Frame-to-Frame Visual Odometry: The Importance of Local Transformations

Aleksander Kostusiak(✉)

Institute of Control and Information Engineering,
Poznań University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland
aleksander.m.kostusiak@doctorate.put.poznan.pl

Abstract. Trajectory estimation is of pivotal importance for mobile robots. Visual Odometry (VO) allows localizing a robot from passive vision data in frame-to-frame fashion. The VO problem can be solved in different ways, hence an evaluation of these algorithms in the context of real benchmark data is interesting. We focus on feature-based n-point methods based on RGB images. These methods used in monocular vision allow for camera rotation estimation, but only a few of them provide translation estimates up to the unknown scale. In the context of the use of commodity RGB-D cameras, we also compare these methods with the Kabsch algorithm, which uses full depth information.

Keywords: Sparse visual odometry · Trajectory estimation · Monocular vision · Comparison · Benchmark

1 Introduction

The trajectory estimation problem is in the spotlight for many years, yielding many different solutions. The best approaches belong to the dense solution group and use all the information seen by the camera. Because this is computationally expensive and requires GPU acceleration, it is almost impossible to use such an approach in mobile robotics in a larger scale. For these reasons, we are concerned with the existing feature-based (sparse) approaches. While working with a single passive (RGB) camera it is impossible to obtain full depth information about the scene and in turn to accurately determine the scale of translation between two frames. However, it is still possible to compute rotational components of the trajectory and to use them for constraining the orientation of the robot/camera. This is of high importance, as small errors here can result in large translational errors. It was shown in [2] that a SLAM system can benefit from being augmented with the visual estimation of the frame-to-frame orientation change. In the last few years, the focus in visual trajectory estimation was put on Visual SLAM and Visual Odometry. The former type of approach builds a map, and then estimates the robot position with respect to it. The first working real-time Visual SLAM was demonstrated by Davison and Murray [11]. The latter type of approach

M. Kurzynski et al. (eds.), *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, Advances in Intelligent Systems and Computing 578, DOI 10.1007/978-3-319-59162-9_37

tries to achieve the same goal without building a map, however with a larger number of frame-to-frame measurements. A comprehensive overview of the VO methods is presented in [12,13], and in [14], where VO and SLAM algorithms are nicely and briefly described. The rest of this paper is structured as follows: Sect. 2 presents the used libraries and related work, Sect. 3 briefly describes the n-point relative pose computation algorithms, Sect. 4 details the methodology used to obtain both the quantitative and qualitative results, Sect. 5 describes and comments the obtained results, and Sect. 6 concludes the paper.

2 State of the Art

Currently, there are several libraries that help in obtaining rotation from RGB images, like OpenCV [8], OpenMVG and OpenGV. All of the mentioned libraries but OpenCV contain the 5-point Nister [3,4], Stewenius [5], and 7-point and 8-point methods, which are in OpenCV, for finding the essential matrices from 2D-2D correspondences. OpenCV has two last methods to return fundamental matrix only. OpenMVG contains a 5-point method that uses Nister and Stewenius constraint expansions that find the essential matrix. The richest library of that kind of algorithms is OpenGV allowing for [9]: calculation pose of the camera from 2D-3D correspondences between points in the world frame and bearing vectors in the camera frame (this states as central absolute pose problem), and in multiple camera frames (non-central absolute pose case), finding the pose of one camera with respect to another camera given a number of 2D-2D correspondences between the bearing vectors in the camera frames (central relative pose), and in multiple camera frames (non-central relative pose problems). Some algorithms may return only rotation, only translation, several different probable rotations, rotation and translation, several probable rotations and translations, essential matrix, several probable essential matrices, or several probable complex essential matrices. In the past Murphy et. al. presented [14] an evaluation of three VO techniques: patch-based, Structure from Motion, and the VO approach of [15], but in a monocular version. Recently, there is a work in semi-dense approaches [1] that labels itself as VO method claiming frame-to-frame trajectory estimation based on semi-dense inverse depth map based. Nonetheless, to create those semi-dense depth maps their approach uses the history of RGB frames to find a possibly oldest frame, in which the pixel from the current frame was seen. That leads to building a form of map, and to the indirect use of it. Consequently, this type of solution does not fit well to the VO definition, being partially a SLAM system, yet without trajectory optimization. Worth mentioning are the experiments trying to combine both worlds of visual SLAM and VO. Earlier, Schmidt et al. [2] proposed to augment SLAM method by replacing orientation change every few steps of EKF prediction function with the one provided by separately running VO system. A comparison of SLAM and VO approaches for RGB-D data was presented in [6]. As for now, there is very little work done concerning evaluation of methods that estimate the frame-to-frame orientation change. Recently, Hartman et. al. [7] used 5-point Stewenius algorithm with a

modified RANSAC approach. Ground truth data was collected with the use of IMU and GPS devices while the one used in experiments presented here were collected by a high-resolution multi-camera system [21], or the Vicon motion capture system [19].

3 Used Algorithms

In this section, we briefly describe used algorithms for solving geometric vision problem. We used SURF, ORB and AKAZE feature detector-descriptor pairs for preliminary image processing. 8-point [16,17] algorithm is here the oldest algorithm for computing essential or fundamental matrix, depending if the calibration has been performed, based on 8 or more matched points. The essential matrix can be defined as:

$$\mathbf{u'Fu} = 0 \quad \text{(a)} \quad \text{or} \quad \mathbf{u'Eu} = 0 \quad \text{(b)} \tag{1}$$

where u' and u are matched homogeneous image points, because of that the last element equals to 1, and E and F stand for the Essential and Fundamental matrix (without calibration data). The entry points are normalized to ensure that all entries of respective matrices will be treated approximately equally. After representing matrix Eq. 1 as a vector, set of linear equations is obtained:

$$\mathbf{Af} = 0 \quad \text{(a)} \quad \text{or} \quad \mathbf{Ae} = 0 \quad \text{(b)} \tag{2}$$

where A is the equation matrix, e and f are 9-vector containing the entries of the matrix E or F. The equation is solved under the singularity constraint ‖f‖or ‖e‖= 1, where ‖e‖and ‖f‖are the norms, by checking if it is of rank 2 and if it is singular and by enforcing if it is not the case. Finally, the result matrix is replaced by closes singular matrix under Frobenius norm. Nister's [3,4] algorithm computes Essential matrix given 5 correspondences, which gives a rise to essential matrix constraint of the following form:

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0 \tag{3}$$

After representing matrix Eq. 3 as vector, set of the linear equation a 5×9 matrix is obtained. Next, four vectors $\tilde{\mathrm{X}}$, $\tilde{\mathrm{Y}}$, $\tilde{\mathrm{Z}}$, $\tilde{\mathrm{W}}$ that spans the right nullspace of that matrix are calculated.

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + w\mathbf{W} \tag{4}$$

They correspond directly to four matrices constraining the form of essential matrix 4 which is in turn inserted into nine cubic constraints 3. Calculations are performed by using Gröwner basis. Scalars x, y, z, w are defined up to a common scale factor and it is assumed that w = 1. As the output algorithm can return up to 10 essential matrices, but some of them are internally rejected. Kneip [18] proposed a direct method for finding exact rotation between two

images independently of the translation. This results from the fact that rotational optical flow is fundamentally different from translational one. The constraints on the rotation are then as follows:

$$|(f_1 \times \mathbf{R}f_1')(f_2 \times \mathbf{R}f_2')(f_3 \times \mathbf{R}f_3')| = 0 \tag{5}$$

Because rotation encodes 3° of freedom at least 3 epipolar plane normal coplanarity constraints are required to fully constrain rotation. Two additional features allow for building the necessary system of equation:

$$\begin{aligned} |(f_1 \times \mathbf{R}f_1')(f_2 \times \mathbf{R}f_2')(f_3 \times \mathbf{R}f_3')| &= 0 \\ |(f_1 \times \mathbf{R}f_1')(f_2 \times \mathbf{R}f_2')(f_4 \times \mathbf{R}f_4')| &= 0 \\ |(f_1 \times \mathbf{R}f_1')(f_2 \times \mathbf{R}f_2')(f_5 \times \mathbf{R}f_5')| &= 0 \end{aligned} \tag{6}$$

To solve this Eq. 6 Gröne basis is used. As the output algorithm can return up to 20 different rotation matrices, resulting from 10 essential matrices, but some of them are internally rejected. Eigensolver method tries to find the eigenvalue that minimizes the fallowing function 7:

$$\mathbf{R} = argmin_{\mathbf{R}} \lambda_{M,min} \tag{7}$$

where R is the rotation transforming the i-th bearin vector f_i to the corresponding one seen from the second frame f_i'. M has rank at most 2 and is a real symmetric and positive- defined matrix of fallowing form: $M = \sum_{i=1}^{n}(f_i \times \mathbf{R}f_i')(f_i \times \mathbf{R}f_i')^T$. At best a non-linear optimization over three parameters is needed because rotation has 3° of freedom. To solve the problem Kneip et.al. [18] are enforcing the first-order partial derivates of $\lambda_{M,min}$ to be zero and then are using Levenberg-Marquardt scheme. Also, this algorithm computes translation direction (but does not explicitly return it) as automatically given by the eigenvector that corresponds to the smallest eigenvalue. We have implemented the Kabsch algorithm [22,23] with the help of information found in [24]. As this algorithm computes transformation of two aligned points sets, the camera movement is the opposite.

4 Experimental Methodology

The whole program is constructed similarly to the work presented in [10]: firstly detector/descriptor algorithm is used to extract silent features. Then RANSAC technique, with varying ejection threshold (and for Eigensolver method also varying confidence) is performed twice to obtain best results. At this stage, two different error measures are used: reprojection error and a simple Euclidean distance between bearing vectors. The second method omits the translation, assuming it is sufficiently small. Finally, a trajectory is computed with all the matches remaining. Kneip algorithm is an exception- it only accepts from 5 to 8 bearing vectors pairs. It is also the only algorithm that does not compute translation, thereby to be able to compute reprojection error we employ

a two-point algorithm, that is also contained in the OpenGV library. Because some algorithms, namely Kneip, Nister and Eightpoint return several results, the best one is chosen based on appropriate error measure. In a case of an empty result set, resulted from in-algorithm suppression, 8 points are randomly chosen from the inliers set to compute rotation and translation as long as it does not exceed maximal RANSAC iterations. If the number of remaining inliers is insufficient for trajectory estimation the whole pipeline breaks as unable to correctly recover trajectory. For experiments we used 2 different trajectories: *putkk_Dataset_1_Kin_1* data set described in [20], is further referred as dataset1, and *fr3_long_office_household* which will be referred as dataset2 from the TUM RGB-D Benchmark [19]. Dataset1 was collected by the Kinect sensor mounted on the moving in the laboratory wheeled robot. Ground truth was obtained from multicamera vision system PUT Ground Truth (PUT GT) [21]. The acquisition time of robot and GT cameras have been synchronized so no interpolation was performed. Dataset2 was collected by the hand-held Kinect camera. Ground truth was also acquired by a multicamera system but a time of acquisition was not enforced what resulted in the need for data interpolation. Timestamps are not perfectly aligned in the ground truth and data collected by the camera. To measure the quality of retrieved trajectory we use well known Absolute Trajectory Error (ATE) and Relative RPE metrics as described in [19]. The first error is the Euclidean distance between corresponding points of the ground truth trajectory and estimated one. RPE is a relative translational or rotational error of successive frames. As for evaluation Root Mean Squared Error (RMSE) of mentioned metrics are used.

5 Results

In this section, in order to fit the data into the tables, we abbreviated some names. The translation recovery approaches that are based on the use of Centroid are abbreviated to C. and the Scale-based methods (depth is used only in the first two frames to determine the scale) are denoted as S. Consequently, the Reprojection error is denoted as R. and the Euclidean error as E. In the first experiment, it occurred that the scene was too demanding, or the found points were insufficient for Eigensolver and Nister algorithms to retrieve full trajectory in 3 out of 4 cases. It is possible, that those algorithms could restore trajectory if a different set of points was provided but regardless of this fact it shows that those algorithms are not as robust as rest of the investigated methods. With respect to rotational RPE RMSE errors, it is questionable if Kneip algorithm managed to retrieve the trajectory. The best of n-point relative pose algorithms was the 8-point, achieving best results with a constant scale, mainly if it was ignoring the translational part in RANSAC scheme. While this depends on the initialization stage and the detected points, the differences are not large. Still, it is outperformed by Kabsch algorithm, which is using full depth information, what is demonstrated in Fig. 1. This method combined with SURF detector/descriptor works twice better in ATE error terms than with other detector/descriptor pair.

Table 1. Comparison of absolute trajectory errors (ATE RMSE) and relative pose errors (RPE RMSE) for the putkk_Dataset_1_Kin_1

Algorithm	AKAZE			ORB			SURF		
	ATE [m]	RPE [m]	RPE [deg]	ATE [m]	RPE [m]	RPE [deg]	ATE [m]	RPE [m]	RPE [deg]
C.E.Eigensolver	7.267	0.021	0.372	7.292	0.021	0.372	7.261	0.021	0.372
C.E.Kneip	3.866	3.863	114.451	—	—	—	3.583	3.885	114.153
C.E.nister	—	—	—	—	—	—	—	—	—
C.E.eight-point	4.044	0.047	4.683	4.067	0.042	0.928	4.064	0.042	0.926
C.R.Eigensolver	—	—	—	—	—	—	—	—	—
C.R.Kneip	4.316	3.835	115.062	3.833	3.772	114.803	3.725	3.869	116.144
C.R.nister	4.090	0.283	9.031	3.759	0.230	6.189	4.123	0.355	10.094
C.R.eight-point	3.843	0.063	4.691	4.033	0.042	0.973	3.968	0.042	0.966
S.E.Eigensolver	2.043	0.014	0.372	2.044	0.014	0.372	2.032	0.014	0.372
S.E.Kneip	11.806	0.932	115.418	1.858	0.212	115.817	3.699	0.434	115.101
S.E.nister	—	—	—	2.044	0.014	3.900	1.974	0.014	4.375
S.E.eight-point	1.745	0.018	0.922	1.735	0.0176	0.9236	1.715	0.018	4.685
S.R.Eigensolver	—	—	—	—	—	—	—	—	—
S.R.Kneip	2.651	0.424	113.133	4.553	0.5737	113.945	2.869	0.277	114.681
S.R.nister	2.143	0.021	11.260	4.463	0.063	9.325	1.932	0.017	8.919
S.R.eight-point	1.861	0.025	0.956	1.768	0.023	0.963	1.699	0.021	0.955
Kabsch	0.724	0.019	0.413	0.714	0.020	0.430	0.470	0.020	0.417

The full comparison is further given in Table 1. In the second experiment, all but Nister coupled with AKAZE detector/descriptor were able to reconstruct trajectory. The best results were achieved by the Kabsch algorithm with the use of depth data. The best algorithm for finding a rotation from solely RGB data is 8-point algorithm if ATE error is considered but if the RPE metrics is evaluated, the Eigensolver seems to be a bit better. It seems as 8-point algorithm works slightly better with proposed in-RANSAC Euclidean error measurement than with reprojection error. This is not the case for Eigensolver algorithm, as it benefits if combined with SURF algorithm, and suffers if combined with AKAZE in the context of constant scale. Also, Eigensolver sometimes returns no solution and thus is not as reliable as the 8-point algorithm. Kneip algorithm has the biggest RPE error from all of the investigated algorithms. This is probably due to its inability to perform computations on more than 8 points. It can be assumed that it "reconstructed" trajectory, as Fig. 1 shows, when was working without depth data, having only a scale and reprojection error but this is not the case if it is used with depth information in centroid fashion or Euclidean error. Still, RPE errors are high. The reason for this are residual outliers, gaining a lot of importance in the final computation of the estimate. Other algorithms perform some form of least-squares optimization, and thus the residual outliers are strongly suppressed. The results for all investigated algorithms for fr3_long_office_household sequence are collected in Table 2 and enhanced with Fig. 2.

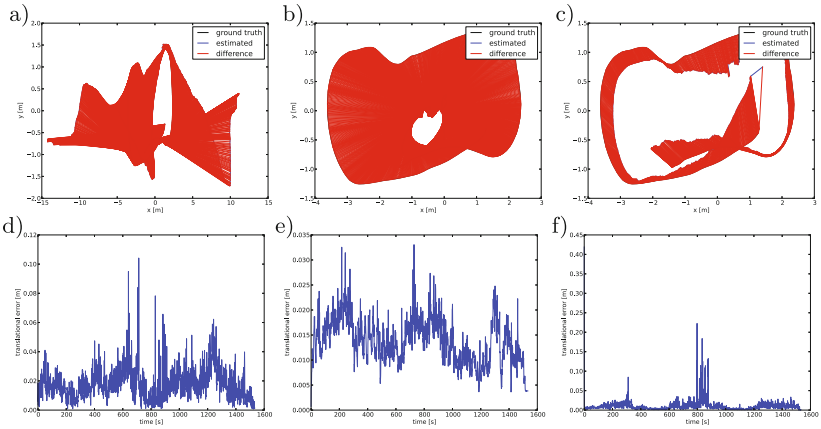

Fig. 1. Trajectories estimated for the putkk_Dataset_1_Kin_1 (a) ORB ATE C.E. Eigensolver, (b) SURF ATE S.E. Nister, (c) SURF ATE Kabsch, (d) ORB RPE C.E. Eigensolver, (e) SURF RPE S.E. Nister, (f) SURF RPE Kabsch

Table 2. Comparison of absolute trajectory errors (ATE RMSE) and relative pose errors (RPE RMSE) for the fr3_long_office_household sequence

Algorithm	AKAZE			ORB			SURF		
	ATE [m]	RPE [m]	RPE [deg]	ATE [m]	RPE [m]	RPE] [deg]	ATE [m]	RPE [m]	RPE [deg]
C.E.Eigensolver	1.683	0.376	19.313	1.822	0.320	11.743	1.790	0.302	11.743
C.E.Kneip	2.571	2.628	131.423	3.211	2.925	131.514	3.334	2.918	130.940
C.E.nister	—	—	—	2.066	1.224	43.236	1.636	1.128	35.555
C.E.eight-point	2.745	0.808	22.853	1.957	0.503	29.023	1.615	0.489	22.757
C.R.Eigensolver	1.467	0.371	18.975	2.290	1.275	48.219	3.211	0.924	46.888
C.R.Kneip	2.933	2.653	132.805	3.996	3.016	131.065	3.579	2.949	131.599
C.R.nister	2.028	1.131	45.637	2.459	1.842	73.429	2.316	1.138	42.402
C.R.eight-point	2.865	0.815	22.956	2.177	0.515	37.691	2.152	0.490	29.017
S.E.Eigensolver	1.972	0.260	105.122	1.984	0.269	11.743	1.913	0.286	11.743
S.E.Kneip	1.957	0.367	131.377	1.737	0.465	132.218	1.920	0.508	131.092
S.E.nister	—	—	—	1.857	0.349	42.416	1.600	0.400	33.549
S.E.eight-point	1.181	0.368	22.819	1.467	0.352	22.895	1.202	0.358	22.777
S.R.Eigensolver	1.803	0.244	21.157	1.998	0.253	49.985	1.935	0.269	49.098
S.R.Kneip	7.865	3.266	131.034	2.076	0.623	131.848	2.105	1.173	132.467
S.R.nister	1.916	0.345	46.895	2.140	0.543	71.752	1.869	0.513	39.916
S.R.eight-point	1.246	0.354	22.887	1.786	0.394	29.294	1.925	0.391	29.788
Kabsch	0.560	0.042	1.267	0.576	0.046	1.372	0.619	0.044	1.434

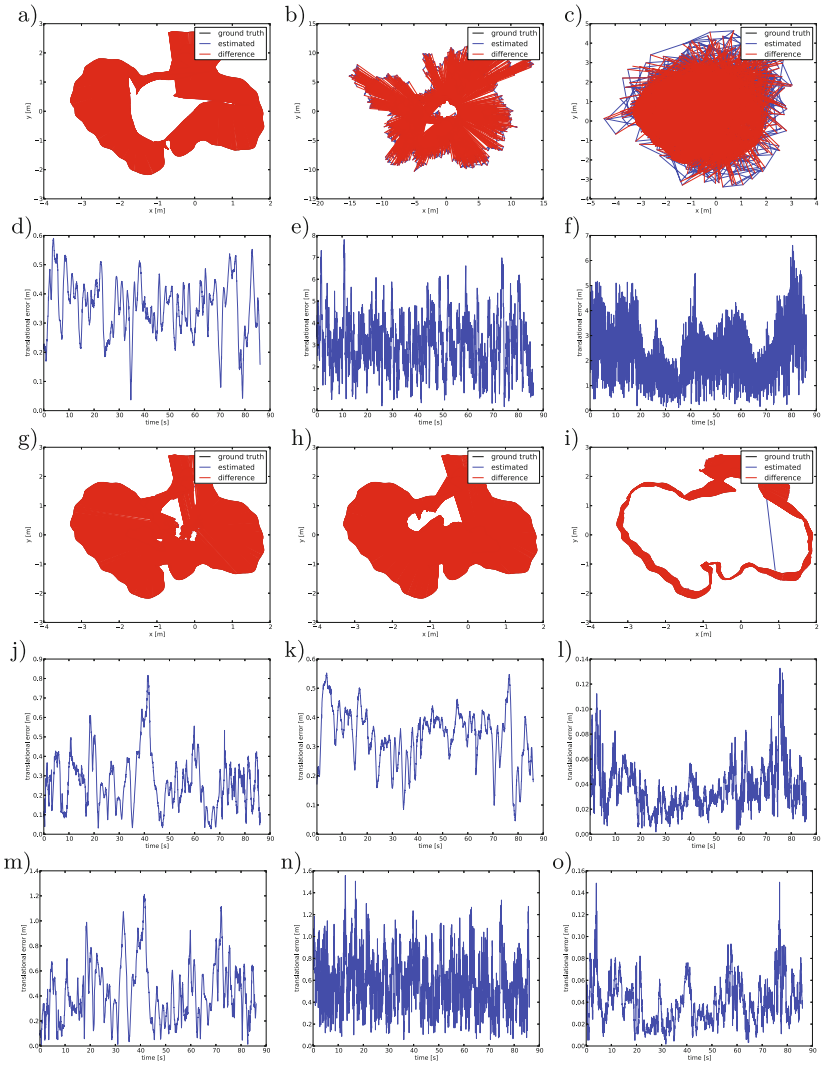

Fig. 2. Trajectories estimated for the fr3_long_office_household, (a) AKAZE ATE S.E. 8-point, (b) AKAZE ATE S.R. Kneip, (c) AKAZE ATE C.R. Kneip, (d) AKAZE RPE S.E. 8-point, (e) AKAZE RPE S.R. Kneip, (f) AKAZE RPE C.R. Kneip, (g) SURF ATE S.E. Eigensolver, (h) ORB ATE S.E. Nister, (i) AKAZE ATE Kabsch, (j) SURF RPE S.E. Eigensolver, (k) ORB RPE S.E. Nister, (l) AKAZE RPE Kabsch, (m) SURF RPE C.E. 8-point, (n) ORB RPE S.R. Kneip, (o) SURF RPE Kabsch

6 Conclusions

Taking into account the presented experimental results we can conclude, that the best algorithm for solving the frame-to-frame motion estimation problem is

the Kabsch algorithm, which however requires depth data, e.g. from an RGB-D sensor. In passive monocular systems, the 8-point algorithm won the competition, as the Eigensolver, although quite accurate, not always was able to deliver a solution. Whenever no depth data is available, the 8-point algorithm can restore a reasonable trajectory in comparison with other methods. The remaining algorithms usually performed by far worse than the two previously mentioned methods. The superiority of the 8-point algorithm may result from a better least-square optimization. Also, when working with real, imperfect data, the assumptions underlying the 8-point algorithm are more realistic than the stricter assumptions of the 5-point algorithm. It seems that the choice of the detector/descriptor has little influence compared to the error metrics used in RANSAC. There is no significant difference in performance resulting from using different detector/descriptor algorithms. The Kabsch algorithm in one scene works better with SURF, but in another scene, AKAZE and ORB perform slightly better. This shows that for different scenes, a different approach to select the salient point features is needed and that there is no perfect feature detector yet. It seems that the 8-point algorithm works better with AKAZE or SURF features than with ORB. The ORB is known from its fast computation and thus is considered for the computational efficiency, but its performance in our experiments was rather disappointing. Taking into account that AKAZE is an open-source software, we also advise considering this detector/descriptor pair instead of SURF. In our future work, we plan to consider also some detector/descriptor combinations, such as ORB-AKAZE, which seem to be interesting.

References

1. Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: IEEE International Conference on Computer Vision (ICCV) (2013)
2. Schmidt, A., Kraft, M., Fularz, M., Domagala, Z.: Comparative assessment of point feature detectors and descriptors in the context of robot navigation. J. Autom. Mob. Rob. Intell. Syst. **7**(1), 11–20 (2013)
3. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Trans. Pattern Anal. Mach. Intell. **26**, 756–770 (2004)
4. Nistér, D.: An efficient solution to the five-point relative pose problem, In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), vol. 2, pp. 195–202 (2003)
5. Stewénius, H., Engels, C., Nistér, D.: Recent developments on direct relative orientation. ISPRS J. Photogrammetry Remote Sens. **60**, 284–294 (2006). http://www.vis.uky.edu/ stewe/FIVEPOINT/
6. Belter, D., Nowicki, M., Skrzypczyński, P.: On the performance of pose-based RGB-D visual navigation systems. In: Computer Vision – ACCV 2014. LNCS, vol. 9004, pp. 407–423. Springer (2015)
7. Hartmann, W., Havlena, M., Schindler, K.: Visual gyroscope for accurate orientation estimation. In: Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision (WACV 2015), pp. 286–293. IEEE Computer Society, Washington, DC (2015)

8. Bradski, G.: Opencv_library. Dr. Dobb's J. Softw. Tools (2000). http://opencv.org/
9. Kneip, L., Furgale, P.: OpenGV: a unified and generalized approach to real-time calibrated geometric vision. In: Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China (2014)
10. Kostusiak, A.: The comparison of keypoint detectors and descriptors for registration of RGB-D data. In: Szewczyk, R. et al. (eds.) Challenges in Automation, Robotics and Measurement Techniques. AISC, vol. 440, pp. 609–622. Springer (2016)
11. Davison, A.J., Murray, D.W.: Simultaneous localisation and map-building using active vision. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 865–880 (2002)
12. Fraundorfer, F., Scaramuzza, D.: Visual odometry: Part I the first 30 years and fundamentals. IEEE Rob. Autom. Mag. **18**(4), 80–92 (2011)
13. Fraundorfer, F., Scaramuzza, D.: Visual odometry: Part II matching, robustness and applications. IEEE Rob. Autom. Mag. **19**(2), 78–90 (2012)
14. Murphy, L., Morris, T., Fabrizi, U., Warren, M., Milford, M., Upcroft, B., Bosse, M., Corke, P.: Experimental comparison of odometry approaches. In: Experimental Robotics: The 13th International Symposium on Experimental Robotics. Springer Tracts in Advanced Robotics, vol. 88, pp. 877–890 (2013)
15. Warren, M., McKinnon, D., He, H., Upcroft, B.: Unaided stereo vision based pose estimation. In: Australasian Conference on Robotics and Automation, Brisbane, ARAA (2010)
16. Longuet-Higgins, H.: Readings in Computer Vision: Issues, Problems, Principles, and Paradigms. Morgan Kaufmann Publishers Inc., San Francisco (1987)
17. Hartley, R.: In defense of the eight-point algorithm. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) **19**(6), 580–593 (1997)
18. Kneip, L., Siegwart, R., Pollefeys, M.: Finding the exact rotation between two images independently of the translation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2012)
19. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D., A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE RSJ International Conference on Intelligent Robots and Systems, Vilamoura, pp. 573–580 (2012)
20. Kraft, M., Nowicki, M., Schmidt, A., Fularz, M., Skrzypczyński, P.: Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect. Mach. Vis. Appl. **28**(1), 61–74 (2017)
21. Schmidt, A., Kraft, M., Fularz, M., Domagala, Z.: The registration system for the evaluation of indoor visual SLAM and odometry algorithms. J. Autom. Mob. Rob. Intell. Syst. **7**(2), 46–51 (2013)
22. Kabsch, W.: A solution for the best rotation to relate two sets of vectors. Acta Crystallogr. **32**, 922 (1976)
23. Kabsch, W.: A discussion of the solution for the best rotation to relate two sets of vectors. Acta Crystallogr. **A34**, 827–828 (1978)
24. Nghia, H.: Finding optimal rotation and translation between corresponding 3D points (2011). http://nghiaho.com/?page_id=671