Self-Supervised Deep Visual Odometry with Online Adaptation — Supplementary Material —

Anonymous CVPR submission

Paper ID 2089

1. Network Architecture

The detailed network architectures of DepthNet, PoseNet and MaskNet are shown in Table 1, 2, 3. All convolutions, deconvolutions and convLSTM are followed by layer normalization (LN) and ReLU activation except for the output layer.

2. Analysis of online adaptation for VO

The problem of online adaptation for VO is very hard, especially when we require that the estimated trajectory in the unseen environment should be close to the ground truth.

Despite some recent works on stereo matching and stereo depth estimation [5, 4, 3] claim that they are able to get reasonable matching and depth in unseen environments after hundreds of online refinement process, these methods use stereo images and the baselines between two cameras are known, which significantly simplifies the online learning problem. In contrast, our method uses only monocular images and the relative pose between consecutive images are unknown. The network needs to adapt online for better pose and depth estimation simultaneously, which is much harder than stereo matching or depth estimation.

Besides, even if the VO network adapts to the new environment after hundreds of updating (maybe 500-2000 frames), previous estimations still have large errors. As the errors are accumulated, the trajectories are getting more and more drifted with time (shown in Fig. 2). This indicates that in order to get a better trajectory, the estimation error should be small at the *beginning* of online adaptation, and the network has to adapt itself as *fast* as possible so as to reduce accumulated error. It can be seen from Fig. 2 that our method achieves promising online adaptation performance compared with other baselines. We select some complicated and challenging trajectories from KITTI [2] 00-10, including multiple turns, loops and sharp turns. If the pose estimation goes wrong at one turn, the trajectory will drift afterwards, no matter how accurate it estimates later. These results demonstrate that our method is able to get small estimation error at the beginning of domain shift, and adapts quickly to the new environment.

3. Pretrain on Carla

We use Carla [1] simulator to create virtual images under different conditions in the virtual city (shown in Fig. 1). We use these collected images to pretrain our network together with other baselines, and test them on KITTI odometry [2] dataset. The results are shown in Fig. 2.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 3

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 1

[3] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Unsupervised adaptation for deep stereo. In *ICCV*, 2017. 1

[4] Alessio Tonioni, Oscar Rahnama, Thomas Joy, Luigi Di Stefano, Thalaiyasingam Ajanthan, and Philip HS Torr. Learning to Adapt for Stereo. In *CVPR*, 2019. 1

[5] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-Time Self-Adaptive Deep Stereo. In *CVPR*, 2019. 1

[6] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*, 2017. 2

Block	Operation	Filter size	Stride	Output size (height×width×channel)
Input I				128×416×3
Encoder	Convolution	7×7	2	64×208×32
	Convolution	7×7	1	64×208×32
	Convolution	5×5	2	32×104×64
	Convolution	5×5	1	32×104×64
	Convolution	3×3	2	16×52×128
	Convolution	3×3	1	16×52×128
	Convolution	3×3	2	8×26×256
	Convolution	3×3	1	8×26×256
	ConvLSTM	3×3	1	8×26×256
	Convolution	3×3	2	4×13×256
	Convolution	3×3	1	4×13×256
	ConvLSTM	3×3	1	4×13×256
	Convolution	3×3	2	2×7×512
	Convolution	3×3	1	2×7×512
	ConvLSTM	3×3	1	2×7×512
Decoder	Deconvolution	3×3	2	4×13×512
	Concatenation	-	-	4×13×768
	Convolution	3×3	1	4×13×512
	Deconvolution	3×3	2	8×26×256
	Concatenation	-	-	8×26×512
	Convolution	3×3	1	8×26×256
	Deconvolution	3×3	2	16×52×128
	Concatenation	-	-	16×52×256
	Convolution	3×3	1	16×52×128
	Convolution	3×3	1	16×52×1
	Sigmoid (get disparity d_1)	-	-	16×52×1
	$d_1 = 100 \times d_1 + 0.01$	-	-	16×52×1
	Deconvolution	3×3	2	32×104×64
	Bilinear upsampling from d_1	-	2	32×104×1
	Concatenation	-	-	32×104×129
	Convolution	3×3	1	32×104×64
	Convolution	3×3	1	32×104×1
	Sigmoid (get disparity d_2)	-	-	32×104×1
	$d_2 = 100 \times d_2 + 0.01$	-	-	32×104×1
	Deconvolution	3×3	2	64×208×32
	Bilinear upsampling from d_2	-	2	64×208×1
	Concatenation	-	-	64×208×65
	Convolution	3×3	1	64×208×32
	Convolution	3×3	1	64×208×1
	Sigmoid (get disparity d_3)	-	-	64×208×1
	$d_3 = 100 \times d_3 + 0.01$	-	-	64×208×1
	Deconvolution	3×3	2	128×416×32
	Bilinear upsampling from d_3	-	2	128×416×1
	Concatenation	-	-	128×416×65
	Convolution	3×3	1	128×416×32
	Convolution	3×3	1	128×416×1
	Sigmoid (get disparity d_4)	-	-	128×416×1
	$d_4 = 100 \times d_4 + 0.01$	-	-	128×416×1
Output d_4				128×416×1

Table 1. Detailed DepthNet architecture, which is basically the same as the DepthNet of SfMLearner [6] but much more lightweighted. The output is disparity d_4 (inverse depth). Then depth $\hat{D} = \frac{1}{d_4}$.

Block	Operation	Filter size	Stride	Output size (height×width×channel)
Input $I_t, \hat{D}_t, I_{t-1}, \hat{D}_{t-1}$				128×416×8
Shared encoder	Convolution	7×7	2	64×208×16
	ConvLSTM	3×3	1	64×208×16
	Convolution	5×5	2	32×104×32
	ConvLSTM	3×3	1	32×104×32
Translation estimation t_x, t_y, t_z	Convolution	3×3	2	16×52×64
	ConvLSTM	3×3	1	16×52×64
	Convolution	3×3	1	16×52×3
	Fully Connected	-	-	1×1× 3
Rotation estimation r_x, r_y, r_z	Convolution	3×3	2	16×52×64
	ConvLSTM	3×3	1	16×52×64
	Convolution	3×3	1	16×52×3
	Fully Connected	-	-	1×1× 3
Output 6-DoF pose $t_x, t_y, t_z, r_x, r_y, r_z$				1×1×6

Table 2. Detailed PoseNet architecture.

Block	Operation	Filter size	Stride	Output size (height×width×channel)
Input warping residual				128×416×3
Encoder	Convolution	7×7	1	128×416×16
	Convolution	5×5	1	128×416×32
	Convolution	5×5	1	128×416×32
	Convolution	3×3	1	128×416×1
	Sigmoid	-	-	128×416×1
Output mask $\hat{M}$				128×416×1

Table 3. Detailed MaskNet architecture.

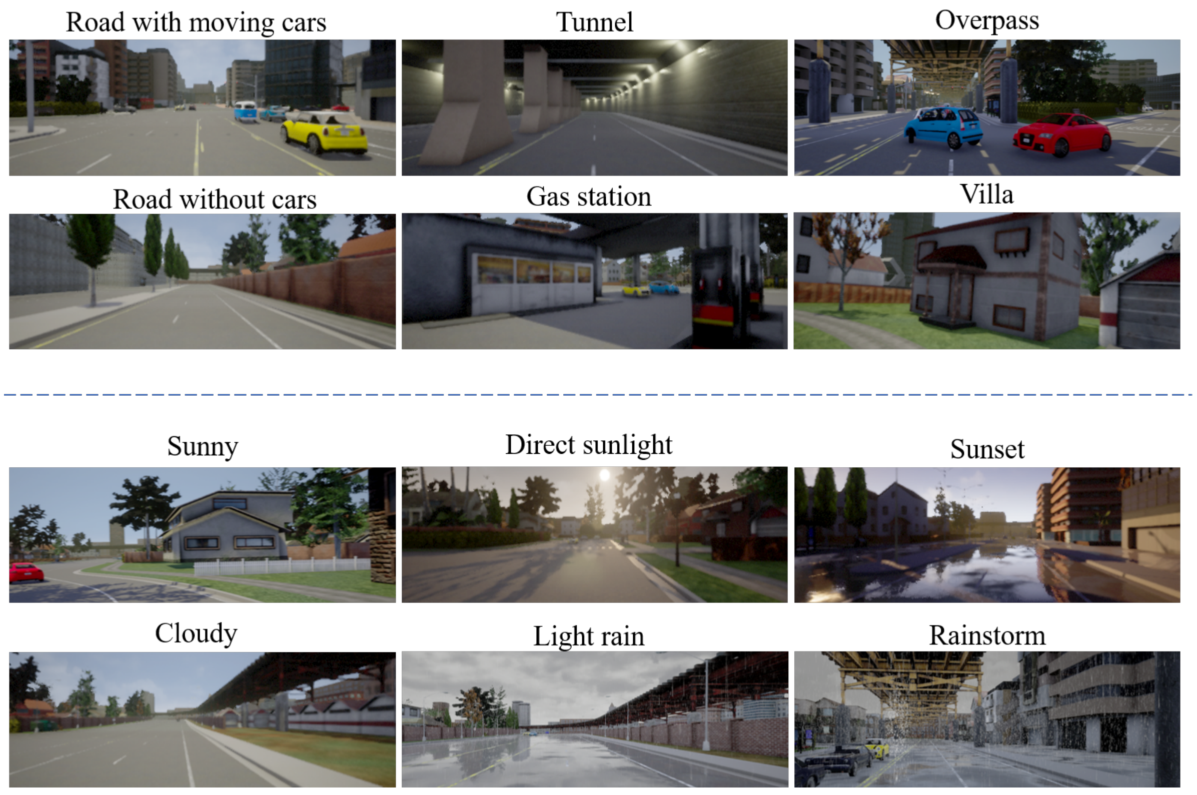

Figure 1. Different types of scenes we collected by Carla [1] simulator.

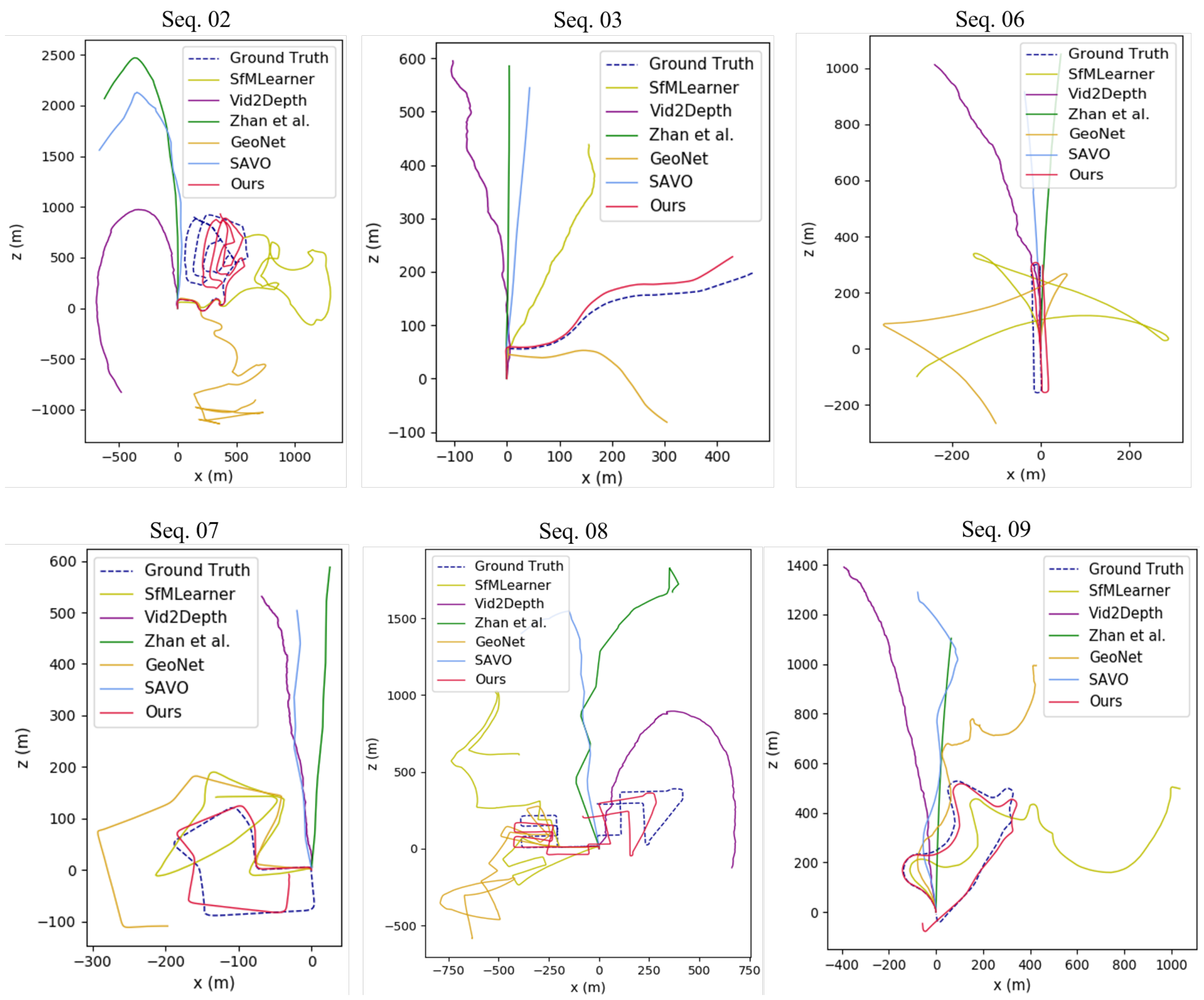

Figure 2. Results of various methods pretraining on Carla and online testing on KITTI 00-10. We select some complicated and challenging trajectories from these 11 sequences, including multiple turns, loops and sharp turns.