

000 054  
 001 055  
 002 056  
 003 057  
 004 **VOLDOR: Visual Odometry from Log-logistic Dense Optical flow Residuals** 058  
 005 Appendix 059  
 006 060  
 007 061  
 008 062  
 009 Anonymous CVPR submission 063  
 010 064  
 011 Paper ID 5888 065  
 012 066  
 013 067  
 014 In the appendix, we first describe our ground plane estimation algorithm coupled with KITTI benchmark in §1. Also, 068  
 015 we provide additional performance comparisons and a detailed performance study in §2. Then we provide implementation 069  
 016 details in §3. Finally, we visualize our results under various scenes using different optical flow inputs in §4. 070  
 017 071  
 018 **1. Ground Plane Estimation** 072  
 019 In Table 1, we detail our ground plane estimation algorithm coupled with KITTI odometry benchmark. 073  
 020 074

---

**Input:** Estimated depth map  $\theta$  from VOLDOR  
**Output:** Ground plane height  $h$   
 Ground normal vector  $n$

---

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

Method	Seq.09		Seq.10	
	Trans. (%)	Rot. (deg/m)	Trans. (%)	Rot. (deg/m)
ORB-SLAM [1]	15.30	<b>0.003</b>	3.68	0.005
GeoNet [4]	43.76	0.160	35.60	0.138
Zhou <i>et al.</i> [6]	17.84	0.068	37.91	0.178
Zhan <i>et al.</i> [5]	11.92	0.036	12.63	0.034
DeepVO [3]	-	-	8.11	0.088
Wang <i>et al.</i> [2]	9.30	0.034	7.21	0.039
VOLDOR (Ours)	<b>1.61</b>	0.004	<b>1.44</b>	<b>0.004</b>

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

108

## 2.2. Detail Performance Study

162

109

163

110

164

111

165

112

166

113

167

114

168

115

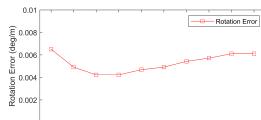
169

116

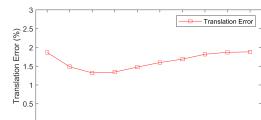
170

117

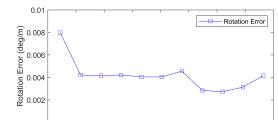
171



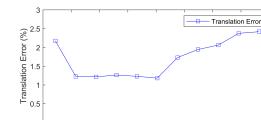
(a) Rotation error over frame



(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 registerable 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.

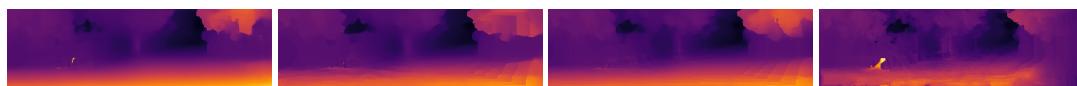
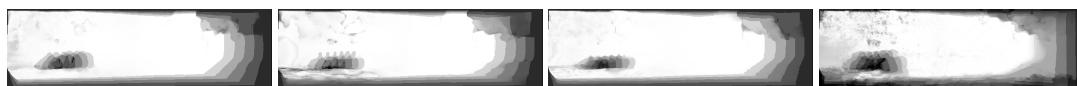
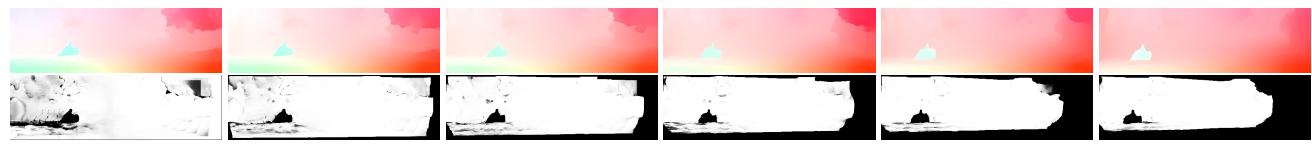
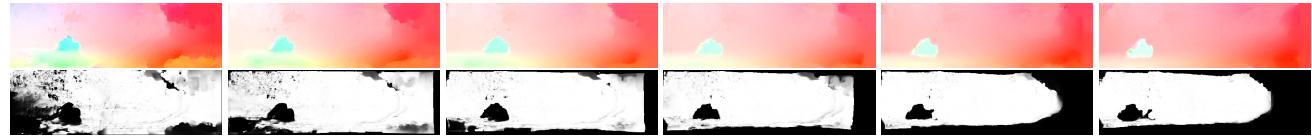
## 3. Implementation Details

**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  $500 > \frac{\theta^j}{t} > 0.1$ , where  $\theta^j$  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 be 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.

216 **4. Results** 270  
217218 In this section, we visualize our results under various scenes using four different optical flows[?, ?, ?, ?]. 271  
219220 **4.1. Result - KITTI - Dynamic Foreground** 273  
221222 When there are moving objects at foreground, rigidness maps can correctly segment the object and exclude them from 223 the estimation of depth map. However, the result depth map can still provide correct background depth estimation at regions 224 occluded by dynamic foreground objects using the information from frames where the region is not occluded. 275  
225226 **Figure 2: Input image sequence.** 281  
227228 **Figure 3: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.** 286  
229230 **Figure 4: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.** 290  
231232 **Figure 5: Optical flow fields from PWC-Net and inferred rigidness maps.** 297  
233234 **Figure 6: Optical flow fields from FlowNet2 and inferred rigidness maps.** 305  
235236 **Figure 7: Optical flow fields from EpicFlow and inferred rigidness maps.** 311  
237238 **Figure 8: Optical flow fields from C2F-Flow and inferred rigidness maps.** 317  
239

324

## 4.2. Result - KITTI - Forward Motion

325

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.

329



330

Figure 9: Input image sequence.

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

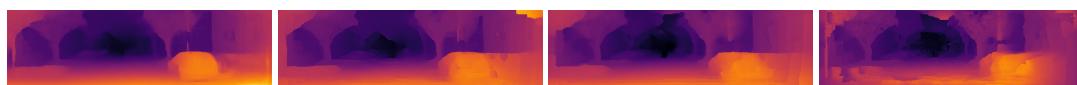


Figure 10: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

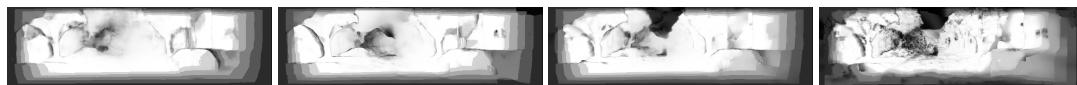


Figure 11: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

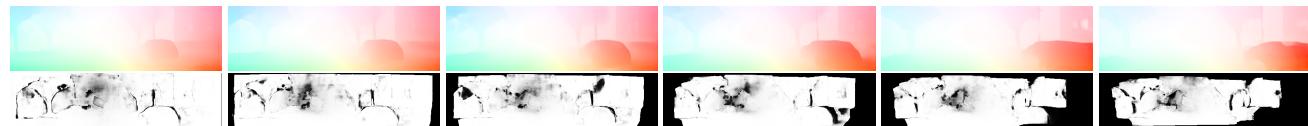


Figure 12: Optical flow fields from PWC-Net and inferred rigidness maps.

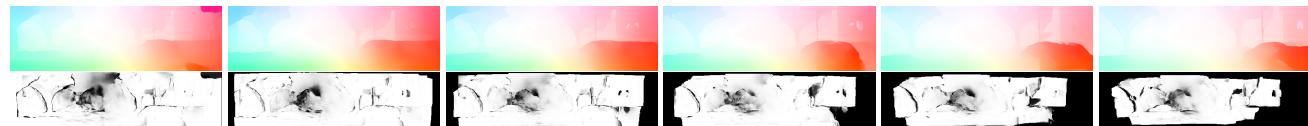


Figure 13: Optical flow fields from FlowNet2 and inferred rigidness maps.

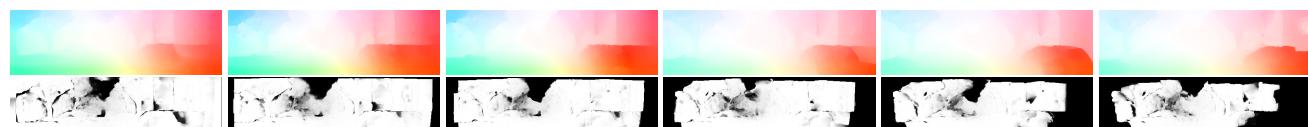


Figure 14: Optical flow fields from EpicFlow and inferred rigidness maps.

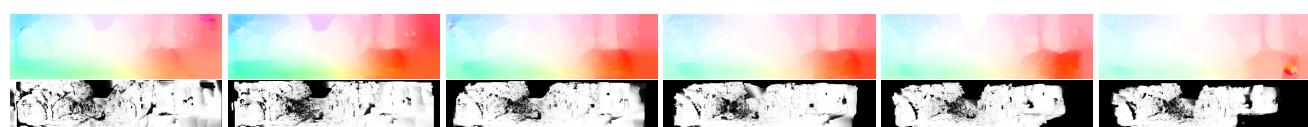


Figure 15: Optical flow fields from C2F-Flow and inferred rigidness maps.

432

### 4.3. Result - KITTI - Rotating Motion

433  
434  
435  
436

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.

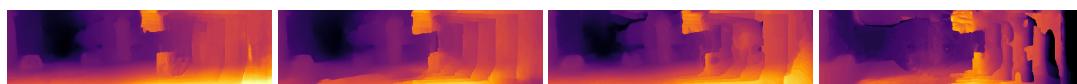
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539"/>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.

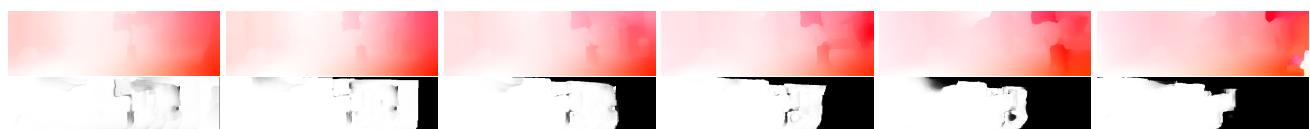


Figure 21: Optical flow fields from EpicFlow and inferred rigidness maps.



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

540

#### 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.

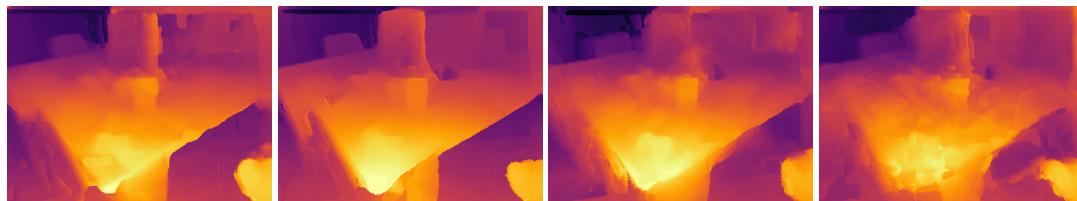


Figure 24: Inferred depth map of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

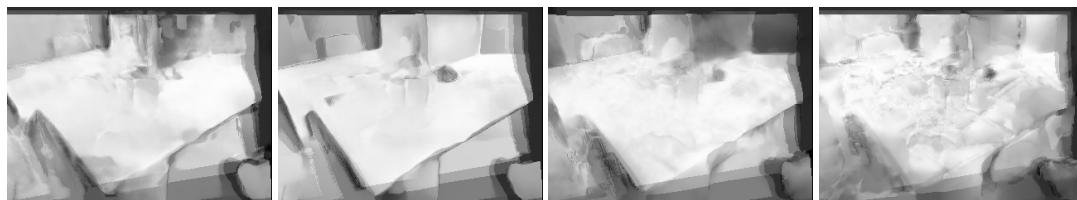


Figure 25: Rigidness map sum of (in order) PWC-Net, FlowNet2, EpicFlow, C2F-Flow.

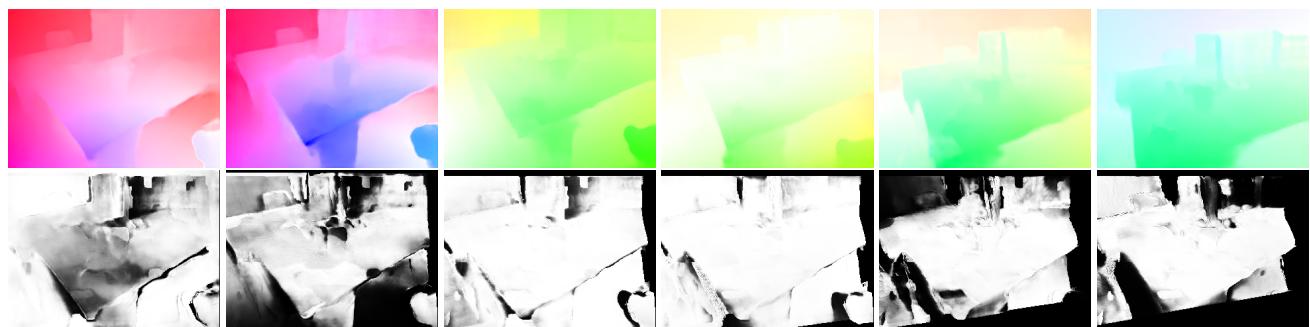


Figure 26: Optical flow fields from PWC-Net and inferred rigidness maps.

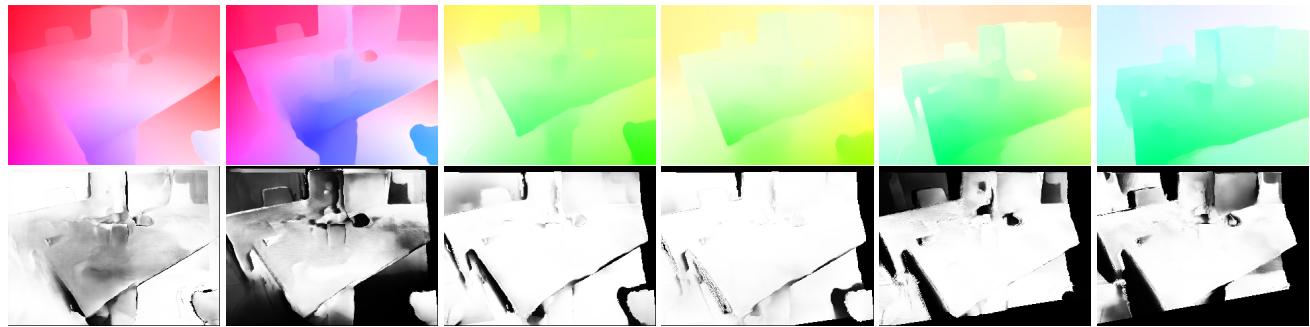
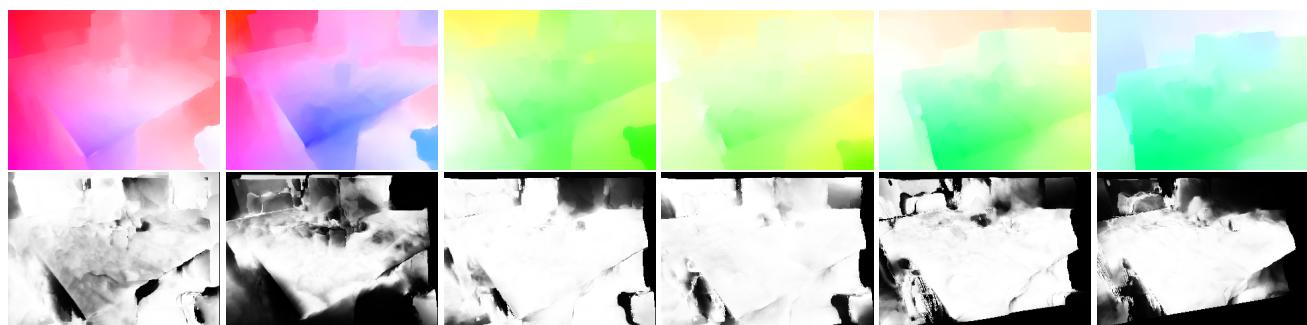
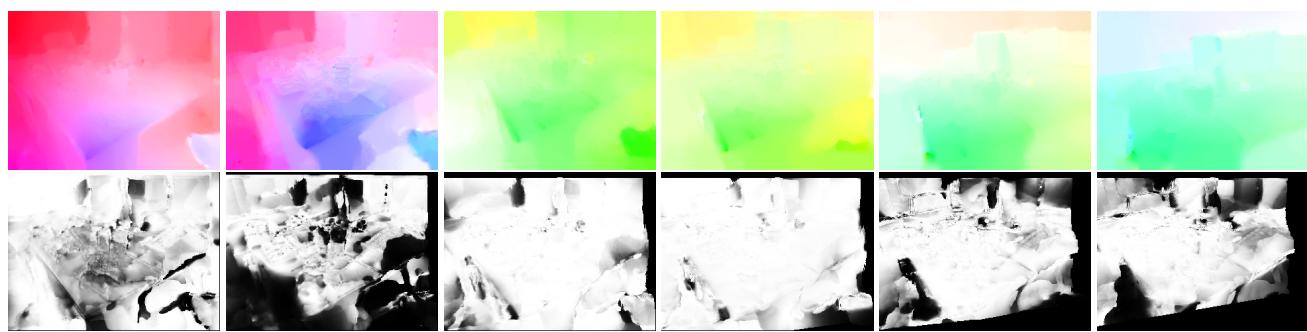


Figure 27: Optical flow fields from FlowNet2 and inferred rigidness maps.

648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
  
Figure 28: Optical flow fields from EpicFlow and inferred rigidness maps.671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
  
Figure 28: Optical flow fields from EpicFlow and inferred rigidness maps.714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755

756

#### 4.5. Result - TUM RGB-D - Indoor 2

757

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

760



761

Figure 30: Input image sequence.

762



763

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

764



765

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

766

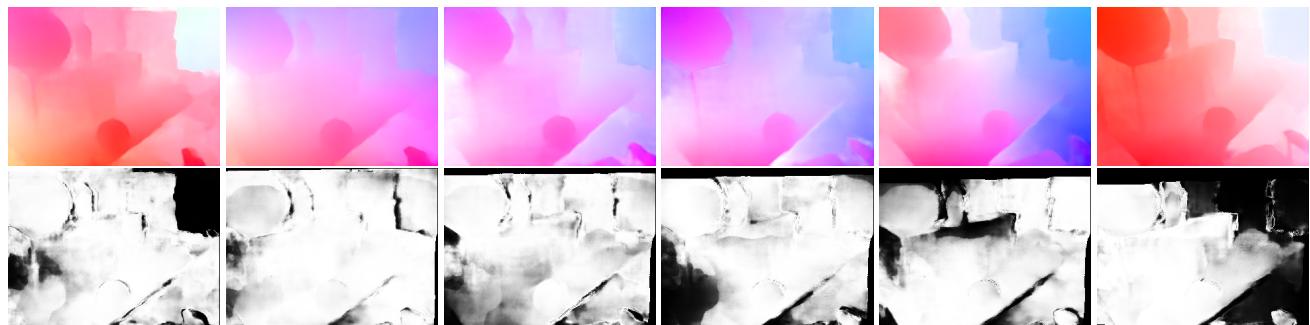


Figure 33: Optical flow fields from PWC-Net and inferred rigidness maps.

767



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



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

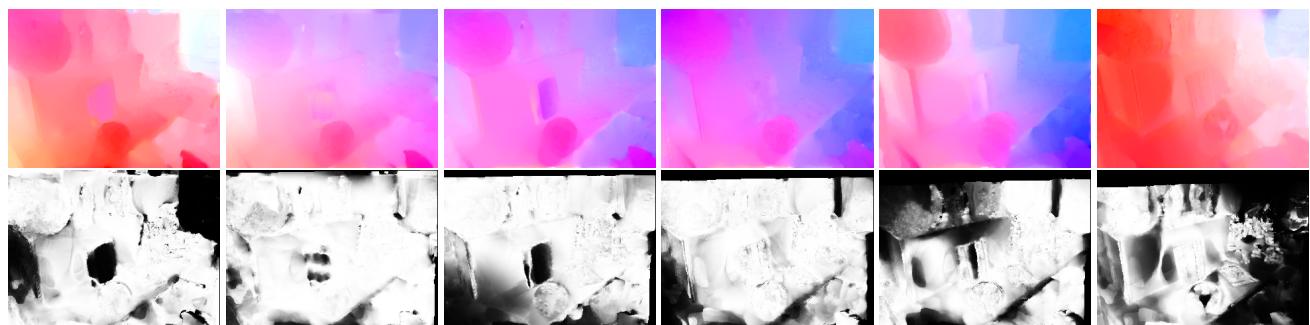


Figure 36: Optical flow fields from C2F-Flow and inferred rigidness maps.

864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886

918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971

972	<b>References</b>	1026
973		1027
974	[1] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. <i>IEEE Transactions on Robotics</i> , 33(5):1255–1262, 2017. 1	1028
975		1029
976	[2] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth. In <i>Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition</i> , pages 5555–5564, 2019. 1	1030
977		1031
978	[3] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In <i>2017 IEEE International Conference on Robotics and Automation (ICRA)</i> , pages 2043–2050. IEEE, 2017. 1	1032
979		1033
980	[4] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In <i>Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition</i> , pages 1983–1992, 2018. 1	1034
981		1035
982	[5] 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 <i>Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition</i> , pages 340–349, 2018. 1	1036
983		1037
984	[6] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In <i>Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition</i> , pages 1851–1858, 2017. 1	1038
985		1039
986		1040
987		1041
988		1042
989		1043
990		1044
991		1045
992		1046
993		1047
994		1048
995		1049
996		1050
997		1051
998		1052
999		1053
1000		1054
1001		1055
1002		1056
1003		1057
1004		1058
1005		1059
1006		1060
1007		1061
1008		1062
1009		1063
1010		1064
1011		1065
1012		1066
1013		1067
1014		1068
1015		1069
1016		1070
1017		1071
1018		1072
1019		1073
1020		1074
1021		1075
1022		1076
1023		1077
1024		1078
1025		1079