Stereo camera tracking for mobile devices

Simone Gasparini, Pascal Bertolino
GIPSA-Lab, Université de Grenoble, CNRS
Grenoble Campus BP46,
F-38402 St Martin d'Hères cedex, France
simone.gasparini@gmail.com, pascal.bertolino@gipsa-lab.fr

Abstract

We present our current work on a camera tracking algorithm designed for a mobile device equipped with a stereo camera. The tracker runs in real-time on a prototype mobile platform and it can be used as the core engine of augmented reality applications. In order to cope with the limited resources available, we design an algorithm that relies on the stereo camera only for the 3D reconstruction of points, while the point tracking is performed only on one of the two images, thus reducing the computational effort. We show some preliminary results in which the camera tracker as been validated in a realistic scenario and it is proved to have an adequate robustness for an augmented reality application.

1. Introduction

In the last decade the wide diffusion of mobile devices equipped with cameras, such as smart-phones and tablets, contributed to the growth of computer vision applications for these new devices. The most notable application is Augmented Reality [18], which enhances the user experience by rendering virtual objects over the images. This can find application in a wide range of domains and contexts, from displaying additional information about a given product during shopping, gaming, up to urban navigation for pedestrians [20].

In order to properly render the virtual objects inside the image, the position, the orientation and the movement of the device must be known at any time instant, so that the virtual objects are perceived by the user as part of the surrounding reality [6]. In the case of mobile phones with cameras, different solutions have been proposed to track the motion of the device according to the required level of accuracy. Accelerometers and magnetometers can be coupled in order to provide the motion and the orientation of the device at any time instant. Unfortunately the inertial sensors alone cannot achieve high accuracy, since the measurements can be noisy for slow motions and magnetic fields may interfere with the magnetometer [7].

A common approach to retrieve the motion and orientation of the device is to use the visual information of the on-board camera to track fixed points in the scene and exploit their apparent motion to estimate the motion [10]. This technique is known as *Camera Tracking* and it's mainly composed of four main steps: (*i*) detection of interest points (features) in different images, (*ii*) finding of corresponding points among the images and 3D reconstruction, (*iii*) tracking of the points in new images, and (*iv*) estimation of the motion and orientation of the camera. Eventually, a Bundle Adjustment (BA) process [30] is run to maintain a global consistency between the 3D points and the camera positions.

Visual tracking can provide a more accurate and stable estimate of the camera motion, especially for smooth movements of the camera. On the other hand, other factors such as change of light conditions, object occlusions, motion blur due to abrupt movements of the camera and texture-less environments can affect the robustness of the tracking. Often visual tracking and motion sensors are coupled together [15, 33] to combine the benefits of the two approaches and limit their drawbacks. In this case the camera and the motion sensors must be mutually calibrated [14].

In a pure visual tracking approach, the detection of interesting features in the scene is often eased by using special markers that are highly recognizable in the image [5, 8]. Camera tracking with markers is very robust and it is a common approach for many augmented reality applications. Optimized methods [32] allow mobile devices to easily detect markers in real-time. The major drawback of this solution, though, is that the scene must be set up with markers, not always a practical solution for some particular applications, *e.g.* tracking in wide spaces or outdoor navigation.

Some marker-less approaches for hand-held devices have been proposed. Klein and Murray adapted and ported their previous work based on a SLAM framework [16] to an iPhone device [17]: they showed that the computational burden of the BA can be optimized and limited by using a dedicated background thread and by reducing the dimensionality of the optimization problem (less tracked points). The method proposed by Wagner *et al.* [31] tracked natural features (based on optimized versions of SIFT [21] and FERN [26]) on known textured planar targets. Lately, Kurz and Benhimane [19] proposed to enrich the feature descriptors with the gravity direction, thus improving the feature matching in marker-less tracking algorithms.

In this paper we present the first results of our on-going work on a camera tracker for a mobile device equipped with a stereo camera. In the last years 3D vision has become popular and some mobile phones have been introduced in the market sporting a stereo camera coupled with auto-stereoscopic screens. This opens the way to a new interesting and challenging range of applications that can further improve the user experience and interaction with the real world. In particular we are working on a mobile platform prototype which is being developed by ST-Ericsson and we are developing a camera tracker method that exploits the stereo configuration and that can be used for stereo augmented reality applications.

Stereo cameras are interesting in the context of camera tracking as they can provide at any time instant the 3D depth of the scene. On the other hand, they require more computational power since two images must be processed at the same time. There are some works in the literature that adapted the VisualSLAM techniques to the stereo case [25, 22]. These works are mainly focused on robotic applications and deal with camera tracking in large environments. The methods proposed by Nister *et al*. [25] uses a tracking by matching approach: the features extracted in the stereo pair are used to perform a first reconstruction of the points, and then, for each new (stereo) frame new features are extracted and matched with the previous ones to estimate the motion of the camera. They also fuse the data coming from the inertial sensors and the GPS with the visual estimation, in a bundle adjustment process. Overall the system runs at $\sim$13fps on a standard PC and it is not suitable for a mobile device. The method proposed by Mei *et al*. [22], instead, uses the reprojection of the 3D points reconstructed at the previous step to estimate the position of the stereo camera and, for each frame, it also reconstructs new 3D points that are added to the set of the tracked points. The image features are represented with SIFT descriptors in order to guarantee the relocalization of the robot whenever the track is lost ("kidnapped robot"). The method requires a Quad Core PC to run in real-time.

Developing a camera tracker for a mobile device requires to design the algorithm in order to meet the computational constraints, yet maintaining a level of robustness adequate for the application. The main contribution of our work is then a novel algorithm that is able to estimate the motion of the stereo camera in real-time and in a robust way. To the best of our knowledge this is the first attempt to develop a stereo camera tracking algorithm for a mobile device.

The paper is organized as follows: §2 presents the main aspects of our approach to stereo camera tracking and some details of the proposed algorithm. §3 presents some preliminary experiments and validation of our method running on the prototype device. §4 concludes the paper with some discussions and future works.

2. Stereo Camera Tracking

The tracking algorithm is composed of two main steps, the initialization step and the tracking step. The initialization step is performed only once for the first frame and it mainly consists in the first 3D stereo reconstruction of the points that will be tracked in the next frames. It is important to note that, in order to limit the computational effort, we process the stereo pair only when necessary, namely when features are lost during the tracking and new 3D points need to be reconstructed, while we process only one of the two images to track the points (in our case we arbitrarily chose the left one).

Stereo calibration (intrinsics, extrinsics and lens distortion) is performed off-line by our Android stand-alone application implementing the method proposed by [29].

2.1. Initilization

The initialization step occurs only at the beginning of the tracking and allows to detect the features with the stereo camera and reconstruct them, so that they can be tracked in the subsequent frames and used to estimate the motion and the pose of the camera.

In the monocular case the initialization step normally requires the user to move the camera in order to have at least a pair of images with a significant displacement between them, which guarantees a better and more reliable first 3D reconstruction [23, 16, 17]. Using a stereo camera, instead, the initialization step does not require any user interaction, since the features can be reconstructed by the stereo camera. We extract the features from both images and we match them in order to obtain a set of corresponding points in both images. These points can be reconstructed by triangulation, thus giving a set of 2D-3D points association.

Feature extraction Recently, a new range of feature descriptors and feature extractors have been proposed in order to overcome the computational limitation of SIFT [21] and SURF [1], yet maintaining a good and comparable recognition performance. A detailed review and a comparison of such feature descriptors for camera tracking applications is proposed in [11]. In our algorithm we chose a combination of FAST [28] features and BRIEF [4] descriptors, since in our experiments it was the combination that guarantees the best trade-off between speed and recognition rate. FAST is well known to be a very fast corner extractor with a good repeatability and robustness to light changes. BRIEF descriptors have been introduced recently and they to describe the feature patch with binary descriptors, as results of a series of random (binary) tests on the pixels of the patch. The resulting descriptors are compact strings composed of few bits (128) that can be matched using the Hamming distance, which is quite fast to compute.

Finally, in order to have an uniform distribution of features covering the whole image, we divide the images in cells (in our case with a 4×4 grid) and for each cell we consider at most the first `MaxFeat` detected features.

Feature Matching Once the features are extracted and their descriptors are computed on both images, the descriptors are matched to find corresponding points in the stereo pair. For each feature on one image, the BRIEF descriptors is compared with the descriptors of the other image and the descriptor with the smallest distance is considered as

a candidate match[1]. In order to improve the robustness of the matching process we consider only those candidate pairs that are mutually a good match. For each feature in both images, we compute the k nearest neighbors using the FLANN algorithm [24]. Then, for each feature descriptor on the left image $\mathbf{d}_i^l$, we consider its k neighbors $\mathbf{d}_j^r$ on the right image and we select as the matching candidate the descriptor $\mathbf{d}_j^r$ that has $\mathbf{d}_i^l$ among its k neighbors. In our experiments we use $\mathtt{k} = 4$.

3D Reconstruction The corresponding points found at the previous step are triangulated to geometrically reconstruct the 3D point in the scene[13]. Since the set of corresponding points may still contains some incorrect correspondences and outliers, we further filter out the pairs having a reprojection error on the two images larger than a given threshold $\mathtt{repThreshold} = 2$pix, and those whose 3D point does not lie in front of both cameras, *i.e*. it is not a physically feasible solution (*cheirality test*).

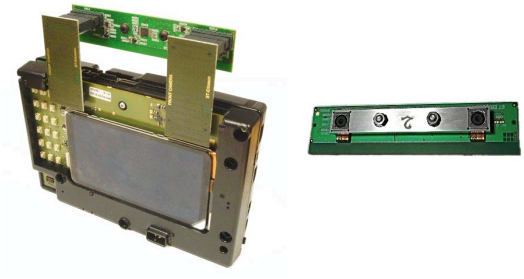
2.2. Tracking

In order to reduce the computational cost only the left images are used to track the features; the tracking is performed with Lucas-Kanade (LK) tracker based on the optical flow only. The new position of the feature in the image is used to solve the resection problem and estimate the camera position and orientation.

During the tracking process, features may be lost due to occlusions, large movements of the camera or lost by the LK tracker. In order to maintain a sufficient number of tracked features, thus preserving the reliability of the motion estimation, whenever the number of tracked features decreases below a given threshold $\mathtt{minFeat} = 20$, we refill the set of tracked features with new ones. As in the initialization step, we rely on the stereo pair to extract new features, find new stereo correspondences and reconstruct new 3D points. Unlike the initialization step, though, in this step we also implemented a re-association algorithm in order to re-associate the new features with already known 3D points that are visible from the camera but that were no longer tracked in the image.

Feature Tracking Since the optical flow tracking is usually a computationally intensive algorithm, we used a sparse version of the Lucas-Kanade tracker using pyramids [2], which computes the optical flow only in a patch surrounding the locations of the features to be tracked. Moreover, using pyramids, different scale factors can be used in order to take into account changes in scale due, *e.g*., to the zooming movement of the camera. In our experience we found that a good trade-off between speed and robustness can be achieved when the optical flow is computed in a 11×11 patch around the locations of the features using a 2-level pyramid.

[1]At the moment the prototype device can only provide raw images for each camera; upcoming hardware upgrades will allow to work directly with rectified images, thus simplifying and speeding-up the search for correspondences. In our early experiments, software rectification introduced a significant overhead in our pipeline.



(a) The mobile platform (b) The stereo camera

Figure 1. The mobile prototype (a) used to run the camera tracker is equipped with a stereo rig (b) having a baseline of 6.5cm.

Pose Estimation Once the new positions of the features are computed, the camera motion and orientation can be estimated by solving the resection problem using the 2D-3D points association. In our implementation we use the method proposed by Grunert [12] to compute an initial estimate of the pose with a minimal set of 3 points. A RANSAC [9] process is used in order to filter out possible outliers and refine the initial solution.

Feature Re-association If the number of tracked features is below the threshold minFeat the stereo pair is used to detect new features and reconstruct new 3D points. We follow a similar procedure as in the initialization step (see §2.1). We extract the FAST corners and compute their BRIEF descriptors, and we keep only those features that are at a distance of at least $\mathtt{minDist} = 15$pix from the current tracked features. This prevents "redundant" points and it guarantees a more uniform distribution of the features in the image.

Then we consider the set of all the 3D points that have been reconstructed so far and that are visible from the camera but not anymore associated to any tracked feature. The 3D point visibility is computed by considering the actual position and orientation of the camera and its field of view, according to the calibration data. We project these 3D points on the left image: if the projected image point is within a distance repThreshold from a new detected feature we then assign that feature to that 3D point and we add it to the set of tracked features. In order to speed-up the search process we use the new left detected features to build a KD-Tree and then we use the FLANN algorithm to query the tree with a projected point in order to find its closest point among the new left detected features.

It is worth to note that this re-association algorithm may generate false associations, since, *e.g*., occlusions among 3D points are not taken into account. However the system is overall robust to false re-associations: when false re-associated points are tracked in the next frame, the points are discarded by the RANSAC process of the pose estimation algorithm, which is robust to outliers.

The remaining new features are then matched with those of the right image, triangulated and added to the set of tracked features as described in §2.1.

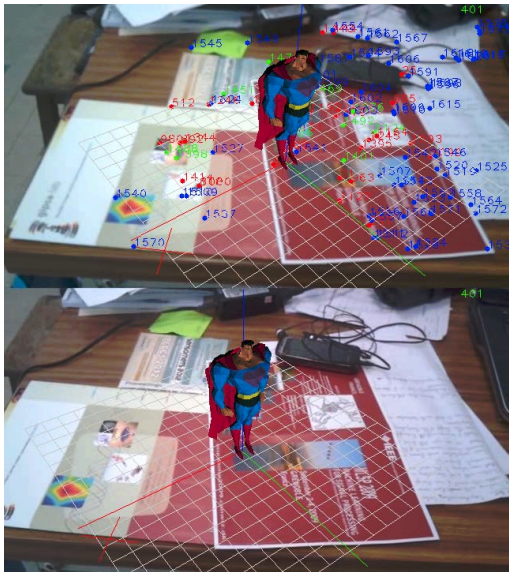


Figure 2. To visually assess the quality of the tracking we render a 3D object over the dominant plane of the scene, which is determined during the initialization step by robustly fitting the 3D points to a plane in a RANSAC process.

3. Experimental Results

The camera tracker has been implemented on a mobile platform prototype developed by ST-Ericsson (see Figure 1). The platform is powered by U8500 application processor, containing a dual-core ARM® CORTEXTM-A9 CPU with NEON running at 800MHz, and a ARM MALI-400 GPU running at 400MHz. The platform is running Android OS 2.4.3 and is equipped with two 5.3Mpix cameras that form a stereo rig with a baseline of 6.5cm. The platform is able to provide a stereo video stream composed of two images of 640×480 pixels at a frame rate of 30fps in a top-down layout.

We implemented the camera tracker in C++ relying on some functions of the OpenCV libraries [3] (feature detection, descriptor computation, and LK tracking), while the main JAVA application on the mobile device processes the stereo image calling a native JNI function of the C++ camera tracker library. It must be noted that at the moment no GPU and multi-threading optimization have been done, we only took advantage of the TBB [27] parallelization of the OpenCV function computing the sparse optical flow.

In order to visually assess the quality and the stability of the camera tracking, we run some preliminary tests drawing a 3D object in the stereo image as depicted in Figure 2. The videos with some preliminary results can be found at the following url http://goo.gl/YuNRw.

In our preliminary experiments the camera tracker ran in real-time up to 15 frames per seconds. Table 1 shows a break down of the algorithm into the most relevant functions and for each of them the average execution time. The execution times have been computed by taking the clock time before and after the function call. It can be noted that during a typical execution of the algorithm the LK tracking process

Function	Mean	Max	Min
Feature extraction	30.19	77.53	13.11
Descriptor computation	24.96	33.32	21.19
Matching	17.49	21.8	10
LK tracking	58.36	126.89	37.01
Pose estimation	16.03	83.87	0.43
Re-association	147.43	188.81	115.06

Table 1. Statistics of the execution time of some functions of our algorithm. Times expressed in [ms].

is the most demanding one with an average execution time of ∼60ms. This is not surprising as the optical flow computation is computationally intensive. The execution time also depends on the number of features that are currently tracked, as it can be noted in Figure 3: the LK tracking time decreases as features are lost and it reaches its maximum after new features are added by the re-association procedure (which corresponds to the spikes in the graph). In our settings we normally use up to 300 features. An alternative approach to feature tracking is to extract the features for each frame, compute the descriptors and match them with the ones of the last frame. As shown in Table 1 these three operations can be as expensive as the optical flow approach. However when we tried this approach we found out that the tracking by matching approach is less robust than the optical flow as it generates more outliers and false stereo correspondences.

The re-association process is the most computationally expensive operation, as it requires to process the two stereo images in order to extract the features and compute their descriptors: as reported in Table 1, these two operations require ∼50ms for each image and up to ∼20ms to compute the correspondences by matching the descriptors. On the other hand the re-association is not performed at every frame but only when enough features are lost during the tracking. This is related to the movement of the camera, hence for smooth movements of the camera higher framerates can be achieved. In our experiments we noted that typically the re-association procedure is called on average every $50 - 60$ frames depending on the camera motion (see the spikes in Figure 3).

In order to numerically assess the quality of the camera tracker we tried to evaluate the error between the estimated starting position and the estimated ending position of the camera when it moves in a loop. Since no device that could guarantee a repeatable and controlled movement (*e.g.* a robotic arm) was available, we manually moved the camera in a loop path, trying to bring the camera back to the initial position over a reference landmark. In our experiments the error between the starting and the ending position typically ranges between 20mm and 80mm with an average path of 4m (see Table 2). Figure 4 shows some samples of the reconstructed trajectories.

4. Conclusion and future work

In this paper we presented our on going work on developing a camera tracker for a mobile device equipped with

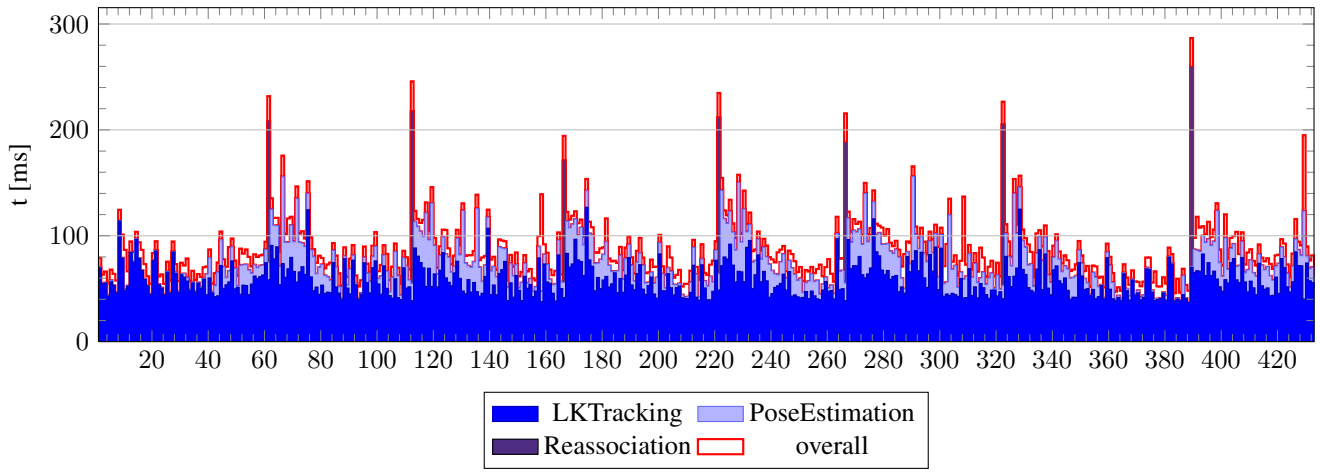
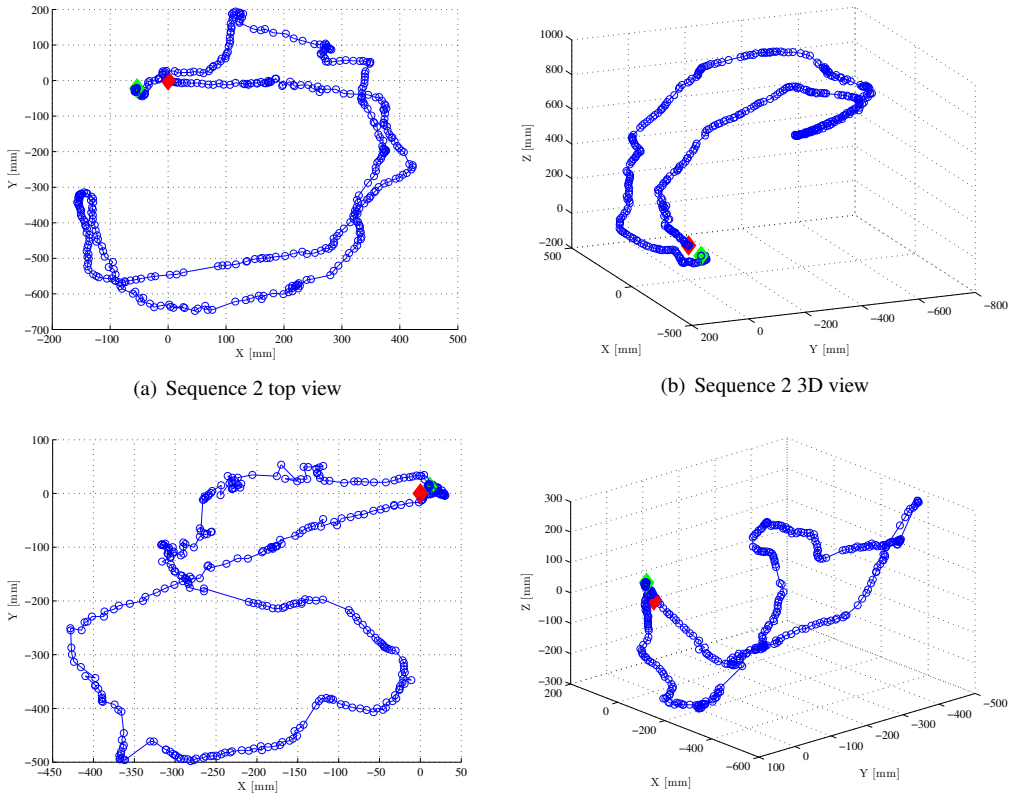


Figure 3. An example of the execution times of our algorithm, with the partial execution time of the main functions



(a) Sequence 2 top view

(b) Sequence 2 3D view

(c) Sequence 4 top view

(d) Sequence 4 3D view

Figure 4. The top view of the XY plane ((a) and (c)) and a 3D view ((b) and (d)) of the trajectories of the Sequence 2 and Sequence 4, with the starting point (in red) and the end point (in green).

Sequence	# Frames	Distance	Error
Sequence 1	412	4.62m	50.74mm
Sequence 2	454	5.37m	73.09mm
Sequence 3	328	3.54m	29.11mm
Sequence 4	409	3.61m	18.56mm

Table 2. Errors in closing a loop path with different video sequences.

a set of stereo cameras. The proposed algorithm is tailored for the limited computational power of the mobile environment. After an initialization step where the stereo images are used to reconstruct a first set of 3D points, the tracking of the frame is then limited to the left image, in order to limit the processing time. The stereo pairs are used to detect and reconstruct new feature points, whenever the number of features decreases below a given threshold. We showed that the proposed camera tracker can run on in real-time on a

prototype of a mobile device equipped with a stereo camera showing a good level of stability and robustness, adequate for a real augmented reality application.

There are several research directions that we are addressing to improve and further develop our algorithm. We are planning to add a re-localization module that allows the tracker to recognize a location that has been already visited in other session. Re-localization is also important to recover the position of the device whenever the tracking is lost due to occlusions or large and sudden motions of the device. A related direction is the optimization module, based on the bundle adjustment, which will contribute to improve the loop closure problem by optimizing the 3D structure and generate a more accurate map of the scene. We are investigating a solution inspired by [17], with a background thread performing the optimization.

The integration and the fusion of the data from the inertial sensors is another interesting direction of investigation, as they can improve the quality of tracking, especially when irregular and large motions are involved, which typically are not properly handled by the visual sensors.

5. Acknowledgments

This work has been conducted withing the Moov3D project, supported by MINALOGIC and IMAGINOVE *pole de compétitivité* and founded by the French state bodies OSEO and region Rhône-Alpes.

References

[1] H. Bay, T. Tuytelaars, L. Van Gool, and L. Van Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proceedings of the European Conference Computer Vision (ECCV 2006)*, volume 3951, pages 404–417. Springer, 2006. 2

[2] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000. 3

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 4

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision (ECCV 2010)*, September 2010. 2

[5] L. Calvet, P. Gurdjos, and V. Charvillat. Camera tracking using concentric circle markers: paradigms and algorithms. In *Proceedings of the IEEE International Conference on Image Processing 2012*, 2012. 1

[6] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51(1):341–377, 2010. 1

[7] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26(6):519–535, 2007. 1

[8] M. Fiala. Artag fiducial marker system applied to vision based spacecraft docking. *Robot Vision for Space Applications*, page 35, 2005. 1

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communication of ACM*, 24(6):381–395, 1981. 3

[10] A. W. Fitzgibbon and A. Zisserman. Automatic camera tracking. In M. Shah and R. Kumar, editors, *Video Registration*, chapter 2, pages 18–35. Kluwer, 2003. 1

[11] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335–360, 2011. 2

[12] J. Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, pages 238–248, 1841. 3

[13] R. Hartley and A. Zisserman. *Multiple View Geometry*, volume 3. Cambridge University Press, 2003. 3

[14] J. Hol. Modeling and calibration of inertial and vision sensors. *International Journal of Robotics Research*, 29(2-3):1–24, 2010. 1

[15] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56, 2011. 1

[16] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. 1, 2

[17] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, October 2009. 1, 2, 6

[18] D. W. F. V. Krevelen and R. Poelman. A survey of augmented reality technologies , applications and limitations. *International Journal of Virtual Reality*, 9(2):1–20, 2010. 1

[19] D. Kurz and S. Benhimane. Handheld augmented reality involving gravity measurements. *Computers & Graphics*, 36(7):866–883, 2012. 1

[20] F. Liarokapis, V. Brujic-Okretic, and S. Papakonstantinou. Exploring urban environments using virtual and augmented reality. *Journal of Virtual Reality and Broadcasting*, 3(5):1–13, 2007. 1

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1, 2

[22] C. Mei, G. Sibley, M. Cummins, P. M. Newman, and I. D. Reid. A constant-time efficient stereo slam system. In *Proceedings of the British Machine Vision Conference (BMVC 2009)*, 2009. 2

[23] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193, July 2009. 2

[24] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340. INSTICC Press, 2009. 3

[25] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:2006, 2006. 2

[26] M. Özuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 2007. 1

[27] C. Pheatt. Intel® threading building blocks. *J. Comput. Sci. Coll.*, 23(4):298–298, Apr. 2008. 4

[28] E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010. 2

[29] P. F. Sturm and S. J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pages 1432–1437, 1999. 2

[30] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, 2000. 1

[31] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 125–134, Washington, DC, USA, 2008. IEEE Computer Society. 1

[32] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, pages 127–, Washington, DC, USA, 2003. IEEE Computer Society. 1

[33] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, 1999.*, pages 260 – 267, 1999. 1