VOLDOR: Visual Odometry from Log-logistic Dense Optical flow Residuals Appendix

In the appendix, we first describe our ground plane estimation algorithm coupled with KITTI benchmark in §1. Also, we provide additional performance comparisons and a detailed performance study in §2. Then we provide implementation details in §3. Finally, we visualize our results under various scenes using different optical flow inputs in §4.

1. Ground Plane Estimation

In Table 1, we detail our ground plane estimation algorithm coupled with KITTI odometry benchmark.

Input: Estimated depth map $\boldsymbol{\theta}$ from VOLDOR
Output: Ground plane height h Ground normal vector $\boldsymbol{n}$
Crop a region of interest (ROI) at the image bottom. Compute a normal vector $\boldsymbol{n_j}$ for each pixel j in the ROI, where $\|\boldsymbol{n_j}\| = 1$. Estimate a height h_j for each pixel, where $h_j = \boldsymbol{n_j} \cdot \theta^j \boldsymbol{K^{-1}} [x_j y_j 1]^T$ Compute median height h_{med}. Meanshift on the 4-vector space of $[\boldsymbol{n_j}, h_j/h_{med}]$ with initial start mean $[0, -1, 0, 1]$. If result mean $[\hat{\boldsymbol{n}}, \hat{h}/h_{med}]$ has near perpendicular normal vector that $\hat{\boldsymbol{n}} \cdot [0, -1, 0] < \epsilon$ Return the result height $\hat{h}$ Else Ground plane registration failed

Table 1: **Ground plane estimation algorithm.** We estimate a normal vector and height for each pixel in the ROI and scale the height with its median. Meanshift is applied to find the mode with prior knowledge of the ground normal and the height scale (median). Finally, we actively check the estimated normal vector to determine if the ground estimate is correct.

2. Additional Experiments

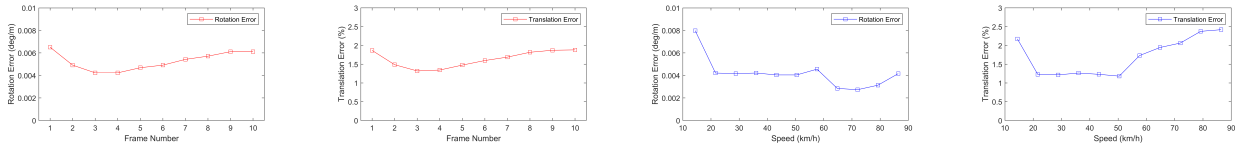
2.1. Comparison with deep-learning VO

As Table 2 shows, we compare our visual odometry result with recent SOTA deep-learning VO methods, where ORB-SLAM is used as baseline.

	Seq.09		Seq.10	
Method	Trans. (%)	Rot. (deg/m)	Trans. (%)	Rot. (deg/m)
ORB-SLAM [3]	15.30	**0.003**	3.68	0.005
GeoNet [8]	43.76	0.160	35.60	0.138
Zhou *et al.* [10]	17.84	0.068	37.91	0.178
Zhan *et al.* [9]	11.92	0.036	12.63	0.034
DeepVO [7]	-	-	8.11	0.088
Wang *et al.* [6]	9.30	0.034	7.21	0.039
VOLDOR (Ours)	**1.61**	0.004	**1.44**	**0.004**

Table 2: **Comparison with recent deep-learning methods on KITTI visual odometry benchmark sequence 09, 10.**

2.2. Detail Performance Study



(a) Rotation error over frame (b) Translation error over frame (c) Rotation error over speed (d) Translation error over speed

Figure 1: **Performance analysis over frame and speed on KITTI benchmark.**

Further, we detail the performance of our method. Due to the camera motion, our geometry representation of first frame's depth map causes the coverage of registrable depth pixels to vary with the frames within the optimization window. Some of the depth pixels can only be registered in a small number of frames, this will yield less reliable depth estimates since they are triangulated from less observations. However, those depth pixels will increase the depth coverage of the frames where they are observed. With different balance between the reliability and coverage of depth pixels, the performance of each frame within the optimization window varies. In Figure 1, we test the performance over 10 frames separately using KITTI dataset. As the result shows, the third and fourth frames usually have the best performance. We will utilize this property when fusing segment estimates as will be described in §3.

Figure 1 (c) (d) shows the performance over different camera moving speed in the KITTI dataset. Our method can stably work with a wide range of speed at 10-50 km/h. When the speed is low, camera baselines are small and the triangulation has large uncertainty, which decreases the performance. When the speed is high, frames have limited overlap and provide little mutual information that also causes a decrease of performance.

3. Implementation Details

Depth update. In the depth update process, we sample a depth value from an uniform distribution with inverse depth representation for each pixel, while the best depth value is kept. This process can be done multiple times to achieve better convergence. In practice, we sample one depth value for each pixel when testing for camera pose accuracy, while two times when testing depth accuracy. Finally, we do the propagation scheme from four directions only one time to spread the depth values. While computing the likelihood of a depth value, if a pixel falls out the boundary, we simply set its rigidness from that time step to zero. We use bilinear interpolation to obtain a flow vector on continuous position of observed optical flow field.

Pose update. In the pose sampling process, besides only referring to the rigidness map, we also select pixels fall in a certain range of depth. We pick pixels with depth $500 > \frac{\theta^j}{t} > 0.1$, where θ^j is the pixel depth value, and t is the translation vector magnitude. In meanshift process, since we observed a low variance of pose weights, we binarize the weight with a threshold at 0.5 for a faster meanshift implementation. For meanshift initialization, we use the estimated camera pose in the previous iteration. In the case of first iteration, we start with the camera pose of previous time step if the initialization can obtain a kernel weight larger than 0.1. Otherwise, we randomly pick 10 start point and start with the start point with largest kernel weight.

Truncation. In the whole workflow, for robustness, we actively detect badly registered frame and truncate the optimization window size. This mainly happens when the camera motion is large, such that the depth map of first frame cannot been registered to the tail frames. We use two criteria to determine a truncation at time step t, 1) when rigidness map at time t has less than 2000 inlier pixels, 2) after meanshift for camera pose at time t converges, the kernel weight is less than 0.01. The truncation truncates the optimize window size to $t-1$. and force the algorithm to run 3 more iterations after truncation to ensure convergence.

Fusion. Our method treats each optimization window independently. Thus, we apply a fusion as post-processing to obtain the full visual odometry result. In KITTI experiment, we run the sliding window of window size 6 with step 1, resulted in obtaining 6 pose candidates for each time step. As shown in Fig. 1, the pose quality of each frame in the optimization window varies. In light of this, we pick poses of better quality with higher priority according to the plot.

4. Results

In this section, we visualize our results under various scenes using four different optical flows[5, 4, 1, 2].

4.1. Result - KITTI - Dynamic Foreground

When there are moving objects at foreground, rigidness maps can correctly segment the object and exclude them from the estimation of depth map. However, the result depth map can still provide correct background depth estimation at regions occluded by dynamic foreground objects using the information from frames where the region is not occluded.

Figure 2: Input image sequence.

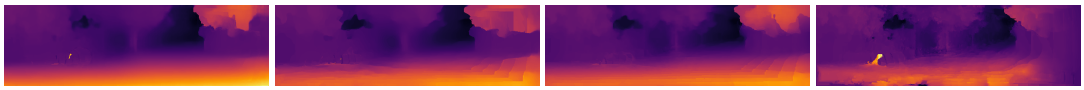


Figure 3: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

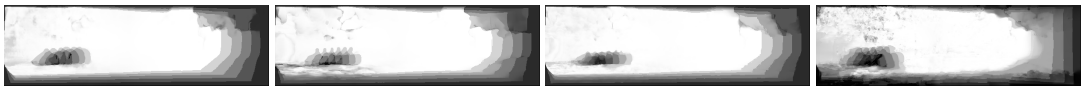


Figure 4: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

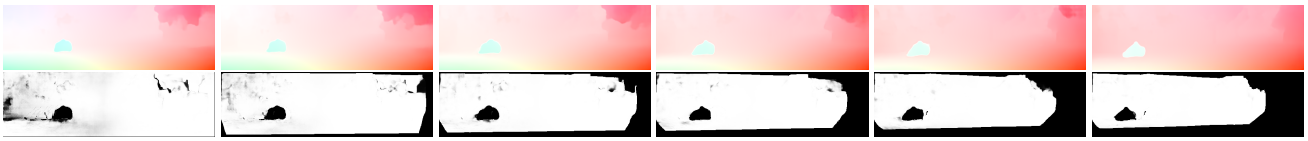


Figure 5: Optical flow fields from PWC-Net and inferred rigidness maps.

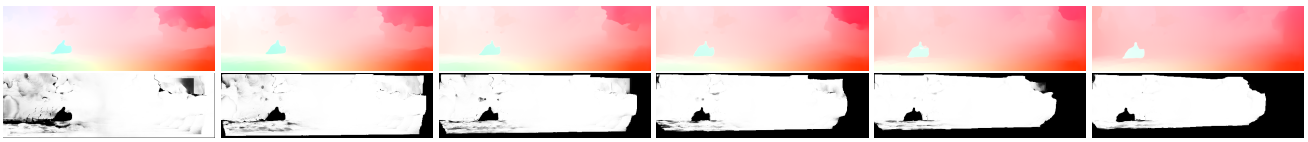


Figure 6: Optical flow fields from FlowNet2 and inferred rigidness maps.

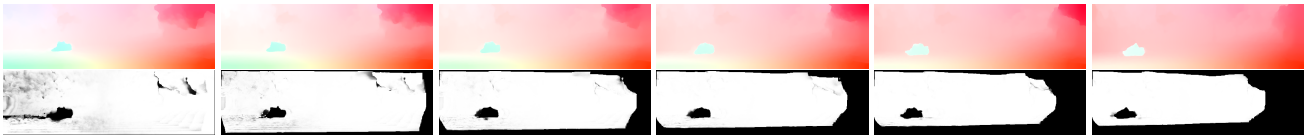


Figure 7: Optical flow fields from EpicFlow and inferred rigidness maps.

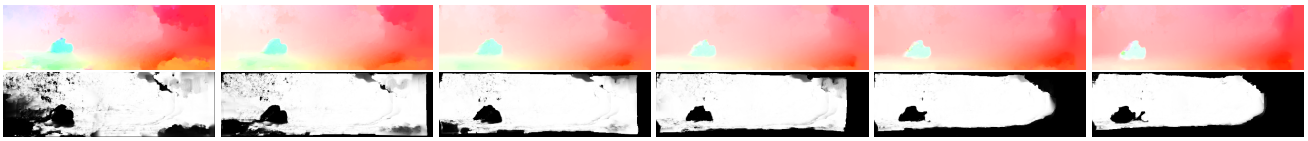


Figure 8: Optical flow fields from C2F-Flow and inferred rigidness maps.

4.2. Result - KITTI - Forward Motion

With forward motion, depth pixels become invisible by exiting the field of view at the image boundary, which is reflected in the rigidness maps. Also, occlusions as well as outlier pixels (along object boundaries and the sky region of C2F-Flow and EpicFlow) are clearly indicated by rigidness maps.

Figure 9: Input image sequence.

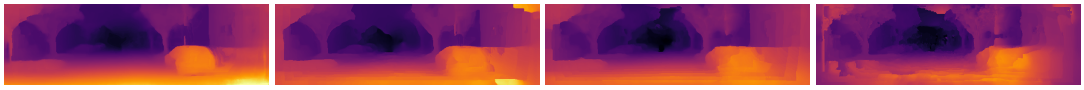


Figure 10: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

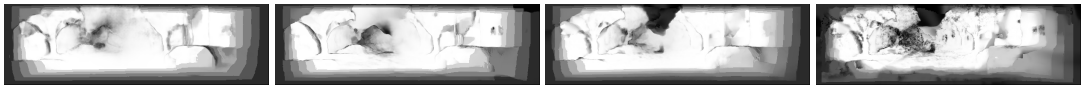


Figure 11: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

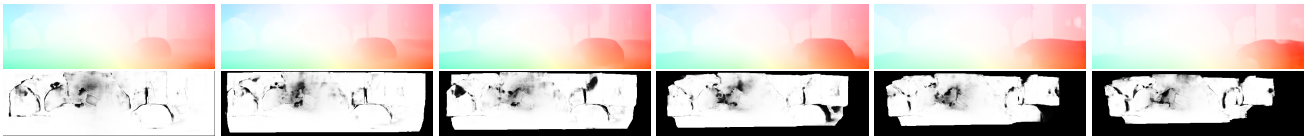


Figure 12: Optical flow fields from PWC-Net and inferred rigidness maps.

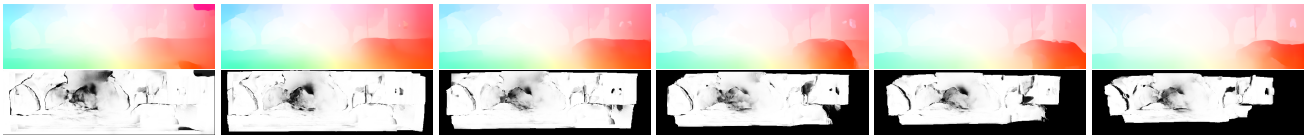


Figure 13: Optical flow fields from FlowNet2 and inferred rigidness maps.

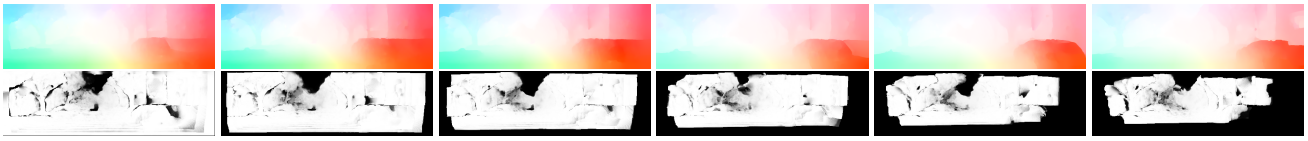


Figure 14: Optical flow fields from EpicFlow and inferred rigidness maps.

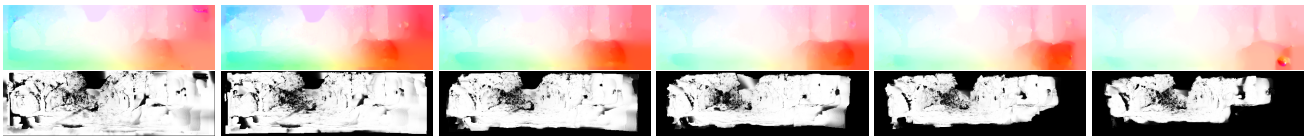


Figure 15: Optical flow fields from C2F-Flow and inferred rigidness maps.

4.3. Result - KITTI - Rotating Motion

With rotating motion, depth pixels becomes invisible from the side opposite the rotation, which also is reflected in the rigidness maps. Good optical flow methods can usually provide more consistent output depth map along the object and frame boundaries.

Figure 16: Input image sequence.

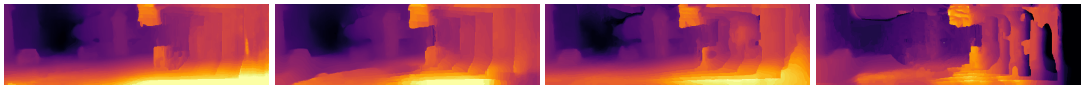


Figure 17: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

Figure 18: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

Figure 19: Optical flow fields from PWC-Net and inferred rigidness maps.

Figure 20: Optical flow fields from FlowNet2 and inferred rigidness maps.

Figure 21: Optical flow fields from EpicFlow and inferred rigidness maps.

Figure 22: Optical flow fields from C2F-Flow and inferred rigidness maps.

4.4. Result - TUM RGB-D - Indoor 1

Indoor capture scenario exhibiting smaller camera motions and diverse motion patterns. Our methods also works well and triangulates consistent depth map.

Figure 23: Input image sequence.

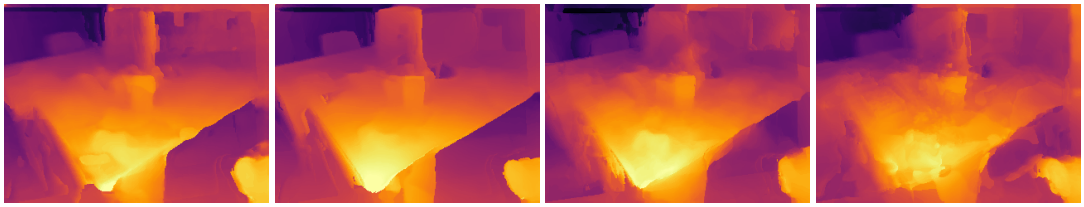


Figure 24: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

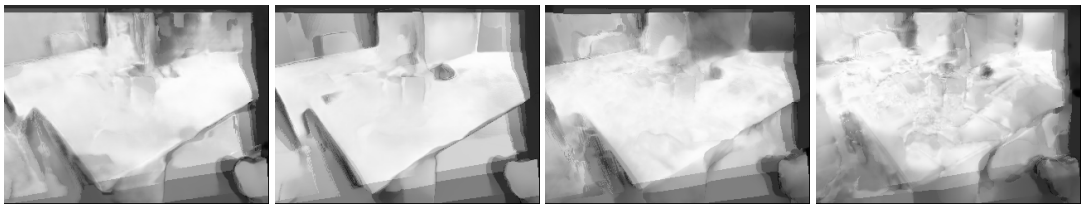


Figure 25: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

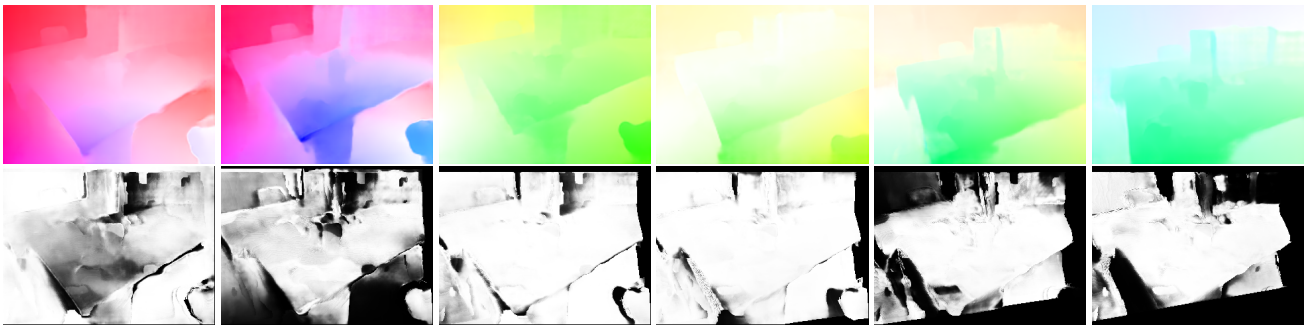


Figure 26: Optical flow fields from PWC-Net and inferred rigidness maps.

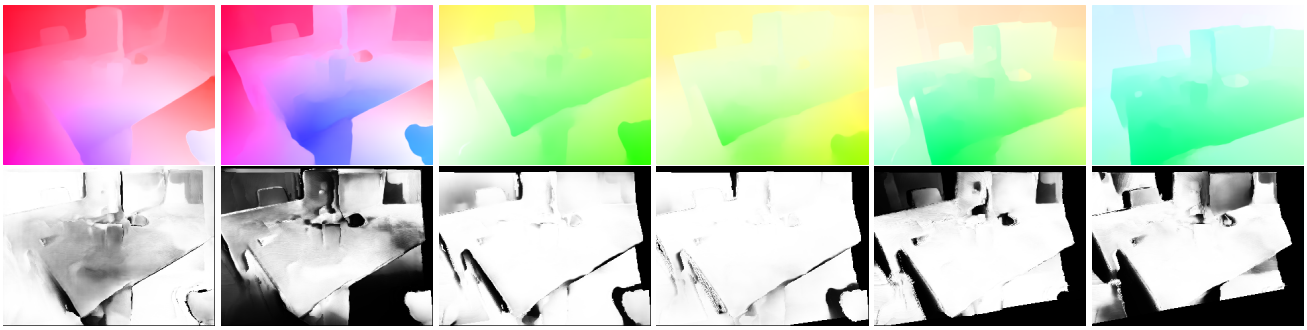


Figure 27: Optical flow fields from FlowNet2 and inferred rigidness maps.

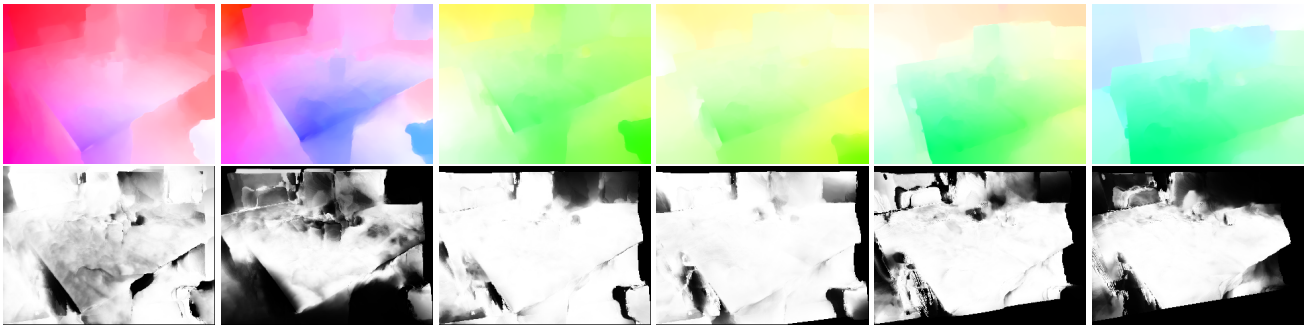


Figure 28: Optical flow fields from EpicFlow and inferred rigidness maps.

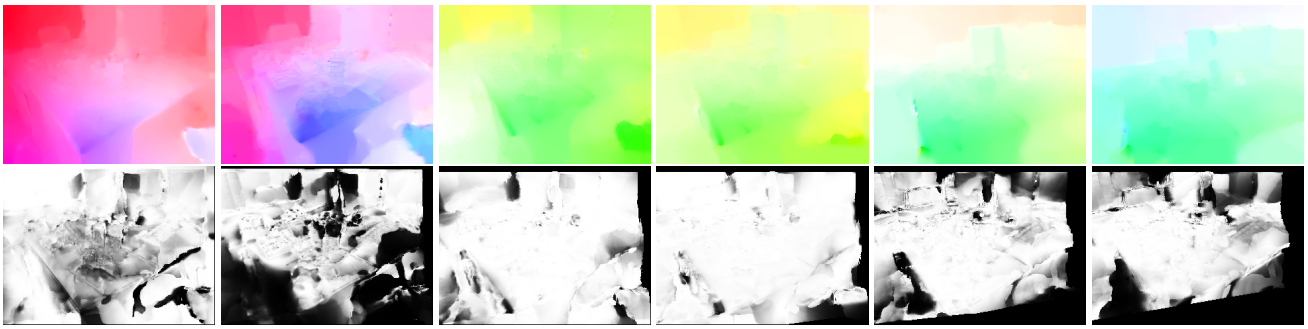


Figure 29: Optical flow fields from C2F-Flow and inferred rigidness maps.

4.5. Result - TUM RGB-D - Indoor 2

Indoor capture scenario exhibiting smaller camera motions and diverse motion patterns. Our methods also works well and triangulates consistent depth map.

Figure 30: Input image sequence.

Figure 31: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

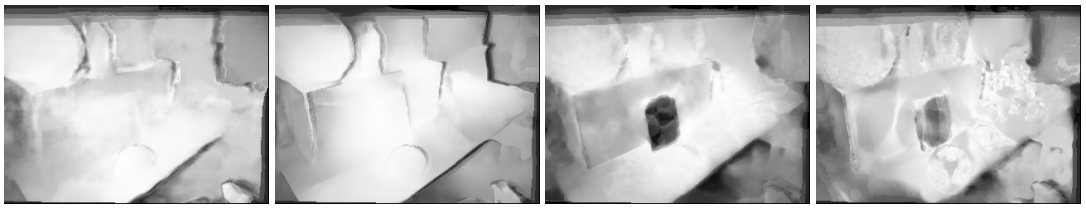


Figure 32: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

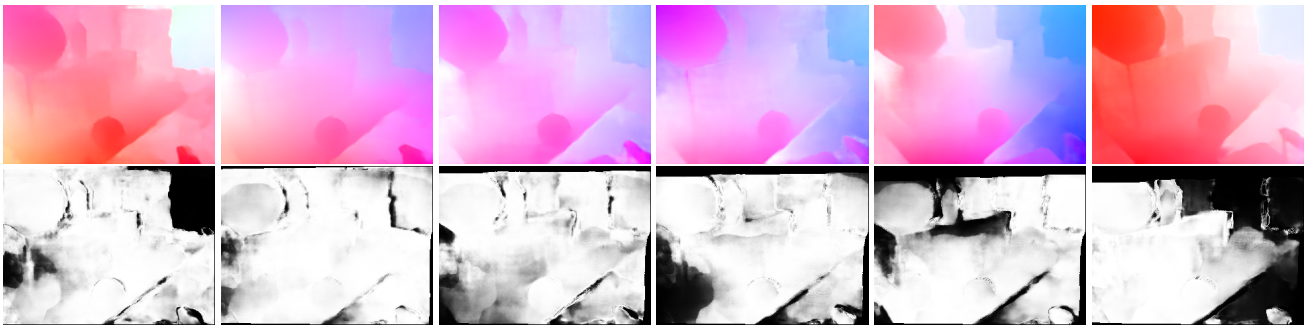


Figure 33: Optical flow fields from PWC-Net and inferred rigidness maps.

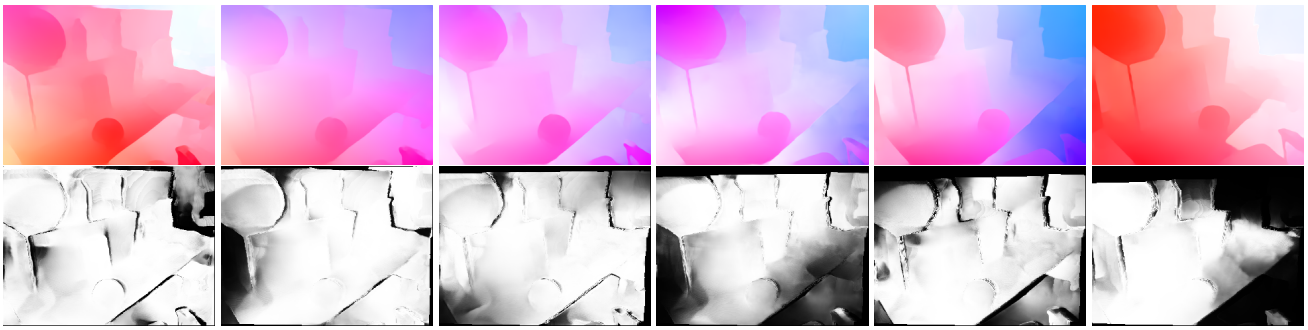


Figure 34: Optical flow fields from FlowNet2 and inferred rigidness maps.

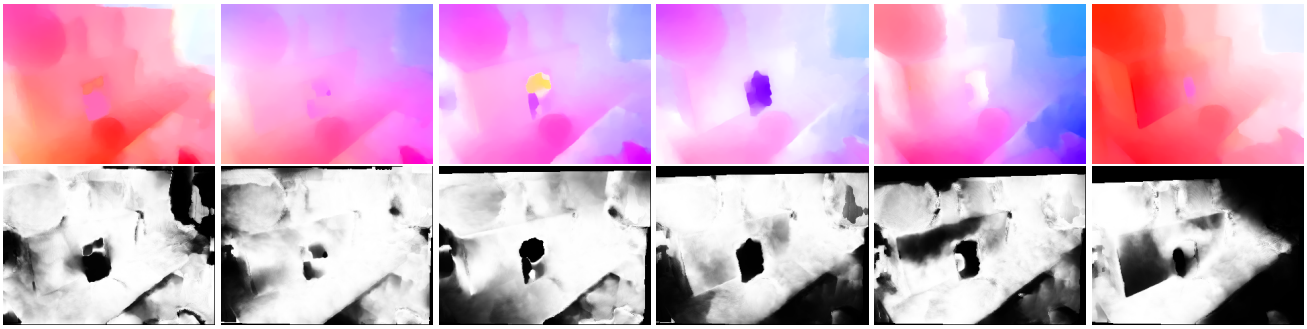


Figure 35: Optical flow fields from EpicFlow and inferred rigidness maps.

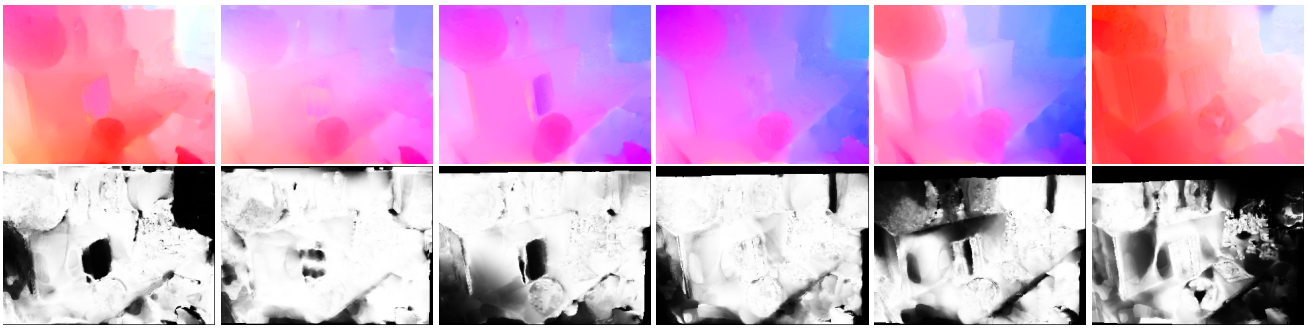


Figure 36: Optical flow fields from C2F-Flow and inferred rigidness maps.

References

[1] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 3

[2] Ce Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009. 3

[3] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 1

[4] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1164–1172, 2015. 3

[5] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 3

[6] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5555–5564, 2019. 1

[7] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017. 1

[8] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018. 1

[9] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018. 1

[10] 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. 1