ORB-SLAM with Near-infrared images and Optical Flow data

Antonio Buemi Arcangelo Bruna Sylvain Petinot Nicolas Roux

STMicroelectronics
(antonio.buemi, arcangelo.bruna, sylvain.petinot, nicolas.roux)@st.com

Abstract

The algorithms designed to solve the Simultaneous Localization And Mapping (SLAM) problem have to be often executed on embedded platforms in order to become part of complex robotics systems. Despite the continuous growth of their computational capabilities, the embedded devices still have considerable limitations, especially in terms of memory. This paper presents a modified version of the well known ORB-SLAM algorithm which improves its performance thanks to the use of Hardware-generated Optical Flow (HW-OF). The ORB-SLAM has been modified in order to run into the Stereo-cam embedded system by STMicroelectronics. The Stereo-cam includes the VD56G3 sensor, able to provide Near Infrared (NIR) images and OF data computed by a hardware accelerator. The experiments showed an improvement of the ORB-SLAM performances in terms of memory consumption and frame rate.

1. Introduction

Simultaneous Localization And Mapping (SLAM) is the set of algorithms aiming to construct and update a map of an unknown environment. At the same time, they have to keep track of the system positioning in the map. Nowadays cameras are low cost, light-weight and low power consumption sensors. Such features make them a good choice for SLAM [6] solutions for real-world applications, in particular for indoor navigation in unknown environment, the typical vacuum cleaner scenario. SLAM algorithms can be applied also in autonomous vehicles [1], smartphone-based and head-mounted devices, Augmented, Virtual and Mixed Reality [25], [28], [7] and commercial drones [9]. Each of these applications requires different levels of precision, accuracy, efficiency, success rate. Most of them cannot be addressed just using visual information, so several kind of additional sensors are used: laser scanners, Inertial Measurement Unit (IMU) [21], GPS receivers, etc. Adding information from different sources takes the problem complexity to a higher level because this information must be synchronized and because the algorithm must be able to use them efficiently. Although some sensors offer good performances at low prices, a multi-modal system is still more expensive than a monocular one, also in terms of power consumption. Since SLAM solutions are usually complex and require high computational and memory resources, implementation on embedded devices remains an interesting challenge. This paper presents a specific case of study: the implementation of a visual slam algorithm exploiting NIR and HW-OF generated by an STMicroelectronics' device on a low power Microprocessor Unit (MPU).
The rest of the paper is organized as follows. Section 2 describes the main blocks of a typical SLAM algorithm; Section 3 presents an overview of existing methods and the related classification; Section 4 introduces the ORB-SLAM algorithm and the proposed modified version. The Section 5 discusses the experimental results. Final remarks in Section 6 end the paper.

2. SLAM algorithm basic design

A SLAM system is a pipeline of algorithms devoted to settle specific sub-problems in order to concur on solving the global SLAM problem. For sake of simplicity, a typical SLAM system can be represented as a pipe of four main blocks (see Figure 1): Input search, Pose tracking, Mapping and Loop Closure, discussed in details in the following subsections.

2.1. Input Search

The Input Search step aims to extract suitable information from sensors'measures. Some approaches use the pixel intensity values in order to match different frames and extract the motion information. They are called "direct" methods. Other techniques extract salient features from the images, i.e. corners or edges, and then they look for matching among such features. The SLAM approaches based on features extraction are known as "indirect" or "feature-based" methods. They require the use of robust feature detectors

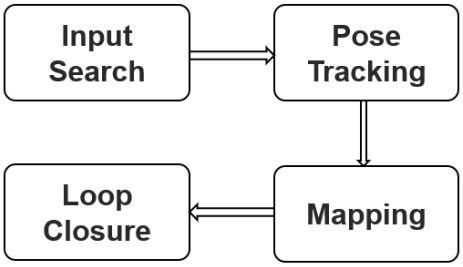

Figure 1. SLAM system basic blocks.

and descriptors. The literature offers a wide choice of descriptors: Harris [4], SURF [12] , FAST [10], BRIEF [19], SIFT [16], ORB [11], each of which presents advantages and disadvantages in terms of performances and reliability. Ideally a feature descriptor should be invariant to translation, to rotation, to scale , to the point of view and brightness variations.

2.2. Pose Tracking

The Pose Tracking step provides an estimate of the current device position, updating it in dependence on the new observations. This block allows to obtain the device trajectory estimation. It can be also referred as odometry and, in particular, Visual Odometry (VO) [21], when this information is extracted from the image. Sometimes SLAM and Visual Odometry are treated as synonyms in the literature, because they solve similar problems, but VO is a building block of SLAM because it doesn't provide global mapping. Depending on whether the features used to perform frames matching are 2D or 3D, there are several methods to implement the VO. Pure VO directly matches the 2D features extracted from consecutive frames. Another option is the 2D-3D method. In this case the pose is estimated from a set of 3D points, previously mapped and projected into the current frame. The less common method is the 3D-3D, that is possible only in the case of stereo cameras: in this case the localization of the newly detected 3D features can be performed directly.

2.3. Mapping

The mapping is the action of correctly localize a new detected feature in the environment mapped by the system so far. The mapping block creates and updates the map of the environment where the device is moving in. Such map can be a 2D or 3D map and it can be dense or sparse, depending on SLAM implementation, i.e. using direct or indirect methods.

2.4. Loop Closure

SLAM system must be able to recognize when the device is in an area of the environment already mapped. The Loop Closure is a two-phases process: it starts with the Loop Detection step and then it performs the Loop Closure itself. Even if the sensors provides accurate information and the odometry and mapping algorithm are reliable, noise and approximation introduce errors into the pose and map estimation processes, and such errors are propagated long the time. In these cases the Loop Closure block exploits the new information to correct the errors accumulated in both the map and the camera position estimations. This implies a correction strategy. This is one of the most important step in the whole SLAM process.
There are two large classes of approaches performing such step: Bundle Adjustment (BA) [3] methods and filter methods. The BA re-evaluates all the camera poses and feature poses in the map from scratch at every time step as a maximum likelihood estimate, whilst the filter methods represent the estimate of the device and feature poses as a probability density function, correcting such estimates incrementally by taking into account the new observations. The Extended Karman Filters (EKF), which gives a Bayesian solution to the state estimate update depending on the new observations, is the typical solution used for this step. Another approach consists on modelling the probabilities for the state estimate by multivariate Gaussian Particle Filters, which do not assume the shape of the probability density function. Even if apparently the filter methods reduce the computational costs reusing previous estimations, the BA can be efficiently implemented looking for the optimal solution on a subset of past camera poses and features. More precisely, keyframes which present meaningful features are chosen at regular intervals and only these will be used to correct the map reducing the computational costs. Computational complexity of real-time BA can be reduced also using other techniques, such as Pose Graph Optimisation [17]. Note that the loop closure is the element that distinguishes SLAM from Visual Odometry, along with the mapping phase.

3. SLAM algorithms classification

SLAM algorithms can be classified in several ways. The most common choice is the classification performed accordingly with their input and output domains. The difference between direct and indirect methods is related to the input domain, while the type of output determines whether the map is dense (the whole frame is processed) or sparse (just a cloud of sparse features is processed). In between there are the semi-sparse methods, defined as the dense methods where only a particular zone of interest of the frame

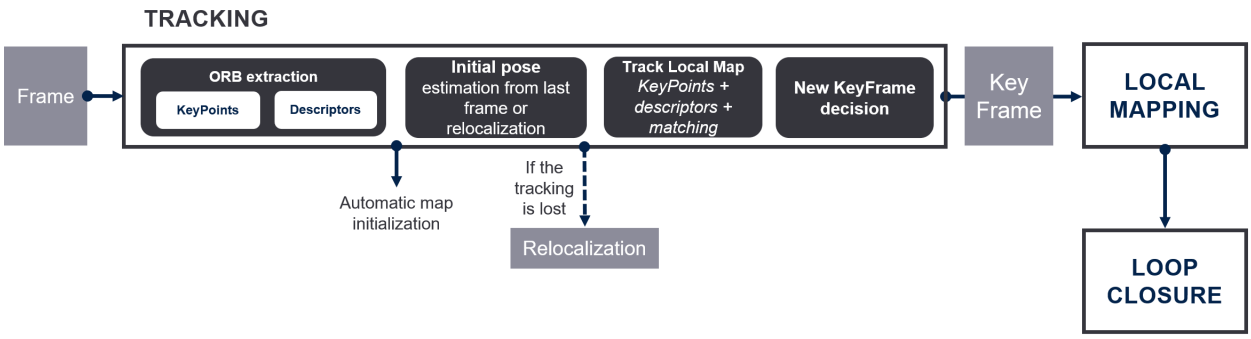

Figure 2. ORB-SLAM basic scheme

is mapped. So, four possible combination of algorithms' categories can be identified: Direct/Dense, Direct/Sparse, Indirect/Dense, Indirect/Sparse.

The direct methods usually are able to provide a dense map, whilst the indirect approaches usually outputs sparse maps. Direct/Sparse and Indirect/Dense methods are rare. The former use the photometric error (Direct Sparse Odometry [14] is the most popular example of this kind of techniques), the latter consider the geometric error as a deviation from the optical flow relating to the same frame [24]. The choice between direct and indirect methods depends on the application key requirements, e.g., the level of environment understanding required, the available computational resources, the environment features and the target accuracy. The feature-based approaches are robust to image noise. Their main drawback is that, once the features are extracted, the global image information is lost, so these methods are less effective in environments with too many or too few salient features, or where such features can be made less recognizable by external factor (e.g., atmospheric phenomena). Moreover, the feature extraction step requires additional computational efforts. On the contrary, the direct methods do not present such overload, they maintain the global information, but they suffer in case of light changes and they are sensitive to large motion, too. In applications exploiting basic hardware, such as webcam or smartphone, an indirect method is likely the best choice due to its robustness. When a more sophisticated device is available, e.g. global shutter cameras, a direct approach may be preferred. Note also that in same case a 3D reconstruction of the environment is a must (e.g. in Augmented or Virtual Reality applications), but it often requires an heavy parallelization on GPU.

A lot of SLAM direct and indirect algorithms have been developed in the latest years. The set of most popular feature-based methods includes the monocular EKF-SLAM MonoSLAM [2], the monocular FastSLAM [20], the Parallel Tracking And Mapping (PTAM) [15] and the ORB-SLAM [22], [23], that will be discussed in detail in the next section. The first dense monocular SLAM direct method is the Dense Tracking And Mapping (DTAM) [26]. A direct approach able to provide a semi-dense map is the Large Scale Semi Direct SLAM (LSD-SLAM) [13], whilst ROVIO [18] and Direct Sparse Odometry (DSO) [14] are examples of direct methods providing sparse maps. Note that several approaches based on deep-learning have been proposed (see [27] for an accurate overview).

4. ORB-SLAM

The ORB-SLAM [22] is one of the most popular feature-based solution for the SLAM problem because it presents several strengths. First of all, the source C++ code includes several versions of the algorithm [23]: monocular or stereo cameras, depth sensor, etc., so it is suitable for many applications. Moreover, one more version, able to exploit IMU, has been recently introduced [5]. The ORB-SLAM implements the main blocks of the system in an efficient manner, so it can be executed just exploiting the CPU. The algorithm itself is quite flexible: the performances maintain the same level in both indoor and outdoor environments. Moreover, the ORB-SLAM is not only a complete SLAM system, i.e. it includes a Loop Closure block, but it also presents a relocalization module able to resume the tracking when the algorithm doesn't find a sufficient number of keypoints for some reason. In order to extract the salient features from the current frame, it uses the ORB detector. ORB [11] is a fusion of oriented FAST [10] keypoints detector and rotated BRIEF [19] descriptors. It is a free alternative to SIFT [16] and SURF [12] and it overcomes them in computation cost and performance. The basic scheme of the monocular ORB-SLAM is shown in the Figure 2. The pipeline consists of three main blocks: the Tracking, the Local Mapping and the Loop Closure. The tracking starts with the ORB features extraction and it includes a self-initialization sub-system able to provide the first pose guess and to fix the scale. To perform the initialization the system requires a regular

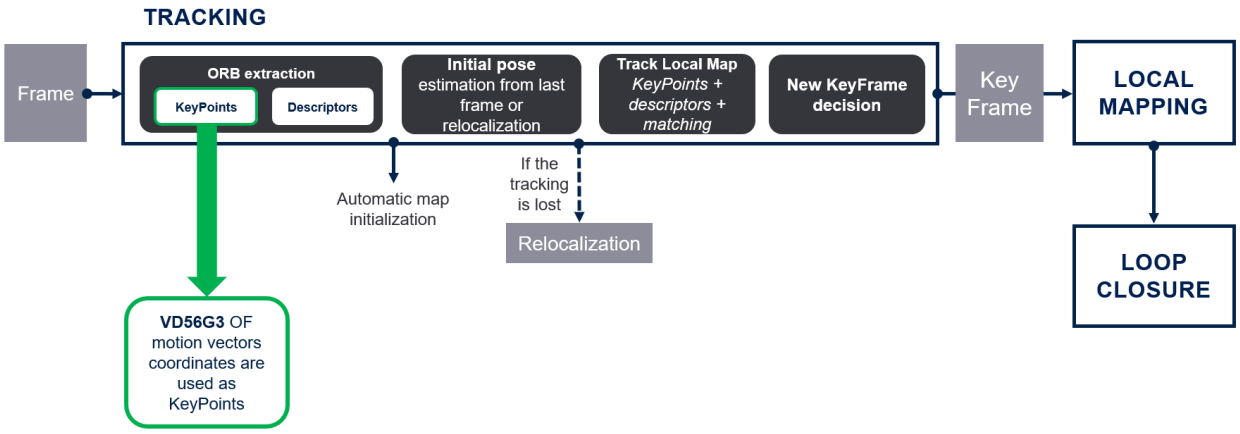

Figure 3. Modified ORB-SLAM basic scheme: the OF motion vectors replace the FAST keypoints.

motion in an environment including a sufficient number of detectable corners and edges in order to matches keypoints of consecutive frames and to perform triangulation. Once that the initial map is created and the first pose estimation is obtained, the tracking starts. When a new frame is acquired, the ORB features are extracted and compared to the existing map in order to update the map itself and the pose. Moreover, keyframes are added when the system recognizes a meaningful change in the scene. This step is the Local Mapping block task. Note also that, thanks to the keyframe-based organization and a Bag of Words (BoW) [8] vocabulary updated on the fly, the relocalization block works efficiently. The Loop Closure is the last main block making the system able to recognize when the device is moving into an already visited zone. In this case the new observations are also exploited to correct the pose and map estimation.

4.1. Modified ORB-SLAM

The purpose of the activity presented in this paper is the exploitation of STMicroelectronics' VD56G3 sensor in 3D-SLAM exploiting the built-in Optical Flow generation. To achieve this goal, the ORB-SLAM algorithm has been slightly modified to make it able to process VD56G3 data and to exploit it during the feature extraction step (see Figure 3). The following section describes more in details the VD56G3 and the hardware system used to perform the tests. The most important information to know is that the VD56G3 is able to provide NearInfraRed (NIR) images and related Optical Flow (OF), computed very efficiently thanks to a hardware accelerator. The NIR data are represented as grayscale images, so no modification was needed to correctly load and process them. While the classic ORB-SLAM uses the ORB to extract the images features, our version uses the coordinates of the motion vectors as keypoints. This allows to remove research of the salient points step in the whole frame, even if the computation of the related descriptors remains unchanged, being necessary for the following matching among the currently extracted features and those already present in the map. A further step to improve the efficiency of the algorithm could be not to use the descriptors as well, by modifying the matching process properly. This is a not a trivial change because it impacts the whole mapping module and it will be the objective of future investigation.

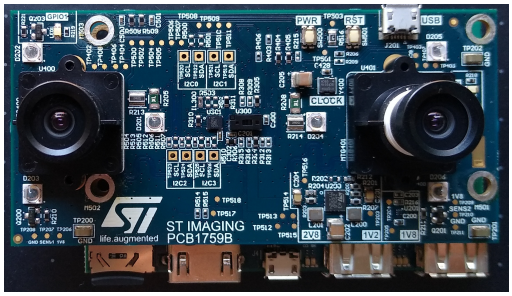

Figure 4. The Stereo-Cam with two VD56G3 mounted on the DragonBoard through the 96connector used for the experiments.

5. Experimental results

The data used for the tests are two KITTI [1] sequences (KITTI-00 and KITTI-04) and an VD56G3 indoor sequence acquired moving the device by hand. The KITTI-00 is the most complex sequences, it includes more than 2000 frames and several rotations. The KITTI-04 sequence presents a more regular trajectory and it consists of 270 frames. The VD56G3 sequence includes straight, slow motion and rotations+translation motion and it includes 800 frames On the contrary, the KITTI data are acquired at high speed in urban environment using a car equipped with several sensors.

Sequence	ORB-SLAM std		**ORB-SLAM OF-based**		Gap	
	Frame rate (fps)	Memory (MiB)	Frame rate (fps)	Memory (MiB)	Frame Rate (%)	Memory (%)
KITTI 00	3.86	590	3.97	445	+2.85	-24.58
KITTI 04	2.92	610	4.30	560	+ 47.26	-8.20
VD56G3	2.46	745	4.38	485	+78.05	-34.90

Table 1. Comparative analysis of ORB-SLAM standard and the proposed, OF-based version.

5.1. Experiments environment

The experiments have been done using a prototype board by STMicroelectronics called Stereo-Cam. It doesn't contains a microcontroller, but it can be connected to any 96board by a 96connector (see Figure 4). It also contains two Near-InfraRed (NIR) imaging VD56G3 sensor with hardware Optical Flow accelerator. Note that just one of the two VD56G3 sensor is used. The VD56G3 is an Advanced 1.5 Mpixel, backside illuminated Global Shutter, ultra compact sensor, optimized for near infrared scenes. The sensor captures up to 98 frames per second in a 1124 x 1364 resolution format. Moreover the prototype board has two IMU (3D accelerometer, 3D gyroscope) and an a VL53L5CX Time of Flight laser-ranging sensor. Such additional modules are not used to improve the SLAM. The VD56G3 add-on has been connected to a Dragonboard 410 C equipped with a Qualcomm® Snapdragon (four cores @1.4GHz). Since the purpose of the experiments is to evaluate the algorithm efficiency improved achieved replacing the FAST keypoints detector with the OF motion vectors, the latter have been calculated by software off-line in the case of KITTI sequences. The comparison between the original ORB-SLAM and our OF-based version have been performed measuring the heap memory consumption (in MiB) and the frame rate (frames per second) achieved by the two approaches on the test sequences.

It's important to point out that the ORB-SLAM2 is a desktop-designed algorithm, so it requires high power calculation. It should also be noted that no specific optimizations were made to the original algorithm in order to reduce such requirements. The Table 1 summarizes the results. The experiments on the KITTI-00 sequence shows that the results in terms of both frame rates are similar with the two approaches (3.86 versus 3.97), whilst the OF-based method performs better in terms of memory occupation: the peak in the case of the original algorithm is 590 MiB, the modified approach registered a maximum of 445 MiB. Exploiting the OF allows to achieve better performances also in terms of frame rate in the other sequences: more than 4 frame per second, versus less than 3 fps of the standard ORB-SLAM. The improvement has been achieved because the OF algorithm performs better on this kind of scenes and it provides a considerable amount of motion vectors suitable for the map construction in a faster manner compared to the ORB-based approach. The improvement is more evident looking at the last two columns of the Table 1, that represents the percentage gap. Considering the KITTI data, it is important to point out that they are images acquired at high speed, so the sequences includes pure rotation, a kind of motion that provides sudden scene changes, faster than new map points triangulation, so it often causes tracking lost too. Ideal camera motion to new areas consists of slow rotation and simultaneous translation. This helps to add new map points in the current area and, in turn, it helps tracking in the new area.

6. Conclusion and future works

Execution of SLAM algorithm originally designed for desktop PC always requires an effort to adapt them to different platform. This paper presented a modified version of the ORB-SLAM able to process near infrared images and Optical Flow data and a comparative analysis between such version and the original algorithm. Experiments performed using the Stereo-CAM equipped with the VD56G3 sensor by STMicroelectronics, having an ad-hoc optical flow HW accelerator, showed a performances improvement in terms of both frame rate and memory occupation. The next steps of this activity focus on the possibility of improvement of the algorithm efficiency and accuracy. In terms of efficiency, there are several opportunities to improve the performances. First of all, thanks to its features, the Stereo-cam can be connected to more recent 96 board taking advance by a richer set of resources, without any software modification. On the other side, the Stereo-cam's equipment includes two VD56G3 sensors, so the possibility of acquire stereo images, IMU and ToF sensors, that provides very useful information to be exploited in a SLAM system. The major challenge here is to maintain a temporal synchronization among different recorded data streams.

References

[1] R. Urtasun A. Geiger, P. Lenz. Are we ready for autonomous driving? the kitti vision benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] N. D. Molton O. Stasse A. J. Davison, I. D. Reid. On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[3] R. Hartley A. Fitzgibbone B. Triggs, P. Mclauchlan. Bundle adjustment, a modern synthesis. *International Workshop on Vision Algorithms, Sep 2000, Corfu, Greece*, page 298–372, 2010.

[4] M. Stephens C. Harris. A combined corner and edge detection. *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[5] Carlos Campos, Richard Elvira, Juan J. Gomez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020.

[6] Henry Carrillo Yasir Latif Davide Scaramuzza Jose Neira Ian Reid John J. Leonard Cesar Cadena, Luca Carlone. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age, 2016.

[7] A. Rabinovich D. DeTone, T. Malisiewicz. Toward geometric deep slam. *https://arxiv.org/abs/1707.07410*, 2017.

[8] J. D. Tardos D. Gálvez-López. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[9] J. Delmerico and D. Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, page 2502–2509, Brisbane, QLD, Australia, May 2018.

[10] T. Drummond E. Rosten. Machine learning for highspeed corner detection. *Computer Vision – European Conference on Computer Vision (ECCV) 2006. Lecture Notes in Computer Science, vol 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds., Springer, Berlin, Heidelberg,*, page 430–443, 2006.

[11] K. Konolige G. Bradski E. Rublee, V. Rabaud. Orb: an efficient alternative to sift or surf. *Proceedings of The Fourth Alvey Vision Conference*, pages 2564–2571, 2011.

[12] L. Van Gool H. Bay, T. Tuytelaars. Distinctive image features from scale-invariant keypoints. *Computer Vision – European Conference on Computer Vision (ECCV)*, page 404–417, 2006.

[13] D. Cremers J. Engel, T. Schöps. Robust visual inertial odometry using a direct ekf-based approach. *Proceedings of the Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Springer, Cham.*, page 298–304, 2014.

[14] D. Cremers J. Engel, V. Koltun. Direct sparse odometry. *https://arxiv.org/abs/1607.02565*, 2016.

[15] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, page 225–234, Nara, Japan, November 2007.

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

[17] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[18] M. Hutter R. Siegwart M. Bloesch, S. Omari. Robust visual inertial odometry using a direct ekf-based approach. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 298–304, Hamburg, Germany, September 2015.

[19] C. Strecha P. Fua M. Calonder, V. Lepetit. Brief: binary robust independent elementary features. *in Computer Vision – ECCV 2010. ECCV 2010, K. Daniilidis, P. Maragos, and N.Paragios, Eds.*, 6314of Lecture Notes in Computer Science, Springer:1052–1067, Berlin, Heidelberg, 2010.

[20] D. Koller B. Wegbreit M. Montemerlo, S. Thrun. Fastslam: a factored solution to the simultaneous localization and mapping problem. *Proceedings of the AAAI National Conference on Artificial Intelligence*, page 593–598, 2002.

[21] A. Dupuis N. Antigny M. Servières, V. Renaudin. Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking, 2021.

[22] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[23] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[24] J. Almaz'an L. M. Bergasa P. F. Alcantarilla, J. J. Yebes. On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments. *Proceedings of the IEEE International Conference on Robotics and Automation*, page 1290–1297, Saint Paul, MN, USA, May 2012.

[25] B. Hu F. Zhu S. Shen P. Li, T. Qin. Monocular visualinertial state estimation for mobile augmented reality. *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, page 11–21, 2017.

[26] A. J. Davison R. A. Newcombe, S. Lovegrove. Dtam: dense tracking and mapping in real-time. *Proceedings of the International Conference on Computer Vision*, 8690:2320–2327, Barcelona, Spain, November 2011.

[27] Dongbing Gu Ruihao Li, Sen Wang. Ongoing evolution of visual slam from geometry to deep learning: Challenges and opportunities. *Cognitive Computation*, 10(6):875–889, 2018.

[28] D. Cremers T. Schöps, J. Engel. Semi-dense visual odometry for ar on a smartphone. *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, page 145–150, 2014.