of non-occluded pixels as the self-supervision signal to guide our optical flow learning of occluded pixels.

3. Method

In this section, we present our self-supervised approach to learning optical flow from unlabeled data. To this end, we train two CNNs (NOC-Model and OCC-Model) with the same network architecture. The former focuses on accurate flow estimation for non-occluded pixels, and the latter learns to predict optical flow for all pixels. We distill reliable non-occluded flow estimations from NOC-Model to guide the learning of OCC-Model for those occluded pixels. Only OCC-Model is needed at testing. We build our network based on PWC-Net [43] and further extend it to multi-frame optical flow estimation (Figure 2). Before describing our approach in detail, we first define our notations.

3.1. Notation

Given three consecutive RGB images I_{t-1}, I_t, I_{t+1}, our goal is to estimate the forward optical flow from I_t to I_{t+1}. Let $\mathbf{w}_{i\to j}$ denote the flow from I_i to I_j, e.g., $\mathbf{w}_{t\to t+1}$ denotes the forward flow from I_t to I_{t+1}, $\mathbf{w}_{t\to t-1}$ denotes the backward flow from I_t to I_{t-1}. After obtaining optical flow, we can backward warp the target image to reconstruct the reference image using Spatial Transformer Network [17, 46]. Here, we use $I^w_{j\to i}$ to denote warping I_j to I_i with flow $\mathbf{w}_{i\to j}$. Similarly, we use $O_{i\to j}$ to denote the occlusion map from I_i to I_j, where value 1 means the pixel in I_i is not visible in I_j.

In our self-supervised setting, we create the new target image $\widetilde{I}_{t+1}$ by injecting random noise on superpixels for occlusion generation. We can inject noise to any of three consecutive frames and even multiple of them as shown in Figure 1. For brevity, here we choose I_{t+1} as an example.

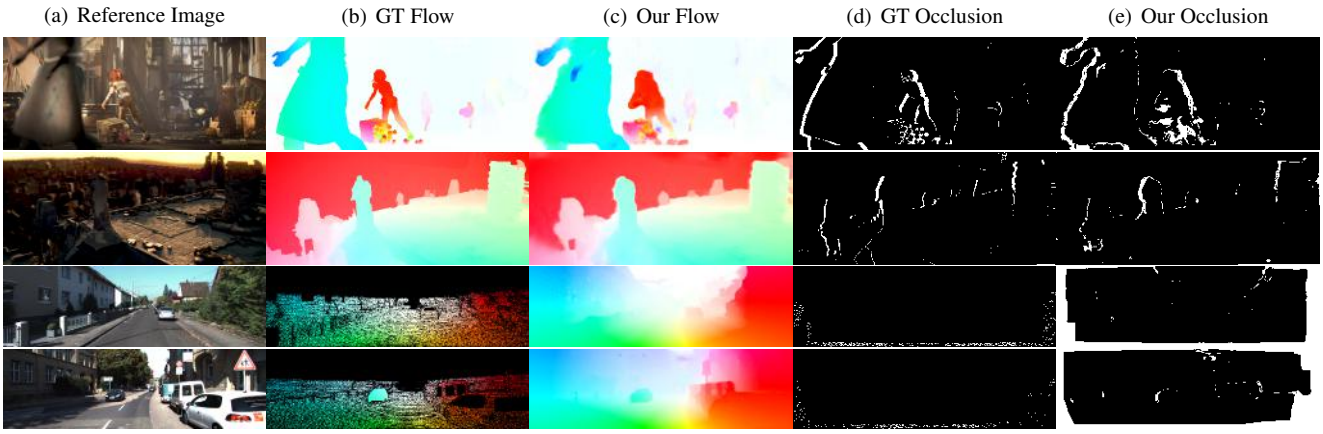

Figure 4. Sample unsupervised results on Sintel and KITTI dataset. From top to bottom, we show samples from Sintel Final, KITTI 2012 and KITTI 2015. Our model can estimate both accurate flow and occlusion map. Note that on KITTI datasets, the occlusion maps are sparse, which only contain pixels moving out of the image boundary.

If we let I_{t-1}, I_t and $\widetilde{I}_{t+1}$ as input, then $\widetilde{\mathbf{w}}, \widetilde{O}, \widetilde{I}^w$ represent the generated optical flow, occlusion map and warped image respectively.

3.2. CNNs for Multi-Frame Flow Estimation

In principle, our method can utilize any CNNs. In our implementation, we build on top of the seminar PWC-Net [43]. PWC-Net employs pyramidal processing to increase the flow resolution in a coarse-to-fine manner and utilizes feature warping, cost volume construction to estimate optical flow at each level. Based on these principles, it has achieved state-of-the-art performance with a compact model size.

As shown in Figure 2, our three-frame flow estimation network structure is built upon two-frame PWC-Net with several modifications to aggregate temporal information. First, our network takes three images as input, thus produces three feature representations F_{t-1}, F_t and F_{t+1}. Second, apart from forward flow $\mathbf{w}_{t\to t+1}$ and forward cost volume, out model also computes backward flow $\mathbf{w}_{t\to t-1}$ and backward cost volume at each level simultaneously. Note that when estimating forward flow, we also utilize the initial backward flow and backward cost volume information. This is because past frame I_{t-1} can provide very valuable information, especially for those regions that are occluded in the future frame I_{t+1} but not occluded in I_{t-1}. Our network combines all this information together and therefore estimates optical flow more accurately. Third, we stack initial forward flow $\dot{\mathbf{w}}^l_{t\to t+1}$, minus initial backward flow $-\dot{\mathbf{w}}^l_{t+1\to t}$, feature of reference image F^l_t, forward cost volume and backward cost volume to estimate the forward flow at each level. For backward flow, we just swap the flow and cost volume as input. Forward and backward flow estimation networks share the same network structure and weights. For initial flow at each level, we upscale optical flow of the next level both in resolution and magnitude.

3.3. Occlusion Estimation

For two-frame optical flow estimation, we can swap two images as input to generate forward and backward flow, then the occlusion map can be generated based on the forward-backward consistency prior [44, 29]. To make this work under our three-frame setting, we propose to utilize the adjacent five frame images as input as shown in Figure 3. Specifically, we estimate bi-directional flows between I_t and I_{t+1}, namely $\mathbf{w}_{t\to t+1}$ and $\mathbf{w}_{t+1\to t}$. Similarly, we also estimate the flows between I_t and I_{t-1}. Finally, we conduct a forward and backward consistency check to reason the occlusion map between two consecutive images.

For forward-backward consistency check, we consider one pixel as occluded when the mismatch between the forward flow and the reversed forward flow is too large. Take $O_{t\to t+1}$ as an example, we can first compute the reversed forward flow as follows,

$$\hat{\mathbf{w}}_{t\to t+1} = \mathbf{w}_{t+1\to t}(\mathbf{p} + \mathbf{w}_{t\to t+1}(\mathbf{p})), \tag{1}$$

A pixel is considered occluded whenever it violates the following constraint:

$$|\mathbf{w}_{t\to t+1} + \hat{\mathbf{w}}_{t\to t+1}|^2 < \alpha_1(|\mathbf{w}_{t\to t+1}|^2 + |\hat{\mathbf{w}}_{t\to t+1}|^2) + \alpha_2, \tag{2}$$

where we set $\alpha_1 = 0.01$, $\alpha_2 = 0.05$ for all our experiments. Other occlusion maps are computed in the same way.

3.4. Occlusion Hallucination

During our self-supervised training, we hallucinate occlusions by perturbing local regions with random noise. In a newly generated target image, the pixels corresponding to noise regions automatically become occluded. There are many ways to generate such occlusions. The most

	Method	Sintel Clean		Sintel Final		KITTI 2012			KITTI 2015	
		train	test	train	test	train	test	test(Fl)	train	test(Fl)
Unsupervised	BackToBasic+ft [20]	–	–	–	–	11.3	9.9	–	–	–
	DSTFlow+ft [37]	(6.16)	10.41	(6.81)	11.27	10.43	12.4	–	16.79	39%
	UnFlow-CSS [29]	–	–	(7.91)	10.22	3.29	–	–	8.10	23.30%
	OccAwareFlow+ft [46]	(4.03)	7.95	(5.95)	9.15	3.55	4.2	–	8.88	31.2%
	MultiFrameOccFlow-None+ft [18]	(6.05)	–	(7.09)	–	–	–	–	6.65	–
	MultiFrameOccFlow-Soft+ft [18]	(3.89)	7.23	(5.52)	8.81	–	–	–	6.59	22.94%
	DDFlow+ft [26]	(2.92)	**6.18**	3.98	7.40	2.35	3.0	8.86%	5.72	14.29%
	Ours	**(2.88)**	6.56	**(3.87)**	**6.57**	**1.69**	**2.2**	**7.68%**	**4.84**	**14.19%**
Supervised	FlowNetS+ft [10]	(3.66)	6.96	(4.44)	7.76	7.52	9.1	44.49%	–	–
	FlowNetC+ft [10]	(3.78)	6.85	(5.28)	8.51	8.79	–	–	–	–
	SpyNet+ft [35]	(3.17)	6.64	(4.32)	8.36	8.25	10.1	20.97%	–	35.07%
	FlowFieldsCNN+ft [2]	–	3.78	–	5.36	–	3.0	13.01%	–	18.68 %
	DCFlow+ft [49]	–	3.54	–	5.12	–	–	–	–	14.83%
	FlowNet2+ft [15]	(1.45)	4.16	(2.01)	5.74	(1.28)	1.8	8.8%	(2.3)	11.48%
	UnFlow-CSS+ft [29]	–	–	–	–	(1.14)	1.7	8.42%	(1.86)	11.11%
	LiteFlowNet+ft-CVPR [14]	(1.64)	4.86	(2.23)	6.09	(1.26)	1.7	–	(2.16)	10.24%
	LiteFlowNet+ft-axXiv [14]	**(1.35)**	4.54	(1.78)	5.38	(1.05)	1.6	7.27%	(1.62)	9.38%
	PWC-Net+ft-CVPR [43]	(2.02)	4.39	(2.08)	5.04	(1.45)	1.7	8.10%	(2.16)	9.60%
	PWC-Net+ft-axXiv [42]	(1.71)	3.45	(2.34)	4.60	(1.08)	**1.5**	6.82%	(1.45)	7.90%
	ProFlow+ft [27]	(1.78)	**2.82**	–	5.02	(1.89)	2.1	7.88%	(5.22)	15.04%
	ContinualFlow+ft [31]	–	3.34	–	4.52	–	–	–	–	10.03%
	MFF+ft [36]	–	3.42	–	4.57	–	1.7	7.87%	–	**7.17%**
	Ours+ft	(1.68)	3.74	**(1.77)**	**4.26**	**(0.76)**	**1.5**	**6.19%**	(**1.18**)	8.42%

Table 1. Comparison with state-of-the-art learning based optical flow estimation methods. Our method outperforms all unsupervised optical flow learning approaches on all datasets. Our supervised fine-tuned model achieves the highest accuracy on the Sintel Final dataset and KITTI 2012 dataset. All numbers are EPE except for the last column of KITTI 2012 and KITTI 2015 testing sets, where we report percentage of erroneous pixels over all pixels (Fl-all). Missing entries (-) indicate that the results are not reported for the respective method. Parentheses mean that the training and testing are performed on the same dataset. Bold fonts highlight the best results among unsupervised and supervised methods respectively.

straightforward way is to randomly select rectangle regions. However, rectangle occlusions rarely exist in real-world sequences. To address this issue, we propose to first generate superpixels [1], then randomly select several superpixels and fill them with noise. There are two main advantages of using superpixel. First, the shape of a superpixel is usually random and superpixel edges are often part of object boundaries. The is consistent with the real-world cases and makes the noise image more realistic. We can choose several superpixels which locate at different locations to cover more occlusion cases. Second, the pixels within each superpixel usually belong to the same object or have similar flow fields. Prior work has found low-level segmentation is helpful for optical flow estimation [49]. Note that the random noise should lie in the pixel value range.

Figure 1 shows a simple example, where only the dog extracted from the COCO dataset [25] is moving. Initially, the occlusion map between I_t and I_{t+1} is (g). After randomly selecting several superpixels from (e) to inject noise, the occlusion map between I_t and $\widetilde{I}_{t+1}$ change to (h). Next, we describe how to make use of these occlusion maps to guide our self-training.

3.5. NOC-to-OCC as Self-Supervision

Our self-training idea is built on top of the classical photometric loss [29, 46, 18], which is highly effective for non-occluded pixels. Figure 1 illustrates our main idea. Suppose pixel p_1 in image I_t is not occluded in I_{t+1}, and pixel p'_1 is its corresponding pixel. If we inject noise to I_{t+1} and let I_{t-1}, I_t, $\widetilde{I}_{t+1}$ as input, p_1 then becomes occluded. Good news is we can still use the flow estimation of NOC-Model as annotations to guide OCC-Model to learn the flow of p_1 from I_t to $\widetilde{I}_{t+1}$. This is also consistent with real-world occlusions, where the flow of occluded pixels can be estimated based on surrounding non-occluded pixels. In the example of Figure 1, self-supervision is only employed to (i), which represents those pixels non-occluded from I_t to I_{t+1} but become occluded from I_t to $\widetilde{I}_{t+1}$.

3.6. Loss Functions

Similar to previous unsupervised methods, we first apply photometric loss L_p to non-occluded pixels. Photometric

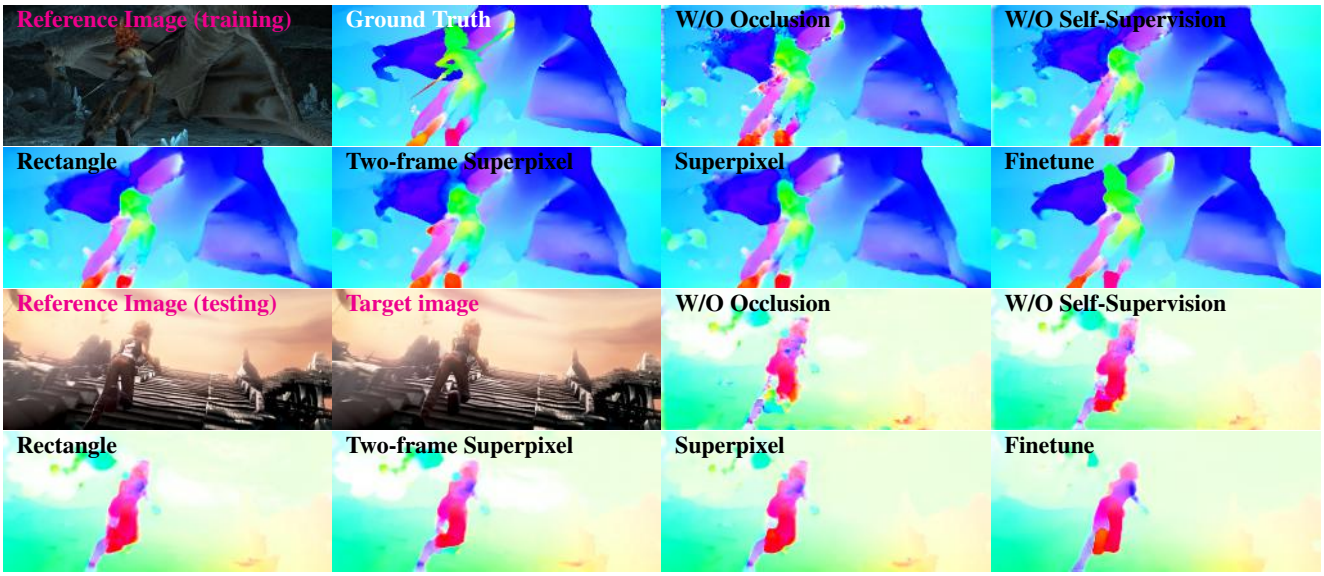

Figure 5. Qualitative comparison of our model under different settings on Sintel Clean training and Sintel Final testing dataset. Occlusion handling, multi-frame formulation and self-supervision consistently improve the performance.

loss is defined as follows:

$$L_p = \sum_{i,j} \frac{\sum \psi(I_i - I^w_{j\to i}) \odot (1 - O_i)}{\sum (1 - O_i)} \tag{3}$$

where $\psi(x) = (|x|+\epsilon)^q$ is a robust loss function, $\odot$ denotes the element-wise multiplication. We set $\epsilon = 0.01$, $q = 0.4$ for all our experiments. Only L_p is necessary to train the NOC-Model.

To train our OCC-Model to estimate optical flow of occluded pixels, we define a self-supervision loss L_o for those synthetic occluded pixels (Figure 1(i)). First, we compute a self-supervision mask M to represent these pixels,

$$M_{i\to j} = \text{clip}(\widetilde{O}_{i\to j} - O_{i\to j}, 0, 1) \tag{4}$$

Then, we define our self-supervision loss L_o as,

$$L_o = \sum_{i,j} \frac{\sum \psi(\mathbf{w}_{i\to j} - \widetilde{\mathbf{w}}_{i\to j}) \odot M_{i\to j}}{\sum M_{i\to j}} \tag{5}$$

For our OCC-Model, we train with a simple combination of $L_p + L_o$ for both non-occluded pixels and occluded pixels. Note our loss functions do not rely on spatial and temporal consistent assumptions, and they can be used for both classical two-frame flow estimation and multi-frame flow estimation.

3.7. Supervised Fine-tuning

After pre-training on raw dataset, we use real-world annotated data for fine-tuning. Since there are only annotations for forward flow $\mathbf{w}_{t\to t+1}$, we skip backward flow estimation when computing our loss. Suppose that the ground truth flow is $\mathbf{w}^{gt}_{t\to t+1}$, and mask V denotes whether the pixel has a label, where value 1 means that the pixel has a valid ground truth flow. Then we can obtain the supervised fine-tuning loss as follows,

$$L_s = \sum (\psi(\mathbf{w}^{gt}_{t\to t+1} - \mathbf{w}_{t\to t+1}) \odot V) / \sum V \tag{6}$$

During fine-tuning, We first initialize the model with the pre-trained OCC-Model on each dataset, then optimize it using L_s.

4. Experiments

We evaluate and compare our methods with state-of-the-art unsupervised and supervised learning methods on public optical flow benchmarks including MPI Sintel [7], KITTI 2012 [11] and KITTI 2015 [30]. To ensure reproducibility and advance further innovations, we make our code and models publicly available at `https://github.com/ppliuboy/SelFlow`.

4.1. Implementation Details

Data Preprocessing. For Sintel, we download the Sintel movie and extract $\sim 10,000$ images for self-training. We first train our model on this raw data, then add the official Sintel training data (including both "final" and "clean" versions). For KITTI 2012 and KITTI 2015, we use multi-view extensions of the two datasets for unsupervised pre-training, similar to [37, 46]. During training, we exclude the image pairs with ground truth flow and their neighboring frames (frame number 9-12) to avoid the mixture of training and testing data.

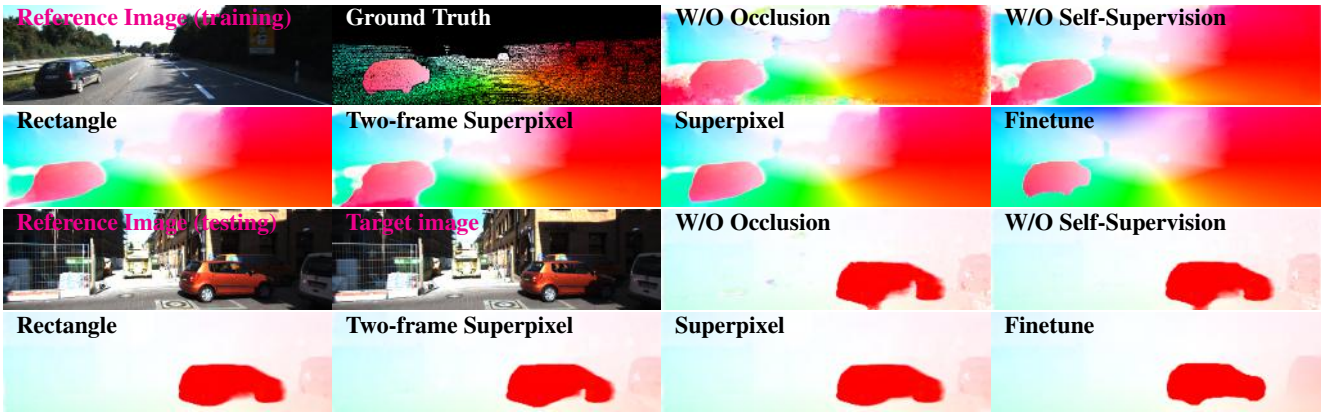

Figure 6. Qualitative comparison of our model under different settings on KITTI 2015 training and testing dataset. Occlusion handling, multi-frame formulation and self-supervision consistently improve the performance.

We rescale the pixel value from [0, 255] to [0, 1] for unsupervised training, while normalizing each channel to be standard normal distribution for supervised fine-tuning. This is because normalizing image as input is more robust for luminance changing, which is especially helpful for optical flow estimation. For unsupervised training, we apply Census Transform [51] to images, which has been proved robust for optical flow estimation [12, 29].

Training procedure. We train our model with the Adam optimizer [23] and set batch size to be 4 for all experiments. For unsupervised training, we set the initial learning rate to be 10^{-4}, decay it by half every 50k iterations, and use random cropping, random flipping, random channel swapping during data augmentation. For supervised fine-tuning, we employ similar data augmentation and learning rate schedule as [10, 15].

For unsupervised pre-training, we first train our NOC-Model with photometric loss for 200k iterations. Then, we add our occlusion regularization and train for another 500k iterations. Finally, we initialize the OCC-Model with the trained weights of NOC-Model and train it with $L_p + L_o$ for 500k iterations. Since training two models simultaneously will cost more memory and training time, we just generate the flow and occlusion maps using the NOC-Model in advance and use them as annotations (just like KITTI with sparse annotations).

For supervised fine-tuning, we use the pre-trained OCC-Model as initialization, and train the model using our supervised loss L_s with 500k iterations for KITTI and $1,000$k iterations for Sintel. Note we do not require pre-training our model on any labeled synthetic dataset, hence we do not have to follow the specific training schedule (FlyingChairs [10]$\rightarrow$ FlyingThings3D [28]) as [15, 14, 43].

Evaluation Metrics. We consider two widely-used metrics to evaluate optical flow estimation: average endpoint error (EPE), percentage of erroneous pixels (Fl). EPE is the ranking metric on the Sintel benchmark, and Fl is the ranking metric on KITTI benchmarks.

4.2. Main Results

As shown in Table 1, we achieve state-of-the-art results for both unsupervised and supervised optical flow learning on all datasets under all evaluation metrics. Figure 4 shows sample results from Sintel and KITTI. Our method estimates both accurate optical flow and occlusion maps.

Unsupervised Learning. Our method achieves the highest accuracy for unsupervised learning methods on leading benchmarks. On the Sintel final benchmark, we reduce the previous best EPE from 7.40 [26] to 6.57, with 11.2% relative improvements. This is even better than several fully supervised methods including FlowNetS, FlowNetC [10], and SpyNet [35].

On the KITTI datasets, the improvement is more significant. For the training dataset, we achieve EPE=1.69 with 28.1% relative improvement on KITTI 2012 and EPE=4.84 with 15.3% relative improvement on KITTI 2015 compared with previous best unsupervised method DDFlow. On KITTI 2012 testing set, we achieve Fl-all=7.68%, which is better than state-of-the-art supervised methods including FlowNet2 [15], PWC-Net [43], ProFlow [27], and MFF [36]. On KITTI 2015 testing benchmark, we achieve Fl-all 14.19%, better than all unsupervised methods. Our unsupervised results also outperform some fully supervised methods including DCFlow [49] and ProFlow [27].

Supervised Fine-tuning. We further fine-tune our unsupervised model with the ground truth flow. We achieve state-of-the-art results on all three datasets, with Fl-all=6.19% on KITTI 2012 and Fl-all=8.42% on KITTI 2015. Most importantly, our method yields EPE=4.26 on the Sintel final dataset, achieving the highest accuracy on the Sintel benchmark among all submitted methods. All these show that our method reduces the reliance of pre-training with syn-

Occlusion Handling	Multiple Frame	Self-Supervision Rectangle	Self-Supervision Superpixel	Sintel Clean			Sintel Final			KITTI 2012			KITTI 2015		
				ALL	NOC	OCC	ALL	NOC	OCC	ALL	NOC	OCC	ALL	NOC	OCC
✗	✗	✗	✗	(3.85)	(1.53)	(33.48)	(5.28)	(2.81)	(36.83)	7.05	1.31	45.03	13.51	3.71	75.51
✗	✓	✗	✗	(3.67)	(1.54)	(30.80)	(4.98)	(2.68)	(34.42)	6.52	1.11	42.44	12.13	3.47	66.91
✓	✗	✗	✗	(3.35)	(1.37)	(28.70)	(4.50)	(2.37)	(31.81)	4.96	0.99	31.29	8.99	3.20	45.68
✓	✓	✗	✗	(3.20)	(1.35)	(26.63)	(4.33)	(2.32)	(29.80)	3.32	0.94	19.11	7.66	2.47	40.99
✓	✗	✗	✓	(2.96)	(1.33)	(23.78)	(4.06)	(2.25)	(27.19)	1.97	0.92	8.96	5.85	2.96	24.17
✓	✓	✓	✗	(2.91)	(1.37)	(22.58)	(3.99)	(2.27)	(26.01)	1.78	0.96	7.47	5.01	2.55	21.86
✓	✓	✗	✓	**(2.88)**	**(1.30)**	**(22.06)**	**(3.87)**	**(2.24)**	**(25.42)**	**1.69**	**0.91**	**6.95**	**4.84**	**2.40**	**19.68**

Table 2. Ablation study. We report EPE of our unsupervised results under different settings over all pixels (ALL), non-occluded pixels (NOC) and occluded pixels (OCC). Note that we employ Census Transform when computing photometric loss by default. Without Census Transform, the performance will drop.

Unsupervised Pre-training	Sintel Clean	Sintel Final	KITTI 2012	KITTI 2015
Without	1.97	2.68	3.93	3.10
With	**1.50**	**2.41**	**1.55**	**1.86**

Table 3. Ablation study. We report EPE of supervised fine-tuning results on our validation datasets with and without unsupervised pre-training.

thetic datasets and we do not have to follow specific training schedules across different datasets anymore.

4.3. Ablation Study

To demonstrate the usefulness of individual technical steps, we conduct a rigorous ablation study and show the quantitative comparison in Table 2. Figure 5 and Figure 6 show the qualitative comparison under different settings, where "W/O Occlusion" means occlusion handling is not considered, "W/O Self-Supervision" means occlusion handling is considered but self-supervision is not employed, "Rectangle" and "Superpixel" represent self-supervision is employed with rectangle and superpixel noise injection respectively. "Two-Frame Superpixel" means self-supervision is conducted with only two frames as input.

Two-Frame vs Multi-Frame. Comparing row 1 and row 2, row 3 and row 4 row 5 and row 7 in Table 2, we can see that using multiple frames as input can indeed improve the performance, especially for occluded pixels. It is because multiple images provide more information, especially for those pixels occluded in one direction but non-occluded in the reverse direction.

Occlusion Handling. Comparing the row 1 and row 3, row 2 and row 4 in Table 2, we can see that occlusion handling can improve optical flow estimation performance over all pixels on all datasets. This is due to the fact that brightness constancy assumption does not hold for occluded pixels.

Self-Supervision. We employ two strategies for our occlusion hallucination: rectangle and superpixel. Both strategies improve the performance significantly, especially for occluded pixels. Take superpixel setting as an example, EPE-OCC decrease from 26.63 to 22.06 on Sintel Clean, from 29.80 to 25.42 on Sintel Final, from 19.11 to 6.95 on KITTI 2012, and from 40.99 to 19.68 on KITTI 2015. Such a big improvement demonstrates the effectiveness of our self-supervision strategy.

Comparing superpixel noise injection with rectangle noise injection, superpixel setting has several advantages. First, the shape of the superpixel is random and edges are more correlated to motion boundaries. Second, the pixels in the same superpixel usually have similar motion patterns. As a result, the superpixel setting achieves slightly better performance.

Self-Supervised Pre-training. Table 3 compares supervised results with and without our self-supervised pre-training on the validation sets. If we do not employ self-supervised pre-training and directly train the model using only the ground truth, the model fails to converge well due to insufficient training data. However, after utilizing our self-supervised pre-training, it converges very quickly and achieves much better results.

5. Conclusion

We have presented a self-supervised approach to learning accurate optical flow estimation. Our method injects noise into superpixels to create occlusions, and let one model guide the another to learn optical flow for occluded pixels. Our simple CNN effectively aggregates temporal information from multiple frames to improve flow prediction. Extensive experiments show our method significantly outperforms all existing unsupervised optical flow learning methods. After fine-tuning with our unsupervised model, our method achieves state-of-the-art flow estimation accuracy on all leading benchmarks. Our results demonstrate it is possible to completely reduce the reliance of pre-training on synthetic labeled datasets, and achieve superior performance by self-supervised pre-training on unlabeled data.

6. Acknowledgment

This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund). We thank anonymous reviewers for their constructive suggestions.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *CVPR*, 2017.

[3] Michael J Black and Padmanabhan Anandan. Robust dynamic motion estimation over time. In *CVPR*, 1991.

[4] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Blind video temporal consistency. *ACM Trans. Graph.*, 34(6):196:1–196:9, Oct. 2015.

[5] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

[6] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 33(3):500–513, 2011.

[7] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[8] Abhishek Kumar Chauhan and Prashant Krishan. Moving object tracking using gaussian mixture model and optical flow. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 2013.

[9] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017.

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[12] David Hafner, Oliver Demetz, and Joachim Weickert. Why is the census transform good for robust optic flow computation? In *International Conference on Scale Space and Variational Methods in Computer Vision*, 2013.

[13] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[14] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018.

[15] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[16] Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *ICCV*, 1999.

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[18] Joel Janai, Fatma Güney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018.

[19] Joel Janai, Fatma Güney, Jonas Wulff, Michael J Black, and Andreas Geiger. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *CVPR*, 2017.

[20] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV*, 2016.

[21] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *arXiv preprint arXiv:1902.06162*, 2019.

[22] Ryan Kennedy and Camillo J Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 364–377. Springer, 2015.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[26] Pengpeng Liu, Irwin King, Michael R. Lyu, and Jia Xu. Ddflow: Learning optical flow with unlabeled data distillation. In *AAAI*, 2019.

[27] D. Maurer and A. Bruhn. Proflow: Learning to predict optical flow. In *BMVC*, 2018.

[28] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[29] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, New Orleans, Louisiana, 2018.

[30] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015.

[31] Michal Neoral, Jan ochman, and Ji Matas. Continual occlusions and optical flow estimation. In *ACCV*, 2018.

[32] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[33] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[34] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[35] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017.

[36] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B. Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *IEEE Winter Conference on Applications of Computer Vision*, 2019.

[37] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[38] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015.

[39] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[40] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[41] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *NIPS*, 2010.

[42] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *arXiv preprint arXiv:1809.05571*, 2018.

[43] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[44] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010.

[45] Sebastian Volz, Andres Bruhn, Levi Valgaerts, and Henning Zimmer. Modeling temporal coherence for optical flow. In *ICCV*, 2011.

[46] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, 2018.

[47] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[48] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. In *CVPR*, 2017.

[49] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *CVPR*, 2017.

[50] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *TPAMI*, 34(9):1744–1757, 2012.

[51] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, 1994.

Supplementary Material

1. Overview

In this supplement, we first show occlusion estimation performance of SelFlow. Then we present screenshots (Nov. 23, 2018) of our submission on the public benchmarks, including MPI Sintel final pass, KITTI 2012, and KITTI 2015.

2. Occlusion Estimation

Following [46, 18, 26], we also report the occlusion estimation performance using F-measure, which is the harmonic mean of precision and recall. We estimate occlusion map using forward-backward consistency check (no parameters to learn).

We compare our occlusion estimation performance with MODOF [50], OccAwareFlow [46], MultiFrameOccFlow-Soft [18] and DDFlow. Note KITTI datasets only have sparse occlusion maps. As shown in Table 1, we achieve the best occlusion estimation performance on Sintel Clean and Sintel Final, and comparable performance on KITTI 2012 and 2015.

Method	Sintel Clean	Sintel Final	KITTI 2012	KITTI 2015
MODOF	–	0.48	–	–
OccAwareFlow	(0.54)	(0.48)	**0.95***	0.88*
MultiFrameOccFlow-Soft	(0.49)	(0.44)	–	**0.91***
DDFlow	**(0.59)**	**(0.52)**	0.94 *	0.86 *
Ours	**(0.59)**	**(0.52)**	**0.95** *	0.88*

Table 1. Comparison of occlusion estimation with F-measure. * marks cases where the occlusion annotation is sparse.

3. Screenshots on Benchmarks

Figure 1 shows the screenshot of our submission on the MPI Sintel benchmark. Our unsupervised entry (CVPR-236) outperforms all the exiting unsupervised learning method, even outperforming supervised methods including FlowNetS+ft+v, FlowNetC+ft+v and SpyNet+ft. At the time of writing, our supervised fine-tuned entry (CVPR-236+ft) is the No. 1 among all submitted methods. In addition to the main ranking metric EPE-all, our method also achieves the best performance on EPE-matched, d10-60, s0-10, s10-40, and very competitive results on remaining metrics. This clearly demonstrates the effectiveness of our method. Figure 2 and Figure 3 show the screenshots of KITTI 2012 and KITTI 2015 benchmark. Again, our unsupervised entry (CVPR-236) outperforms all the exiting unsupervised learning method on both benchmarks. On KITTI 2012, our unsupervised entry (CVPR-236) even outperforms the most recent fully supervised methods including ProFlow, ImpPB+SPCI, Flow-FieldCNN, IntrpNt-df. Our supervised fine-tuned entry (CVPR-236+ft) is the second best compared to published monocular optical flow estimation methods (only second to LiteFlowNet), while achieving better Out-All and Ave-All. On KITTI 2015, our unsupervised entry (CVPR-236) also outperforms several recent supervised methods including DCFlow, ProFlow, FlowFields++ and FlowFieldCNN. Our supervised fine-tuned entry (CVPR-236+ft) is the third best compared to published monocular optical flow estimation methods, only behind the concurrent work MFF, and the extended version of PWC-Net.

Final | Clean

	EPE all	EPE matched	EPE unmatched	d0-10	d10-60	d60-140	s0-10	s10-40	s40+	
GroundTruth [1]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Visualize Results
CVPR-236+ft [2]	4.262	2.040	22.369	4.083	1.715	1.287	0.582	2.343	27.154	Visualize Results
ContinualFlow_ROB [3]	4.528	2.723	19.248	5.050	2.573	1.713	0.872	3.114	26.063	Visualize Results
MFF [4]	4.566	2.216	23.732	4.664	2.017	1.222	0.893	2.902	26.810	Visualize Results
PWC-Net [5]	4.596	2.254	23.696	4.781	2.045	1.234	0.945	2.978	26.620	Visualize Results
CompactFlow_mix [6]	4.691	2.307	24.142	4.327	1.933	1.458	0.863	2.552	28.873	Visualize Results
IRR [7]	4.751	2.278	24.910	4.271	1.919	1.369	0.876	2.541	29.325	Visualize Results
ProgFlow-dcf [8]	4.808	2.192	26.132	4.799	1.988	1.269	1.087	3.224	27.096	Visualize Results
SF_Net [9]	4.860	2.301	25.732	4.121	1.991	1.493	0.812	2.606	30.402	Visualize Results
CompactFlow [10]	4.884	2.246	26.410	4.174	1.861	1.505	0.897	2.588	30.241	Visualize Results
PWC-Net_ROB [11]	4.903	2.454	24.878	4.636	2.090	1.517	0.799	3.029	29.800	Visualize Results
ProFlow_ROB [12]	5.015	2.659	24.192	4.985	2.185	1.771	0.964	2.989	29.987	Visualize Results
ProFlow [13]	5.017	2.596	24.736	5.016	2.146	1.601	0.910	2.809	30.715	Visualize Results
TIMCflow [14]	5.049	2.094	29.134	4.738	1.812	1.221	0.922	3.226	29.926	Visualize Results
LiteFlowNet2 [15]	5.057	2.361	27.057	4.111	2.055	1.569	0.760	2.546	32.471	Visualize Results
DCFlow [16]	5.119	2.283	28.228	4.665	2.108	1.440	1.052	3.434	29.351	Visualize Results
FlowFieldsCNN [17]	5.363	2.303	30.313	4.718	2.020	1.399	1.032	3.065	32.422	Visualize Results
MR-Flow [18]	5.376	2.818	26.235	5.109	2.395	1.755	0.908	3.443	32.221	Visualize Results
LiteFlowNet [19]	5.381	2.419	29.535	4.090	2.097	1.729	0.754	2.747	34.722	Visualize Results
HTC [20]	5.385	2.635	27.808	3.985	2.135	2.019	0.876	2.351	35.104	Visualize Results
F3-MPLF [60]	6.274	3.093	32.210	5.045	2.859	2.082	1.083	3.227	39.409	Visualize Results
EpicFlow [61]	6.285	3.060	32.564	5.205	2.611	2.216	1.135	3.727	38.021	Visualize Results
Devon [62]	6.350	3.234	31.775	5.338	2.878	2.297	1.120	3.834	38.382	Visualize Results
FF++_ROB [63]	6.496	2.990	35.057	5.319	2.540	2.045	1.030	4.182	39.191	Visualize Results
ResPWCR_ROB [64]	6.530	3.849	28.371	5.565	3.396	2.876	1.306	3.848	38.892	Visualize Results
CVPR-236 [65]	6.571	3.119	34.721	5.275	2.834	2.092	1.358	3.883	38.945	Visualize Results
FGI [66]	6.607	3.101	35.158	5.432	2.970	2.131	1.152	3.986	39.985	Visualize Results
FDFlowNet [67]	6.640	3.450	32.625	5.339	3.305	2.562	1.528	3.843	38.744	Visualize Results
TF+OFM [68]	6.727	3.388	33.929	5.544	3.238	2.551	1.512	3.765	39.761	Visualize Results
Deep+R [69]	6.769	2.996	37.494	5.182	2.770	2.064	1.157	3.837	41.687	Visualize Results
PatchBatch-CENT+SD [70]	6.783	3.507	33.498	6.080	3.408	2.103	0.725	3.064	45.858	Visualize Results

1

Figure 1. Screenshot of the Sintel benchmark on November 23th, 2018.

Additional information used by the methods

- Stereo: Method uses left and right (stereo) images
- Multiview: Method uses more than 2 temporally adjacent images
- Motion stereo: Method uses epipolar geometry for computing optical flow
- Additional training data: Use of additional data sources for training (see details)

Error threshold 3 pixels ▾ Evaluation area All pixels ▾

	Method	Setting	Code	Out-Noc	Out-All	Avg-Noc	Avg-All	Density	Runtime	Environment	Compare
1	DM-Net-i2		code	0.00 %	0.00 %	0.0 px	0.0 px	0.00 %	0.90 s	1 core @ 2.5 Ghz (C/C++)	
2	PRSM		code	2.46 %	4.23 %	0.7 px	1.0 px	100.00 %	300 s	1 core @ 2.5 Ghz (Matlab + C/C++)	
	C. Vogel, K. Schindler and S. Roth: 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. ijcv 2015.										
3	HTC			2.55 %	7.84 %	0.8 px	1.6 px	100.00 %	0.03 s	1 core @ 2.5 Ghz (C/C++)	
4	LiteFlowNet2.1			2.72 %	6.30 %	0.7 px	1.4 px	100.00 %	0.0486	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: . .										
5	VC-SF			2.72 %	4.84 %	0.8 px	1.3 px	100.00 %	300 s	1 core @ 2.5 Ghz (Matlab + C/C++)	
	C. Vogel, S. Roth and K. Schindler: View-Consistent 3D Scene Flow Estimation over Multiple Frames. Proceedings of European Conference on Computer Vision. Lecture Notes in, Computer Science 2014.										
6	SPS-StFl			2.82 %	5.61 %	0.8 px	1.3 px	100.00 %	35 s	1 core @ 3.5 Ghz (C/C++)	
	K. Yamaguchi, D. McAllester and R. Urtasun: Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. ECCV 2014.										
7	CompactFlow			2.98 %	6.54 %	0.8 px	1.5 px	100.00 %	0.05 s	Nvidia Tesla V100 (Python)	
8	SF-Net			3.00 %	6.92 %	0.8 px	1.5 px	100.00 %	0.03 s	GPU, GTX 1080Ti	
9	LiteFlowNet2			3.07 %	6.89 %	0.8 px	1.5 px	100.00 %	0.0396 s	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: . .										
10	LiteFlowNet		code	3.27 %	7.27 %	0.8 px	1.6 px	100.00 %	0.0885 s	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.										
11	CVPR-236+ft			3.32 %	6.19 %	0.9 px	1.5 px	100.00 %	0.09 s	NVIDIA GPU	
12	SPS-Fl			3.38 %	10.06 %	0.9 px	2.9 px	100.00 %	11 s	1 core @ 3.5 Ghz (C/C++)	
	K. Yamaguchi, D. McAllester and R. Urtasun: Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. ECCV 2014.										
13	PWC-Net		code	3.41 %	6.82 %	0.8 px	1.5 px	100.00 %	0.03 s	NVIDIA Pascal Titan X	
	D. Sun, X. Yang, M. Liu and J. Kautz: PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. CVPR 2018.										
14	OSF		code	3.47 %	6.34 %	1.0 px	1.5 px	100.00 %	50 min	1 core @ 3.0 Ghz (Matlab + C/C++)	
	M. Menze and A. Geiger: Object Scene Flow for Autonomous Vehicles. Conference on Computer Vision and Pattern Recognition (CVPR) 2015.										
15	PR-Sf+E			3.57 %	7.07 %	0.9 px	1.6 px	100.00 %	200 s	4 cores @ 3.0 Ghz (Matlab + C/C++)	
	C. Vogel, K. Schindler and S. Roth: Piecewise Rigid Scene Flow. International Conference on Computer Vision (ICCV) 2013.										
16	PCBP-Flow			3.64 %	8.28 %	0.9 px	2.2 px	100.00 %	3 min	4 cores @ 2.5 Ghz (Matlab + C/C++)	
	K. Yamaguchi, D. McAllester and R. Urtasun: Robust Monocular Epipolar Flow Estimation. CVPR 2013.										
17	PR-Sceneflow			3.76 %	7.39 %	1.2 px	2.8 px	100.00 %	150 sec	4 core @ 3.0 Ghz (Matlab + C/C++)	
	C. Vogel, K. Schindler and S. Roth: Piecewise Rigid Scene Flow. International Conference on Computer Vision (ICCV) 2013.										
18	SDF			3.80 %	7.69 %	1.0 px	2.3 px	100.00 %	TBA s	1 core @ 2.5 Ghz (C/C++)	
	M. Bai*, W. Luo*, K. Kundu and R. Urtasun: Exploiting Semantic Information and Deep Matching for Optical Flow. ECCV 2016.										
19	MotionSLIC			3.91 %	10.56 %	0.9 px	2.7 px	100.00 %	11 s	1 core @ 3.0 Ghz (C/C++)	
	K. Yamaguchi, D. McAllester and R. Urtasun: Robust Monocular Epipolar Flow Estimation. CVPR 2013.										
20	AugmentedFlowNetDCSS			3.97 %	7.48 %	0.9 px	1.5 px	100.00 %	0.07 s	GPU @ 2.5 Ghz (C/C++)	
21	SfM-PM			4.02 %	6.15 %	1.0 px	1.5 px	100.00 %	69 s	3 cores @ 3.6 Ghz (C/C++)	
	D. Maurer, N. Marniok, B. Goldluecke and A. Bruhn: Structure-from-Motion-Aware PatchMatch for Adaptive Optical Flow Estimation. ECCV 2018.										
22	MFF			4.19 %	7.87 %	0.9 px	1.7 px	100.00 %	0.05 s	1 core @ 2.5 Ghz (C/C++)	
23	UnFlow		code	4.28 %	8.42 %	0.9 px	1.7 px	100.00 %	0.12 s	GPU @ 1.5 Ghz (Python + C/C++)	
	S. Meister, J. Hur and S. Roth: UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. AAAI 2018.										
24	CVPR-236			4.31 %	7.68 %	1.0 px	2.2 px	100.00 %	0.09 s	GPU @ 2.5 Ghz (Python)	
25	MirrorFlow		code	4.38 %	8.20 %	1.2 px	2.6 px	100.00 %	11 min	4 core @ 2.2 Ghz (C/C++)	
	J. Hur and S. Roth: MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation. ICCV 2017.										
26	ProFlow			4.49 %	7.88 %	1.1 px	2.1 px	100.00 %	112 s	GPU+CPU @ 3.6 Ghz (Python + C/C++)	
	D. Maurer and A. Bruhn: ProFlow: Learning to Predict Optical Flow. BMVC 2018.										

Figure 2. Screenshot of the KITTI 2012 benchmark on November 23th, 2018.

Evaluation ground truth: All pixels ▾ Evaluation area: All pixels ▾

	Method	Setting	Code	Fl-bg	Fl-fg	Fl-all	Density	Runtime	Environment	Compare
7	SPOSF			5.41 %	15.96 %	7.16 %	100.00 %	10 min	1 core @ 3.5 Ghz (Matlab + C/C++)	
8	MFF			7.15 %	7.25 %	7.17 %	100.00 %	0.05 s	NVIDIA Pascal Titan X (Python)	
	Z. Ren, O. Gallo, D. Sun, M. Yang, E. Sudderth and J. Kautz: A Fusion Approach for Multi-Frame Optical Flow Estimation. IEEE Winter Conference on Applications of Computer Vision 2019.									
9	OSF 2018		code	5.38 %	17.61 %	7.41 %	100.00 %	390 s	1 core @ 2.5 Ghz (Matlab + C/C++)	
	M. Menze, C. Heipke and A. Geiger: Object Scene Flow. ISPRS Journal of Photogrammetry and Remote Sensing (JPRS) 2018.									
10	CompactFlow			6.94 %	11.28 %	7.66 %	100.00 %	0.05 s	Nvidia Tesla V100 (Python)	
11	LiteFlowNet2.1			7.85 %	**7.20 %**	7.74 %	100.00 %	0.0486	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: . .									
12	OSF		code	5.62 %	18.92 %	7.83 %	100.00 %	50 min	1 core @ 2.5 Ghz (C/C++)	
	M. Menze and A. Geiger: Object Scene Flow for Autonomous Vehicles. Conference on Computer Vision and Pattern Recognition (CVPR) 2015.									
13	PWC-Net		code	7.87 %	8.03 %	7.90 %	100.00 %	0.03 s	NVIDIA Pascal Titan X	
	D. Sun, X. Yang, M. Liu and J. Kautz: PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. CVPR 2018.									
14	CompactFlow_mix			7.37 %	13.45 %	8.38 %	100.00 %	0.05 s	GPU @ 2.5 Ghz (Python)	
15	CVPR-236+ft			7.61 %	12.48 %	8.42 %	100.00 %	0.09 s	GPU @ 2.5 Ghz (Python)	
16	AugmentedFlowNetDCSS			8.36 %	9.62 %	8.57 %	100.00 %	0.07 s	GPU @ 2.5 Ghz (C/C++)	
17	LiteFlowNet2			8.82 %	7.73 %	8.64 %	100.00 %	0.0396 s	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: . .									
18	CVPR 2597			8.82 %	8.53 %	8.77 %	100.00 %			
19	SF-Net			8.80 %	10.10 %	9.01 %	100.00 %	0.21 s	GPU, GTX 1080Ti	
20	LiteFlowNet		code	9.66 %	7.99 %	9.38 %	100.00 %	0.0885 s	NVIDIA GTX 1080	
	T. Hui, X. Tang and C. Loy: LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.									
21	HTC			9.15 %	12.76 %	9.75 %	100.00 %	0.03 s	1 core @ 2.5 Ghz (C/C++)	
22	ContinualFlow_ROB			8.54 %	17.48 %	10.03 %	100.00 %	0.15 s	GPU - NVidia 1080Ti	
	M. Neoral, J. Šochman and J. Matas: Continual Occlusions and Optical Flow Estimation. 14th Asian Conference on Computer Vision (ACCV) 2018.									
23	MirrorFlow		code	8.93 %	17.07 %	10.29 %	100.00 %	11 min	4 core @ 2.2 Ghz (C/C++)	
	J. Hur and S. Roth: MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation. ICCV 2017.									
24	RIMM-SF			9.68 %	16.18 %	10.76 %	100.00 %	150 s	4 cores @ 3.5 Ghz (C/C++)	
25	SS-SF			8.17 %	25.20 %	11.00 %	100.00 %	3 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
26	SDF			8.61 %	23.01 %	11.01 %	100.00 %	TBA	1 core @ 2.5 Ghz (C/C++)	
	M. Bai*, W. Luo*, K. Kundu and R. Urtasun: Exploiting Semantic Information and Deep Matching for Optical Flow. ECCV 2016.									
27	LFNet_ROB			11.18 %	10.20 %	11.01 %	100.00 %	0.0985 s	NVIDIA 1080 (Python + C/C++)	
28	UnFlow		code	10.15 %	15.93 %	11.11 %	100.00 %	0.12 s	GPU @ 1.5 Ghz (Python + C/C++)	
	S. Meister, J. Hur and S. Roth: UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. AAAI 2018.									
29	FSF+MS			8.48 %	25.43 %	11.30 %	100.00 %	2.7 s	4 cores @ 3.5 Ghz (C/C++)	
	T. Taniai, S. Sinha and Y. Sato: Fast Multi-frame Stereo Scene Flow with Motion Segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017) 2017.									
30	CNNF+PMBP			10.08 %	18.56 %	11.49 %	100.00 %	45 min	1 cores @ 3.5 Ghz (C/C++)	
	F. Zhang and B. Wah: Fundamental Principles on Learning New Features for Effective Dense Matching. IEEE Transactions on Image Processing 2018.									
31	PWC-Net_ROB		code	11.22 %	13.69 %	11.63 %	100.00 %	0.03 s	NVIDIA Pascal Titan X	
	D. Sun, X. Yang, M. Liu and J. Kautz: PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. CVPR 2018.									
32	SfM-PM			9.66 %	22.73 %	11.83 %	100.00 %	69 s	3 cores @ 3.6 Ghz (C/C++)	
	D. Maurer, N. Marniok, B. Goldluecke and A. Bruhn: Structure-from-Motion-Aware PatchMatch for Adaptive Optical Flow Estimation. ECCV 2018.									
33	MR-Flow		code	10.13 %	22.51 %	12.19 %	100.00 %	8 min	1 core @ 2.5 Ghz (Python + C/C++)	
	J. Wulff, L. Sevilla-Lara and M. Black: Optical Flow in Mostly Rigid Scenes. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2017.									
34	semflow			11.49 %	18.12 %	12.60 %	100.00 %	10 s	6 cores @ 2.5 Ghz (C/C++)	
35	Mono-SF			11.40 %	19.64 %	12.77 %	100.00 %	41 s	1 core @ 3.5 Ghz (Matlab + C/C++)	
36	SceneFFields			10.58 %	24.41 %	12.88 %	100.00 %	65 s	4 cores @ 3.7 Ghz (C/C++)	
	R. Schuster, O. Wasenmüller, G. Kuschk, C. Bailer and D. Stricker: SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences. IEEE Winter Conference on Applications of Computer Vision (WACV) 2018.									
37	CSF			10.40 %	25.78 %	12.96 %	100.00 %	80 s	1 core @ 2.5 Ghz (C/C++)	
	Z. Lv, C. Beall, P. Alcantarilla, F. Li, Z. Kira and F. Dellaert: A Continuous Optimization Approach for Efficient and Accurate Scene Flow. European Conf. on Computer Vision (ECCV) 2016.									
38	PR-Sceneflow		code	11.73 %	24.33 %	13.83 %	100.00 %	150 s	4 core @ 3.0 Ghz (Matlab + C/C++)	
	C. Vogel, K. Schindler and S. Roth: Piecewise Rigid Scene Flow. ICCV 2013.									
39	CVPR-236			12.68 %	21.74 %	14.19 %	100.00 %	0.09 s	GPU @ 2.5 Ghz (Python)	

Figure 3. Screenshot of the KITTI 2015 benchmark on November 23th, 2018.