Experimental Comparison of Open Source Vision-Based State Estimation Algorithms

A. Quattrini Li, A. Coskun, S. M. Doherty, S. Ghasemlou, A. S. Jagtap, M. Modasshir, S. Rahman, A. Singh, M. Xanthidis, J. M. O'Kane, and I. Rekleitis

Computer Science & Engineering Department, University of South Carolina, 315 Main, Columbia, SC, 29208, USA
{acoskun,dohertsm,sherving,ajagtap, modasshm,srahman,akanksha,mariosx}@email.sc.edu
{albertoq,jokane,yiannisr}@cse.sc.edu

Abstract. The problem of state estimation using primarily visual data has received a lot of attention in the last decade. Several open source packages have appeared addressing the problem, each supported by impressive demonstrations. Applying any of these packages on a new dataset however, has been proven extremely challenging. Suboptimal performance, loss of localization, and challenges in customization have not produced a clear winner. Several other research groups have presented superb performance without releasing the code, sometimes materializing as commercial products. In this paper, ten of the most promising open source packages are evaluated, by cross validating them on the datasets provided for each package and by testing them on eight different datasets collected over the years in our laboratory. Indoor and outdoor, terrestrial and flying vehicles, in addition to underwater robots, cameras, and buoys were used to collect data. An analysis on the motions required for the different approaches and an evaluation of their performance is presented.

Keywords: Vision based State Estimation, Localization, SLAM

1 Introduction

One of the most significant challenges in robot autonomy is state estimation, specifically the dual problems of tracking the pose of the robot as it moves through its environment and of mapping that environment as the robot moves. In the last decade, the wide availability of camera sensors, coupled with progress in computer vision, has given rise to a variety of vision-based techniques for these problems, known as *visual odometry* or *visual SLAM*. Scaramuzza and Fraundorfer [28, 11] presented a comprehensive overview this work, from the fundamentals of Visual Odometry to recent research challenges and applications. Fuentes-Pacheco *et al.* [12] recently surveyed Visual SLAM methods.

Vision based state estimation can be divided into a few broad appoaches. One line of research uses probabilistic filters, such as the Extended Kalman Filter (EKF), to fuse visual features with other data. For example, some influential works that fuse data from a camera and an inertial measurement unit (IMU) include those of Mourikis and Roumeliotis [25], Jones and Soatto [17], and Kelly and Sukhatme [18]. Another group of approaches builds on Structure from Motion (SfM) methods and Visual Odometry (VO), in which images are processed to extract features to be tracked, and the poses are estimated by minimizing the re-projection error deriving from the reconstruction of such tracked features. Such approaches include the work of Davison et al. [7] on real time accurate 3D structure reconstruction and motion estimation of a monocular camera, moving in a constrained indoor space. Konolige et al. [20], Furgale and Barfoot [13] have shown real-time visual odometry systems that are capable of accurately localizing terrestrial robots over tens-of-kilometers-long trajectories. Computationally expensive global optimization schemes, often termed bundle adjustment (BA) [31, 24], can also be used. BA can be further subdivided by whether features (sparse methods) or pixel intensities (direct methods) are considered for tracking.

In the recent years, several open source software packages for visual state estimation have become available, each supported by impressive demonstrations. However, the comparative evaluation of such methods, when available, is usually limited to only a few of them at a time, e.g., [32], making it difficult to select a reliable and robust method. Also, due to both algorithmic limitations, such as, number of and sensitivity to parameters, special initialization motions, *etc.*, and software engineering challenges, such as, diverse input formats, undisclosed software dependencies, *etc.*, applying these packages on new datasets can be remarkably difficult. In addition, several other research groups have presented superb performance without releasing the code, sometimes materializing as commercial products—e.g., [16], thus making it hard to evaluate and use.

The objective of this paper is to bring clarity to the landscape of visual state estimation software. Specifically, we evaluate eleven open source packages on eight new datasets. The datasets span a variety of environments (including indoor, outdoor, and underwater) and vehicle types (including terrestrial, airborne, marine surface, and underwater platforms). We present an analysis on the motions required for each approach, together with an evaluation of their performance on each dataset. The main contribution of this paper is to provide, based on this analysis, insights on which package to choose according to the problem at hand, and to highlight some of the open challenges that are still not fully addressed. Good practices for producing replicable results are also discussed.

This paper is structured as follows. The next section briefly describes the tested algorithms. Section 3 shows the datasets used in the evaluation. Section 4 presents the results and Section 5 discusses them, concluding the paper.

2 Methods Evaluated

Tables 1 and 2 list the open source vision-based state estimation packages analyzed in this paper together with a qualitative evaluation on different datasets. This section briefly introduces each of those methods, without any attempt to provide a comprehensive discussion of their details; please refer to the original papers for more information.

Kalman filter-based methods: **MonoSLAM** [5] is based on an incremental EKF, where the state contains the map and the camera pose. The state vector is updated with the prediction step, assuming a constant motion model that follows a Gaussian profile. The update is performed according to measurements derived from the detected features in the images. Feature points are detected with an active search algorithm that restricts the search space to the most probable area, according to a window and an estimated motion.

SfM-based methods: Packages that based on the Structure from Motion (SfM) approach include **libVISO** [15], which is a library that provides a sparse visual odometry method. Parallel Tracking and Matching (**PTAM**) [19], which is also a sparse method, is designed for augmented reality applications in *s*mall workspaces. It works with input images from a monocular camera. PTAM performs state estimation in two steps. First, a tracking phase, in which new frames are compared with the current map using features. Second, a map updating phase, which utilizes a set of keyframes. An initialization phase where the same features are seen from different point of views is required.

ORB-SLAM [26] is a monocular SLAM system, with a recent extension to stereo visual input, that uses ORB features for tracking, mapping, relocalizing, and loop closing. Semi-direct Visual Odometry (**SVO**) [10] extracts features only when a new keyframe is added to the map and matches the features in the successive frames as an implicit result of direct motion estimation. Outliers are filtered out with a Bayesian filter. Large-Scale Direct Monocular SLAM (**LSD-SLAM**), instead of using key-points, operates on intensities of images from a monocular camera [9], both for tracking and mapping, allowing a dense 3D reconstruction. Finally, **RatSLAM** [2] takes inspiration from the neural processes in rodent brains for navigation. Given images from a monocular camera and odometric information, the method matches scenes according to their appearance and constructs a semi-metric topological map.

Global optimization methods: Some of the above real-time solutions utilize global optimization packages to smooth the resulting trajectories. The open source packages: **g^{2}o** [21] and **Ceres** [1] are both graph optimization frameworks working with nonlinear error functions. They can model and efficiently solve large optimization problems.

A very recent solution that involves a complete visual pipeline is **COLMAP** [29] that allows a reconstruction of ordered or un-ordered sets of image. It utilizes the Ceres [1] framework over the whole set of images, resulting in impressive, albeit very slow, reconstructions of the camera trajectory and the environment.

3 Experimental Datasets

Although standard datasets are important for reproducibility and repeatability in experimental evaluation, existing datasets for state estimation typically capture only a single scenario, such as a university campus (e.g. Rawseeds [4]) or an urban environment (e.g. Kitti [14]).

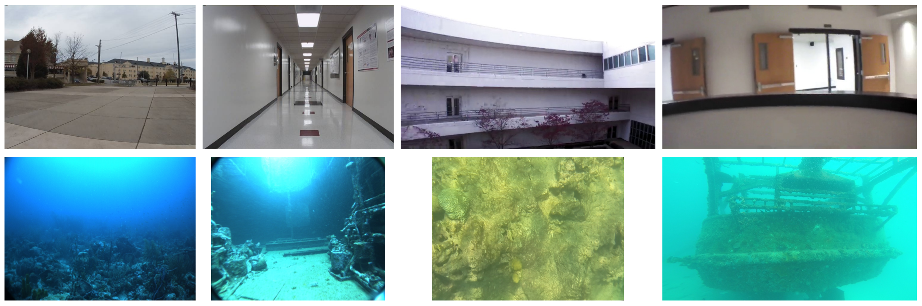


Fig. 1. Characteristic images from the evaluated datasets. Top row from left: UGV outdoors, UGV indoors, UAV outdoors, UAV indoors; Bottom row from left: AUV over a coral reef, AUV inside a wreck, Drifter, Camera moved manually underwater.

To test the visual state estimation packages discussed above on a richer set of scenarios, we collected datasets in the form of ROS bag files[1] in different environments using a diverse set of robotic platforms:

– UGV outdoor (H/Out) and indoor (H/In): A Clearpath Husky unmanned ground vehicle (UGV), equipped with GPS, IMU, and monocular camera (30fps, 640×480), moving both outside and inside a building at the University of South Carolina campus. The camera was mounted forward facing and lateral facing in different experiments.
– UAV outdoor (Q/Out) and indoor (Q/In): A Parrot AR-drone 2.0 quadrotor, with front (30fps, 640×360) and bottom cameras (60fps, 320×240) and an IMU, in the same environment as above. The forward facing camera was used for the evaluation. During the indoor experiments, the UAV experienced several abrupt rotations which resulted in loss of localization in most of the packages.
– AUV over coral reefs and inside a shipwreck: An Aqua2 autonomous underwater vehicle (AUV), equipped with an IMU and a forward facing camera (15fps, 870×520), operating off the coast of Barbados.
– Drifter: A custom made passive drifter [3] equipped with GPS, IMU, and a 10fps 640×480 camera, deployed also off the coast of Barbados. The camera is downward facing and the motion of the asset was caused only by the wave action. The bobbing motion of the camera resulted in an expanded field of view up to 120°. The low quality of the camera, the constantly changing lighting condition, and the continuous rotations made this dataset the most challenging among all.
– Manual underwater: A pair of GoPro Hero3+ cameras (30fps, 1920×1080) in a 3D Dual Hero System stereo configuration, deployed off the coast of Barbados. The stereo rig was operated by a diver inspecting inside and around shipwrecks and coral reefs.

[1] http://wiki.ros.org/Bags

Table 1. Qualitative Analysis: Performance of the different open source packages using the provided datasets from every other package. The legend is as follows: red–failure, i.e., the algorithm does not localize the robot with the tested parameters; orange–partial failure, i.e., the algorithm is able to track the robot in portions of the trajectory; yellow–partial success, i.e., the algorithm is able to track the robot until the end, but the trajectory contains some errors; green–success, i.e., the method produces an accurate trajectory.

Package-Dataset	[5]	[15]	[19]	[26]	[10]	[9]	[2]	[29]
MonoSLAM [5]	🟩	🟨	🟩	🟨	🟩	🟩	N/A	N/A
libVISO [15]	🟩	🟩	🟨	🟨	🟨	🟨	N/A	N/A
PTAM [19]	🟩	🟨	🟩	🟩	🟩	🟩	N/A	N/A
ORB-SLAM [26]	🟩	🟩	🟥	🟩	🟩	🟩	N/A	N/A
SVO [10]	🟥	🟥	🟩	🟨	🟩	🟩	N/A	N/A
LSD-SLAM [9]	🟩	N/A	🟩	🟩	🟩	🟩	N/A	N/A
RatSLAM [2]	🟥	🟥	🟩	🟨	🟧	🟧	🟩	N/A
ColMap [29]	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟩

The datasets together with detailed instructions on the usage of each package can be found online at `http://afrl.cse.sc.edu/afrl/resources/datasets/` so future packages could be tested and evaluated.

4 Results

The software packages described above were evaluated using the provided datasets from each package (cross-validation) and also the eight datasets discussed above. The tests were performed on a computer equipped with an Intel i7-4770 3.4 GHz CPU, 16 GB RAM, under Ubuntu 14.04 and ROS Indigo Igloo. The cameras were calibrated and the intrinsic parameters were provided to each package. In addition, the specific parameters of all packages were manually tuned for each dataset. The parameters were initially set to the package's default values, and tuned to improve the performance. All available suggestions from the packages' authors for parameter selection were followed. To test the global optimization frameworks, as they do not provide a complete SLAM system, input graphs were obtained by saving the pre-optimized resulting graph at the end of the best run of ORB-SLAM, which already relies on g^{2}o for local optimization. Repeated trials were conducted for each package-dataset pair; we report the best observed result for each pair from all the trials.

Table 1 shows a qualitative summary of the cross-validation experiments which test each package against the datasets provided by every other package. The cell colors indicate performance, utilizing the best parameters arising after extensive tuning. *Green* illustrates that the results were accurate. *Yellow* means that the robot was localized for the whole experiment, but the resulting trajectory deviated significantly from the general structure of the observed behavior. *Orange* shows that the method tracked the robot pose in some portions of the

Table 2. Qualitative Analysis: Performance of the different open source packages using the new datasets. Datasets: Husky outdoors (H/Out); Husky indoor (H/In); Quadrotor outdoor (Q/Out); Quadrotor indoor (Q/In); Aqua on coral reef (A/Out); Aqua inside wreck (A/In); drifters on coral reef (D/UW); GoPro stereo on the outside of a shipwreck (G/UW). The legend is as in Table 1.

Package	H/Out	H/In	Q/Out	Q/In	A/Out	A/In	D/UW	G/UW
MonoSLAM [5]	🟥	🟥	🟧	🟧	🟥	🟥	🟥	🟥
libVISO [15]	🟥	🟨	🟨	🟨	🟨	🟨	🟨	🟥
PTAM [19]	🟩	🟨	🟨	🟧	🟩	🟩	🟨	🟩
ORB-SLAM [26]	🟩	🟩	🟥	🟥	🟩	🟩	🟥	🟨
SVO [10]	🟧	🟧	🟥	🟧	🟧	🟥	🟧	🟧
LSD-SLAM [9]	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟥
RatSLAM [2]	🟨	🟧	🟩	🟧	🟧	🟩	🟥	🟨
COLMAP [29]	🟩	🟨	🟨	🟧	🟩	🟩	🟨	🟧
g^2o [21]	🟩	🟩	🟥	🟥	🟩	🟩	🟥	🟨
Ceres [1]	🟩	🟩	🟥	🟥	🟩	🟩	🟥	🟨

trajectory. *Red* indicates that the package was not able to localize the robot. The majority of the datasets provided have short duration, usually covering a small workspace inside a lab.

Table 2 presents a qualitative summary of the results from the eight diverse datasets collected by the authors. The same colors were used as in the previous table. In several occasions packages exhibited different performance in repeated trials under identical conditions. In all cases the best performance was used. In addition, for PTAM the datasets were tested to provide a starting point which resulted in initializing the tracking and the package was evaluated using the hand-tuned (trimmed) trajectory. Figure 2 shows examples of trajectories from the H/Out and H/In datasets, for each package rated Yellow or Green on that dataset.

Finally, Table 3 shows quantitative results evaluating the produced trajectory of each package for select datasets where a good estimate of the trajectory is available from other sources (GPS or LIDAR sensor). That trajectory is used as ground truth. In particular, for H/Out the GPS information is available, while for H/In the ground truth trajectory was obtained using gmapping[2] on the odometric, inertial, and LIDAR data. The metrics considered are:

Er the accuracy, measured in terms of error between ground truth and the produced trajectory [22]. In particular, the metric is based on the relative displacement between robot poses. More formally, the error of a trajectory $x_{1:T}$ with respect to the ground truth trajectory $x^*_{1:T}$ is calculated as:

$$\epsilon(\delta) = \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2$$

[2] `http://wiki.ros.org/gmapping`

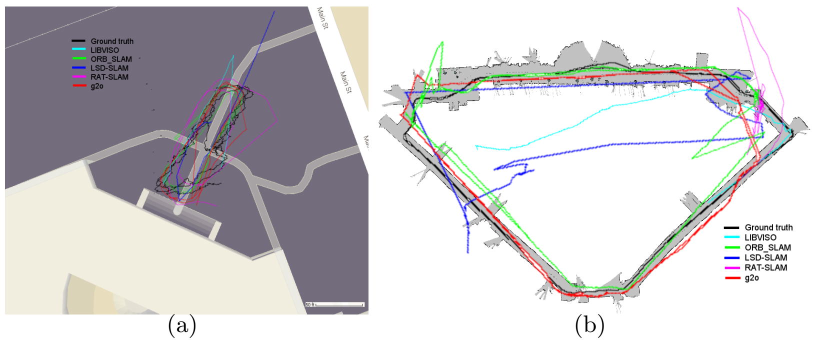


Fig. 2. Trajectories resulting from the tested methods in H/Out and H/In, together with the GPS trace (outdoor) and gmapping (laser-based) trajectory (indoor).

where $\delta_{i,j}$ and $\delta^*_{i,j}$ are the relative relation between two consecutive poses at time i, j for the estimated trajectory and ground truth trajectory, respectively, N is the number of relative relations, and *trans* considers the translation component. The error is reported in meters

TL track loss percentage, that is the ratio between the time in which the system is not localized and the total time of the dataset; lower numbers are better.

Mem the maximum amount of memory used by the package during a run; reported in megabytes (MB).

Note that as a monocular setup is considered in almost all the packages, a post-processing step is performed on the produced trajectory to fit/align to the ground truth minimizing the distance between them. In particular, the vision-based trajectory is rotated and scaled in order to coincide with the ground truth trajectory, at least at the starting moments. Some of the packages failed finding a trajectory, thus the resulting error displays a very large value. In H/Out$_1$, the robot traveled outdoor in the grass with bushes and trees, while in H/Out$_2$ the robot was moving on the sidewalk. Images in H/In were collected inside the Computer Science and Engineering department.

ORB-SLAM is the package that provides the best result in terms of accuracy among the sparse methods, and using g^{2}o at the very end of the dataset does not improve much the trajectory, highlighting its reliability. MonoSLAM is not able to localize the robot for most of the trajectory. Packages perform better in structured datasets (H/Out$_2$ and H/In) than unstructured ones, because features can be more easily identified. Memory usage does not show any specific pattern considering the different classes of visual SLAM methods, although for most of them it grows linearly over time. The difference between online and offline approaches is illustrated in Fig. 3 which shows the results from ORB-SLAM and COLMAP for one dataset collected outdoors using a Husky UGV. The global optimization method provides visually better results compared to the realtime one; however, it took more than a day for COLMAP to find the presented solution.

Table 3. Quantitative Evaluation of the different open source packages for the selected datasets with ground truth. Er measures the accuracy of the trajectory and is reported for packages which were partially successful, TL is the percentage of track loss, and Mem is the maximum memory usage. N/A stands for not applicable, e.g., calibration parameters were not reported for a dataset.

Package-Dataset		H/Out$_1$ tot. length: 413m			H/Out$_2$ tot. length: 438m			H/In tot. length: 413m		
		Er m	TL %	Mem MB	Er m	TL %	Mem MB	Er m	TL %	Mem MB
MonoSLAM	[5]	N/A	95.7%	73	N/A	90.6%	646	N/A	97.3%	102
libVISO	[15]	112.2	9.5%	155	98.7	3.0%	130	67.8	8.3%	165
PTAM[1]	[19]	33.4	7.6%	1543	24.0	15.9%	718	23.4	3.5%	437
ORB-SLAM	[26]	12.0	33.9%	5537	11.2	6.5%	2089	10.1	0.0%	4222
SVO[1]	[10]	36.7	18.8%	904	20.0	64.9%	244	18.0	63.5%	261
LSD-SLAM	[9]	38.8	0.1%	2728	27.6	12.0%	1376	15.1	78.6%	1067
RatSLAM	[2]	37.4	N/A	402	24.4	N/A	444	17.9	N/A	333
ColMap	[29]	23.7	N/A	N/A	9.2	N/A	N/A	29.6	N/A	N/A
g^2o	[21]	12.0	N/A	N/A	11.2	N/A	N/A	10.1	N/A	N/A

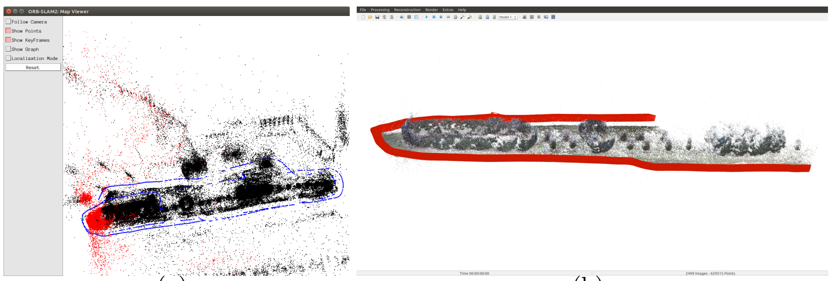
[1] The error reported is only for a large part of the trajectory

5 Main Experimental Insights

Comparing the behavior and performance of such a diverse set of vision based estimation packages provided multiple insights. One of the main challenges is to find the fine balance between computational efficiency and result accuracy. Many parameters, such as the number of tracked features and the number of RANSAC iterations, can improve the accuracy at the expense of added computational load. A slight change in some of the parameters could lead to very different behaviors.

Some of the packages, such as SVO, restricted the operating space to a small area during their demonstrations. This allows the method to produce very good trajectories in a limited workspace, as it is possible to run a global optimization algorithm on all of the keyframes in the map. As a result, SVO was only able to track the trajectory in the tested datasets partially. The cross validation and the new datasets results show that in testing a proposed approach more challenging scenarios should be considered to validate it. Indeed, most of the attached datasets with the packages are from experiments performed inside a single laboratory, many times just over a single desk.

The images' quality is another important factor influencing the results. The quality depends on the amount of texture in the images, illumination variations, and the presence of blur, both out of focus and motion blur. As most packages tested rely on tracking features, the quality of the detected features depends on the image quality. For example, sharp rotations are a type of motion that authors of some packages, such as ORB-SLAM, suggested to avoid as it could result in losing track of the detected features. As a matter of fact, the most successful



(a) (b)

Fig. 3. Resulting trajectories/reconstruction from (a) ORB-SLAM; (b) COLMAP.

package, ORB-SLAM, failed for Q/Out, which contains continuous rotations. Moreover, many packages failed in the underwater datasets, due to the difficult visual conditions, which led to features not detected and also to several wrong loop closures. This is especially true for the dataset from the drifting sensors, in which the camera has the lowest frame-rate compared to the other datasets.

Note that, since monocular cameras cannot recover depth from a single frame, one open issue affecting the performance of methods working with monocular images is the initialization step. Some packages explicitly reported a required initial motion to initialize the SLAM algorithm. In many vehicles such motion might not be feasible for the robotic platform —e.g., PTAM requires an initial translation along the x-axis of the camera, however, many robotic platforms have forward-facing cameras to enable navigation and lateral motion is not possible. In H/Out, PTAM succeeded, because the camera was rotated to face laterally.

Furthermore, for several online packages, an inconsistent behavior was observed in the results between successive runs of the same dataset with the same parameters; a behavior reported in the papers. For example, H/In resulted in repeated failures of ORB-SLAM before producing an accurate trajectory and scene reconstruction. There are several causes, including the realtime constraint, where some of the frames could be dropped according to the load of the computing unit, and the random nature of RANSAC.

RatSLAM, utilizes a learning process for adjusting how neurons are triggered, thus improving the trajectory as the robot visits the same place multiple times; e.g., in Q/Out it is able to produce a good result, given the spiral motion.

Global optimization methods improve the resulting trajectory; e.g., running g^{2}o on the complete graph from ORB-SLAM on H/In, the χ-squared test showed an improvement from $\chi^2 = 183068$ to $\chi^2 < 10^{-9}$. However, being an expensive operation, ORB-SLAM usually runs g^{2}o only on a fixed number of keyframes. It is interesting to note that, if a general optimization frameworks is tailored for a specific package, such as in ORB-SLAM, the number of iterations required for convergence drops—e.g., for g^{2}o used in conjunction with ORB-SLAM, it takes on average in the order of tens of iterations, while using Ceres "straight out of the box" takes tens of thousands of iterations. COLMAP, which provides a complete pipeline for SfM problems utilizing Ceres, shows very promising results, although the time to get the estimated trajectory can be very long—e.g., for 700 images, 7-8 hours.

In addition to the packages reported above, several more packages were tested. In particular, preliminary tests of the following global optimization packages: Bundler [30], SBA [24], parallaxBA [33], and GTSAM [8] did not produce acceptable results. In particular in most cases they failed to reliably track features for most of the datasets and the global optimization converged into a local minima. Ongoing work includes the study of the effects of changing parameters, collection of data focusing on different type of motions, and the investigation of more open-source packages on the same datasets, including DTAM [27], DPP-TAM [6], OKVis [23].

References

1. Agarwal, S., Mierle, K., Others: Ceres Solver. `http://ceres-solver.org` (2015)
2. Ball, D., Heath, S., Wiles, J., Wyeth, G., Corke, P., Milford, M.: OpenRatSLAM: an open source brain-based SLAM system. Auton. Robot. **34**(3) (2013) 149–176
3. Boydstun, D., Farich, M., III, J.M., Rubinson, S., Smith, Z., Rekleitis, I.: Drifter sensor network for environmental monitoring. In: 12th Conf. on Computer Robot Vision. (Jun. 2015) 16–22
4. Ceriani, S., Fontana, G., Giusti, A., Marzorati, D., Matteucci, M., Migliore, D., Rizzi, D., Sorrenti, D.G., Taddei, P.: RAWSEEDS ground truth collection systems for indoor self-localization and mapping. Auton Robot **27**(4) (2009) 353–371
5. Civera, J., Grasa, O.G., Davison, A.J., Montiel, J.M.M.: 1Point RANSAC for Extended Kalman Filtering: Application to Real-time Structure from Motion and Visual Odometry. Journal of Field Robotics **27**(5) (2010) 609–631
6. Concha, A., Civera, J.: DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. (2015)
7. Davison, A., Reid, I., Molton, N., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Tran. on Pattern Analysis and Machine Intelligence **29**(6) (jun. 2007) 1052 –1067
8. Dellaert, F., Kaess, M.: Square Root SAM: Simultaneous localization and mapping via square root information smoothing. The Int. Journal of Robotics Research **25**(12) (2006) 1181–1203
9. Engel, J., Schps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., eds.: European Conf. on Computer Vision (ECCV). Volume 8690 of Lecture Notes in Computer Science. Springer Int. Publishing (2014) 834–849
10. Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: Fast semi-direct monocular visual odometry. In: IEEE Int. Conf. on Robotics and Automation. (2014) 15–22
11. Fraundorfer, F., Scaramuzza, D.: Visual odometry: Part II: Matching, robustness, optimization, and applications. IEEE Robotics & Automation Magazine **19**(2) (2012) 78–90
12. Fuentes-Pacheco, J., Ruiz-Ascencio, J., Rendón-Mancha, J.M.: Visual simultaneous localization and mapping: A survey. Artificial Intelligence Review **43** (2015) 55–81
13. Furgale, P.T., Barfoot, T.D.: Stereo mapping and localization for long-range path following on rough terrain. In: ICRA. (2010) 4410–4416

14. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets Robotics: The KITTI Dataset. The Int. Journal of Robotics Research **32**(11) (2013) 1231–1237
15. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: Dense 3d reconstruction in real-time. In: Intelligent Vehicles Symposium (IV). (2011)
16. Hesch, J., Kottas, D., Bowman, S., Roumeliotis, S.: Consistency Analysis and Improvement of Vision-aided Inertial Navigation. IEEE Tran. on Robotics **30**(1) (2014) 158–176
17. Jones, E.S., Soatto, S.: Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. The Int. Journal of Robotics Research **30**(4) (2011) 407–430
18. Kelly, J., Sukhatme, G.S.: Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. The Int. Journal of Robotics Research **30**(1) (2011) 56–79
19. Klein, G., Murray, D.: Parallel tracking and mapping for small ar workspaces. In: IEEE and ACM Int. Symp. on Mixed and Augmented Reality. (2007) 225–234
20. Konolige, K., Agrawal, M., Solà, J.: Large scale visual odometry for rough terrain. In: Int. Symposium on Research in Robotics (ISRR). (November 2007)
21. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g^2o: A general framework for graph optimization. In: IEEE Int. Conf. on Robotics and Automation. (2011) 3607–3613
22. Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., Kleiner, A.: On measuring the accuracy of SLAM algorithms. Autonomous Robots **27**(4) (2009) 387–407
23. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. The Int. Journal of Robotics Research **34**(3) (2015) 314–334
24. Lourakis, M.A., Argyros, A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Trans. Math. Software **36**(1) (2009) 1–30
25. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: IEEE Int. Conf. on Robotics and Automation, IEEE (2007) 3565–3572
26. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Trans. Robot. **31**(5) (2015) 1147–1163
27. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense Tracking and Mapping in Real-time. In: Int. Conf. on Computer Vision. ICCV, Washington, DC, USA, IEEE Computer Society (2011) 2320–2327
28. Scaramuzza, D., Fraundorfer, F.: Visual odometry [tutorial]. IEEE Robotics Automation Magazine **18**(4) (2011) 80–92
29. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: IEEE Conf. on Computer Vision and Pattern Recognition. (2016)
30. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. The Int. Journal of Computer Vision **80**(2) (2008) 189–210
31. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle Adjustment — A Modern Synthesis. In: Vision Algorithms: Theory and Practice: Int. Workshop on Vision Algorithms, Corfu, Greece (2000) 298–372
32. Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., Tardós, J.: A comparison of loop closing techniques in monocular SLAM. Robotics and Autonomous Systems **57**(12) (2009) 1188–1197
33. Zhao, L., Huang, S., Sun, Y., Yan, L., Dissanayake, G.: Parallaxba: bundle adjustment using parallax angle feature parametrization. The Int. Journal of Robotics Research **34**(4-5) (2015) 493–516