PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume
SUPPLEMENTARY MATERIAL

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz
NVIDIA

Section 1 provides more ablation and visual results. Section 2 summarizes the details of our network. Section 3 shows the screenshot of the MPI Sintel final pass, KITTI 2012, and KITTI 2015 public tables at the time of submission (November 15th, 2017). Section 4 shows the learned features at the first level of the feature pyramid extractor.

1. More Ablation and Visual Results

Figure 1 shows the enlarged images of Figure 1 in the main manuscript. PWC-Net outperforms all published methods on the MPI Sintel final pass benchmark in both accuracy and running time. It also reaches the best balance between size and accuracy among existing end-to-end CNN models.

Table 1 shows more ablation results, in particular, the full results for models trained on FlyingChairs (Table 1a) and then fine-tuned on FlyingThings3D (Table 1b). To further test the dilated convolutions, we replace the dilated convolutions of the context network with plain convolutions. Using plain convolutions has worse performance on Chairs and Sintel, and is slightly better on KITTI. We also have independent runs of the same PWC-Net that only differ in the random initialization. As shown in Table 1d, the two independent runs lead to models that have close performances, although not exactly the same.

Figures 2 and 3 provide more visual results by PWC-Net on the MPI Sintel final pass and KITTI 2015 test sets. PWC-Net can recover sharp motion boundaries in the presence of large motion, severe occlusions, and strong shadow and atmospheric effects. However, PWC-Net tends to produce errors on objects with thin structures that rarely occur in the training set, such as the wheels of the bicycle in the third row of Figure 3.

2. Network Details

Figure 4 shows the architecture for the 7-level feature pyramid extractor network used in our experiment. Note that the bottom level consists of the original input images. Figure 5 shows the optical flow estimator network at pyramid level 2. The optical flow estimator networks at other levels have the same structure except for the top level, which does not have the upsampled optical flow and directly computes cost volume using features of the first and second images. Figure 6 shows the context network that is adopted only at pyramid level 2.

3. Screenshots of MPI Sintel and KITTI Public Table

Figures 7-9 respectively show the screenshots of the MPI Sintel final pass, KITTI 2015, and KITTI 2012 public tables at the time of submission (November 15th, 2017). Among all optical flow methods, PWC-Net is ranked 1st on both MPI Sintel final and KITTI 2015, and 2nd on KITTI 2012. Note that the 1st-ranked method on KITTI 2012, SDF [1], assumes a rigidity constraint for the background, which is well-suited to the static scenes in KITTI 2012. PWC-Net performs better than SDF on KITTI 2015 that contains dynamic objects and is more challenging.

4. Learned Features

Figure 10 shows the learned filters for the first convolution layer by PWC-Net and the feature responses to an input image. These filters tend to focus on regions of different properties in the input image. After training on FlyingChairs, fine-tuning on FlyingThings3D and Sintel does not change these filters much.

References

[1] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 7

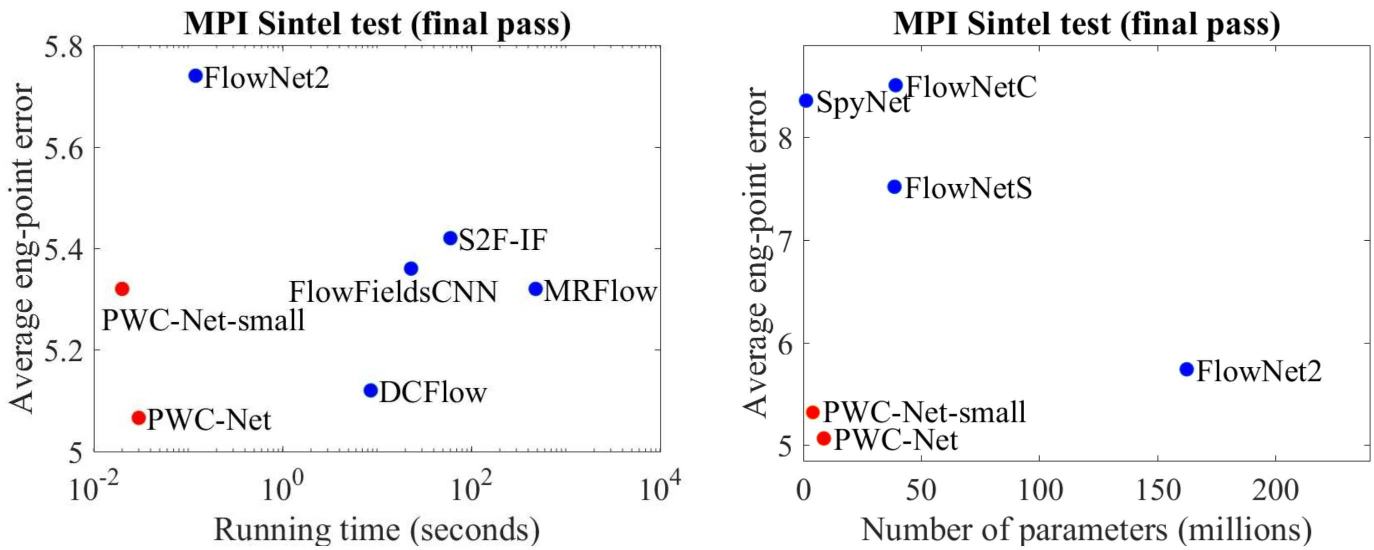


Figure 1. Left: PWC-Net outperforms all published methods on the MPI Sintel final pass benchmark in both accuracy and running time. Right: PWC-Net reaches the best balance between size and accuracy among existing end-to-end CNN models.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	**2.00**	3.33	4.59	5.14	28.67%	13.20	41.79%
No context	2.06	**3.09**	**4.37**	**4.77**	**25.35**%	**12.03**	**39.21**%
No DenseNet	2.23	3.47	4.74	5.63	28.53%	14.02	40.33%
Neither	2.22	3.15	4.49	5.46	28.02%	13.14	40.03%

(a) **Trained on FlyingChairs.**

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	**2.30**	**2.55**	**3.93**	**4.14**	**21.38**%	**10.35**	**33.67**%
No context	2.48	2.82	4.09	4.39	21.91%	10.82	34.44%
No DenseNet	2.54	2.72	4.09	4.91	24.04%	11.52	34.79%
Neither	2.65	2.83	4.24	4.89	24.52%	12.01	35.73%

(b) **Fine-tuend on FlyingThings3D after FlyingChairs.**

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Dilated conv	**2.00**	**3.33**	**4.59**	**5.14**	28.67%	13.20	41.79%
Plain conv	2.03	3.39	4.85	5.29	**25.86**%	**13.17**	**38.67**%

(c) **Dilated vs plain convolutions** for the context network.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Run 1	2.00	3.33	**4.59**	5.14	28.67%	13.20	41.79%
Run 2	**2.00**	**3.33**	4.65	**4.81**	**27.12**%	**13.10**	**40.84**%

(d) **Two independent runs** result in slightly different models.

Table 1. **More ablation experiments**. Unless explicitly stated, the models have been trained on the FlyingChairs dataset.

Figure 2. More PWC-Net results on the MPI Sintel final pass dataset.

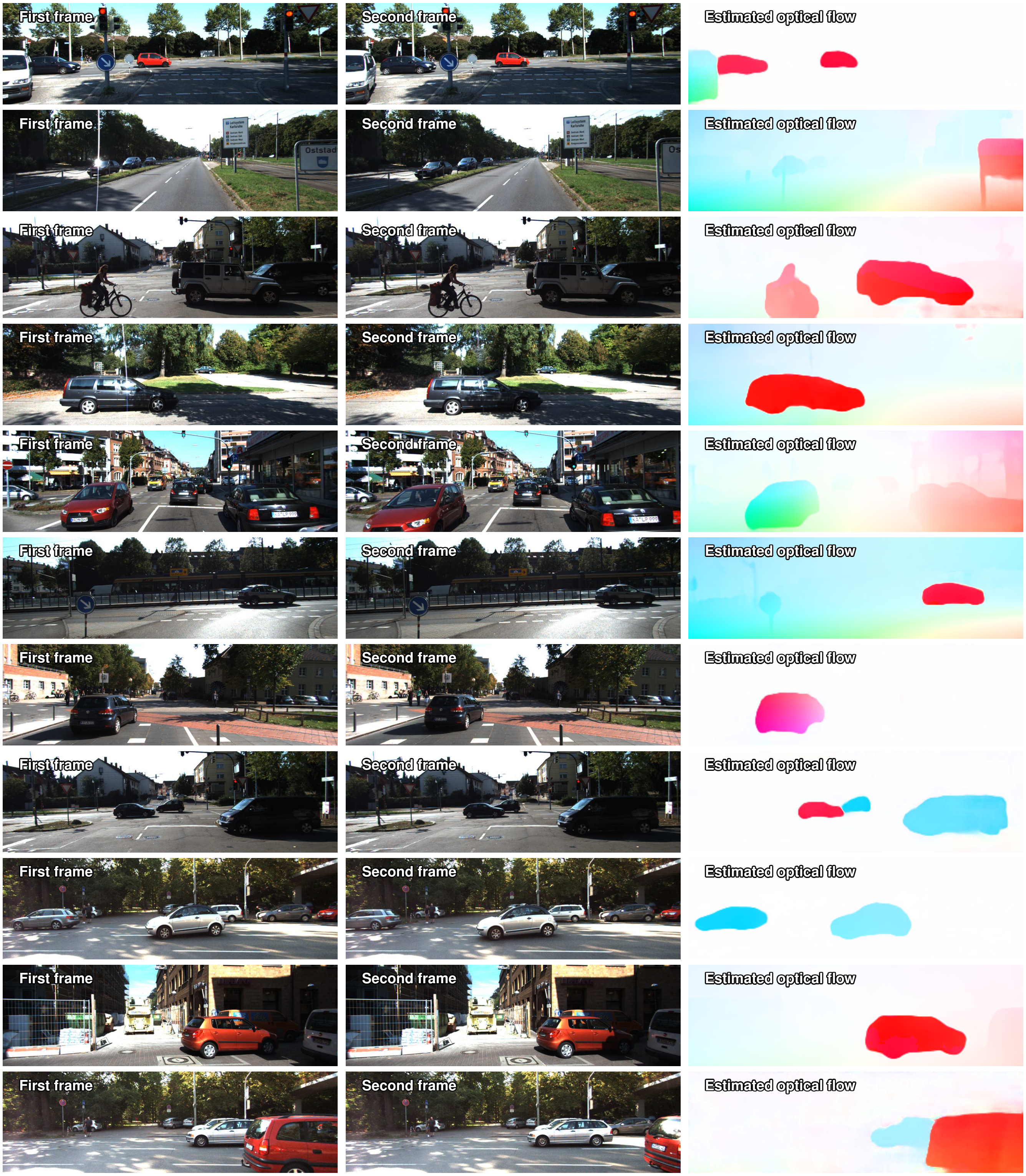


Figure 3. More PWC-Net results on KITTI 2015 test set. PWC-Net can recover sharp motion boundaries despite large motion, strong shadows, and severe occlusions. Thin structures, such as the bicycle, are challenging to PWC-Net, probably because the training set has no training samples of bicycles.

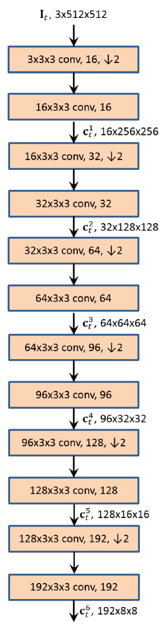


Figure 4. The feature pyramid extractor network. The first image ($t = 1$) and the second image ($t = 2$) are encoded using the same Siamese network. Each convolution is followed by a leaky ReLU unit. The convolutional layer and the $\times 2$ downsampling layer at each level is implemented using a single convolutional layer with a stride of 2. $\mathbf{c}_t^l$ denotes extracted features of image t at level l;

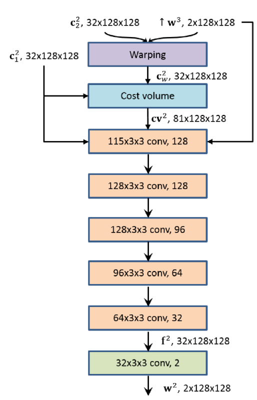


Figure 5. The optical flow estimator network at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow.

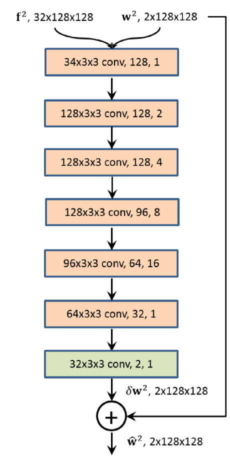


Figure 6. The context network at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow. The last number in each convolutional layer denotes the dilation constant.

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
PWC-Net [2]	5.042	2.445	26.221	4.636	2.087	1.475	0.799	2.986	31.070	Visualize Results
DCFlow [3]	5.119	2.283	28.228	4.665	2.108	1.440	1.052	3.434	29.351	Visualize Results
FlowFieldsCNN [4]	5.363	2.303	30.313	4.718	2.020	1.399	1.032	3.065	32.422	Visualize Results
MR-Flow [5]	5.376	2.818	26.235	5.109	2.395	1.755	0.908	3.443	32.221	Visualize Results
FTFlow [6]	5.390	2.268	30.841	4.513	1.964	1.366	1.046	3.322	31.936	Visualize Results
S2F-IF [7]	5.417	2.549	28.795	4.745	2.198	1.712	1.157	3.468	31.262	Visualize Results
InterpoNet_ff [8]	5.535	2.372	31.296	4.720	2.018	1.532	1.064	3.496	32.633	Visualize Results
PGM-C [9]	5.591	2.672	29.389	4.975	2.340	1.791	1.057	3.421	33.339	Visualize Results
RicFlow [10]	5.620	2.765	28.907	5.146	2.366	1.679	1.088	3.364	33.573	Visualize Results
InterpoNet_cpm [11]	5.627	2.594	30.344	4.975	2.213	1.640	1.042	3.575	33.321	Visualize Results

Figure 7. Screenshot of the MPI Sintel final pass public table. PWC-Net has the lowest average end-point error (EPE) among all evaluated methods as of November 15th, 2017.

Evaluation ground truth: All pixels   Evaluation area: All pixels

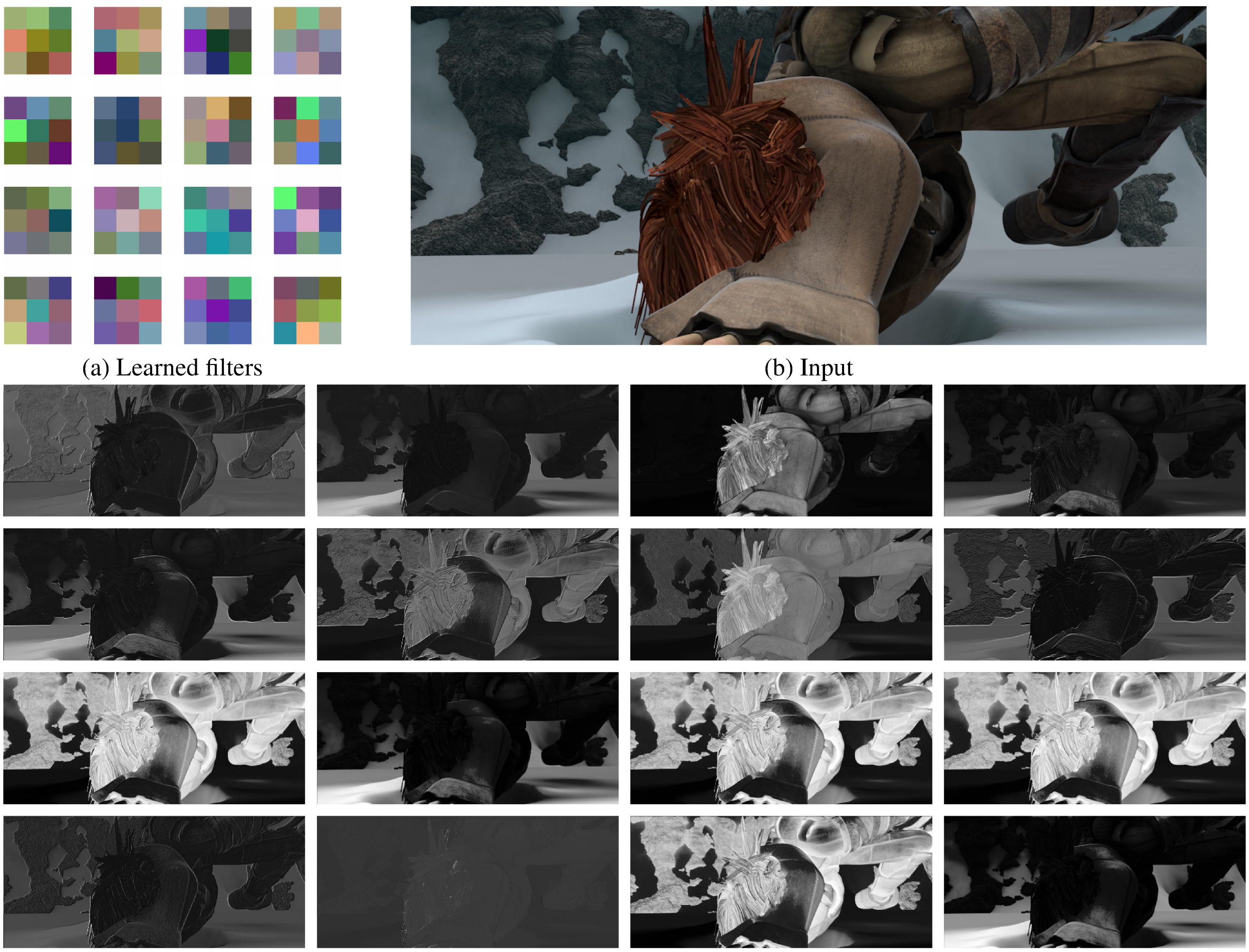
	Method	Setting	Code	Fl-bg	Fl-fg	Fl-all	Density	Runtime	Environment	Compare
1	PSPO			**4.35 %**	15.21 %	**6.15 %**	100.00 %	5 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
2	ISF			5.40 %	10.29 %	6.22 %	100.00 %	10 min	1 core @ 3 Ghz (C/C++)	
	A. Behl, O. Jafari, S. Mustikovela, H. Alhaija, C. Rother and A. Geiger: Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?. International Conference on Computer Vision (ICCV) 2017.									
3	PRSM		code	5.33 %	13.40 %	6.68 %	100.00 %	300 s	1 core @ 2.5 Ghz (C/C++)	
	C. Vogel, K. Schindler and S. Roth: 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. ijcv 2015.									
4	OSF+TC			5.76 %	13.31 %	7.02 %	100.00 %	50 min	1 core @ 2.5 Ghz (C/C++)	
	M. Neoral and J. Šochman: Object Scene Flow with Temporal Consistency. 22nd Computer Vision Winter Workshop (CVWW) 2017.									
5	SSF			5.63 %	14.71 %	7.14 %	100.00 %	5 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
	Z. Ren, D. Sun, J. Kautz and E. Sudderth: Cascaded Scene Flow Prediction using Semantic Segmentation. International Conference on 3D Vision (3DV) 2017.									
6	SOSF			5.42 %	17.24 %	7.39 %	100.00 %	55 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
7	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.									
8	PWC-Net			9.66 %	9.31 %	9.60 %	100.00 %	0.03 s	NVIDIA Pascal Titan X	
9	MirrorFlow			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.									
10	FlowNet2			10.75 %	**8.75 %**	10.41 %	100.00 %	0.12 s	GPU Nvidia GeForce GTX 1080	
11	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.									
12	UnFlow			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.									
13	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.									
14	CNNF+PMBP			10.08 %	18.56 %	11.49 %	100.00 %	45 min	1 cores @ 3.5 Ghz (C/C++)	
15	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.									

Figure 8. Screenshot of the KITTI 2015 public table. PWC-Net has the lowest percentage of error (Fl-all) among all optical flow methods, only inferior to scene flow methods that use additional stereo input information.

Error threshold 3 pixels ▼ Evaluation area All pixels ▼

	Method	Setting	Code	Out-Noc	Out-All	Avg-Noc	Avg-All	Density	Runtime	Environment	Compare
1	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.											
2	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.											
3	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.											
4	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.											
5	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.											
6	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.											
7	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.											
8	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.											
9	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.											
10	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.											
11	PWC-Net			4.22 %	8.10 %	0.9 px	1.7 px	100.00 %	0.03 s	NVIDIA Pascal Titan X	
12	TBR			4.24 %	7.50 %	0.9 px	1.5 px	100.00 %	1750 s	4 cores @ 2.5 Ghz (Matlab + C/C++)	
13	UnFlow			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.											
14	MirrorFlow			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.											

Figure 9. Screenshot of the KITTI 2012 public table. SDF [1] is the only optical flow method that has lower percentage of outliers in non-occluded regions (Out-Noc) than PWC-Net. However, SDF assumes a rigidity constraint for the background, which is well-suited for the static scenes in the KITTI 2012 set.



(a) Learned filters (b) Input

(c) Feature responses

Figure 10. Learned filters at the first convolutional layer of PWC-Net and the filter responses to an input image.