Deep Two-View Structure-from-Motion Revisited
– *Supplemental Material* –

[*]Jianyuan Wang[1], [*]Yiran Zhong[1], Yuchao Dai[2], Stan Birchfield[3],
Kaihao Zhang[1], Nikolai Smolyanskiy[3], Hongdong Li[1]
[1]Australian National University, [2]Northwestern Polytechnical University, [3]NVIDIA

In this supplemental material, we provide additional implementation details of our framework as well as more details of the Eigen SfM Split. We show more qualitative and quantitative results of full sequence odometry and depth estimation on the KITTI VO dataset. We also attach a video of our depth results of the KITTI VO sequences. We implement our framework on an autonomous driving car and provide some qualitative results of the depth estimation in open-world scenarios

1. Additional Implementation Details

1.1. Network and Hyper-parameter Selection

Our framework consists of two matching modules: an optical flow module and a depth estimation module. We select the current off-the-shelf state-of-the-art network, DICL-Flow [10], as our optical flow module and the DPSNet [5] with our proposed scale-invariant modification as our depth estimation module.

We use the SIFT keypoint locations to mask the optical flow before feeding into the RANSAC. For the sake of completeness, we also provide the results of using different keypoint detectors in Table 1, including SURF [1] and FAST [8]. While all of them achieve state-of-the-art performance in all three datasets, the SIFT keypoints have the overall best performance.

For essential matrix estimation, we use 512 threads on each GPU to compute essential matrix hypotheses. Each GPU thread randomly selects 5 points from the masked matching points and estimates an essential matrix hypothesis using the 5-point algorithm [6]. We choose the hypothesis with the most inliers using the RANSAC scheme. We empirically set the RANSAC inlier error threshold $\tau = 0.0001$ and set a maximum iteration $\theta = 20$ for MVS, Scenes11 and SUN3D datasets, and $\theta = 5$ for the KITTI dataset.

[*] indicates equal contribution, listed in alphabetical order. Yiran is the corresponding author. Work was partially done when Yiran was an intern at NVIDIA, Redmond, WA.

Table 1. **Various Keypoint Detection Methods for Optical Flow Masking.** We keep the predicted optical flow the same while using the keypoint locations of FAST, SURF, and SIFT to mask the flow correspondences.

Model	MVS		Scenes11		Sun3D	
	Rot	Tran	Rot	Tran	Rot	Tran
DeepSFM [11]	2.824	9.881	0.403	5.828	1.704	13.107
Our-FAST	1.832	3.849	0.327	**1.492**	1.489	11.430
Our-SURF	**1.744**	**3.643**	0.372	1.501	1.587	12.226
Our-SIFT	2.417	3.878	**0.276**	2.041	**1.391**	**10.757**

For the depth estimation module, we set the number of matching candidates $L = 96$ for KITTI, MVS, Scenes11, and SUN3D datasets. We use a normalized minimum depth $d_{\text{min}} = 1.0$ for the KITTI dataset and $d_{\text{min}} = 0.5$ for MVS, Scenes11, and SUN3D datasets.

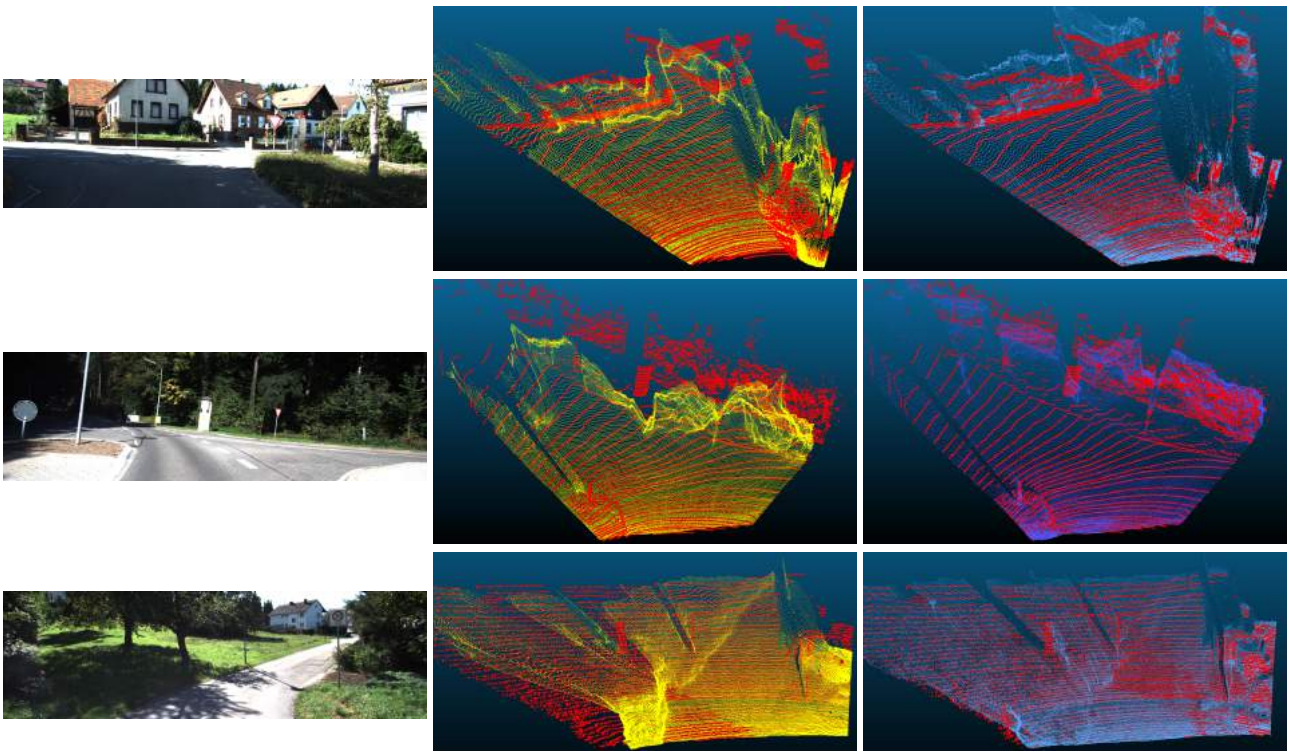
1.2. Training

We implement our framework in PyTorch with Automatic Mixed Precision[1]. For KITTI dataset, we use a crop size of $[256, 768]$ and a batch size of 32, and train the network for 10 epochs. The initial learning rate is set to 0.0005 and dropped by half at 3 and 8 epochs. The total training time is 40 hours on eight NVIDIA Tesla V100 GPUs. For the MVS, Scenes11 and SUN3D datasets, we leverage the same training protocol as provided in [11]. The crop size is set to $[256, 384]$ and the batch size is set to 64. We jointly train our network on all three datasets for 10 epochs and use the same initial learning rate of 0.0005 and drop by half at the fifth epoch. The total training time for these three datasets is 85 hours. Our data augmentation includes random flipping, color jittering, resizing, and cropping.

1.3. Processing time

We provide the time cost of our method in Table 2 on a NVIDIA V100 GPU. The depth estimator module occupies more than over 71% of the total processing time. However, we can easily swap the DPSNet [5] architecture to any other

[1]Code will be released



Reference Image Monodepth2-sup [4] Ours

Figure 1. **Point Cloud Comparison.** Red: LiDAR; Yellow: Monodepth2 [4] with ground truth supervision; Blue: Ours. Our point clouds are more aligned with the ground truth LiDAR points while the Monodepth2-sup often wrongly estimate depth for grass and trees.

Table 2. **Timing of Pipeline Components.** We provide the average processing time of each component on a pair of 1242×375 images.

	Flow	SIFT	RANSAC	Depth	Total
Time (s)	0.09	0.01	0.15	0.62	0.87

light-weighted plane sweep based MVS networks for faster processing time.

2. More KITTI VO Results

2.1. Full Sequence Odometry

We compare our visual odometry results on the 9^{th} and 10^{th} sequences of KITTI VO dataset with the state-of-the-art SfM methods in Table 3 using common visual odometry evaluation criteria. We also adopt the metric absolute trajectory error (ATE) on full sequence and relative pose error (RPE) in meters and degrees. For all results, we align the predicted trajectories to the ground truth via least square optimization [9]. Our approach performs better than all other methods over these five metrics.

2.2. Depth Estimation

To compare the depth estimation results on KITTI VO dataset, we use the state-of-the-art monocular depth estimation network Monodepth2 [4] as our main competitor.

For a fair comparison, we re-train both our method and the Monodepth2 [4] with ground truth depth supervision on the first 9 sequences (Seq 00 to Seq 08) of the KITTI VO dataset. We add a suffix "-sup" to Monodepth2 to denote that the model is trained with ground truth depth. We report their depth estimation accuracy on the 9^{th} and 10^{th} sequences. As shown in Table 4, our method reduces the error rate of D1_all metric for more than 73%, comparing with the Monodepth2-sup [4]. We provide a point cloud comparison in Fig. 1, where our point clouds are more aligned with the ground truth LiDAR points while the Monodepth2-sup often wrongly estimate depth for grass and trees. We also attach a video of our depth results on the Seq.09 and Seq.10 of the KITTI VO dataset. In the video, we convert the depth maps to disparity maps for better visualization.

2.3. Challenging Case.

We add a qualitative example in Fig. 2 to support the claim that compared with conventional algorithms, deep learning can handle challenging cases better. The example contains large textureless areas. The state-of-the-art classical SfM method COLMAP fails in these areas while ours succeeds.

Table 3. **Full Sequence Visual Odometry on KITTI VO.** Note that our network is trained on synthetic datasets and compute camera poses from only two consecutive frames while other methods are fine-tuned on the KITTI VO dataset and take multiple frames to estimate the camera poses. The ATE here measures the root-mean-square error over a full sequence while the ATE in the Table 3 of main paper evaluates over each five frames. Bold indicates the best.

Method	Seq 09			Seq 10		
	ATE	RPE (m)	RPE (°)	ATE	RPE (m)	RPE (°)
SfMLearner [12]	24.31	0.099	0.140	20.87	0.120	0.154
SC-SfMLearner [2]	15.02	0.095	0.102	20.19	0.105	0.107
CCNet [7]	29.00	0.095	0.088	13.77	0.097	0.116
Ours	**6.87**	**0.016**	**0.034**	**2.26**	**0.010**	**0.040**

Table 4. **Quantitative Results on KITTI VO dataset.** The monodepth2-sup model was trained with ground truth depth maps.

Method	Abs Err (m)	Abs Rel	Sq Rel	RMSE (m)	RMSE_{log}	D1-all	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2-sup [4]	1.7958	0.0935	0.4842	3.6140	0.1478	26.1221	0.8933	0.9751	0.9938
Ours	**0.9294**	**0.0442**	**0.1618**	**2.2164**	**0.0789**	**7.0159**	**0.9766**	**0.9948**	**0.9981**

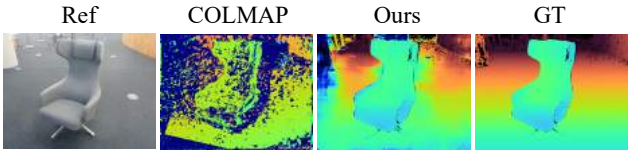


Figure 2. **Challenging Case** which contains large textureless areas, *e.g.*,, the carpet.

3. Additional Open World Qualitative Result

We implement our framework in an autonomous driving car and test it in open world scenarios. As shown in Fig. 3, our framework is able to recover visually convincing depth maps, which demonstrates the generalization ability of our framework.

Figure 3. **Depth Estimation on Open World Scenarios.**

4. More Dataset Details

4.1. Eigen SfM Split

Eigen Split [3] is primarily designed for evaluating monocular depth estimation, which does not take camera motions and dynamic objects into account. For the SfM task, the close to static camera motions and dynamic objects will lead to ill-posed situations. To better evaluate the performance of SfM algorithms on Eigen Split, we build the Eigen SfM Split that mostly satisfies two-view SfM assumptions. Specifically, we first pair each frame with its next frame then manually remove these pairs with small relative translations (less than 0.5 meters) or contain large dynamic objects[2]. We use the remaining 256 frames to construct our Eigen SfM Split.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 1

[2] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in neural information processing systems*, pages 35–45, 2019. 3

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 3

[4] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *Int. Conf. Comput. Vis.*, October 2019. 2, 3

[5] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: End-to-end deep plane sweep stereo. *International Conference on Learning Representations*, 2019. 1

[6] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004. 1

[7] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2019. 3

[8] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005. 1

[2]We define a dynamic object which occupies more than 20% pixels of a scene as a large dynamic object.

[9] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 376–380. IEEE, 1991. 2

[10] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-invariant matching cost learning for accurate optical flow estimation. In *Adv. Neural Inform. Process. Syst.*, 2020. 1

[11] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. *Eur. Conf. Comput. Vis.*, 2020. 1

[12] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017. 3