On the Application of RGB-D SLAM Systems for Practical Localization of the Mobile Robots

Submitted: 2nd January 2017; accepted: 18th April 2017

Aleksander Kostusiak, Michał Nowicki, Piotr Skrzypczyński

DOI: 10.14313/JAMRIS_2-2017/17

Abstract:

This paper considers the practical application of the RGB-D Simultaneous Localization and Mapping (SLAM) techniques for localization of mobile robots. We attempt to answer the question: how the quality of the estimated sensor trajectory depends on the approach to RGB-D data processing in the SLAM system when RGB-D frames acquired on a real mobile robot are used. Experiments are performed on data obtained from robots of different classes, and from different environment types to present the problems characteristic to RGB-D data. Conclusions as to the robustness of particular architectures and solutions applied in SLAM are drawn on the basis of experimental results. Publicly available data sets and well-established performance metrics are used to ensure that the results are verifiable, reproducible and relevant.

Keywords: *SLAM, RGB-D, mobile robot, evaluation*

1. Introduction

Recent progress in understanding the structure of the Simultaneous Localization and Mapping problem, and the availability of efficient non-linear optimization algorithms in open-source software libraries enabled a paradigm shift in SLAM research. In the last couple of years the traditional SLAM algorithms, based on filtration have been replaced by methods based on optimization of some (often graph-like) representation of the SLAM problem [33]. The introduction of commodity RGB-D (RGB and Depth) sensors shifted the focus of SLAM research from passive vision to more reliable systems exploring direct depth measurements [23]. Also, the RGB-D sensors turned out to be practical in the localization of mobile robots, even those of limited size, payload, and computing power [24]. However, no standard algorithm or a dominant SLAM architecture has evolved, in both the visual and the RGB-D SLAM domains [31].

Another new trend in SLAM research are the publicly available RGB-D data sets. These benchmarks allow for comparison of new architectures to the solutions already known from the literature [4, 21, 22, 37]. This kind of evaluation, however, usually involves RGB-D data sequences acquired by handheld sensors (Kinect or Xtion) in relatively confined spaces [35] or simulated RGB-D images [14]. Thus, a comparative assessment of the SLAM algorithms and architectures on such benchmarks does not allow to select the solutions that are robust to such factors as motion blur in images, sudden motions of the sensor, lack of texture in the field of view, occlusions, shadows, and illumination changes. These factors are to various extent present whenever real mobile robots have to be localized in real-time.

There are few examples of publicly available RGB-D data sets obtained using mobile robots. In the TUM RGB-D Benchmark [35] some sequences were taken by a sensor mounted on the Pioneer wheeled robot. However, in those sequences often no objects are present within depth measurement range of the Kinect sensor, rendering them unsuitable to evaluate SLAM algorithms that rely on depth data. The large scale MIT Stata Center data set [12] contains RGB-D frames acquired using a Kinect sensor mounted on a mobile robot, but the ground truth trajectories are obtained by aligning 2D laser scans with a floor plan of the building, providing the accuracy of about 3 cm [12], which is below the accuracy achieved by the state-of-the-art SLAM systems [6, 23].

Whereas the literature is rich in papers evaluating feature detectors and descriptors, also with respect to various aspects of mobile robot localization [7, 16, 28], very few authors studied the influence of particular SLAM system architectures on the accuracy and reliability of robot trajectory estimation. Strasdat [34] compared several versions of his visual SLAM system, however, working mostly with the passive cameras. Mur-Artal *at al.* extensively evaluated their monocular ORB-SLAM [22], comparing it also to other architectures, and pointing out advantages of the feature-based approach. In the domain of RGB-D-based systems, we have presented a comparative study on pose-based localization approaches [3], however, using typical benchmark data sets. To the extent of our knowledge, no study has been yet published evaluating and comparing the recent approaches to SLAM in the specific context of real mobile robot localization.

Therefore, in this article, we attempt to compare the representative architectures of RGB-D SLAM systems in the context of indoor localization of real mobile robots, extending our recent conference paper [17]. We use two data sets that were recently published by the Mobile Robots Lab of Poznań University of Technology (PUT). One of these data sets was acquired using a wheeled robot in a typical laboratory room [19], whereas the other one was obtained from a sensor mounted on a six-legged robot traversing a simple terrain mockup [5]. These data sequences are supplemented by one sequence from the TUM RGB-D Benchmark, to show the difference between results on a standard benchmark, and the robot-specific data sets.

Most of the SLAM systems that are included in the comparative assessment exploit point features. We believe that the feature-based approach is more useful in practical mobile robot localization than the direct approaches, which estimate the transformation between images minimizing the difference between the actual and the predicted measurement in the intensity or depth domain [31]. The core of the comparison is constituted by four SLAM systems: the PUT SLAM [4], the ORB-SLAM2 [23], the CCNY_RGBD [9], and the RGB-D SLAM v2 [11]. The first of these systems, PUT SLAM, was developed at the Mobile Robots Lab at PUT specifically for localization of various mobile robots. The remaining three systems represent significantly different approaches to the problem of SLAM with RGB-D data. Besides these four SLAM systems, two other solutions to the localization problem with RGB-D data are evaluated. These are a RGB-D visual odometry system, which represents the simplest possible solution to the use of RGB-D data in the localization problem, and the KinFu Large Scale system, which is a featureless approach, based on the recently popular KinectFusion architecture. All used in this research systems are open-source. Also, used data sequences are publicly available.

We analyze the results using the well-established quantitative performance metrics introduced in [35], but we also show qualitative results in the form of dense, volumetric environment maps obtained from the RGB-D frames registered by PUT SLAM. The PUT SLAM system, developed in our lab, is of special interest in this study, and drawing conclusions from the evaluation experiments we also define the directions for further development of PUT SLAM, to ensure its effectiveness in the localization of mobile robots belonging to different classes.

The rest of this paper is structured as follows: section 2 briefly presents the evaluated localization systems, section 3 details used methodology to obtain both the quantitative and the qualitative results, section 4 describes and comments these results, and section 5 concludes this paper.

2. Systems and Their Architectures

2.1. RGB-D VO

The investigated architectures of localization systems are using various techniques to eliminate the trajectory drift, which occurs in the Visual Odometry (VO) [27]. In comparison to SLAM, the VO is a simpler approach to localization based on the same data. In order to investigate and visualize how important are the drift reduction techniques for achieving the accurate trajectory estimation, a simple RGB-D VO pipeline has been proposed, based on the procedures available in the OpenCV library [16]. The RGB-D VO system searches salient point features in the RGB image from the current frame and tries to match them with the features from the previous frame. The position of these points in the 3-D space is determined with the use of the corresponding depth image. Next, the transformation $[\mathbf{R}, \mathbf{t}]^T \in \mathrm{SE}(3)$ between the two sets of matched 3-D points is computed using a least squares estimation method [10]. The correctness of the estimated transformation is verified by applying the RANSAC approach. Only those pairs of features are accepted as inliers, for which the residual Euclidean error is smaller than a given threshold. The threshold is increased gradually if no transformation satisfying the criteria can be computed. To calculate the final translation and rotation estimation all accepted pairs of points (all inliers) are used. The block diagram of the RGB-D VO algorithm is presented in Fig. 1.

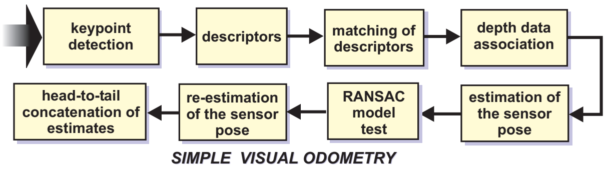

Fig. 1. Block scheme of the simple RGB-D VO system

2.2. PUT SLAM

The PUT SLAM system employs photometric as well as depth images. From the RGB images, it only extracts the keypoints with the use of the ORB detector [26], which was chosen due to the good trade-off between the performance in visual navigation and the computational efficiency [28]. The keypoints are extracted with respect to the scene depth data availability at the given point, avoiding artifacts in depth images [18]. In comparison with our earlier localization systems [3,18], an entirely new SLAM architecture has been introduced in [4].

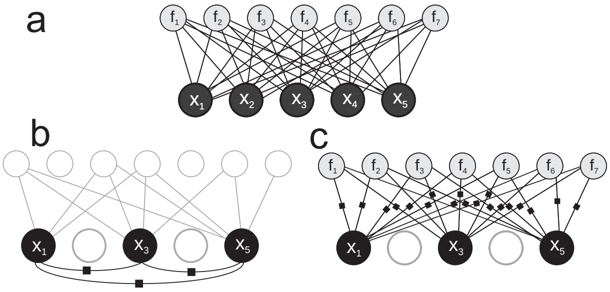

Fig. 2. Markov random field for the SfM problem (a). Factor graphs for the pose-based SLAM (b), and the Bundle Adjustment SLAM (c). Larger black circles are sensor poses, smaller white circles are features, while links denote measurements, and black rectangles stand for factors. Elements shown in gray are either marginalized or not used

The typical approach to graph-based SLAM, which dates back to 2D SLAM systems employing laser scanners [13], is based on the optimization of a graph of sensor poses and explicit detection of loop closures. Instead, PUT SLAM is based on a map of 3-D point features and borrows ideas from purely visual mapping systems. As shown in [33] the SLAM problem can be considered in terms of inference on a graph. The most general form, which is known in the computer vision community as the Structure from Motion (SfM) problem assumes that all the historic poses of the sensor are related to the observed features by measurements. This can be visualized as a random Markov field with the pose variables **x** and feature variables **f** (Fig. 2a). However, if SLAM is used for robot localization this graph grows when new poses are added as the robot moves, while more features appear whenever the robot explores new parts of the environment. To avoid the need to process a very large graph of constraints the pose-based SLAM systems marginalize all historic features (thus no explicit map), and keep only a graph of poses with the constraints between them (Fig. 2b). These constraints stem from the motion of the robot computed upon the locally co-observed features (shown in gray in Fig. 2b). The constraints may also represent loop closures, i.e. relations between locations that are distant in the chain of historic poses but are spatially close enough to co-observe some features. These loop closure constraints are essential is SLAM, as they enable the system to reduce the trajectory drift. The constraints are represented in the graph as factors, shown in Fig. 2b by small black rectangles.

The new algorithm implemented in PUT SLAM, presented in more detail in [4] and [6], uses also the non-linear estimation techniques from the g^{2}o library [20] to optimize a graph of constraints. It exploits, however, a much larger number of factors directly binding the positions of point features with the poses of the sensor (Fig. 2c). The graph of constraints has two kinds of vertices: **x** representing sensor poses, and **f** representing point features. The $\mathbf{t}_{ij} \in \mathbb{R}^3$ edge represents a constraint resulting from the RGB-D sensor measurement between the i-th pose and the j-th point feature. The uncertainty of each constraint is represented by its information matrix $\mathbf{\Omega}$, which can be determined by inverting the covariance matrix of a particular measurement [6]. The PUT SLAM closes loops locally by frequent matching of the incoming RGB-D frames to the map, without identifying explicitly the already seen places. This approach is similar to the Bundle Adjustment (BA) method used to efficiently solve the SfM problem [36] and applied recently to real-time visual SLAM [22] and RGB-D-based reconstruction [21]. However, an important difference between the typical BA algorithm and the approach taken in PUT SLAM is that in PUT SLAM the Euclidean errors in the positions of features are minimized, whereas in vision-only BA the re-projection error of features onto images is minimized.

A number of original concepts have been implemented in the PUT SLAM architecture. The most important of them is the use of a computationally efficient VO pipeline for the frame-to-frame tracking of the sensor pose. This subsystem is based on the Lucas-Kanade algorithm [1] and uses only the RGB images [18]. This solution allows to better estimate the sensor pose with respect to the map than a camera motion model applied in most of the visual SLAM systems [22]. A novel solution is also the use of several ORB descriptors [26] for each feature in the map. On the basis of the RGB images collected during the sensor motion, a number of descriptors are established for a single point feature, which represent this feature as seen from different views [4]. This solution improves the robustness of the feature-to-map matching process if a feature is re-observed from a significantly different angle than the original angle of observation.

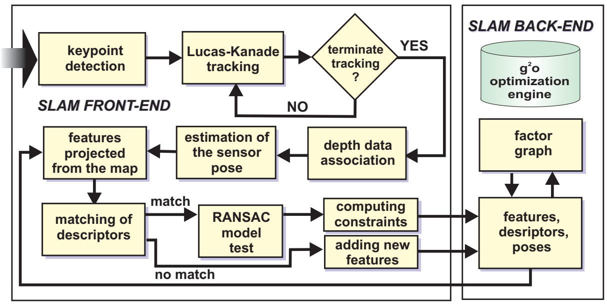

Fig. 3. Block scheme of the PUT SLAM system

The architecture of PUT SLAM system takes an advantage of multi-thread processing (Fig. 3), which ensures good efficiency for contemporary multi-core processors. The system is divided into the front-end implementing RGB-D data processing, sensor tracking (employing VO), and matching between the current frame and the map, and the back-end storing the map structure and implementing optimization of the graph. The synchronization of threads for data exchange occurs only at defined events, e.g. the end of an optimization cycle by g^{2}o. The PUT SLAM, thanks to such an architecture, works in real-time without the need of hardware acceleration. The PUT SLAM is open-source software available at GitHub[1].

2.3. ORB-SLAM2

The ORB-SLAM2 is the newest variant of the visual SLAM system presented in [22]. The ORB-SLAM emerged as a monocular SLAM that used the BA concept and point features optimization. The second version [23] can, however, use RGB-D data, thus it avoids a typical problem of initializing map from photometric features, for which the depth information is not available. The factor graph optimization is implemented in ORB-SLAM2 with the use of g^2o library, likewise in PUT SLAM. Also very similar is the overall architecture of both systems, which is divided into the front-end and the back-end. The most important differences between these two architectures result from the fact, that ORB-SLAM2 retains all properties of monocular SLAM, taking into account features lacking depth information and optimizing the map using only the re-projection error. This allows to substantially increase the number of features used in the matching process between the map and the current perception, e.g. including features lying beyond the depth measurement range of a RGB-D sensor. Moreover, in ORB-SLAM2 the loop closure detection is implemented using appearance-based place recognition, applying the bag-of-words technique [22]. This allows for closing loops of an arbitrary size, not only local ones, though this process for very large loops is not real-time. In the ORB-SLAM2 the modified, fully multi-scale ORB features are used in the whole system: to track the sensor motion, to match the current perception to the map (creating constraints), and in the loop closure procedures. However, tracking uses a simple sensor motion model, assuming constant speed motion, which makes ORB-SLAM2 (and the vision-only ORB-SLAM) vulnerable to sudden changes in the motion of the robot.

2.4. CCNY_RGBD

This algorithm, presented in [9] uses the photometric information only to detect keypoints without creating their descriptors (one can choose different detectors – in the presented evaluation the ORB detector was used). On the basis of the keypoints and information of scene depth, a global map of 3-D point features is created. Each feature has assigned uncertainty of its position expressed by a covariance matrix. The keypoints obtained from successive frames are matched to the features in the map with the use of the sparse Iterative Closest Points (ICP) algorithm. This allows determining the sensor motion. In order to improve the quality of feature matching, a method of estimating the depth image uncertainty from the Kinect sensor model was proposed in [9]. Matches between new feature points and the map are used to update the positions of features in the map with the use of a Kalman filter. The yet unmatched, new features are added to the map. In the CCNY_RGBD system an additional, off-line sensor pose graph optimization is available. This optimization may significantly improve the quality of the map, if accurate map building is considered as the task at hand [38]. However, as we are focused on robot localization, and we assume that all the investigated SLAM systems work in real-time, this possibility was not used, and results of the direct, real-time Kalman-filter-based approach are demonstrated.

2.5. RGB-D SLAM v2

The algorithm presented in [11] employs the color information as well as depth measurements. Similarly to PUT SLAM it uses point features and their descriptors but assumes a different concept of a map, which is a graph of poses. In the front-end, frame-by-frame visual odometry is used to estimate the sensor displacement between the successive RGB-D frames. To eliminate the mismatched feature pairs in VO the RANSAC approach is applied. To reduce the error accumulation (drift), the current frame is compared with previous frames to find possible loop closures. This allows for closing relatively large loops and setting constraints between distant frames, to suppress the drift. However, the search for loop closures is heuristic, still without explicit appearance-based place recognition. The preliminary trajectory obtained by VO (but with loop closures) is transformed into a pose graph with motion-related factors and optimized with the use of the g^2o library. In the RGB-D SLAM v2, it is possible to use several different detector/descriptor pairs. In the experiments the SURF detector/descriptor [2] was used, as the preliminary tests with the ORB features demonstrated that ORB is inefficient for loop closures. This result was also independently confirmed in [7].

2.6. KinFu Large Scale

This algorithm, available in the PCL library [25], is an open-source re-implementation of the Kintinuous [37] algorithm, which, in turn, is an improved version of the well-known KinectFusion concept. In contrast to the rest of the evaluated systems, this one uses only the depth information from RGB-D frames and does not use any features. It creates a dense, volumetric map of the local environment, which is a cube of 8 m^3 volume. This map is created using the Truncated Signed Distance Function (TSDF) and is stored in the graphic card memory (GPGPU accelerator is essential for this system to work). The sensor pose is determined with respect to the volumetric map with the use of a modified ICP algorithm. If the sensor moves close to the limits of the map area, a new cube around the current sensor pose is created, to enable mapping of larger space. Data, which do not fit in the new map are transformed into a triangular mesh, is used to reconstruct the volumetric representation whenever the sensor re-enters the already mapped area.

3. Concept and Methodology of Experiments

In order to assess the performance of the tested systems in the context of mobile robot localization three experiments have been performed. Two of them involved data collected by a real wheeled or crawling robot[2]. To show the difference in performance between the localization task of real robots, and a typical benchmark task, the `fr3_long_office_hausehold` sequence from the TUM RGB-D Benchmark [35] was used as well. This sequence, called experiment no. 1, is characterized by a considerable length of 2486 frames, however, the Kinect sensor moved by hand observes only a small area in a room, while its motion is slow and smooth (Fig. 4a). The experiment no. 2 refers to the sequence `putkk_1` (1537 frames) obtained by a wheeled robot with the Kinect sensor moving in the laboratory as a part of the data set described in [19]. A set of trajectories encompassing considerable part of the room and including closed loops have been registered – the robot returns to its start position (Fig. 4b). Data used in the experiment no. 3 have been registered on the six-legged walking robot Messor II with the Asus Xtion PRO Live sensor. The robot was traversing a simple terrain mockup with small bumps, following a trajectory of a roughly square shape (Fig. 4c). The registered trajectories differ in the used gait type and speed of the robot. A thorough analysis of the influence of the gait type on the trajectory estimation results for the PUT SLAM system has been presented in [5]. Thus, here we consider only the `messor2_2` sequence (1500 frames), which was obtained using the default tripod gait of the robot at the translational speed of 0.09 m/s.

Ground truth trajectories for the RGB-D data sets, collected by mobile robots were obtained from the multi-camera vision system PUT Ground Truth (PUT GT) [29]. This system uses five high-resolution came-

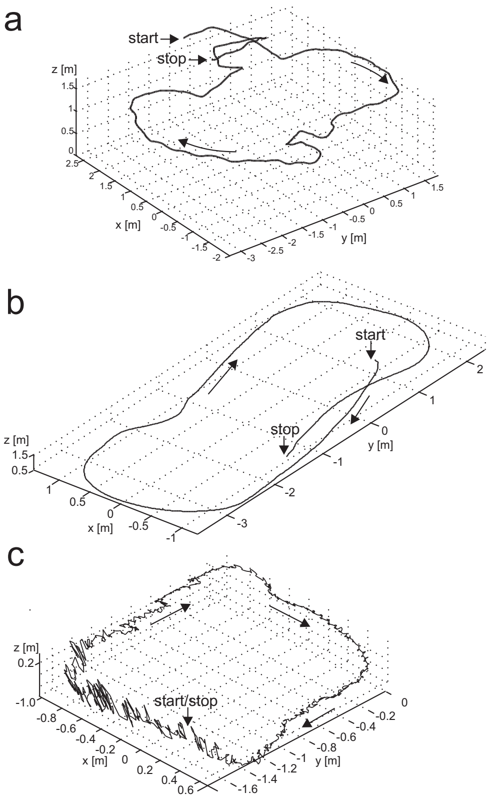

Fig. 4. Ground truth trajectories: for the handheld Kinect sensor (a), for the wheeled robot (b), and for the legged robot (c)

ras mounted to the ceiling of a lab room and requires a passive marker in the form of a chessboard to be mounted on the tracked object (robot). The transformations between the coordinate system of the cameras, the coordinate system of the marker, and the coordinate system of the sensor are obtained through appropriate calibration [30]. The calibration procedure that involves at least two cameras seeing the marker at the same time ensures high accuracy of the resulting transformation, and eventually, together with the large size of the multi-field chessboard marker, contributes to the accuracy of the ground truth trajectories. The PUT GT system ensures tight time synchronization between the images from the cameras, and the RGB-D frames collected from the on-board sensor of the robot [19].

To evaluate the accuracy of the recovered trajectories the methods of assessing the Absolute Trajectory Error (ATE), and Relative Pose Error (RPE) are applied. These methods, and the error metrics related to them, have been introduced in [35], and now are commonly used in the robotics community. The ATE value is the Euclidean distance between the corresponding points of the estimated, and the ground truth trajectory. Thus, the ATE metrics allows to determine how far away from the reference pose is the estimated pose on the trajectory. For the whole trajectory, the Root Mean Squared Error (RMSE) of the ATE metrics is calculated. The RPE value determines a relative translational or rotational error between the successive RGB-D frames on the estimated trajectory. Assuming that we have two trajectories: the ground truth $\mathbf{T}^{\mathrm{gt}} = \{\mathbf{T}^{\mathrm{gt}}_1, \mathbf{T}^{\mathrm{gt}}_2, \ldots, \mathbf{T}^{\mathrm{gt}}_n\}$, and the estimated one $\mathbf{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$, with the same number of n poses. and $\mathbf{T}_i$ and $\mathbf{T}^{\mathrm{gt}}_i$ are given as 4×4 homogeneous matrices, we can compute the ATE metrics for the i-th frame:

$$\mathbf{E}^{\mathrm{ATE}}_i = \left(\mathbf{T}^{\mathrm{gt}}_i\right)^{-1}\mathbf{T}_i, \tag{1}$$

and then obtain the ATE value for the whole trajectory from the RMSE of (1) for all nodes. Note that to obtain correct ATE, the trajectories have to be alligned prior to computing (1) by finding a transformation that minimizes the distance between the two rigid sets of points representing these trajectories [35]. Similarly, the RPE metrics for i-th frame is given by:

$$\mathbf{E}^{\mathrm{RPE}}_i = \left((\mathbf{T}^{\mathrm{gt}}_i)^{-1}\mathbf{T}^{\mathrm{gt}}_{i+1}\right)^{-1}\left(\mathbf{T}_i^{-1}\mathbf{T}_{i+1}\right). \tag{2}$$

Taking the translational or rotational part of $\mathbf{E}^{\mathrm{RPE}}_i$ we obtain the translational $\mathrm{RPE}_{t(i)}$ or the rotational $\mathrm{RPE}_{\theta(i)}$, respectively. The RMSE of the RPE_t or RPE_θ metrics for the whole trajectory are computed from the respective part of (2) for all nodes.

Although in this experimental study we focus on the accuracy of robot trajectory estimation, the accurate information about the pose is usually not enough for robot operation. Unfortunately, the sparse maps generated by feature-based SLAM cannot be used for motion planning or object recognition. Thus, separate dense mapping frameworks are used [38], typically relying for data registration on the accurate pose estimates from SLAM. Therefore, to demonstrate that the achieved localization accuracy is sufficient for dense environment mapping we produce textured triangle mesh 3-D maps from all the RGB-D sequences used for trajectory recovery.

The dense maps are computed using the FastFusion system described in [32]. FastFusion uses the octree data structure, which efficiently represents the volumetric data at different scales, as in the OctoMap algorithm [15]. However, in contrast to the OctoMap framework voxels are created only in a narrow band around the surface of the observed objects, which makes the map much more compact and allows it to grow dynamically as new RGB-D frames arrive. This approach runs in real-time on a standard CPU, without any hardware acceleration, which makes it well suited for mobile robots with limited computational power. Map visualization in FastFusion is based on a triangle mesh generated from the Signed Distance Function [8], which is continuously updated within the octree representation. The triangle mesh is computed in a separate thread to enable real-time rendering.

4. Experimental Results

Results of the first experiment provide a reference for the tests based on data obtained from mobile robots. In this experiment, all SLAM systems using point features (PUT SLAM, ORB-SLAM2, CCNY_RGBD, RGB-D SLAM v2) correctly recovered the sensor trajectory with ATE error within 10 cm (numerical results are presented in Tab. 1). SLAM architectures using optimization of a map of features, i.e. PUT SLAM (Fig. 5b) and

ORB_SLAM2 (Fig. 5c) achieved very high trajectory estimation accuracy with ATE error at the level of the uncertainty in the Kinect sensor measurement range.

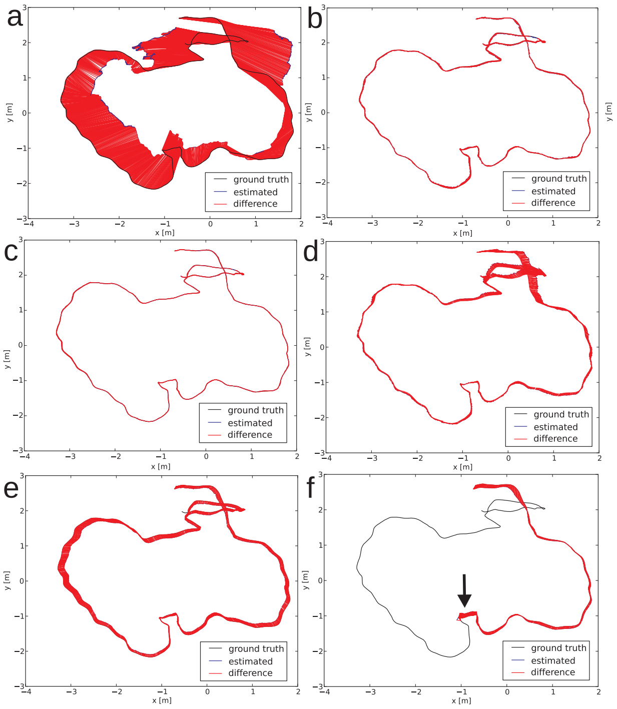

Fig. 5. Estimated trajectories for `fr3_long_office_household`***: RGB-D VO (a), PUT SLAM (b), ORB-SLAM2 (c), CCNY_RGBD (d), RGB-D SLAM v2 (e), KinFu LS (f)***

The slight advantage of the ORB-SLAM2 system results probably from more accurate feature localization through the modified, multi-scale ORB detector. Both feature-based systems, that do not use feature map optimization, the CCNY_RGBD relying on Kalman filtering (Fig. 5d), and the RGB-D SLAM v2, which uses pose graph optimization (Fig. 5e) recorded larger errors, what demonstrates the benefits of the BA-like approach that jointly optimizes the sensor poses and the feature positions. The KinFu Large Scale system (hereinafter called KinFu LS), despite the small ATE error at the beginning of the run, was not able to recover the entire trajectory, losing sensor tracking during the dynamic motion of the sensor (this point is marked by the arrow in Fig. 5f). The trajectory was entirely recovered by the simple RGB-D VO system (Fig. 5a), for which, however, the accumulating drift led to very large ATE error, while the translational RPE is still comparable to the results of the other systems.

The second experiment using data from a wheeled robot revealed the role of loop closing in the SLAM architecture. Visual odometry, despite the efficient matching of successive frames (small RPE$_t$, Tab. 1) produced large ATE error caused by the drift (Fig. 6a). A similar ATE value was observed for the CCNY_RGBD (Fig. 6d), what suggests that the pose correction for measurements of individual features without optimization of the whole map is much less effective than the optimization-based approach for a map encompassing a larger area, where feature re-observations take place after accumulating a significant trajectory drift. The RGB-D SLAM v2 was not able to recover the whole trajectory in this experiment. This system stopped to track features at the place, where the incoming RGB frames were characterized by a small number of features (the place is marked by an arrow in Fig. 6e). However, while processing next frames the RGB-D SLAM v2 detected a loop closure when the robot entered the area near the start position, and then resumed sensor tracking (Fig. 6e). The ATE value for the entire trajectory is, in this case, unreliable and was omitted in Tab. 1. The PUT SLAM system (Fig. 6b) achieved ATE of about 10 cm using the BA-like architecture, without explicit loop closures. However, in this case, ORB-SLAM2 (Fig. 6c) achieved a definitely better trajectory estimation accuracy. This result is caused by two factors – the use of explicit, appearance-based loop closure detection, which allowed this system to correct the trajectory in its final fragment, and the ability to exploit features without associated depth data, which enabled ORB-SLAM2 to use very distant features in the large room. The KinFu LS system (Fig. 6f) recovered only a part of the trajectory, breaking off tracking at the first turn.

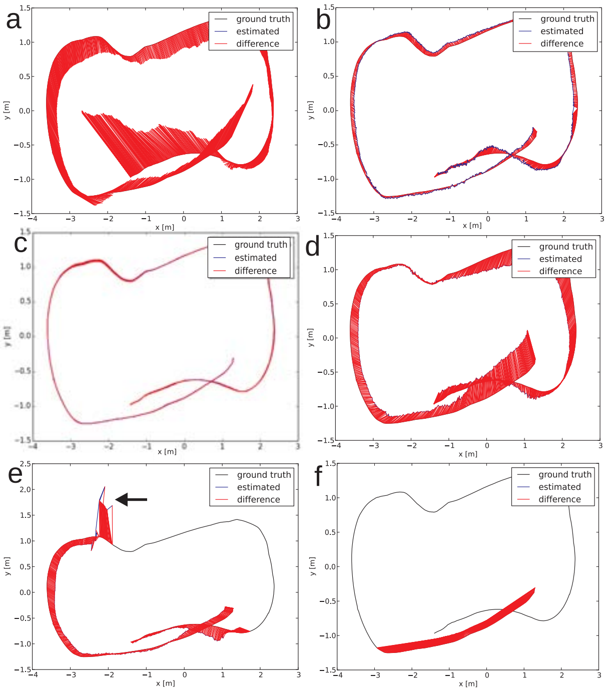

Fig. 6. Estimated trajectories for `putkk_1` ***sequence: RGB-D VO (a), PUT SLAM (b), ORB-SLAM2 (c), CCNY_RGBD (d), RGB-D SLAM v2 (e), KinFu LS (f)***

The third experiment allowed us to determine the robustness of the investigated systems to the problems related to sudden sensor motions, in this case, caused by the discrete nature of the legged robot motion and the unavoidable slippages. The blurred RGB images frequently caused inaccurate sensor motion estimation between successive frames. In the case of the simple RGB-D VO, this effect caused the reconstruction of an entirely wrong trajectory (Fig. 7a).

Tab. 1. Comparison of absolute trajectory error (ATE RMSE) and relative pose errors (RPE RMSE) for the evaluated systems

Localization system	exp. 1 RPE$_t$ [m]	exp. 1 RPE$_\theta$ [°]	exp. 1 ATE [m]	exp. 2 RPE$_t$ [m]	exp. 2 RPE$_\theta$ [°]	exp. 2 ATE [m]	exp.3 RPE$_t$ [m]	exp.3 RPE$_\theta$ [°]	exp.3 ATE [m]
RGB-D VO	0.025	10.95	1.098	0.011	0.45	0.612	0.514	22.99	1.384
PUT SLAM	0.009	0.59	0.023	0.016	0.33	0.104	0.041	5.66	0.069
ORB-SLAM2	0.004	0.25	0.009	0.004	0.12	0.018	–	–	–
CCNY_RGBD	0.028	1.07	0.106	0.033	1.95	0.529	0.118	6.39	0.307
RGB-D SLAM v2	0.030	9.8	0.095	–	–	–	0.116	10.54	0.210

Also, the ORB-SLAM2 was unable to recover the full trajectory, losing sensor tracking after the first sharp turn, when a sequence of successive blurred RGB images appears in the data (Fig. 7c). The ORB-SLAM2 system uses only the image re-projection error while establishing the constraints in the map (like monocular SLAM), ignoring the depth data. This causes increased sensitivity to such factors as motion blur and rolling shutter. The PUT SLAM system, always using full RGB-D data, recovered the entire robot trajectory (Fig. 7b). Also, the CCNY_RGBD (Fig. 7d) and RGB-D SLAM v2 (Fig. 7e) were successful, but in their case, ATE errors were significantly larger. The KinFu LS system does not use point features, so RGB image blur does not have any impact on its performance. However, also this time KinFu LS did not recover the entire trajectory, losing the ability to track sensor at the first turn (Fig. 7f).

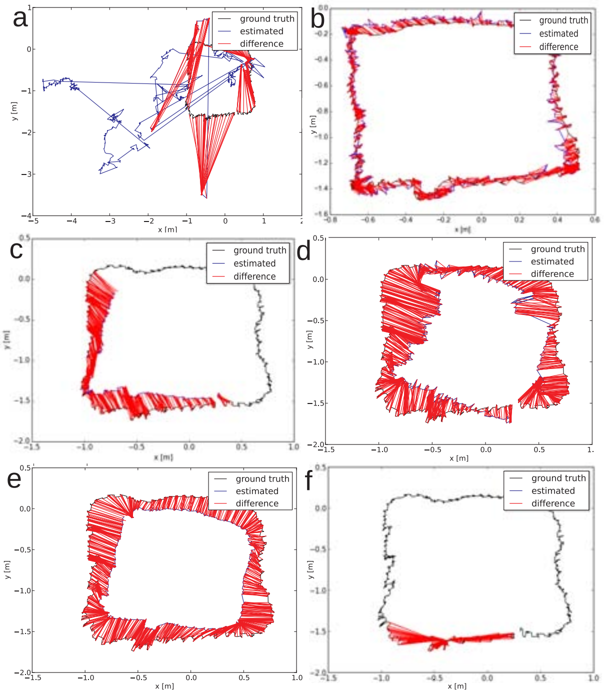

Fig. 7. Trajectories estimated for the* `messor2_2` *sequence: RGB-D VO (a), PUT SLAM (b), ORB-SLAM2 (c), CCNY_RGBD (d), RGB-D SLAM v2 (e), KinFu LS (f)

The ATE trajectories and the numerical results summarized in Tab. 1 are supplemented by the qualitative analysis of dense mapping results. We show the FastFusion maps produced from the RGB-D data registered by PUT SLAM in Fig. 8. Although no scene geometry ground truth (such one as provided in [14]) is available for the used sequences, the well-defined object shapes and sharp edges in these maps indicate the accuracy of the sensor trajectory estimation. This accuracy suggests that dense maps registered with PUT SLAM trajectories could be used to support various tasks of the mobile robot in all the conducted experiments[3].

5. Conclusions

The presented experiments[4] have shown that not all RGB-D SLAM architectures considered as representative for the state-at-the-art are dealing well with the data collected under real working conditions of a mobile robot. Most important problems were caused by sudden and unpredictable motions of the sensor, particularly in the walking robot experiment. Dynamic motion usually invalidates common assumptions as to the availability of point features precisely located on images (because of motion blur), and as to the mathematical motion model used to predict the sensor pose (as the one used in ORB-SLAM2). Problems specific to the second experiment in a large room were frames lacking point features because of the character of the environment (white walls) or due to the insufficient Xtion's depth measurement range. The SLAM architecture exploiting dense depth data – KinFu LS turned out to be extremely unreliable for robot localization. This approach was designed mostly for augmented reality applications and does not cope well with fast sensor/robot motion (especially sharp turns). Apparently, if the distances between the incoming depth images are too large, the ICP-based approach cannot match correctly the new frames to the volumetric map.

In comparison with other systems, PUT SLAM has shown its effectiveness in all experiments. The fast VO subsystem allowed for more accurate tracking of the sensor poses than the mathematical motion model in the cases when the motion was difficult to predict, e.g. for the walking robot (slippages, body vibrations). Also the relatively sudden view changes while turning (the second experiment) were not a challenge. However, in the simpler cases, the PUT SLAM gives way in the trajectory estimation accuracy to the ORB-SLAM2 system. Further development of PUT SLAM will, therefore, aim at implementing loop closures based on appearance-based place recognition, and at improving the management of point features, in order to limit the uncertainty of their location in the map [6]. In the further research, we will also consider the use of features

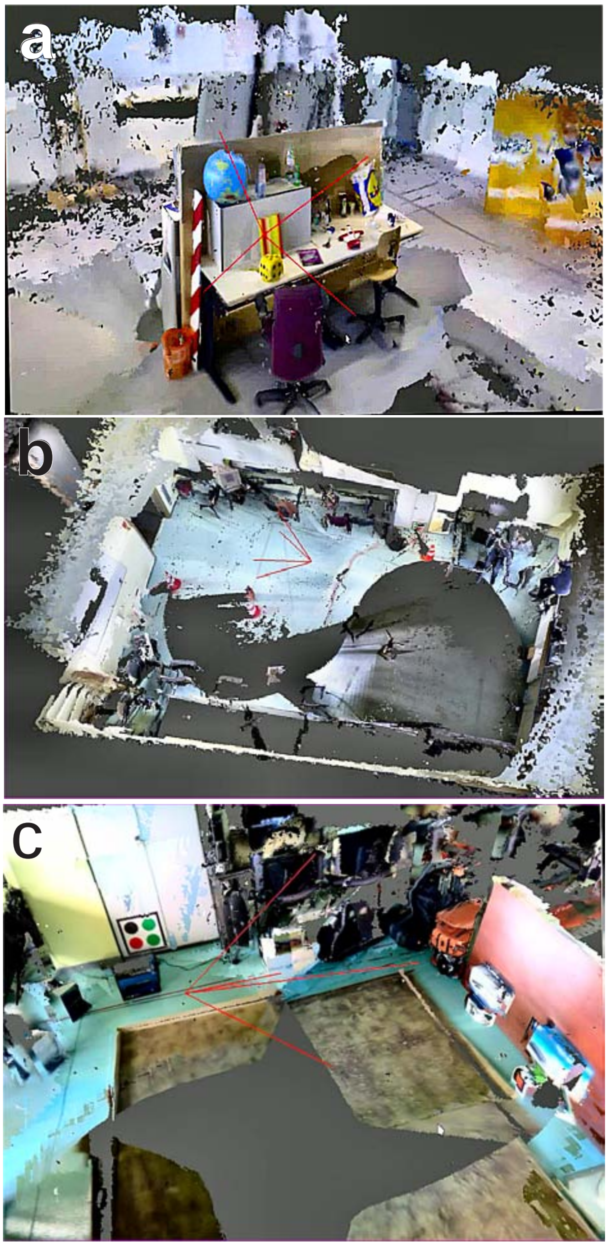

Fig. 8. FastFusion textured triangle mesh produced from RGB-D frames registered by PUT SLAM: `fr3_long_office_hausehold` *(a),* `putkk_1` *(b), and* `messor2_2` *(c)*

without depth information, which can be substantial in practical applications, due to the limited range of the commercially available RGB-D sensors.

Acknowledgements

This research was supported by the National Science Centre, Poland under grant 2013/09/B/ST7/01583. Work of A. Kostusiak was supported by the Poznań University of Technology Faculty of Electrical Engineering grant DSMK/0154.

Notes

[1] `https://github.com/LRMPUT/PUTSLAM/tree/release`

[2] Data sets are publicly available at `http://lrm.put.poznan.pl/putslam/`

[3] A movie is at `http://lrm.cie.put.poznan.pl/jamrismap.wmv`

[4] A movie is at http://lrm.cie.put.poznan.pl/jamrislam.wmv

AUTHORS

Aleksander Kostusiak* – Poznań University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: aleksander.m.kostusiak@doctorate.put.poznan.pl.

Michał Nowicki – Poznań University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: michal.nowicki@put.poznan.pl.

Piotr Skrzypczyński – Poznań University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: piotr.skrzypczynski@put.poznan.pl.

*Corresponding author

REFERENCES

[1] S. Baker, I. Matthews, ”Lucas-Kanade 20 years on: A unifying framework”, *Int. Journal of Computer Vision*, vol. 56, no. 3, 221–255, 2004. DOI: 10.1023/B:VISI.0000011205.11775.fd.

[2] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, ”Speeded-up robust features (SURF)”, *Computer Vision and Image Understanding*, vol. 110, no. 3, 2008, 346–359. DOI: 10.1016/j.cviu.2007.09.014.

[3] D. Belter, M. Nowicki, P. Skrzypczyński, ”On the performance of pose-based RGB-D visual navigation systems”, In: *Computer Vision – ACCV 2014* (D. Cremers *et al.*, eds.), LNCS 9004, Springer, 2015, 407–423. DOI: 10.1007/978-3-319-16808-1_28.

[4] D. Belter, M. Nowicki, P. Skrzypczyński, ”Accurate map-based RGB-D SLAM for mobile robots”, In: *Robot 2015: Advances in Robotics* (L. P. Reis *et al.*, eds.), AISC 418, Springer, 2015, 533–545. DOI: 10.1007/978-3-319-27149-1_41.

[5] D. Belter, M. Nowicki, P. Skrzypczyński, ”Evaluating map-based RGB-D SLAM on an autonomous walking robot”, In: *Challenges in Automation, Robotics and Measurement Techniques*, (R. Szewczyk, *et al.*, eds.), AISC 440, Springer, 2016, 469–481. DOI: 10.1007/978-3-319-29357-8_42.

[6] D. Belter, M. Nowicki, P. Skrzypczyński, ”Improving accuracy of feature-based RGB-D SLAM by modeling spatial uncertainty of point features”. In: *Proc. IEEE Int. Conf. on Robotics and Automation*, Stockholm, 2016, 1279–1284. DOI: 10.1109/ICRA.2016.7487259.

[7] P. Čížek, J. Faigl, ”On localization and mapping with RGB-D sensor and hexapod walking robot in rough terrains”. In: *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, Budapest, 2016, 2273–2278. DOI:: 10.1109/SMC.2016.7844577.

[8] B. Curless, M. Levoy, ”A volumetric method for building complex models from range images”,

Proc. 23rd Conf. on Computer Graphics and Interactive Techniques SIGGRAPH, New Orleans, 1996, 303–312. DOI:10.1145/237170.237269.

[9] I. Dryanovski, R. Valenti, J. Xiao, ”Fast visual odometry and mapping from RGB-D data”. In: *Proc. IEEE Int. Conf. on Robotics & Automation*, Karlsruhe, 2013, 5704–5711. DOI: 10.1109/I-CRA.2013.6630889.

[10] D. W. Eggert, A. Lorusso, R. B. Fisher, ”Estimating 3-D rigid body transformations: a comparison of four major algorithms”. *Machine Vision and Applications*, vol. 9, no. 5–6, 272–290, 1997. DOI: 10.1007/s001380050048.

[11] F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, ”3-D mapping with an RGB-D camera, *IEEE Trans. on Robotics*, vol. 30. no. 1, 2014, 177–187. DOI: 10.1109/TRO.2013.2279412.

[12] M. Fallon, H. Johannsson, M. Kaess, J. J. Leonard, ”The MIT Stata Center dataset”, Int. Journal of Robotics Research, vol. 32, no. 14, 2013, 1695–1699. DOI: 10.1177/0278364913509035.

[13] G. Grisetti, R. Kümmerle, C. Stachniss, W. Burgard, ”A tutorial on graph-based SLAM”, *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, 2010, 31–43. DOI: 10.1109/MITS.2010.939925.

[14] A. Handa, T. Whelan, J. D. McDonald, A. J. Davison, ”A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”, *IEEE Int. Conf. on Robotics & Automation*, Hong Kong, 2014, 1524–1531. DOI: 10.1109/ICRA.2014.6907054.

[15] A. Hornung, K. O. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots, vol. 34, no. 3, 2013, 189–206. DOI: 10.1007/s10514-012-9321-0.

[16] A. Kostusiak, ”The comparison of keypoint detectors and descriptors for registration of RGB-D data”, In: *Challenges in Automation, Robotics and Measurement Techniques*, (R. Szewczyk, *et al.*, eds.), AISC 440, Springer, 2016, 609–622. DOI: 10.1007/978-3-319-29357-8_53.

[17] A. Kostusiak, M. Nowicki, P. Skrzypczyński, ”On the use of RGB-D SLAM for mobile robots localization”, *Zeszyty Naukowe Politechniki Warszawskiej*, no. 195, *Postępy robotyki*, tom 2, 2016, 387–396 (in Polish).

[18] M. Kraft, M. Nowicki, R. Penne, A. Schmidt, P. Skrzypczyński, ”Efficient RGB-D data processing for feature-based self-localization of mobile robots”, *Int. Journal of Applied Mathematics and Computer Science*, vol. 26, no. 1, 2016, 63–79. DOI: 10.1515/amcs-2016-0005.

[19] M. Kraft, M. Nowicki, A. Schmidt, M. Fularz, P. Skrzypczyński, ”Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect”, *Machine Vision and Applications*, vol. 28, no. 1, 2017, 61–74. DOI: 10.1007/s00138-016-0802-6.

[20] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, ”g^{2}o: A general framework for graph optimization”, Proc. IEEE Int. Conf. on Robotics & Automation, Shanghai, 2011, 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[21] R. Maier, J. Sturm, D. Cremers, ”Submap-based bundle adjustment for 3D reconstruction from RGB-D data”, In: *Pattern Recognition: GCPR 2014*, (X. Jiang *et al.*, eds.), LNCS 8753, Springer, 2014, 54–65. DOI: 10.1007/978-3-319-11752-2_5.

[22] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, ”ORB-SLAM: A versatile and accurate monocular SLAM system”, *IEEE Trans. on Robotics*, vol. 31, no. 5, 2015, 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[23] R. Mur-Artal, J. D. Tardós, ”ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras, arXiv preprint, arXiv:1610.06475v1, 2016.

[24] M. Nowicki, P. Skrzypczyński, ”Experimental verification of a walking robot self-localization system with the Kinect sensor”, J*ournal of Automation, Mobile Robotics & Intelligent Systems*, vol. 7, no. 4, 2013, 42–51. DOI: 10.14313/JAMRIS_4-2013/43.

[25] Point Cloud Library, `http://pointclouds.org/`

[26] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ”ORB: an efficient alternative to SIFT or SURF”, IEEE Int. Conf. on Computer Vision, Barcelona, 2011, 2564–2571. DOI:10.1109/ICCV.2011.6126544.

[27] D. Scaramuzza, F. Fraundorfer, ”Visual odometry: Part I the first 30 years and fundamentals”. *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, 2011, 80–92. DOI: 10.1109/MRA.2011.943233.

[28] A. Schmidt, M. Kraft, M. Fularz, Z. Domagala, ”‘Comparative assessment of point feature detectors and descriptors in the context of robot navigation”, *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 7, no. 1, 2013, 11–20.

[29] A. Schmidt, M. Kraft, M. Fularz, Z. Domagala, ”The registration system for the evaluation of indoor visual SLAM and odometry algorithms”, *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 7, no. 2, 2013, 46–51.

[30] A. Schmidt, A. Kasiński, M. Kraft, M. Fularz, Z. Domagala, ”Calibration of the multi-camera registration system for visual navigation benchmarking”, *Int. Journal of Advanced Robotic Systems*, vol. 11, no. 83, 2014. DOI: 10.5772/58471.

[31] P. Skrzypczyński, ”Mobile robot localization: where we are and what are the challenges?”, In: *Automation 2017. Innovation in Automation, Robotics and Measurement Techniques*, (R. Szew-

czyk, *et al.*, eds.), AISC 550, Springer, 2017. DOI: 10.1007/978-3-319-54042-9.

[32] F. Steinbrücker, J. Sturm, D. Cremers, "'Volumetric 3D mapping in real-time on a CPU". In: *Proc. IEEE Int. Conf. on Robotics & Automation*, Hong Kong, 2014, 2021–2028. DOI: 10.1109/ICRA.2014.6907127.

[33] H. Strasdat, J. M. M. Montiel, A. J. Davison, "Visual SLAM: Why filter?", *Image and Vision Computing*, vol. 30, no. 2, 2012, 65–77. DOI: 10.1016/j.imavis.2012.02.009.

[34] H. Strasdat, *Local accuracy and global consistency for efficient visual SLAM*, PhD Dissertation, Imperial College, London, 2012.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Vilamoura, 2012, 573–580. DOI: 10.1109/IROS.2012.6385773.

[36] B. Triggs, P. F. McLauchlan, R. I., Hartley, A. W. Fitzgibbon, "'Bundle adjustment – a modern synthesis", In: *Vision Algorithms: Theory and Practice*, LNCS 1883, Springer, 2000, 298–372. DOI: 10.1007/3-540-44480-7_21.

[37] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, J. B. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion", *Int. Journal of Robotics Research*, vol. 34, no. 4–5, 2015, 598–626. DOI: 10.1177/0278364914551008.

[38] A. Wilkowski, T. Kornuta, M. Stefańczyk, W. Kasprzak, Efficient generation of 3D surfel maps using RGB-D sensors, Int. Journal of Applied Mathematics and Computer Science, vol. 26, no. 1, 2016, 99–122. DOI: 10.1515/amcs-2016-0007.