Camera-Agnostic Monocular SLAM and Semi-dense 3D Reconstruction

Martin Rünz(✉), Frank Neuhaus, Christian Winkens, and Dietrich Paulus

University of Koblenz-Landau, Mainz, Germany
martin.runz.15@ucl.ac.uk

Abstract. This paper discusses localisation and mapping techniques based on a single camera. After introducing the given problem, which is known as *monocular SLAM*, a new **c**amera **a**gnostic **m**onocular SLAM system (CAM-SLAM) is presented. It was developed within the scope of this work and is inspired by recently proposed SLAM-methods. In contrast to most other systems, it supports any central camera model such as for omnidirectional cameras. Experiments show that CAM-SLAM features similar accuracy as state-of-the-art methods, while being considerably more flexible.

1 Introduction

The term *simultaneous localisation and mapping* (*SLAM*) denotes the process of creating or refining a map, while determining the own position at the same time. SLAM techniques that rely on vision are referred to as *visual-SLAM* techniques and their performance increased remarkably in the last decade. This work concentrates on *omnidirectional monocular* visual-SLAM systems, which are SLAM systems that deploy a single camera[1] with a wide field of view. Such a vision system would correspond to that of a rabbit or horse, as the eyes of these animals also maximise their field of view at the expense of having monocular vision. There are several reasons for advocating the use of omnidirectional sensors in computer vision. First, many algorithms with applications to robotics detect visual landmarks. These landmarks will be present in many images when an omnidirectional camera is employed, resulting in a higher robustness. Second, a vision system with a high field of view collects more data than a traditional vision system in the same amount of time, although at a lower resolution. This is particularly useful in robotics or when teleconferencing or performing surveillance. The main contribution of this work is the design of a new SLAM system for omnidirectional cameras. In fact, the newly created system makes little assumptions about the camera model and supports traditional, omnidirectional and every single-view camera model[2]. For this reason, it is called ***c**amera-**a**gnostic **m**onocular SLAM* – CAM-SLAM. Furthermore, CAM-SLAM supports arbitrary salient image features and is also capable of reconstructing the environment semi-densely.

[1] Or a set of cameras with non-overlapping images.

[2] As long as it is possible to extract and track salient image features.

B. Rosenhahn and B. Andres (Eds.): GCPR 2016, LNCS 9796, pp. 285–296, 2016.
DOI: 10.1007/978-3-319-45886-1_23

1.1 Related Work

A substantial amount of the theoretic principles involved in omnidirectional machine vision has been established during the 1990s [1,9], followed by the implementation of thorough omnidirectional vision systems [2,22]. The first decisive monocular SLAM systems emerged in the subsequent decade and naturally inspired omnidirectional monocular SLAM systems.

In a seminal work [5] Andrew Davison applied EKF-SLAM to the domain of computer vision and his method is able to execute mapping as well as tracking in real-time. The system, however, was limited in map size and required noticeable user interaction. These drawbacks were overcome by Klein and Murray in their seminal PTAM (**p**arallel **t**racking **a**nd **m**apping) publication [13]. In contrast to previous work, Klein and Murray separate the mapping and tracking task, which allows fast tracking, while the map is refined using bundle adjustment. An alternative approach is to reconstruct a dense 3D map and to use this for tracking, as proposed by Newcombe et al. [21]. Such methods have the advantage that more data of the image is effectively used by the SLAM system and researchers concluded [6,21] that they are more robust as well as more accurate than keypoint-based methods. Both techniques, are still subjects of active research and in the middle of 2015 the state-of-the-art method in terms of accuracy – ORB-SLAM [19] – was a keypoint-based method.

There has been a trend in recent years to compute visual odometry, i.e. to estimate the robot motion, using omnidirectional vision [14,27,28]. The advantage of estimating motion with omnidirectional sensors lies in the computation of the rotational motion component. While rotations are usually a burden when working with traditional cameras, all methods formerly mentioned exploit omnidirectional peculiarities to compute rotation. Although some of the systems, such as the one of Scaramuzza and Siegwart [27], exhibit a high accuracy, their estimation will diverge from the ground truth eventually. This happens because small errors accumulate over time, resulting in an inevitable drift. Furthermore, some of the systems impose strong constraints to the environment.

In contrast to visual odometry approaches, researchers have also adapted full-fledged SLAM systems to the domain of omnidirectional vision, often based on EKF-SLAM [10,23,26] or Fast-SLAM [7]. This might be due to the fact that landmarks are longer visible when an omnidirectional sensor is used. Hence, the drawback that EKF-SLAM scales badly with respect to map size is less prominent. But these methods would not be applicable for a large-scale SLAM system nevertheless. Even though Fast-SLAM performs more efficiently than EKF-SLAM, their processes of tracking and mapping are strongly coupled. This neglects a more profound map optimisation, since tracking has to be executed at framerate. Other methods, again, put strict assumptions on the environment. While Burbridge et al. [3], for example, only maintain a 2D map of the environment, Schoenbein and Geiger [30] assume an urban Manhattan world and Scaramuzza and Siegwart [27] presume planar motion. Maxime Lhuillier presented a generic offline structure from motion system in 2006 [15] that even

supports non-central cameras. While its level of abstraction is impressive, it is solving a slightly different problem as no real-time constraint is imposed.

2 System Overview

In order to support consumer quality omnidirectional cameras, CAM-SLAM relies on a parallel tracking and mapping bundle-adjustment strategy inspired by PTAM [13] and ORB-SLAM [19]. It internally creates separate *tracking* and *mapping threads*. An additional *system thread* is created, which handles communication with the user thread and performs some computations in order to unload the tracking thread. Being able to semi-densely reconstruct the environment, CAM-SLAM starts at least one more *reconstruction thread*. When compiled with OpenMP support, which is optional, this number can be increased arbitrarily. Figure 1 illustrates the data flow of a single image frame.

After salient image features were created and assuming that the map is already initialised, the tracking thread takes over new frames. Tracking involves bundle-adjustment of the locally visible map, which is accessible by maintaining a *covisibility graph*, as suggested by Mei et al. [17]. In the case that tracking was successful, the mapping thread may further process the frame by converting it to a keyframe, if required by the system. Here, the decision of the system is based on the conditions proposed by Mur-Artal et al. [19]. Keyframes are used during mapping and are subject to more profound optimisation. In this process, new map points are triangulated, bad ones are removed and existing ones are

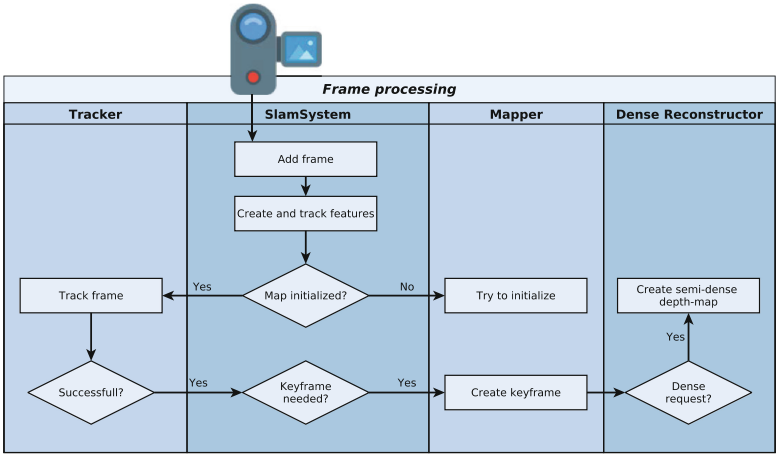


Fig. 1. Frame processing flowchart. While the *Tracker*, *Slam System* and *Mapper* lane each map to one thread in the application, the *Dense Reconstructor* lane maps to at least one thread. New frames arrive in the system thread and are subsequently processed by the tracking, mapping and semi-dense reconstruction thread, according to the system state.

refined with bundle-adjustment. Optionally, the system performs a semi-dense reconstruction of the environment, either online or offline – the latter to increase accuracy.

3 Camera Models

In general, camera models define two projections: From Cartesian 3D coordinates to pixel image coordinates, and vice-versa, from pixel coordinates to a 3D direction. Most visual SLAM systems employ cameras that can be represented with the pinhole camera model. There is, however, a variety of alternative camera models [34] and CAM-SLAM aims at supporting as many models as possible. For this reason it treats the actual camera model as a black-box and solely requires a uniform interface of shared characteristics, namely the aforementioned bidirectional 2D-3D mapping. Additionally, for some camera models one has to implement a function $d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$ computing the offset between two image space coordinates, which can differ from a simple subtraction. In order to support a wide range of camera models out of the box, CAM-SLAM implements the model by Scaramuzza et al. [29], Mei and Rives [16], an equiangular and cylindrical model, and the pinhole camera model. While optimising, each model is processed in exactly the same way, usually by employing numerical optimisations.

4 Mapping

A central role of the mapping thread is to generate new map points. This is accomplished by identifying and triangulating corresponding salient image features of different keyframes. Some popular triangulation methods, like *linear triangulation* [11], are restricted to the pinhole camera model, though. Methods that allow arbitrary camera models include angular triangulation [25], midpoint triangulation and triangulation based on angular disparity [2]. During experiments, triangulation based on angular disparity was rejected rather quickly, as it is more expensive than midpoint triangulation and did not perform better. Tests revealed that CAM-SLAM is able to execute stably with midpoint as well as angular triangulation. This is because triangulated map-point positions only serve as an initialisation and get optimised by bundle-adjustment, whenever a new keyframe is created – including the moment of map initialisation. As long as a triangulated map point is not rejected as an outlier, which happens more frequently with midpoint triangulation, its position will be optimised by the mapper. With respect to efficiency, midpoint triangulation executes around 60× faster than angular triangulation, but both methods are real-time capable. In order to gain robust triangulation, the following pre- and post-triangulation (distinguished by ◀, ▶) checks are performed for each potential map point:

◀ The angle between the back-projections of both image coordinates that are subject to triangulation, has to be larger than a threshold λ_α. This way,

map points with a high depth uncertainty are effectively rejected. An alternative approach is to parametrise map points by their inverse depth, as discussed in [4].

- Inspired by ORB-SLAM, the scale consistency is confirmed by verifying that the ratio of distances to both camera centers corresponds to the feature pyramid levels. For instance, when the features are on the same level, the ratio has to be in a range close to 1.
- The projection of the newly created map point to both image spaces $\boldsymbol{u}_{1,2}$ has to be valid.
- The error between the keypoint positions and $\boldsymbol{u}_{1,2}$ has to be lower than a threshold λ_e which is based on the image-space covariance of the camera model.

Notice that in contrast to regular visual SLAM methods, a maximal re-projection error is enforced instead of the epipolar constraint[3]. Enforcing the epipolar constraint has been avoided, as it would be dependent on a specific camera model and possibly expensive to evaluate.

When CAM-SLAM is newly started or when it lost tracking, it tries to initialise mapping. In the course of this, an essential matrix estimation is performed in a RANSAC-fashion. Even if this approach is less usual than the fundamental matrix estimation, it again offers the advantage of being camera model independent – as long as it is central. The first step of map initialisation is to select a reference keyframe, usually the first input frame. Afterward, each new frame is treated as a potential keyframe and keypoints are matched to the reference keyframe, which is necessary to employ the 8-point algorithm. The matching is based on a feature tracker that locally searches the best matching descriptor in two immediately consecutive frames for each previously matched keypoint of the older frame, and when a match succeeds it is back-propagated to the reference keyframe. This can be understood as a survival of the fittest scheme, as the number of keypoints available for back-tracing is monotonically decreasing. When the number of matches falls below a threshold, a new reference keyframe is selected.

Due to the local search, camera movements should not be too aggressive until being initialised. It is common practice to pre-define a threshold for good features in order to remove outliers. Since this reduces flexibility – thresholds are only valid for one descriptor type and could also be scene dependent – CAM-SLAM follows another strategy. It is assumed that keypoints that were successfully traced back several frames tend to correspond to good matches and that after $\Lambda_i = 4$ consecutive frames without switching the reference keyframe, descriptor distances are approximately normally distributed around a good threshold. Then, after observing Λ_i consecutive frames, the threshold λ_f is learned based on a generous χ^2 test. As soon as the map is initialised, another χ^2 test is performed to refine the threshold. This scheme has been tested successfully with binary ORB and real-valued SURF descriptors.

[3] One could argue that this is an implicit epipolar check, since the re-projected position is located on the epipolar line.

When the map is successfully initialised, the mapper performs local bundle-adjustment to optimise map points and keyframe positions. The implementation is based on the g^{2}o graph optimisation framework and adopts a double window approach [32]. As with ORB-SLAM, the optimisation process is performed twice: At a coarse scale first, that is with few Levenberg-Marquardt iterations, and at a finer scale after outliers are removed.

Map points are created whenever a new keyframe is created. This time, point correspondences are found using FLANN [18], and if applicable, also using the tracking method employed during initialisation.

5 Tracking

The tracking-component is responsible for locating the camera in space and has to operate as fast as possible. Should a new image frame arrive before the last one is processed, the new frame is skipped. Initially, the pose of a new frame is estimated using a constant motion model that also incorporates skipped frames. A more sophisticated guess would be possible, for instance with Kalman filtering, but the constant model showed to be sufficient.

Given the initial estimate, feature matching is performed with the currently selected keyframe by projecting available map points to the new frame. This is possible because the keyframe has keypoints associated to map points and map points, as well as the new frame, are already located in 3D space. The distance between the projected and the actual keypoint position has to fall below a threshold however, and the descriptor distance is tested using the threshold learned during map initialisation. If the number of successful matches is lower than the threshold λ_m, the new frame is labeled as untrackable. In this case, the system tries to create a new keyframe as fast as possible, because this could assist future tracking. After receiving Λ_t untrackable frames in a row, tracking is set to be lost. During experiments $\Lambda_t = 1$ was specified. Hence, tracking was lost at the first untrackable frame.

In the case that enough matches are found, bundle-adjustment is performed to refine the initial pose estimate. This process is again divided into two steps – first, before outlier removal; and second, with outlier removal. In contrast to the bundle-adjustment of the mapping-component, map point positions are unaltered and fewer points are used.

Finally, as with ORB-SLAM, the tracker requests a new keyframe when less than $\Lambda_v N_{fk}$ map points associated to the keyframe were found in the frame, where N_{fk} is the number of map points in the keyframe and $0 < \Lambda_v < 1$, here $\Lambda_v = 0.9$. This can be understood as a visual change condition.

5.1 Semi-dense Reconstruction

A comprehensive representation of the environment is crucial for mobile robots and CAM-SLAM implements a semi-dense reconstruction method for this purpose. The procedure is based on the one introduced by Mur-Artal and Tards [20],

but a generalised algorithm it presented, which only assumes a central camera model. Given this assumption, epipolar line searches are possible, but the parametrisation of these lines is not predefined. Capturing hyperbolic or parabolic mirrors yields conical epipolar lines, while the cylindrical model results in sinusoid, for instance. Our matching, depth estimation and depth fusion follows the approach of Mur-Artal and Tardos, but the epipolar lines are sampled using a line simplification scheme, following the same principle as the Ramer–Douglas–Peucker [24] algorithm. Given the target keyframe K with the inverse depth range $d_{\min} < d_k < d_{\max}$, a reference keyframe K_i and a pixel coordinate $\boldsymbol{u}$ of interest, the back-projection of $\boldsymbol{u}$ at depth $d_{\min}$ and $d_{\max}$ is projected to the image space of K_i and referred to as $\boldsymbol{v}_{1,2}$. Here, the extremes $d_{\min}$ and $d_{\max}$ are chosen as the 5 %- and 95 %-quantiles of the depth values of all map points associated with K, which effectively narrows the search and removes outliers. The direct line between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is then sub-sampled by projecting intermediate 3D points of the declared depth range to K_i. Sub-sampling is repeated iteratively until subsequent line-segments are nearly straight or until the length of segments falls below a threshold. After completing the semi-dense inverse depth-map, hole filling and regularisation are applied to improve the quality, similarly as in LSD-SLAM [6]. While LSD-SLAM averages those depth values in the window, however, that are not further away than 2σ from the current one. CAM-SLAM, on the other hand, performs a χ^2 test at 95 %, which is effectively the same, but allows to avoid some computational expensive divisions in the implementation. Also, CAM-SLAM checks image intensity consistency during hole filling and regularisation. A result of the triangulation procedure is shown in Fig. 2d.

6 Experiments

In order to evaluate the performance of CAM-SLAM in terms of accuracy, speed and robustness, a variety of comparisons to ground-truth data have been carried out. As flexibility is the major objective of CAM-SLAM, experiments have been executed on diverse datasets and omnidirectional data has explicitly been included. The used datasets are characterised as follows:

1. TUM-RGBD datasets [33]: Pinhole camera, handheld
2. KITTI datasets [8]: Pinhole camera, car-motion
3. Catadioptric RGB-D dataset [31]: Catadioptric camera, car-motion
4. *V360 dataset*: Cylindrical camera, handheld
5. *Room dataset*: Equiangular camera, synthetic images and motion
6. *Mars dataset*: Equiangular camera, synthetic images and motion

While datasets 1–3 are publicly available, sets 4–6 were created within the scope of this work. To capture set 4, a cylindrical camera was rigidly coupled with a marker and externally tracked. Datasets 5 and 6 were generated using the computer graphics software Blender with an equiangular camera model. Synthetic datasets have the advantage of providing perfect ground-truth trajectories,

but introduce appearance-related difficulties. For instance, while most real surfaces exhibit structure due to imperfections, synthetic ones can be completely smooth, and hence, featureless. The setup for acquiring set 4, as well as renderings of the synthetic sets are shown in the supplementary thesis.

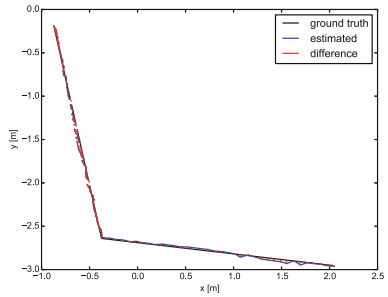
6.1 Accuracy

Experiments are grouped in two categories: Small- and large-scale experiments. A good measure of performance for the former category is the *absolute trajectory error* (ATE), as described by Sturm et al. [33]. It is computed by determining the absolute differences of camera positions, after aligning the ground-truth with the SLAM-produced trajectory. Here, either a 6D- or 7D-alignment is performed, for example, using the method of Horn [12]. The absolute trajectory error is less suited for large-scale data, however. Imagine a SLAM system always produced the perfect trajectory, but failed in one curve so that every subsequent position had a major offset to the ground truth. Compare this to the case in which a system continuously produces errors, but luckily, did not misinterpret the rotation. In a large-scale scenario, the second system might obtain an ATE which is several orders of magnitude less than the system that made one mistake only. As CAM-SLAM – just like other monocular SLAM-methods – suffers from scale-drift, only segments of large-scale datasets are investigated. For a more profound evaluation on large-scale data, loop-closing would be required, which has not been performed during experiments.

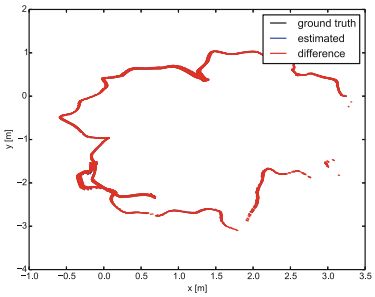
Table 1 compares the *root mean squared* ATE produced by CAM-SLAM to the ones produced by PTAM, LSD-SLAM and ORB-SLAM. Sequences starting with *fr...* belong to the popular TUM-RGBD benchmarking datasets. While the sequences *fr1_xyz* and *fr2_desk* present a static environment with a moving camera, a person is interacting in sequence *fr2_desk_person*. Processing non-static environments is difficult for visual SLAM systems, which is shown in PTAM failing to handle the scene and LSD-SLAM producing large errors. Figures 2b and c

Table 1. Comparison of root mean square absolute trajectory errors in cm, using small-s cale datasets. With the exception of CAM-SLAM measurements, the data is provided by Mur-Artal et al. [19]

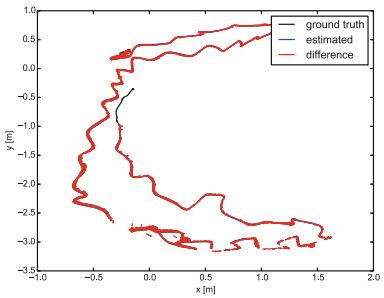
	CAM-SLAM	PTAM	LSD-SLAM	ORB-SLAM
fr1_xyz	2.60	1.15	9.00	**0.90**
fr2_desk	3.53	×	4.57	**0.88**
fr2_desk_person	2.29	×	31.73	**0.63**
fr3_long_office	16.83	×	38.53	**3.45**
synth_room	**2.84**	×	×	×
v360 sequence1	**16.21**	×	×	×
v360 sequence2	**14.92**	×	×	×



(a) Synthetic room dataset (5)



(b) fr2_desk, dataset (1)



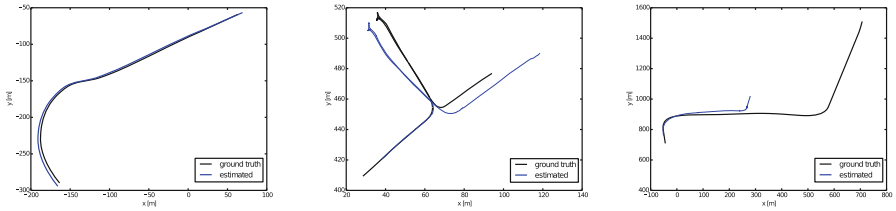
(c) fr2_desk_person, dataset (1)

(d) Reconstruction of fr2_desk

Fig. 2. Plots that highlight the differences between CAM-SLAM estimated trajectories and ground-truth trajectories. While ground-truth trajectories are black, estimated trajectories are blue and differences between associated positions are plotted red. Gaps in the graphs correspond to missing associations, mainly because ground-truth data is missing. Figure (d) shows the result of a semi-dense reconstruction of the environment (Color figure online)

show related plottings of the ground-truth and estimated trajectories as well as their differences.

The results presented in Table 1 show that CAM-SLAM has a comparable performance as state-of-the-art methods. It outperformed LSD-SLAM in all test-sequences and it was robuster than PTAM. ORB-SLAM, on the other hand, consistently produced better results than CAM-SLAM, which is not surprising as it is more finely tuned with respect to the camera model and keypoint descriptors. Furthermore, ORB-SLAM performs loop-closing, while CAM-SLAM did not in the experiments. Nevertheless, CAM-SLAM benefits from its flexibility. Neither of the other methods is able to handle omnidirectional data, while CAM-SLAM performed as well on the omnidirectional synthetic room sequence as on the other sequences. The corresponding trajectory is shown in Fig. 2a.



(a) Synthetic Mars dataset (b) Catadioptric dataset (c) KITTI dataset

Fig. 3. Plotting of large-scale trajectories

In addition to evaluating the accuracy of CAM-SLAM in small-scale sequences, the applicability to large-scale sequences has been tested. Figure 3 presents the corresponding trajectories, which were generated using three different camera models. The according ground-truth trajectory is evenly approximated by CAM-SLAM, despite a modest drift. Due to the more complex camera movement, a more serious drift occurred in the catadioptric dataset. That the estimated trajectory suffers from a drift in scale becomes clear when observing the pulling in and out of the dead-end road in Fig. 3b. One has to consider, however, that the catadioptric camera is not mounted centrally on the car and that a certain offset is expected for this reason. The remaining plot shown in Fig. 3c resembles experiments with the KITTI dataset that uses a pinhole camera. In this dataset, the most severe drift occurred, which might be related to the shorter visibility of map points. Interestingly, SURF feature matching was more reliable than ORB matching, given catadioptric images. This could be related to the higher degree of distortion, which clearly affects keypoint descriptors, but an in-depth analysis remains future work.

7 Conclusion

This work presented CAM-SLAM, a new monocular SLAM system that focuses on flexibility. It supports any type of central camera model and is also able to perform a semi-dense reconstruction of the environment. Being research software though, CAM-SLAM can still be improved. For instance, the initialisation procedure assumes a non-planar environment. Furthermore, Mur-Artal et al. [19] already realised that their method might profit from representing points at infinity, as described by Civera et al. [4]. The same argumentation holds true for CAM-SLAM: Especially on sequences that exhibit camera transformations with a rotational component only, an inverse depth representation during tracking can stabilise the SLAM execution.

Acknowledgements. Martin Rünz has been partly supported by the SecondHands project, funded from the EU Horizon 2020 Research and Innovation programme under grant agreement No. 643950.

References

1. Baker, S., Nayar, S.K.: A theory of single-viewpoint catadioptric image formation. Int. J. Comput. Vis. **35**(2), 175–196 (1999)
2. Bunschoten, R., Krse, B.: Robust scene reconstruction from an omnidirectional vision system. IEEE Trans. Robot. Autom. **19**(2), 351–357 (2003)
3. Burbridge, C., Spacek, L., Condell, J., Nehmzow, U.: Monocular omnidirectional vision based robot localisation and mapping. In: Proceedings of the TAROS (2008)
4. Civera, J., Davison, A.J., Montiel, J.: Inverse depth parametrization for monocular slam. IEEE Trans. Robot. **24**(5), 932–945 (2008)
5. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: Proceedings of Ninth IEEE International Conference on Computer Vision, pp. 1403–1410. IEEE (2003)
6. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part II. LNCS, vol. 8690, pp. 834–849. Springer, Heidelberg (2014)
7. Gamallo, C., Mucientes, M., Regueiro, C.V.: A FastSLAM-based algorithm for omnidirectional cameras. J. Phys. Agents **7**, 12–21 (2013)
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
9. Geyer, C., Daniilidis, K.: Catadioptric camera calibration. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 1, pp. 398–404. IEEE (1999)
10. Gutierrez, D., Rituerto, A., Montiel, J.M.M., Guerrero, J.J.: Adapting a real-time monocular visual slam from conventional to omnidirectional cameras. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 343–350. IEEE (2011)
11. Hartley, R., Gupta, R., Chang, T.: Stereo from uncalibrated cameras. In: 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings CVPR 1992, pp. 761–764, June 1992
12. Horn, B.K., Hilden, H.M., Negahdaripour, S.: Closed-form solution of absolute orientation using orthonormal matrices. JOSA A, **5**(7), 1127–1135 (1988)
13. Klein, G., Murray, D.: Parallel tracking and mapping on a camera phone. In: Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2009, Washington, DC, pp. 83–86. IEEE Computer Society (2009)
14. Labrosse, F.: The visual compass: performance and limitations of an appearance-based method. J. Field Robot. **23**(10), 913–941 (2006)
15. Lhuillier, M.: Effective and generic structure from motion using angular error. In: 18th International Conference on Pattern Recognition (ICPR 2006), vol. 1, pp. 67–70 (2006)
16. Mei, C., Rives, P.: Single view point omnidirectional camera calibration from planar grids. In: 2007 IEEE International Conference on Robotics and Automation, pp. 3945–3950. IEEE (2007)
17. Mei, C., Sibley, G., Newman, P.: Closing loops without places. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3738–3744. IEEE (2010)
18. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: International Conference on Computer Vision Theory and Application VISSAPP 2009, pp. 331–340. INSTICC Press (2009)

19. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. Submitted to IEEE Trans. Robot. (2015). arXiv preprint arXiv:1502.00956
20. Mur-Artal, R., Tards, J.D.: Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In Robotics: Science and Systems (2015)
21. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2320–2327. IEEE (2011)
22. Peri, V., Nayar, S.K.: Generation of perspective and panoramic video from omnidirectional video. In: Proceedings of DARPA Image Understanding Workshop, vol. 1, pp. 243–245. Citeseer (1997)
23. Phan, K.D., Ovchinnikov, A.V.: Indoor slam using an omnidirectional camera. Middle East J. Sci. Res. **16**(1), 88–94 (2013)
24. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Comput. Graph. Image Process. **1**(3), 244–256 (1972)
25. Recker, S., Hess-Flores, M., Joy, K.I.: Statistical angular error-based triangulation for efficient and accurate multi-view scene reconstruction. In: 2013 IEEE Workshop on Applications of Computer Vision (WACV), pp. 68–75. IEEE (2013)
26. Rituerto, A., Puig, L., Guerrero, J.J.: Visual slam with an omnidirectional camera. In: 20th International Conference on Pattern Recognition (ICPR), pp. 348–351. IEEE (2010)
27. Scaramuzza, D., Siegwart, R.: Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. IEEE Trans. Robot. **24**(5), 1015–1026 (2008)
28. Scaramuzza, D.: Omnidirectional vision: from calibration to robot motion estimation. PhD thesis. Citeseer (2008)
29. Scaramuzza, D., Martinelli, A., Siegwart, R.: A toolbox for easily calibrating omnidirectional cameras. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5695–5701. IEEE (2006)
30. Schoenbein, M., Geiger, A.: Omnidirectional 3D reconstruction in augmented manhattan worlds. In: International Conference on Intelligent Robots and Systems, pp. 716–723, Chicago. IEEE, October 2014
31. Schnbein, M., Strauss, T., Geiger, A.: Calibrating and centering quasi-central catadioptric cameras. In: International Conference on Robotics and Automation (ICRA) (2014)
32. Strasdat, H., Davison, A.J., Montiel, J.M.M., Konolige, K.: Double window optimisation for constant time visual slam. In: IEEE International Conference on Computer Vision (ICCV), pp. 2352–2359, November 2011
33. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D slam systems. In: Proceedings of the International Conference on Intelligent Robot Systems (IROS), October 2012
34. Sturm, P.: Camera models and fundamental concepts used in geometric computer vision. Found. Trends Comput. Graph. Vis. **6**(1–2), 1–183 (2010)