Accurate Map-Based RGB-D SLAM for Mobile Robots

Dominik Belter, Michał Nowicki and Piotr Skrzypczyński

Abstract In this paper we present and evaluate a map-based RGB-D SLAM (Simultaneous Localization and Mapping) system employing a novel idea of combining efficient visual odometry and a persistent map of 3D point features used to jointly optimize the sensor (robot) poses and the feature positions. The optimization problem is represented as a factor graph. The SLAM system consists of a front-end that tracks the sensor frame-by-frame, extracts point features, and associates them with the map, and a back-end that manages and optimizes the map. We propose a robust approach to data association, which combines efficient selection of candidate features from the map, matching of visual descriptors guided by the sensor pose prediction from visual odometry, and verification of the associations in both the image plane and 3D space. The improved accuracy and robustness is demonstrated on publicly available data sets.

Keywords SLAM · Point features · Tracking · Factor graph · RGB-D data

1 Introduction

Some types of robots, like quadrotors [19] and legged machines [2, 17] require accurate pose estimates in 3D for reliable navigation. Recently, solutions to the SLAM problem that employ the compact RGB-D sensors, such as Kinect or Xtion, became popular on those robotic platforms. Approaches that use dense depth data, like Kintinuous [25], demonstrate high accuracy, but they require hardware acceleration on high-end GPGPU cards, which cannot be packed on-board in small mobile robots. A viable alternative is the pose-based approach to SLAM with RGB-D data, which computes the sensor motion between consecutive frames in order to estimate the

D. Belter · M. Nowicki · P. Skrzypczyński(✉)
Institute of Control and Information Engineering, Poznań University of Technology,
ul. Piotrowo 3A, 60-965 Poznań, Poland
e-mail: {dominik.belter,michal.nowicki,piotr.skrzypczynski}@put.poznan.pl

L.P. Reis et al. (eds.), *Robot 2015: Second Iberian Robotics Conference*,
Advances in Intelligent Systems and Computing 418,
DOI: 10.1007/978-3-319-27149-1_41

trajectory. The obtained sensor poses and the motion-related constraints are treated respectively as the vertices and edges of a factor graph, which is then optimized [14]. Loop closures detected whenever the robot comes back to an already visited location introduce pose-to-pose constraints that enable the graph optimization in SLAM to correct the trajectory drift. The sensor motion estimate between two frames may be obtained in several ways, e.g. applying dense optical flow [12] or sparse optical flow [16], but the most common approach is to match sparse features by using local visual descriptors [6, 11]. As we have demonstrated in [3] the pose-based approach to RGB-D SLAM results in accurate real-time trajectory estimation without hardware acceleration. However, this approach neglects a large part of the feature-to-pose constraints resulting from frequent re-observations of the features. Because the loop closure detection has linear complexity in the number of locations, in the pose-based SLAM the data associations are usually established only between a relatively small fraction of the previous keyframes memorized along the trajectory.

In contrast, in the bundle adjustment (BA) approach [23], widely employed by modern visual SLAM [13, 21], the feature-to-pose constraints are directly used in optimization. Henry *et al.* [11] already applied the two-view BA to improve the frame-to-frame motion estimation in RGB-D visual odometry (VO), whereas Scherer and Zell [19] presented a solution to the RGB-D SLAM based on the structure of Parallel Tracking and Mapping (PTAM) [13]. The use of local BA defined on submaps for 3D object reconstruction from RGB-D data was presented in [15]. Very recently, the SlamDunk system has been presented [7], which uses a pool of point features (map) to track the sensor motion and then includes these features in optimization.

In this paper we investigate how to efficiently implement a RGB-D SLAM based on factor graph optimization including 3D features. The crucial components of such a solution are fast and accurate sensor tracking, map maintenance, and robust data association between the map and the perceived features. This work contributes a novel architecture of the map-based RGB-D SLAM system, which differs from other similar approaches [7, 15, 19] by employing the VO to track the sensor pose, by defining a flexible structure of the factor graph with a buffered part that enables real-time optimization in the background, and by exploring new techniques for outlier rejection in data association. The experimental evaluation focuses on the influence of the VO pipeline implementation on the accuracy of the trajectory estimation, and on the behavior of features in the map. Moreover, a comparison to selected state-of-the-art approaches to the RGB-D SLAM problem is included.

2 System Architecture

2.1 RGB-D Data Processing

The implementation of our RGB-D SLAM system[1] is divided into the front-end, which implements the VO pipeline and data association procedures, and the

[1] Source code is available at https://github.com/LRMPUT/PUTSLAM/tree/release

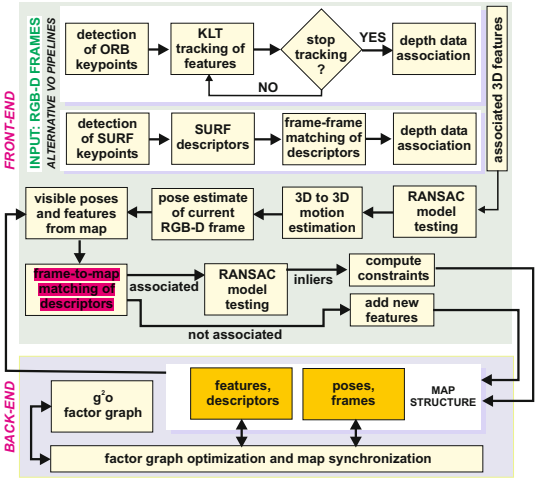

Fig. 1 Block scheme of the feature-based RGB-D SLAM system

back-end implementing the map structure and factor graph optimization. This architecture is presented in Fig. 1.

The main tasks of the front-end are to estimate the local sensor motion and to associate the observed point features to the existing map. In the front-end the sensor displacement guess is obtained from the VO pipeline, which considers 3D-to-3D feature correspondences for frame-to-frame motion estimation, like in stereo vision [17]. The 3D features are obtained by augmenting the 2D features (keypoints) extracted from the RGB images by the readily available depth data. The correspondences between the 2D keypoints can be accomplished either by matching of visual descriptors, or by optical flow tracking. The matching-based approach with local descriptors is popular in SLAM research [8]. The alternative approach we demonstrate here detects the keypoints only in the first RGB image of a sequence of n frames (n is small), and then tracks these features through the n images using the Lucas-Kanade algorithm [20]. This variant provides a very fast VO pipeline, which is also quite accurate [16].

Once the correspondences between the two RGB-D frames are established, the **SE(3)** rigid transformation between these two frames is estimated using the Umeyama algorithm [24]. The estimation procedure is embedded in the RANSAC scheme to make the computed transformation robust to outliers resulting from imperfect tracking or wrong descriptor associations. The RANSAC returns a transformation consistent with the largest number of feature pairs, which are considered inliers. The final transformation is computed using the g^2o optimization library to perform the two-view BA on the set of inlier features.

The resulting sensor motion estimate is added head-to-tail to the last known sensor pose in the factor graph in order to compute the pose estimate of the last RGB-D frame. Using this initial pose guess from the VO, features selected from the map become candidates for matching with the current set of local features. The current

set of features is established by the points that were tracked over the last n frames, or have been detected in the current RGB-D data frame, if the matching-based VO is used. In the implementation demonstrated in this paper we employ ORB features [18] in the tracking-based pipeline, and SURF [1] in the matching-based version. For the sake of computation efficiency the descriptors of the features employed in the VO are then re-used to obtain associations between the local features and the map. The features used by the Lucas-Kanade tracker have to be corner-like, thus ORB is selected. The matching-based version employs SURF, which has a blob-like detector, but provides a more discriminative descriptor.

Regardless of the feature type being used it is necessary to place the keypoints in all parts of the image frame in order to achieve good estimation of the sensor motion. Thus, the RGB images are divided into 80×80 pixels square subimages, and the feature detection is performed individually in each subimage with adaptation of the detector parameters ensuring that roughly the same number of features in each square is obtained. After feature detection the DBScan unsupervised clustering algorithm is used to detect groups of keypoints in the image, and then each group is represented only by a single, strongest keypoint [16].

Having the sensor displacement guess, the front-end attempts to associate the local features and the features already stored in the map by using the visual descriptors (ORB or SURF). Having these associations the front-end computes the **SE(3)** transformation between the current pose and the map (in the global frame), by applying the Umeyama algorithm to the set of corresponding feature pairs. Again, the inliers are determined by RANSAC. For the set of inliers $\mathbb{R}^3$ constraints (translations) are added between the pose of the current frame, and the re-observed features in the map.

2.2 Map Optimization

The back-end holds the map and creates the factor graph, which is then optimized using the g^{2}o library [14]. The map contains a set of 3D point features augmented by visual descriptors, and the sensor poses related to the keyframes. The features are connected to the poses by constraints in $\mathbb{R}^3$. The poses are expressed in the global reference frame, whereas the features are anchored in the coordinate systems of the poses, from which they have been observed for the first time. The sensor poses are represented by their Cartesian positions (x,y,z) and quaternions (q_w,q_x,q_y,q_z) for the sensor orientation. The features $\mathbf{p}_j^f$ $(j = 1 \ldots m)$ are represented as 3D points. The scalability of the system that incorporates a high number of features in the factor graph is limited, because the computational complexity of graph optimization is cubic in the number of variables. Therefore, our architecture adopts the Double Window optimization framework [21] by defining an inner window of keyframes around the current frame in the factor graph, and an outer window of peripheral keyframes. The parts of the feature map outside of the inner window are marginalized out, which renders the factor graph optimization efficient.

However, the graph optimization procedure is still much slower than the front-end. Thus, the front-end and back-end are implemented in separate threads, similarly to the PTAM architecture [13]. The front-end and the back-end work asynchronously, and they get synchronized on specified events, such as a query for visible features and insertion of a new set of constraints. New poses, features, and measurements are buffered in a smaller temporary graph, which is not used by g^{2}o until the back-end finishes the on-going graph optimization session. At this moment the temporary structure is merged with the main graph.

Unlike typical visual SLAM systems, our algorithm does not track the camera directly against the map, but employs an efficient VO pipeline to predict the sensor location. This allows our SLAM system to handle situations when there is a very small overlapping between the current view and the map, and we cannot re-observe the mapped features. In such cases the direct pose-to-pose constraints are introduced to the factor graph to stabilize the optimization in g^{2}o.

The optimal sensor poses $\mathbf{p}_1^c, \ldots, \mathbf{p}_n^c$ and feature positions $\mathbf{p}_1^f, \ldots \mathbf{p}_m^f$ are found by minimizing the function:

$$\underset{\mathbf{p}}{\operatorname{argmin}} F = \sum_{i=1}^{n} \sum_{j=1}^{m} e(\mathbf{p}_i^c, \mathbf{p}_j^f, \mathbf{m}_{ij})^T \Omega_{ij} e(\mathbf{p}_i^c, \mathbf{p}_j^f, \mathbf{m}_{ij}), \tag{1}$$

where $e(\mathbf{p}_i^c, \mathbf{p}_j^f, \mathbf{m}_{ij})$ is the error function between the estimated and the measured pose of the vertex related to the measurement $\mathbf{m}_{ij}$. In our formulation $\mathbf{m}_{ij}$ is $\mathbf{t}_{ij} \in \mathbb{R}^3$ for the feature-to-pose constraints, or $\mathbf{T}_{ij} \in \mathbf{SE(3)}$ for the pose-to-pose constraints. The information matrix $\mathbf{\Omega}$ represents the accuracy of each measurement. For the feature-to-pose constraints this matrix is obtained by inverting the covariance matrix of the feature uncertainty. In this work we assume isotropic spatial uncertainty of the features, so we set $\mathbf{\Omega}$ to an identity matrix, but as demonstrated by our recent research [4] computing $\mathbf{\Omega}$ as inversion of a realistic feature covariance matrix improves the accuracy of trajectory estimation. The $\mathbf{\Omega}$ of a pose-to-pose constraint is also set to identity, but it can be estimated from the uncertainty of the measurements that are marginalized when the transformation between two poses is computed [9].

3 Data Association in the Map

Our system handles local, metric loop closures implicitly, by establishing feature-to-pose constraints in the map. The map contains features that were discovered at various time instances along the sensor trajectory. Only features anchored to the sensor poses that are in the Euclidean neighborhood of the current sensor pose are considered for matching. In the current implementation the system does not account for global loop closures that require purely appearance-based place recognition [5]. However, integration of such a method is considered as future work.

The front-end queries the back-end (which holds the map data structure) for the set of candidate features that could be re-observed from the current pose of the sensor.

The set of features resulting from this query is projected into the current frame. If the projection results in an acceptable position of the feature on the image, this feature is considered for matching. According to the guided matching principle the candidate matches are considered only in a small neighborhood of the predicted feature in the image plane. As a feature from the map is often observed from various viewpoints, we store for a single point feature in the map multiple ORB/SURF descriptors obtained at various observation angles. Then, when matching the features we choose the descriptor with the smallest difference between its stored observation angle and the observation angle from the current frame. The observation angles are represented in the global coordinate system, and the angle between them is computed quickly using the dot product. We reject matches altogether whenever the difference in the observation angle between the most similar viewpoint of the mapped feature and the current sensor viewpoint is larger than 30°.

When we attempt to associate the local features with the map it is important to limit the number of map features that are considered for matching. Thus, we eliminate features that are occluded when observed from the current pose. This is accomplished using the dense depth measurements readily available in the current frame. We compare the distance between each feature projected from the map to the current coordinate frame and the image plane (i.e. "predicted depth" of the feature) with the average depth measured around the projected image coordinates of the map feature. If the measured depth is shorter by more than 0.1 m than the predicted depth, we consider the feature as occluded (Fig. 2a). This approach is more precise than the one based on considering only features visible from the local neighborhood [21] (Fig. 2b).

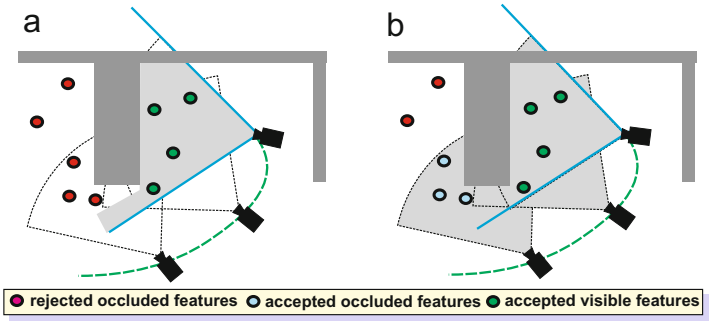

Fig. 2 Discrimination of the occluded features: features within the depth sensor field of view (a), and all features visible from the neighboring keyframes (b)

From the set of visual correspondences determined by descriptor matching, the correct matches (inliers) are estimated using the preemptive RANSAC framework. The outlier rejection test in RANSAC can use either the feature re-projection error in the image plane, or the Euclidean distance between the observed and the predicted feature in 3D. The re-projection error is given as:

$$E_{\text{Rep.}} = \max\left(d_{2D}(\Pi_a(\mathbf{p}_i^f), \Pi_a(\mathbf{p}_j^f)), d_{2D}(\Pi_b(\mathbf{p}_i^f), \Pi_b(\mathbf{p}_j^f))\right), \quad (2)$$

where Π_a represents the operator of projection onto the image plane a, $d_{2D}(x, y)$ represents computation of image distance between two image points x and y, and $\mathbf{p}_i^f$ is the 3D position of the i-th feature. The alternative Euclidean norm between the previously mentioned features is computed as:

$$E_{\text{Euc.}} = d_{3\text{D}}(\mathbf{p}_i^f, \mathbf{p}_j^f), \tag{3}$$

where $d_{3\text{D}}(x, y)$ is the computation of the Euclidean distance between two 3D points x and y. We have implemented both outlier rejection approaches in order to test which one is more efficient in practice.

4 Map Maintenance

If the point features discovered in the current keyframe cannot be matched to the already mapped features, they are added to the map. When adding features we try to avoid the possible aliasing of nearby features, which may further result in false matches. Thus, we add features to the map only from the keyframes, which contribute enough to the environment exploration. A keyframe is selected from the incoming RGB-D data stream if the number of features re-projecting from the map into the current image is below a given threshold (usually set to 80) or the map matching procedure resulted in less feature-to-pose constraints than a preset threshold (usually set to 25). From each keyframe, no more than 40 point features can be added to the map, and a preference is given to the keypoints having strongest detector response. Features that are located too close or too far from the sensor are not added to the map, as their locations may be highly uncertain due to the range measurement errors.

When a new feature is added to the map, we also verify that this new point is distant enough to the features already existing in the map. There is a minimal distance threshold in the Euclidean space, and a threshold on the minimal distance in the image plane. However, these parameters often need to be adjusted for scenes or sensor motions of different characteristics in order to achieve best accuracy.

5 Evaluation and Results

5.1 Investigation of the Map Evolution

We performed a series of experiments on the ICL-NUIM data set [10] in order to understand how the accuracy of estimated trajectories is influenced by such factors as VO-based motion estimation, the choice of error metrics in RANSAC, and the distribution of feature measurements in the map. The choice of ICL-NUIM, which is rendered in a synthetic environment is motivated by the perfect ground-truth sensor trajectories provided in this data set, and the availability of two variants of the depth data: one assuming no noise, and another one with simulated noise. We investigated

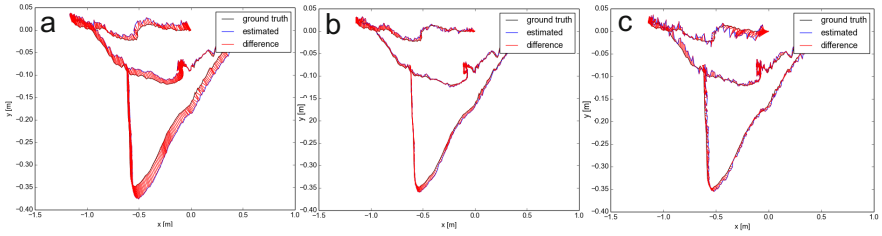

Fig. 3 ICL-NUIM data set `office_room/kt2` trajectory recovered from the noiseless data with two versions of our RGB-D SLAM: the tracking-based version (a), and the matching-based version (b), and the same trajectory recovered from noisy data by using the matching-based SLAM (c)

the evolution of features in the map, and evaluated quantitative results using the well-established ATE (Absolute Trajectory Error) and RPE (Relative Pose Error) metrics [22].

Figure 3 shows trajectories recovered from the exemplary `office_room/kt2` sequence. The trajectory estimated by the tracking-based version (Fig. 3a) is slightly less accurate than the one estimated by the matching-based variant (Fig. 3b). These example trajectories have been obtained from the data assuming no noise, however, the robust feature matching and outlier rejection mechanisms allow our SLAM to obtain similar results on the noisy version of this sequence (Fig. 3c).

Table 1 Performance for the representative configurations of our SLAM system measured on the ICL-NUIM `office_room/kt2` sequence without and with depth noise

RANSAC variant and noise in ICL-NUIM data	Tracking-VO SLAM					Matching-VO SLAM				
	ATE [m]	ATE std dev. [m]	RPE [m]	No. of features	FPS [Hz]	ATE [m]	ATE std dev. [m]	RPE [m]	No. of features	FPS [Hz]
$E_{\text{Euc.}}$ no noise	0.028	0.010	0.006	1566	28.8	0.010	0.006	0.007	2479	2.9
$E_{\text{Rep.}}$ no noise	0.022	0.011	0.007	2279	25.9	0.009	0.005	0.007	3328	2.8
$E_{\text{Euc.}}$ with noise	0.050	0.024	0.016	2027	29.6	0.021	0.009	0.013	2480	2.4
$E_{\text{Rep.}}$ with noise	0.045	0.021	0.015	2032	28.6	0.019	0.009	0.011	2541	2.2

Table 1 summarizes the quantitative results obtained on the `office_room/kt2` sequence by the tracking-based and matching-based versions of our SLAM demonstrating also how the outlier rejection metrics in RANSAC influences the accuracy of the obtained trajectory. All frame rates (FPS) were measured on a PC with Intel Core i7-2600 3.4GHz CPU and 16GB RAM. Figure 4a provides a visualization of the feature-based map for the same sequence[2]. The matching-based version achieves better accuracy (smaller ATE), but at the cost of being much slower. The influence of the error metric in RANSAC is small, but the re-projection error metric constantly provides slightly better accuracy. However, the more complicated computations in eq. (2) slow down the front-end.

[2] A short video clip is available at http://lrm.cie.put.poznan.pl/robot15.mp4

Comparing the ATE RMSE results in Tab. 1 to the figures given in [10] for five SLAM or VO systems, we can see that our matching-based SLAM is more accurate than all of the feature-based approaches evaluated on the `office_room/kt2` sequence in [10]. Only the dense ICP-based Kintinuous achieved smaller ATE on this sequence, but it requires hardware acceleration to run. Even the tracking-based version, which runs at the high frame rate allows us to localize the sensor with a positional error of about 4.5 cm (with noise).

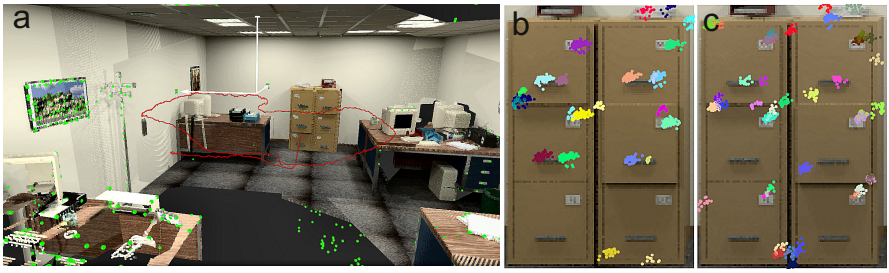

Fig. 4 Exemplary view of the map features (green points) and sensor trajectory (red) recovered from the ICL-NUIM `office_room/kt2` sequence (a), and distribution of the keypoints extracted by the front-end in the tracking-based (b), and matching-based (c) version. Points belonging to different features are shown in different colors

In order to investigate in more detail the behaviour of the features in the map, we conducted an experiment, which shows the distribution of all measured features in the common coordinate system. We run our SLAM algorithm on the same `office_room/kt2` sequence. Whenever the algorithm computes a transformation between two frames we substitute the computed value by the perfectly known transformation from the ground truth. Thus, the error of feature positions is caused only by the imperfect feature detection, and tracking or matching in the front-end. Then, we transform all measured positions (u, v) of features into the first RGB frame using the ground truth data. A close-up of a selected, feature-rich region of this RGB image is presented in Fig. 4b and 4c. The tracking-based front-end yields more measurements per feature, which is advantageous for the factor graph optimization. On the other hand, when the front-end tracks a single feature for a long sequence of frames, the measurements start to drift. The features form an oval shape on the image (Fig. 4b). The features produced by the matching-based front-end are more scattered on the image (Fig. 4c), and in the Euclidean space. The matching-based version re-establishes the features at each frame, and thus is more discriminative in accepting the feature-to-map associations on the basis of their descriptors. Thus, whenever a feature starts to drift, the matching-based front-end creates a new feature. This approach gives better precision in SLAM (smaller drift of features), but finally, the features are close to each other, and the map expands quickly. The larger number of features, together with the relatively slow SURF detector/descriptor, contribute to the low frame rate (cf. Tab. 1) achieved by the matching-based version.

5.2 Accuracy Assessment on Benchmark Data

We evaluated the accuracy of our map-based RGB-D SLAM on the TUM RGB-D benchmark [22], which has been already used to demonstrate performance of many RGB-D SLAM and VO systems [3, 6, 25]. We have tested our SLAM on five sequences, that are representative for indoor navigation: `fr1_desk` (571 frames), `fr1_desk2` (611 frames), `fr1_room` (1352 frames), `fr2_desk` (2217 frames), and `fr3_long_office_household` (2486 frames).

Table 2 Positional ATE and RPE for the two main configurations of our SLAM system measured on five TUM RGB-D benchmark sequences

TUM RGB-D benchmark sequence	Tracking-VO SLAM				Matching-VO SLAM				Known accuracy
	ATE [m]	ATE std dev. [m]	RPE [m]	FPS [Hz]	ATE [m]	ATE std dev. [m]	RPE [m]	FPS [Hz]	ATE RMSE [m]
`fr1_desk`	0.044	0.021	0.020	24.9	0.027	0.013	0.011	3.5	0.026 [6]
`fr1_desk2`	0.084	0.034	0.028	32.7	0.040	0.016	0.015	2.9	0.043 [6]
`fr1_room`	0.155	0.051	0.014	38.6	0.133	0.049	0.010	3.3	0.084 [6]
`fr2_desk`	0.095	0.034	0.012	19.7	0.067	0.016	0.008	2.1	0.076 [15]
`fr3_office`	0.057	0.018	0.013	26.1	0.023	0.012	0.009	3.0	0.035 [15]

The positional ATE RMSE and positional RPE RMSE results achieved on these sequences by our SLAM with two variants of the front-end are summarized in Tab. 2 (note that `fr3_office` stands for `fr3_long_office_household`). In these tests parameters that considerably depend on the characteristics of the data, like the distance thresholds in the map maintenance procedure have been adjusted individually for each sequence. For all five sequences the matching-based version was more accurate than the tracking-based one. In the TUM sequences there are many RGB frames with a considerable amount of motion blur and the tracking-based front-end using the corner-like ORB detector had problems with the drift and perhaps aliasing of keypoints in such frames. The matching-based version managed to produce more repeatable point features using the multi-scale SURF, which resulted in very accurate pose estimation. However, this was at the cost of low frame rate, an order of magnitude smaller than for the tracking-based version. The tracking-based version achieved the trajectory estimation accuracy that is satisfactory for most tasks of a mobile robot, being able to work in real-time without any hardware acceleration. For comparison we provide in Tab. 2 also the positional ATE RMSE values achieved on the respective sequences by other state-of-the-art systems, namely the RGB-D SLAM [6] and the Submap-based BA described in [15]. As far as we can tell from the literature analysis, these were the best ATE results published till now for the respective TUM benchmark sequences. Our approach achieves similar or even smaller positional ATE values. Trajectories recovered for three exemplary sequences are visualized in Fig. 5.

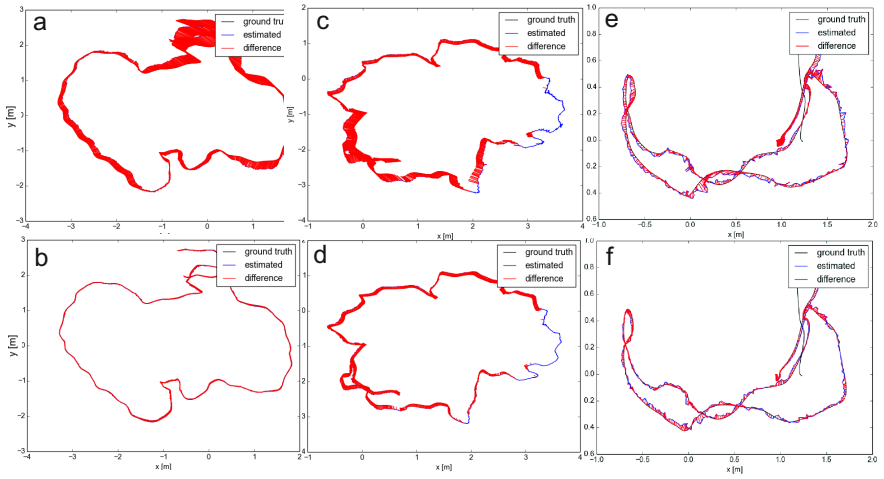

Fig. 5 Estimated trajectories with the ATE error for TUM RGB-D benchmark sequences: `fr3_long_office_household` (a,b), `fr2_desk` (c,d), and `fr1_desk` (e,f) recovered by our SLAM with tracking-based (a,c,e), and matching-based (b,d,f) front-end

6 Conclusions

We have presented a new feature-based RGB-D SLAM system, which employs the factor graph optimization approach, but unlike the more popular pose-based SLAM approaches builds a persistent map of 3D features. The new concept in this system is the use of a VO pipeline to compute the sensor motion guess, instead of direct tracking of the sensor against the map of features. This concept implemented by using the very fast sparse optical flow tracker results in a map-based RGB-D SLAM that can run at the Kinect frame rate without any hardware acceleration on a commodity PC. The high number of feature-to-pose constraints established and maintained by our algorithm enables more precise trajectory reconstruction than in the pose-based SLAM systems. The robustness was improved by new feature management techniques: the detection of occluded features by directly using the dense depth image, and multiple-view-angle descriptors associated with the mapped features. The tracking-based front-end enables real-time operation (up to 38 Hz), but it achieves lower accuracy of the trajectory estimation than the matching-based variant implemented for comparison. This is attributed to the choice of feature type: ORB keypoints are less repeatable, and more prone to errors due to motion blur. Thus, the tracked features drift more than the points generated by SURF. The SURF keypoints are more evenly distributed and stable on the image, but SURF is too slow to compute and match for real-time operation.

Acknowledgement This work was financed by the National Science Centre under decision DEC-2013/09/B/ST7/01583.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding **110**(3), 346–359 (2008)
2. Belter, D., Skrzypczyński, P.: Precise self-localization of a walking robot on rough terrain using parallel tracking and mapping. Industrial Robot: An International Journal **40**(3), 229–237 (2013)
3. Belter, D., Nowicki, M., Skrzypczyński, P.: On the performance of pose-based RGB-D visual navigation systems. In: Cremers, D., et al. (eds.) Computer Vision – ACCV 2014. LNCS, vol. 9004, pp. 407–423. Springer (2015)
4. Belter, D., Skrzypczyński, P.: The importance of measurement uncertainty modeling in the feature-based RGB-D SLAM. In: Proc. Int. Workshop on Robot Motion and Control, Poznań, pp. 308–313 (2015)
5. Cummins, M., Newman, P.: Accelerating FAB-MAP with Concentration Inequalities. IEEE Trans. on Robotics **26**(6), 1042–1050 (2010)
6. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D Mapping with an RGB-D Camera. IEEE Trans. on Robotics **30**(1), 177–187 (2014)
7. Fioraio, N., Di Stefano, L.: SlamDunk: affordable real-time RGB-D SLAM. In: Computer Vision – ECCV 2014 Workshops. LNCS, vol. 8925, pp. 401-414. Springer (2015)
8. Gil, A., Martinez Mozos, O., Ballesta, M., Reinoso, O.: A comparative evaluation of interest point detectors and local descriptors for visual SLAM. Machine Vision and Applications **21**(6), 905–920 (2010)
9. Grisetti, G., Kümmerle, R., Ni, K.: Robust optimization of factor graphs by using condensed measurements. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, Vilamoura, pp. 581–588 (2012)
10. Handa, A., Whelan, T., McDonald, J. B., Davison, A. J.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: IEEE Int. Conf. on Robotics & Automation, Hong Kong, pp. 1524–1531 (2014)
11. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. Int. Journal of Robot. Res. **31**(5), 647–663 (2012)
12. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: Proc. IEEE Int. Conf. on Robotics & Automation, Karlsruhe, pp. 3748–3754 (2013)
13. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Int. Symp. on Mixed and Augmented Reality, Nara, pp. 225–234 (2007)
14. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: IEEE Int. Conf. on Robotics & Automation, Shanghai, pp. 3607–3613 (2011)
15. Maier, R., Sturm, J., Cremers, D.: Submap-based bundle adjustment for 3D reconstruction from RGB-D Data. In: Pattern Recognition. LNCS, vol. 8753, pp. 54–65. Springer (2014)
16. Nowicki, M., Skrzypczyński, P.: Combining photometric and depth data for lightweight and robust visual odometry. In: European Conf. on Mobile Robots, Barcelona, pp. 125–130 (2013)
17. Ozawa, R., Takaoka, Y., Kida, Y., Nishiwaki, K., Chestnutt, J., Kuffner, J., Inoue, H.: Using visual odometry to create 3D maps for online footstep planning. In: IEEE Int. Conf. on Systems, Man and Cybernetics, Hawaii, pp. 2643–2648 (2005)
18. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE Int. Conf. on Computer Vision, pp. 2564–2571 (2011)
19. Scherer, S., Zell, A.: Efficient onboard RGBD-SLAM for autonomous MAVs. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, Tokyo, pp. 1062–1068 (2013)
20. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conf. on Comp. Vis. and Pattern Recog., Seattle, pp. 593–600 (1994)
21. Strasdat, H., Davison, A. J. Montiel, J., Konolige, K.: Double window optimisation for constant time visual SLAM. In: Proc. Int. Conf. on Computer Vision, Los Alamitos, pp. 2352–2359 (2011)

22. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, Vilamoura, pp. 573–580 (2012)
23. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment – a modern synthesis. In: Vision Algorithms: Theory and Practice. LNCS, vol. 1883, pp. 298–372. Springer (2000)
24. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. on Pattern Analysis & Machine Intelligence **13**(4), 376–380 (1991)
25. Whelan, T., Johannsson, H., Kaess, M., Leonard, J., McDonald, J.: Robust real-time visual odometry for dense RGB-D mapping. In: IEEE Int. Conf. on Robotics & Automation, Karlsruhe, pp. 5704–5711 (2013)