Robotica: (2021) volume 39, pp. 277–290. © The Author(s), 2020. Published by Cambridge University Press.
doi:10.1017/S0263574720000363

Constrained RGBD-SLAM

Sylvie Naudet-Collette[†]*, Kathia Melbouci[‡], Vincent Gay-Bellile[‡], Omar Ait-Aider[¶] and Michel Dhome[¶]

[†]CEA LIST, DRT/LIST/DIASI/SIALV, DRT/LIST/DIASI/SIALV, CEA Saclay Nano-INNOV, Gif-sur-Yvette, 91191
[‡]CEA, LIST, Artificial intelligence Language and Vision Laboratory, F-91191 Gif-sur-Yvette, France
E-mails: kathia.melbouci@cea.fr, vincent.gay-bellile@cea.fr
[¶]Pascal Institut, UMR 660, Blaise Pascal University, 63000 Clermont Ferrand, France
E-mails: omar.ait-aider@uca.fr, michel.dhome@uca.fr

(Accepted May 3, 2020. First published online: June 2, 2020)

SUMMARY
This paper introduces a new RGBD-Simultaneous Localization And Mapping (RGBD-SLAM) based on a revisited keyframe SLAM. This solution improves the localization by combining visual and depth data in a local bundle adjustment. Then, it presents an extension of this RGBD-SLAM that takes advantage of a partial knowledge of the scene. This solution allows using a prior knowledge of the 3D model of the environment when this latter is available which drastically improves the localization accuracy. The proposed solutions called RGBD-SLAM and Constrained RGBD-SLAM are evaluated on several public benchmark datasets and on real scenes acquired by a Kinect sensor. The system works in real time on a standard central processing units and it can be useful for certain applications, such as localization of lightweight robots, UAVs, and VR helmet.

KEYWORDS: Simultaneous localization and mapping; RGBD sensor; Bundle adjustment; 3D model; RGBD-SLAM.

1. Introduction

To safely navigate, especially in GPS-denied environments such as indoor environments, a robot must have the ability to localize itself accurately using visual sensor. This problem is referred to as Simultaneous Localization And Mapping (SLAM). Visual SLAM techniques reconstruct a sparse map of an unknown environment and simultaneously localize the sensor in this map by tracking salient image points across frames of video. Over the last few years, several works on real-time sparse and dense SLAM have been published.[1–5] Despite this, several challenges persist, such as localization in challenging environments and the problem of drift over long trajectories.

Indeed, visual SLAM is an incremental process prone to small drifts which when integrated over time, become increasingly significant over large distances. To correct the accumulated error, loop closure can be performed but this solution requires that the sensor returns to a mapped area place in order to relocalize the camera, which is not always possible.

With the emergence of RGBD sensors in particular the Microsoft Kinect, which provide pixel-level depth (D) and visual (RGB) information, the use of the 3D data in SLAM algorithms becomes increasingly widespread. Indeed, the use of depth data may solve the issues cited earlier and may allow reliable SLAM solutions. Nonetheless, the problems of such sensors are their limited range and noisy depth maps with missing data for reflective or highly absorptive surfaces.

To cope with these limitations, the proposed solution is focused on the integration of additional information into an existing monocular visual SLAM system in order to constrain the camera localization and the mapping points. This additional information is the depth provided by a 3D sensor

* Corresponding author. E-mail: sylvie.naudet@cea.fr

(Microsoft Kinect) and geometric information about scene structure. This work has been conducted with an objective to keep the high speed of the initial SLAM process but also with the goal to avoid that the lack of these precited additional constraints should give rise to the failure of the process. Therefore, the proposed solution can work just like a monocular SLAM when these constraints are not available.

We propose a new RGBD-SLAM with the two following contributions. The primary contribution consists in modifying a visual SLAM algorithm[2] to take into account the depth measurement provided by a 3D sensor. It mainly consists in combining the depth and visual data in a local bundle adjustment (BA) process. The second contribution is a solution that uses, in addition to the depth and visual data, constraints lying on points which belong to the 3D model of the scene. We called this solution as Constrained RGBD-SLAM. In contrast to the state-of-the-art RGBD approaches, the proposed method operates in real time on central processing unit (CPU) and it is able to deal with environments where the depth maps are very poor or unavailable using not only the depth information but also the triangulation of visual features. In the same way, when 3D model is unavailable, the system works like an RGBD-SLAM.

This paper brings together the work presented in two previous publications.[6,7] In this paper, we provide a number of additions to that work including new experiments and comparison with the state of the art. We discuss related work in Section 2, describe our RGBD-SLAM in Section 3, and present the Constrained RGBD-SLAM that includes depth and geometric data in Section 4. Section 5 provides a quantitative evaluation using multiple benchmark datasets. Section 6 presents conclusions on the work and future directions of our research.

2. Related Work

Several RGBD-SLAM approaches have been proposed in the state of the art.[8–18] Newcombe et al.[8] in the KinectFusion method proposed to incrementally build a dense model of the scene which is refined by registering every new depth measurement to this model and used to estimate the camera poses by ICP. This approach is limited to small scenes due to its volumetric representation. Whelan et al.[12] with their Kintinuous algorithm extended the KinectFusion algorithm to large scenes using a rolling shutter cyclical buffer, with global optimization. Huang et al.[11] proposed an RGBD-SLAM based on FAST feature correspondences for visual odometry and sparse BA for global consistency. Similarly, Endres et al.[15] proposed a feature-based RGBD-SLAM, where the front-end computes frame-to-frame motion by feature matching and ICP and the back-end performs pose-graph optimization with loop closure. In a similar way, Belter et al.[19] proposes also a feature-based RGBD-SLAM system with a factor graph optimization including 3D features optimization. Kerl et al. proposed the Dense Visual Odometry (DVO):[14] a dense photometric-based RGBD visual odometry system that combines dense tracking with keyframe selection and pose-graph optimization. Most of the above approaches are focused on dense map reconstruction and require graphics processing unit (GPU) to achieve quasi real-time performance. To avoid this limitation, other types of approaches are based on a sparse visual SLAM: they propose to integrate depth measurements into an existing visual keyframe-based SLAM.[16,17,20] Scherer et al.[20] revisit the PTAM-SLAM to exploit depth data and propose its extension in ref. [17] with graph optimization to be able to operate in large environments. Mur-Artal et al.[16] revisit the ORB-SLAM to integrate the depth information of each ORB feature in a BA like a stereo point by synthesizing a right coordinate for each feature using the associated depth value. Their system performs a local BA to optimize the camera pose and points in a local mapping, and after a loop closure to optimize all keyframes and points in a full BA.

In order to reduce the drift inherent in any iterative methods, most of the above approaches propose to incorporate loop closure.[12–16] Unfortunately, this technique cannot be applicable in every environment. It depends on the camera trajectory. Some recent methods propose to use a prior map of the environment to improve the camera localization and limit the drift without using loop closure. A number of them rely on particle filter methods with 2D Lidar data[21] and recently with RGBD data.[22–24] Fallon et al.[22] proposed an RGBD Monte-Carlo localization (MCL) approach for indoor environments. Their technique requires a preliminary mapping phase, where only major plan segments are considered. Using this map as input, visual odometry is used for particle propagation and a likelihood function is computed for each particle by comparing the current depth data with a prediction of the scene geometry. A similar work was presented by Scherer et al..[23] The authors compute the frame-to-frame motion using depth and photometric information. Then, a particle filter-based method is proposed to locate the camera in the given 3D global map. Winterhalter et al.[24] presented an MCL approach for RGBD sensors where odometry is computed from RGBD and IMU data. The likelihood model for the particle filter assigns a probability to the measurement using a function of a distance between the depth measurement and the floor plan given by the environment model.

These MCL methods are usually computationally expensive due to the necessary large number of particles for accurate localization and require GPU processing to operate in a receivable time. Moreover, these approaches exploit only the depth data in the particle filter algorithm and may suffer from the lack of depth information in challenging large indoor environments with reflective surfaces.

In this paper, we first introduce a new RGBD-SLAM based on a revisited keyframe SLAM. Then, we present an extension of this RGBD-SLAM that takes advantage of a partial knowledge of the scene (partial 3D model of the environment, for example, generated from the 2D plan of the building). This information is integrated in a local BA process to provide long-term and globally consistent localization. We call this new approach as Constrained RGBD-SLAM. The proposed solution uses the depth and visual data and, when available, the 3D model of the scene to constrain the localization. In contrast to the MCL approaches cited before, the proposed method operates in real time on CPU and it is able to deal with environments where the depth maps are very poor or unavailable using not only the depth information but also visual feature triangulation. In the same way, when 3D model is unavailable, the system processes like an RGBD-SLAM.

Our contributions are summarized in the following list:

- We first propose an RGBD-SLAM based on a revision of a keyframe SLAM which integrates visual and depth data in a local BA.
- Secondly, we propose an extension called Constrained RGBD-SLAM to integrate the geometric information provided by a prior 3D model along with visual and depth data in a local BA pipeline. The cost function is a combination of different types of residual errors: error based on visual features, error based on depth data, and error based on geometric constraints provided by a 3D model of the environment. Therefore, the proposed approach may not suffer in environments where the RGBD sensor does not provide depth or in environments with an incomplete 3D model since in this case the other visual information are still used.
- The algorithm runs in real time on CPU.

3. The Proposed RGBD-SLAM

3.1. Notation

Let us note the pose of the frame j as:

$$P_j = \begin{bmatrix} R_j & t_j \\ 0_{1\times 3} & 1 \end{bmatrix}$$

We note $q_{i,j}$ the observation of the 3D point Q_i in the camera j such that $q_{i,j} = \pi(K P_j Q_i)$. $\pi(x, y, z)$ describes the perspective projection function such that $\pi(x, y, z) = (x/z, y/z)^\top$, and K denotes the camera calibration matrix.

3.2. Overview of the RGBD-SLAM algorithm

The proposed RGBD-SLAM is a monocular SLAM revisited in order to exploit depth data. It is built on the monocular keyframe SLAM proposed by ref. [2] but it could be built on other monocular keyframe SLAM like.[5,25] The main components are summarized in Fig. 1 that presents how depth information is exploited and which elements of the system are affected.

In this algorithm, two main steps are performed: (1) the estimation of the sensor pose between the consecutive frames and (2) the refinement of the latest poses and the 3D points through a local BA process. As illustrated in Fig. 1, the approach consists in solving in real-time and incrementally conventional computer vision tasks (matching, pose computation, triangulation, and BA). For each incoming frame, key points are detected using the SURF detector[26] and matched with 3D points, resulting in 2D–3D correspondences. The inter-frame pose is estimated by robust minimization of the re-projection error. If the camera has moved at least a predefined distance, or if the number of visible 3D points falls below a threshold, a new keyframe is created. At each new keyframe, new

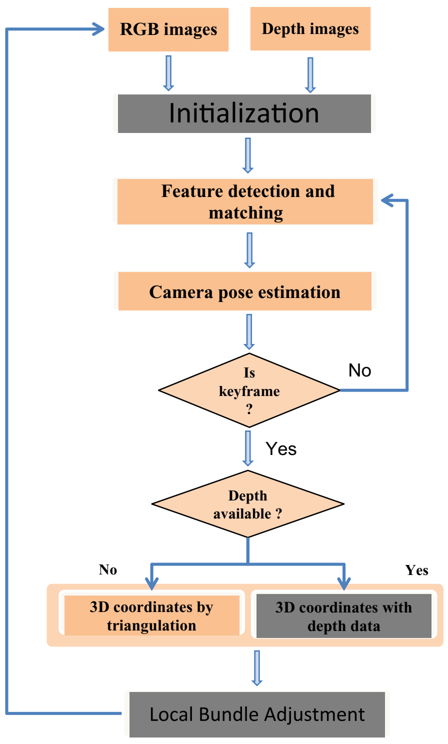

Fig. 1. Schematic overview of our approach. In gray, the elements affected by the integration depth information in the process.

3D points are added to the map using the depth measurement when available or using triangulation otherwise. The camera poses and the 3D map are then refined in a local BA which combines the classical visual error and the depth measurement. These two types of error used in the cost function of the BA are presented below.

We have chosen not to take depth information into account in the inter-frame pose estimation. Indeed as the pose will be optimized in the BA, adding the depth measurements during inter-frame pose estimation would not have a significant impact. This might even degrade the computational performances. That explains why we have chosen to integrate the depth measurement only in the initialization step and in the BA. The choice of SURF has been guided by the study of ref. [27] which compares SURF, SIFT, and ORB techniques in term of processing time and robustness against different kinds of transformations. According to ref. [27], SIFT provides the highest matching rate in the most scenarios but the matchers ORB and SURF are significantly faster than SIFT. ORB and SURF show quasi-equal performances concerning the computational time but SURF is more robust against distortions like intensity variation and rotations, which are mainly encountered in successive images. This last point is confirmed in the recent study of ref. [28].

3.3. BA with visual and depth data

BA optimizes the camera poses and the 3D points they observed by minimizing the 2D re-projection error in the keyframes. This error is the difference between the estimated projection of the point Q_i through the camera P_j, and its corresponding observation $q_{i,j}$. Conventional BA approaches simultaneously optimize the poses of all keyframes and the 3D points they observed using measurements obtained from 2D images. To achieve time-efficient performances, the local BA[2] optimizes only the poses of the N_c latest cameras and the N_p 3D points they observed. The local BA cost function is given by:

$$\varepsilon_{\text{visual}}\left(\{P_j\}_{j=0}^{N_c}, \{Q_i\}_{i=0}^{N_p}\right) = \sum_{i=0}^{N_p} \sum_{j \in A_i} \left(q_{i,j} - \pi(K P_j Q_i)\right) \tag{1}$$

where A_i denotes the set of keyframes indices observing Q_i

To take advantage of the depth measurement, one solution is to integrate it, as a new constraint into BA, in addition to the conventional 2D error $\varepsilon_{\text{visual}}$.

Let $q_{i,k} = (u_{i,k}, v_{i,k}, 1)^\top$ be the observation of the 3D point Q_i in the frame k, and $d_{i,k}$ its depth.

We compute the 3D point $Q_{i,k}$ corresponding to the observation $q_{i,k}$ using the inverse of the projection function π as mentioned in Eq. (2)

$$Q_{i,k} = \pi^{-1}(q_{i,k}, d_{i,k}) = d_{i,k} K^{-1} q_{i,k} \tag{2}$$

In the world coordinate system, this point is expressed as:

$$Q_i = P_k^{-1} \pi^{-1}(q_{i,k}, d_{i,k}) \tag{3}$$

We measure the 2D projection error of the reconstructed 3D point Q_i in each frame j on which it appears, with $(j \in A_i$ and $j \neq k)$. The resulting cost function is given by:

$$\varepsilon_{\text{depth}}\left(\{P_j\}_{j=0}^{N_c}\right) = \sum_{i=0}^{N_p} \sum_{j \in A_i} \sum_{k \in A_i}^{k \neq j} \left(q_{i,j} - \pi\left(K P_j P_k^{-1} \pi^{-1}(q_{i,k}, d_{i,k})\right)\right) \tag{4}$$

Hence, the overall error minimized in the BA of our RGBD-SLAM is summarized as:

$$\varepsilon_{visual.depth} = \varepsilon_{\text{visual}} + \varepsilon_{\text{depth}} \tag{5}$$

4. Constrained RGBD-SLAM

4.1. Motivation

The evaluation of the proposed RGBD-SLAM will be presented in Section 5.1. We will see that this solution drastically reduces drift of a classical visual SLAM but does not allow to totally remove it. Indeed, like many RGBD solutions, the drift recurs over long trajectories, more specially in areas where the depth is not completely available (as illustrated in Fig. 2) and when loop closure is not possible.

To handle this problem, we introduce in the cost function of the BA an additional error which constrains some 3D points to belong to the 3D model of the environment. Note that this 3D model can be easily generated from the 2D plan of the building.

4.2. BA with depth and plane constraints

The proposed solution is inspired from the work of Tamaazousti et al.[29] The authors propose to constrain the mapping of a visual SLAM using the prior knowledge of the 3D model of an interest object in the scene. The main idea is that a 3D point Q_i lying on a plane π_i of the 3D model has only two degrees of freedom, thus this 3D point is forced to move only on its associated plane.

Lets M^{π_i} be the known transfer matrix between the coordinate frame of plane π_i and the world coordinate frame.

Then, $Q_i = M^{\pi_i} Q_i^{\pi_i}$ where $Q_i^{\pi_i} = (X^{\pi_i}; Y^{\pi_i}; 0; 1)^T$ and $(X^{\pi_i}; Y^{\pi_i})$ are the coordinates of Q_i in the plane π_i coordinate frame. Note that the association process to match each 3D point with its corresponding plane is detailed in ref. [29].

This relation is integrated to the optimization process as follows:

$$\varepsilon_{plan}\left(\{P_j\}_{j=0}^{N_c}, \{Q_i\}_{i=0}^{N_p}\right) = \sum_{i=0}^{N_p} \sum_{j \in A_i} \left(q_{i,j} - \pi(K P_j M^{\pi_i} Q_i^{\pi_i})\right) \tag{6}$$

where N_p is the set of the 3D points observed by the A_i cameras and N_c the number of cameras to optimize.

Our contribution is to add the depth information along with the constraint given by the model's plans. The idea is that for each 3D point for which the depth measurement is available and to which

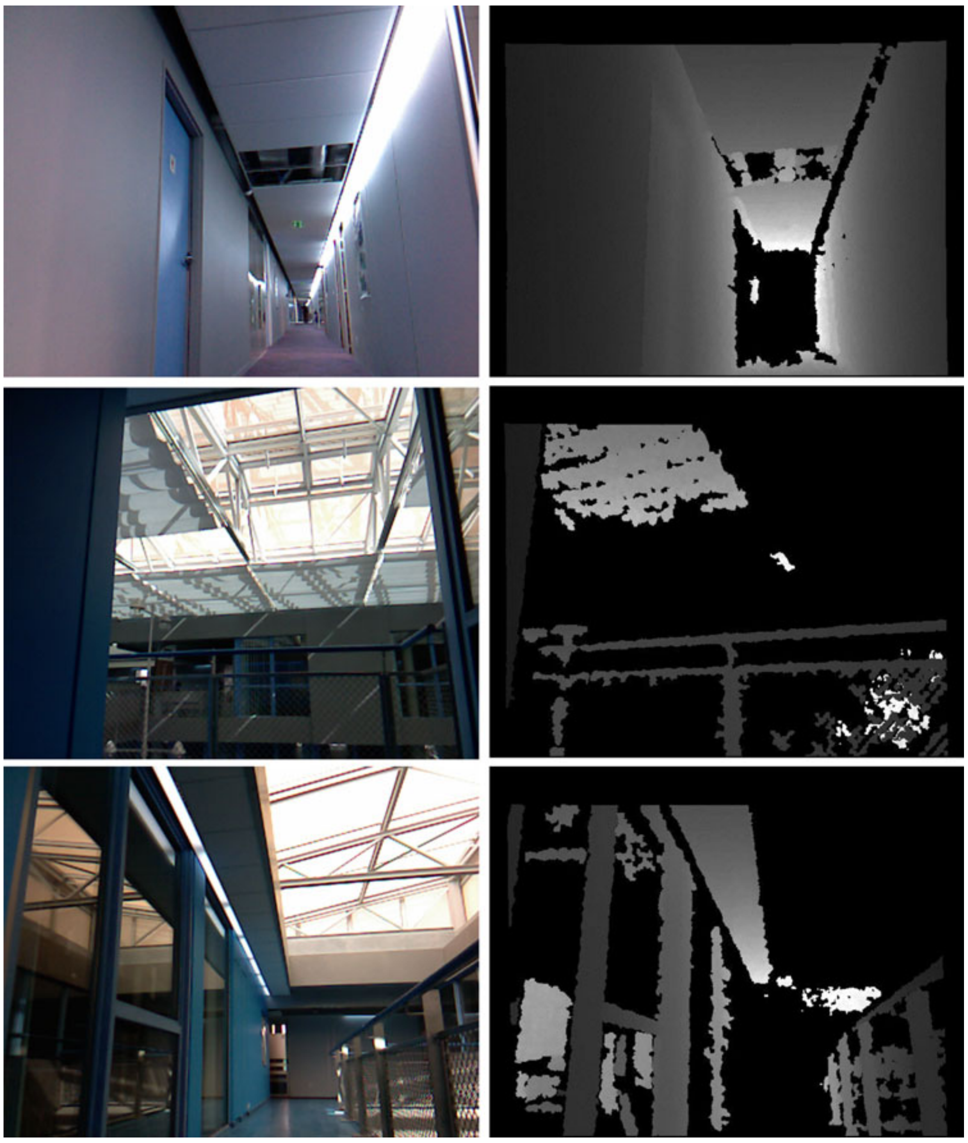

Fig. 2. Localization in challenging environment: illustration of the depth (right) and color images (left) in the textureless environment and in environment with missing depth measurements.

a plane has been associated, we add its depth reprojection error, considering that this 3D point has only two degree of freedom.

$$\varepsilon_{\text{depth-and-plan}}\left(\{P_j\}_{j=0}^{N_c}, \{Q_i\}_{i=0}^{N_p}\right) = \sum_{i=0}^{N_p} \sum_{j \in A_i} \left(\frac{d_{i,j} - [P_j M^{\pi_i} Q_i^{\pi_i}]_z}{\alpha d_{i,j}^2} \right). \tag{7}$$

where α is an experimentally estimated scaling parameter which models the uncertainty of the depth data. The determination of its value is explained in ref. [20]. This parameter could be tuned for each sequence according to its complexity. In our experiments, we found that setting $\alpha = 3, 3310e - 2$ was satisfying for all sequences.

The constraints given by the 3D model of the environment are integrated as

$$\varepsilon_{model} = \varepsilon_{plan} + \varepsilon_{depth.and.plan} \tag{8}$$

The overall error to be minimized in the BA of the Constrained RGBD-SLAM is given by

$$\varepsilon_{visual.depth.model} = \varepsilon_{visual.depth} + \varepsilon_{model} \tag{9}$$

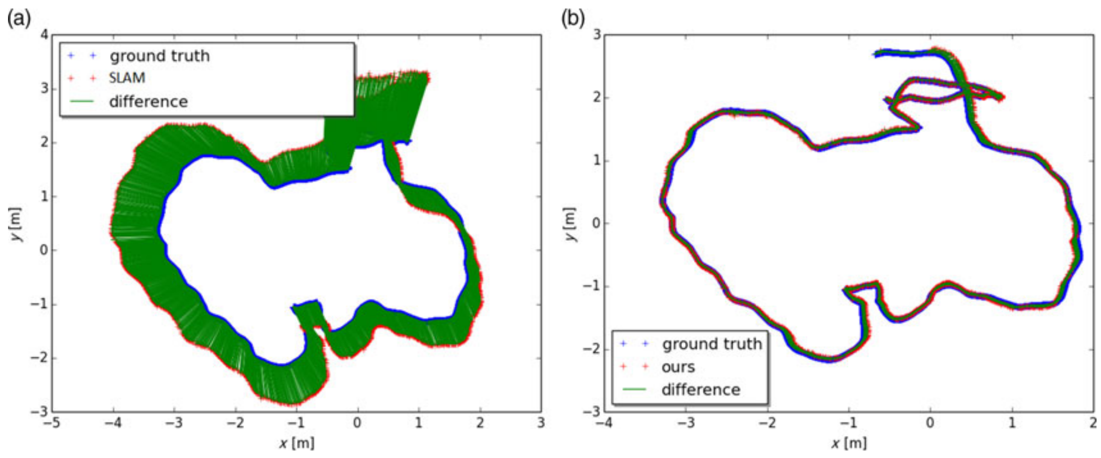

Fig. 3. Estimated trajectories on the freiburg3-long-office-household sequence. Left: SLAM trajectory.[2] Right: RGBD-SLAM trajectory.

Note that in practice, in the Levenberg–Marquardt algorithm used in BA, $L2$ norm in Eqs. (1), (4), (6), and (7) is replaced by a robust norm to deal with outliers (i.e., inconsistent feature matches, wrong data/model association, erroneous depth). The optimization is performed with the Geman–McClure estimator.

Notice that the proposed BA retains the sparse blocks structure of the matrices involved in the optimization. Like in the classical Bundle Ajustment, it is thus possible to implement it efficiently, taking into account sparse structure blocks, as described in ref. [30].

5. Evaluation

This section is divided into two subsections. The first one presents the evaluation of the proposed RGBD-SLAM. The second one is focused on the Constrained RGBD-SLAM.

5.1. RGBD-SLAM evaluation

We have evaluated our RGBD-SLAM on the popular TUM-RGBD dataset[9] which contains indoor sequences from *Microsoft Kinect* ($V1$) sensor with different textures, illuminations, and trajectories. This benchmark provides synchronized ground truth poses for the RGBD sensor moved through an environment, captured with a highly precise motion capture system. Like proposed in ref. [9], we use the absolute trajectory (ATE) root mean square error (RMSE) metric to evaluate our system, which measures the root mean square of the Euclidean distances between all estimated camera poses and the ground truth poses associated by timestamp .

We first evaluate the gain obtained using depth information by comparing our RGBD-SLAM with the original visual SLAM[2] in multiple sequences of the dataset. The results of Fig. 3 and the tests conducted in several sequences reported in Table I demonstrate that integrating depth data in the BA of SLAM process improve significantly the accuracy and reduce the drift.

We also evaluate the computational performances of our system on a sequential single thread using a single core from an Intel Xeon W3570 at 3.20 GHz. We measured that the average run time required to process each frame is about 25 ms (the inter-frame pose estimation requires an average of 20 ms and BA of 38 ms).

Then we compare our method to other state-of-the-art RGBD-SLAM systems, using the results published by the original authors. Table II compares our RGBD-SLAM accuracy with the following state-of-the-art methods: DVO,[14] RGBD,[31] Kintinuous,[12] dSLAM,[20] and ORB-SLAM2.[16] Concerning DVO, we note DVO(1) – the results published by[14] and DVO(2) – the results we obtained with the source code.[1] Boldface highlights the better results.

Note that the results published by DVO(1), RGBD, Kintinuous, and ORB-SLAM2 use loop closure in their optimization to reduce the error accumulation which explains their better accuracy. Loop closure requires that the sensor returns to a mapped area place in order to relocalize the camera using

[1]https://vision.in.tum.de/data/software/dvo.

Table I. Comparison of our RGBD-SLAM with the visual SLAM on TUM-RBGD dataset: absolute trajectory error (ATE) measured in meters (m).

		ATE (m)	
Sequences	**Algorithms**	RMSE	STD
fr1/desk	SLAM	0.631	0.201
	RGBD-SLAM	**0.062**	0.024
fr1/xyz	SLAM	0.183	0.061
	RGBD-SLAM	**0.028**	0.091
fr1/rpy	SLAM	–	–
	RGBD-SLAM	**0.056**	0.025
fr1/plant	SLAM	–	–
	RGBD-SLAM	**0.075**	0.027
fr2/xyz	SLAM	0.044	0.029
	RGBD-SLAM	**0.021**	0.015
fr3/long	SLAM	0.654	0.302
	RGBD-SLAM	**0.034**	0.019

Table II. Comparison of ATE-RMSE (m) in TUM-RGBD dataset. "–" indicates that no results have been published. Boldface highlights the better results.

Seq	**Ours**	**dSLAM**	**DVO(2)**	**RGBD**	**Kinti-nuous**	**DVO(1)**	**ORB-SLAM2**
Loop closure	**no**	**no**	**no**	**yes**	**yes**	**yes**	**yes**
fr1/desk	**0.062**	–	0.229	0.049	0.037	0.021	**0.016**
fr1/xyz	**0.028**	–	0.029	0.021	0.017	**0.011**	–
fr1/rpy	**0.056**	–	0.072	0.042	0.028	**0.020**	–
fr1/plant	**0.073**	–	0.097	0.142	0.041	**0.028**	–
fr2/xyz	**0.021**	–	0.023	–	0.029	0.018	**0.004**
fr3/long	**0.034**	0.136	0.222	0.067	**0.031**	0.035	–

reconstructed points, which is not always possible. Our method does not perform loop closure: we want to evaluate its accuracy in cases where revisiting known parts of the map is not possible. So it is interesting to focus the comparison on methods without loop closure: DSLAM, DVO(2). The results show that our method is more accurate than DVO(2) and dSLAM.

An other important point concerns the computation time. The methods DVO, RGBD, and Kintinuous are dense methods, and they use all the points of the image. Therefore, they are expensive in computation time and require GPU to achieve real-time performance. Our goal is to propose an algorithm working only on CPU for use on a lightweight structure with high autonomy like small robots, UAVs, or VR helmet. The only methods which operate in real time on CPU are dSLAM, ORB-SLAM2, and ours. Note that these three methods are all sparse keyframe SLAM approaches revisited. We indicate in Table III their average processing time using the results published by the original authors. It is difficult to compare their performances because they depend on the type of CPU used but our solution seems to be faster. We can mention that their average time processing is around 25 and 40 ms: these methods run in real time without using GPU. This average processing time depends also on the number of keyframes created by the algorithm: each creation of a keyframe involves a new local BA, so a new mapping process. So we mention also the processing time of each process: tracking and mapping. The mapping process of our solution is significantly faster than the others. This can be explained by the optimization of the sparse blocks structure of the matrices and the low number N_c of estimated poses in our BA (see Section 4.2).

These results show that our solution has globally comparable accuracy than approaches of the state of the art, with the advantage of being less computationally expensive. The second benefit of this solution is that it can work in different types of environments like large open spaces, corridors, indoor, or outdoor environments. Indeed, the proposed cost function in BA allows to perform localization in

Table III. Type of method, type of processor CPU/GPU, and processing time for the sparse methods which operate in real time on CPU.

Method	**Ours**	**dSLAM**	**DVO(2)**	**RGBD**	**Kinti-nuous**	**DVO(1)**	**ORB-SLAM2**
Type	**Sparse CPU**	**Sparse CPU**	**Dense CPU**	**Dense GPU**	**Dense GPU**	**Dense GPU**	**Sparse CPU**
Processor	Intel Xeon W3570 at 3.20 GHz using 1core	Intel Duo P9400 at 2.40 GHz using 1core	–	–	–	–	Intel Core i7-4790 (4 cores @ 3,6 GHz)
Average process time	25 ms	33 ms	–	–	–	–	$<$ 33 ms
Tracking time	20 ms	12 ms	–	–	–	–	26 ms
Mapping time	38 ms	148 ms	–	–	–	–	267 ms

a wide range of environments by automatically switching between two modes: classical monocular SLAM mode when depth information is unavailable or RGBD-SLAM mode, using depth and visual data, when depth information is available. This way our system is agnostic to the input being RGB or RGBD.

5.2. *Constrained RGBD-SLAM evaluation*

This subsection is dedicated to the evaluation of the Constrained RGBD-SLAM. The main objective of this algorithm is to reduce the drift over long trajectories even when loop closures are not possible, by taking advantage of a partial knowledge of the scene.

We evaluate the algorithm on real sequences from the benchmark "CoRBS" dataset.[32] This dataset is composed of different sequences acquired with the *Microsoft KinectV2*. Each sequence includes a 3D model of an interest object in the scene and groundtruth camera poses. Figure 4 illustrates three of them. Unfortunatly, for these datasets, we have no comparison with other RGBD methods of the state of the art using prior knowledge of the scene. For this reason, the presented evaluations concern only the comparisons between the visual SLAM,[2] the Constrained SLAM,[29] our RGBD-SLAM (Eq. (5)), and our Constrained RGBD-SLAM (Eq. (9)). This comparison results from the comparison of performances of our algorithm by taking into account a part or the totality of the terms of the cost function in Eq. (9).

Figure 5 shows the evolution of the position error along the "Human" sequence for four algorithms. The results confirm that adding depth information as an additional constraint in BA of the visual SLAM process significantly improves its precision. Indeed, the visual SLAM suffers from significant drift due to the difficulties like blur and textureless images. If we analyze the evolution of the ATE, we see peaks of errors between the images 1190 and 1194. This can be explained by the high angular velocity of the camera on this part of the sequence that causes motion blur and strongly affects the frame-to-frame motion estimation. This error is compensated by the BA only with the Constrained RGBD-SLAM. The results obtained in the other sequences from the "CoRBS" dataset are summarized in Table IV. They highlight that the best performance is obtained by the Constrained RGBD-SLAM for all sequences. We can note that the difference between the RGBD-SLAM and the Constrained RGBD-SLAM is not very significant on the "Desk" sequence which is the easier sequence (textured sequence without abrupt motion). However, in the "Human" sequence which is a textureless sequence with abrupt motion and high average angular velocity causing motion blur, the Constrained RGBD-SLAM is three times more precise than the RGBD-SLAM.

We have also evaluated Constrained RGBD-SLAM on a real sequence using the Microsoft Kinect V1. The range of this sensor is about 5 m. The experiment is carried out on a difficult sequence of about 60 m without loop closure. The scene is composed of:

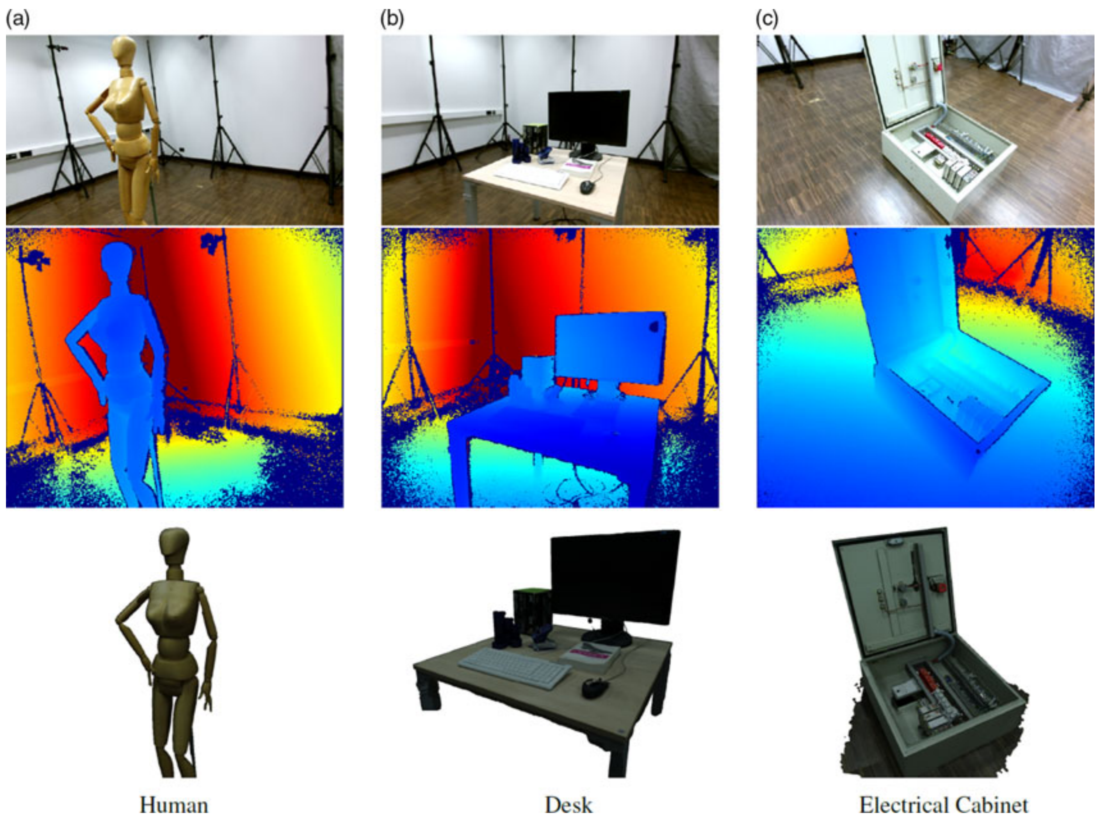

Fig. 4. Sequences “human”, “Desk”, and “Electrical Cabinet” from the “CoRBS” dataset[32] and their associated 3D model. The top row shows the color images, the middle row shows color-coded depth images recorded by the Kinect V2, and the botton row presents the associated 3D models. (a) Human. (b) Desk. (c) Electrical Cabinet.

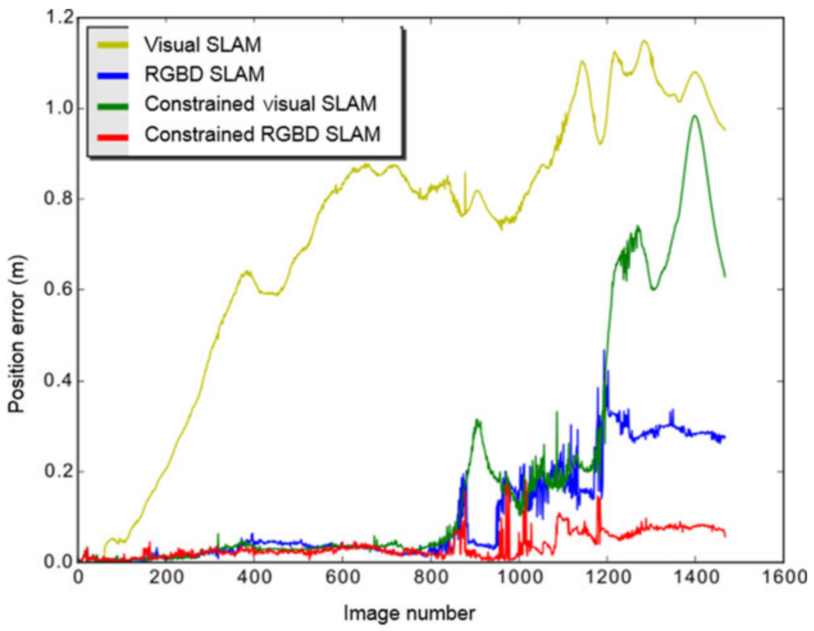

Fig. 5. Comparison of the four algorithms in the “human” sequence of “CoRBS” dataset. The camera trajectory is about 12 m.

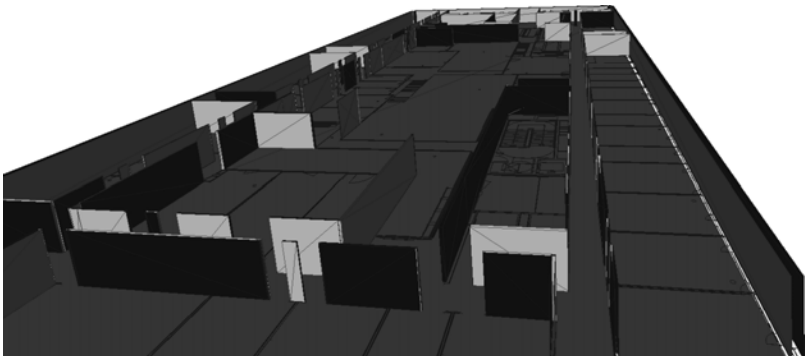

Fig. 6. 3D model of the environment obtained from the 2D plan of building.

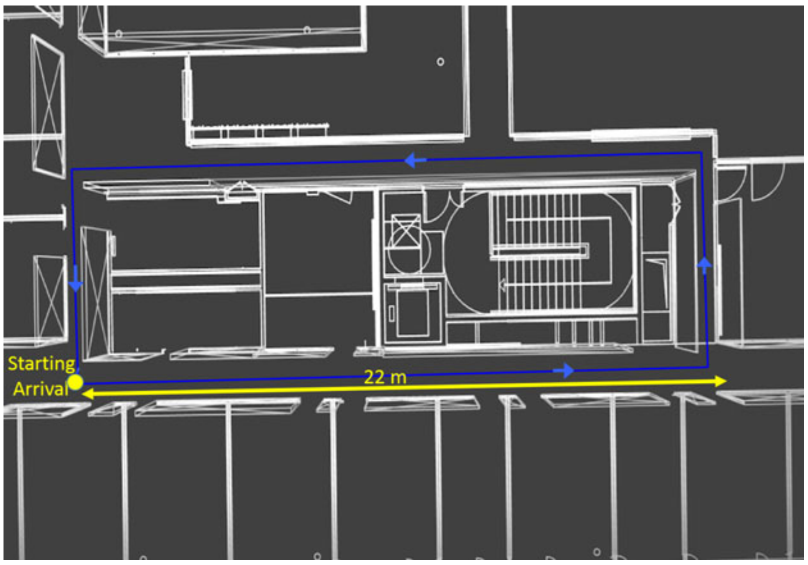

Fig. 7. Illustration of the path followed by the camera in the real scene.

- long (2 m) and narrow (1.5 m) corridors with doors and some walls with glass surfaces.
- large areas where the depth is about 20 m.

Figure 2 shows some images of the real sequence. We note that some areas are textureless and some depth images are very poor (they have missing values) due to the limited range of the sensor or to the reflective surfaces. Unfortunately, no ground truth is available for this trajectory. So we present the resulting trajectory only in the 3D model and we compare it with the representation of the followed path illustrated in Fig. 7. The 3D model is presented in Fig. 6.

This trajectory has been manually measured. During the navigation, we try to keep the sensor at the center of the hallway and we took care to stop the trajectory at the final position identical to the starting position (see Fig. 7). The trajectories estimated with the RGBD-SLAM and the Constrained RGBD-SLAM are shown in Fig. 8. For the RGBD-SLAM, we obtain an error at the end of the trajectory of 5.5 m which is 9% of the global trajectory with a significant deviation of the trajectory. With the Constrained RGBD-SLAM, the final drift is around 1.5 m, which corresponds to 2.5% of the global trajectory. From these results, we can see the positive contributions of the model constraints for reducing the drift and improving the accuracy of the localization.

Indeed, the poor depth maps and the lack of visual features or geometric structures (long corridors) pose big challenges for the localization. The proposed approach overcomes this limitation thanks to the model constraint. Even when the frame-to-frame motion estimation is not precise, the model constraint helps to correct it in the BA process.

Table IV. Statistical errors (ATE) on the "CoRBS" sequences with visual SLAM, RGBD-SLAM, Constrained SLAM, and Constrained RGBD-SLAM. The symbol "–" indicates that the visual SLAM was unable to produce an estimate.

		ATE (m)	
Sequences	**Algorithms**	**RMSE**	**STD**
Human	Visual SLAM	0.583	0.172
	RGBD-SLAM	0.114	0.055
	Constrained visual SLAM	0.283	0.164
	Constrained RGBD-SLAM	**0.036**	**0.017**
Desk	Visual SLAM	0.165	0.041
	RGBD-SLAM	0.016	0.006
	Constrained visual SLAM	0.022	0.008
	Constrained RGBD-SLAM	**0.013**	**0.005**
Electrical	Visual SLAM	–	–
	RGBD-SLAM	0.065	0.020
	Constrained visual SLAM	0.057	0.031
	Constrained RGBD-SLAM	**0.040**	**0.017**

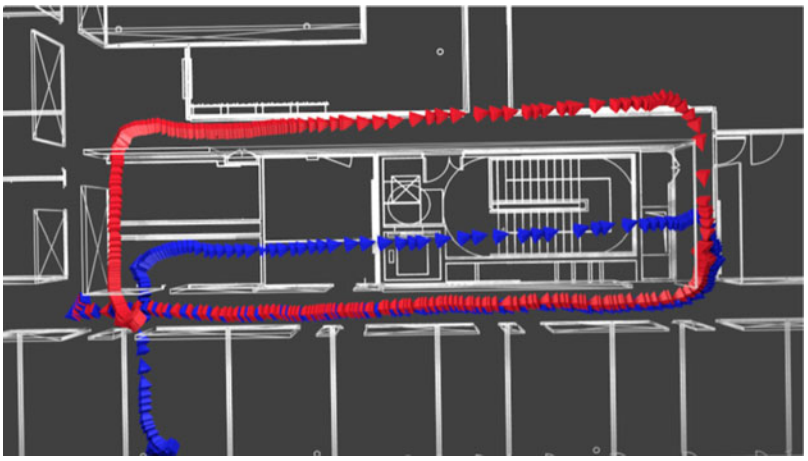

Fig. 8. Camera trajectory obtained with our algorithm projected on the model of the environment. The blue trajectory is the one obtained with the RGBD-SLAM with depth constraints. The red trajectory is the localization obtained by the RGBD-SLAM with model and depth constraints.

6. Conclusion

In this paper, we have proposed a real-time RGBD-SLAM for long-term and global-consistent localization even in case where loop closure is impossible. The proposed RGBD-SLAM is able to take into account prior knowledge of the 3D model of the environment in a local BA to constrain the localization.

The proposed cost function in BA allows to perform localization in a wide range of environments by automatically switching between three modes: classical monocular SLAM mode when depth information is unavailable, RGBD-SLAM mode using depth and visual data, or Constrained RGBD-SLAM mode when a 3D model is available. Doing so, this system is agnostic of the input being RGB or RGBD.

The proposed solution has been evaluated on common benchmarks and on our real sequences, which depict various environments. It has been compared to the state-of-the-art methods. The performances obtained demonstrate that the additional constraints significantly improve the accuracy and the robustness of the SLAM localization. The proposed solution based on BA with RGBD observations allows for accurate trajectory estimation with metric scale and limited-drift localization without loop closure. An additional advantage is to be less computationally expensive than the majority of solutions from the state of the art. It is lightweight and works with standard CPUs with an average frame processing time of 25 ms.

This research has been conducted with Kinect sensors but it can be extended to other sensors like a camera coupled to a Lidar. Our future work might include this extension with some interpolations to deal with sparse Lidar map. Our current implementation is sensitive to highly blured images, so the fusion with an IMU is also in our perspectives.

References

1. A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," *International Conference on Computer Vision* (IEEE, 2003).
2. E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser and P. Sayd, "Real Time Localization and 3d Reconstruction," *International Conference on Computer Vision and Pattern Recognition* (IEEE, 2006).
3. G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *International Symposium on Mixed and Augmented Reality* (IEEE, 2007).
4. R. A. Newcombe, S. J. Lovegrove and A. J. Davison, "Dtam: Dense Tracking and Mapping in Real-Time," *ICCV* (2011).
5. R. Mur-Artal, J. Monteil and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *Trans. Robot.* **31**(5), 1147–1163 (2015).
6. K. Melbouci, S. N. Collette, V. Gay-Bellile, O. Ait-Aider, M. Carrier and M. Dhome, "Bundle Adjustment Revisited for SLAM with RGBD Sensors," *International Conference on Machine Vision Applications* (IEEE, 2015).
7. K. Melbouci, S. N. Collette, V. Gay-Bellile, O. Ait-Aider and M. Dhome, "Model Based RGBD SLAM," *International Conference on Image Processing* (IEEE, 2016).
8. R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges and A. Fitzgibbon, "Kinectfusion: Real-Time Dense Surface Mapping and Tracking," *International Symposium on Mixed and Augmented Reality* (IEEE, 2011).
9. J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," *International Conference on Intelligent Robots and Systems* (IEEE, 2012).
10. P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *Int. J. Robot. Res.* **31**(5), 647–663 (2012).
11. A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," *International Symposium on Robotics Research* (IFRR, 2011).
12. T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *Int. J. Robot. Res.* **34**(4–5), 598–626 (2015).
13. M. Meilland and A. I. Comport, "On Unifying Key-Frame and Voxel-Based Dense Visual SLAM at Large Scales," *International Conference on Intelligent Robots and Systems* (IEEE, 2013).
14. C. Kerl, J. Sturm and D. Cremers, "Dense Visual SLAM for RGB-D Cameras," *International Conference on Intelligent Robots and Systems* (IEEE, 2013).
15. J. S. D. C. F. Endres, J. Hess and W. Burgard, "3-d mapping with an RGB-D camera," *IEEE Trans. Robot.* **30**(1), 177–187 (2014).
16. R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo with RGB-D cameras," *Trans. Robot.* **33**(5), 1255–1261 (2017).
17. S. A. Scherer and A. Zell, "Efficient Onboard RGBD-SLAM for Autonomous MAVs," *International Conference on Intelligent Robots and Systems* (2013).
18. R. Jamiruddin, A. O. Sari, J. Shabbir and T. Anwer, "RGB-depth SLAM review," CoRR, vol. abs/1805.07696 (2018).
19. D. Belter, M. Nowicki and P. Skrzypczyński, "Accurate Map-Based RGB-D SLAM for Mobile Robots," *Robot 2015: Second Iberian Robotics Conference* (L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano and V. Muñoz-Martinez, eds.) (Springer International Publishing, Cham, 2016) pp. 533–545.
20. S. A. Scherer, D. Dube and A. Zell, "Using Depth in Visual Simultaneous Localisation and Mapping," *International Conference on Robotics and Automation* (IEEE, 2012).
21. K. W. Lee, W. S. Wijesoma and J. I. Guzman, "A constrained SLAM approach to robust and accurate localisation of autonomous ground vehicles," *Robot. Auto. Syst.* **55**(7), 527–540 (2007).
22. M. F. Fallon, H. Johannsson and J. J. Leonard, "Efficient Scene Simulation for Robust Monte Carlo Localization Using an RGB-D Camera," *International Conference on Robotics and Automation* (IEEE, 2012).
23. Z. Fang and S. Scherer, "Real-time onboard 6dof localization of an indoor MAV in degraded visual environments using a RGB-D camera," *International Conference on Robotics and Automation* (2015).
24. W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello and W. Burgard, "Accurate Indoor Localization for RGB-D Smartphones and Tablets Given 2d Floor Plans," *International Conference on Intelligent Robots and Systems* (IEEE, 2015).
25. G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *International Symposium on Mixed and Augmented Reality* (2007).

26. H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, “Speeded-up robust features (surf),” *Comput. Vis. Image Underst.* **110**, 346–359 (2008).
27. E. Karami, S. Prasad and M. S. Shehata, “Image matching using sift, surf, BRIEF and ORB: Performance comparison for distorted images,” *Newfoundland Electrical and Computer Engineering Conference* (2015).
28. D. DeTone, T. Malisiewicz and A. Rabinovich, “Self-improving visual odometry,” 2018.
29. M. Tamaazousti, S. Naudet-Collette, V. Gay-Bellile, S. Bourgeois, B. Besbes and M. Dhome, “The constrained slam framework for non-instrumented augmented reality,” *Multimedia Tools Appl.* **75**, 9511–9547 (2016).
30. B. Triggs, P. F. McLauchlan, R. I. Hartley and A. W. Fitzgibbon, “Bundle Adjustment - A Modern Synthesis,” *International Conference on Computer Vision* (IEEE, 2000).
31. F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers and W. Burgard, “An Evaluation of the RGB-D SLAM System,” *International Conference on Robotics and Automation* (IEEE, 2012).
32. O. Wasenmüller, M. Meyer and D. Stricker, “Corbs: Comprehensive RGB-D Benchmark for SLAM Using Kinect v2,” *Winter Conference on Applications of Computer Vision* (IEEE, 2016).