On the Performance of Pose-based RGB-D Visual Navigation Systems

Dominik Belter, Michał Nowicki, Piotr Skrzypczyński

Institute of Control and Information Engineering, Poznań University of Technology, ul. Piotrowo 3A, PL-60-965 Poznań, Poland

Abstract. This paper presents a thorough performance analysis of several variants of the feature-based visual navigation system that uses RGB-D data to estimate in real-time the trajectory of a freely moving sensor. The evaluation focuses on the advantages and problems that are associated with choosing a particular structure of the sensor-tracking front-end, employing particular feature detectors/descriptors, and optimizing the resulting trajectory treated as a graph of sensor poses. Moreover, a novel yet simple graph pruning algorithm is introduced, which enables to remove spurious edges from the pose-graph. The experimental evaluation is performed on two publicly available RGB-D data sets to ensure that our results are scientifically verifiable.

1 Introduction

The introduction of compact and affordable RGB-D sensors, such like Microsoft Kinect and Asus Xtion Pro Live, triggered a new wave of research on visual SLAM (Simultaneous Localization and Mapping) [2, 12, 18] and VO (Visual Odometry) [33, 38] systems that rely on the direct depth measurements. A RGB-D VO system computes the sensor motion between the consecutive keyframes (selected frames of the RGB-D data stream), and estimates the trajectory. It can be paired with a back-end for post-processing of a pose-graph, whose vertices correspond to the sensor poses, and whose edges represent constraints between these poses [9]. A pose-based RGB-D visual navigation system can be implemented in many different forms. However, the diversity of details in the published research on both VO and pose-based visual SLAM makes it hard, or even impossible to tell, which structure is the best one, and how the implementation of particular components influences the performance.

The computer vision literature is rich in papers concerning performance evaluation and comparison of various algorithms for feature detection/description [22, 36], including comparative studies in the context of visual navigation [30]. Also authors of some papers on the RGB-D navigation methods or benchmarking, such like [12, 13, 17, 24] demonstrate the performance of their systems on publicly available data, and in some cases include a comparison to other systems. However, to the extent of our knowledge, no study is available concerning the influence of the particular design choices made to a RGB-D navigation system on its performance. Among the few works that tackle this problem Endres *et al.* [12] evaluate only three different feature descriptors with their RGB-D SLAM, while Strasdat [34] shows several variants of a large-scale visual navigation system, being however concerned mostly with the passive vision sensors.

Hence, this paper attempts to experimentally evaluate several configurations of a pose-based RGB-D navigation system. We implemented a RGB-D VO/SLAM in two main configurations, which respectively are based on the visual tracking of point features, or on the frame-to-frame matching of salient visual features. These two front-ends are shown as pure frame-to-frame VO systems, then enhanced by local trajectory optimization, and finally, they are turned into the full pose-graph SLAM systems by adding the loop closure detection and global pose-graph optimization back-end. Moreover, we demonstrate how to improve the precision of the pose estimates by pruning the pose-graph from the edges that appear to be spurious. The experiments were performed using two publicly available data sets: the well-known TUM RGB-D benchmark [35], and the very recent ICL-NUIM dataset [17].

2 Pose-based RGB-D Visual Navigation System

As the main idea of this paper is to demonstrate the performance of several configurations of the pose-based RGB-D navigation system, we divided our implementation into the separate front-end, which is an implementation of the VO concept, and the back-end implementing pose-graph optimization and the loop closure. The front-end and the back-end are implemented in separate Linux threads, and run asynchronous, exchanging only the necessary data: the pose-graph, and the data regarding the features and local descriptors for loop closure. Having the direct depth measurements from the Kinect or Xtion RGB-D sensor we consider the 3-D-to-3-D feature correspondences for frame-to-frame motion estimation [16]. Although combining the coordinates of the 2-D point features with the depth information we obtain 3-D positions of the point features, we do not keep these features in the system after computing the motion estimate. Optimization of a map structure containing potentially thousands of point features is time consuming, while advantages might be insignificant [13]. The lack of persistent 3-D features makes it impossible to use the sliding window bundle adjustment, implemented as in [11] or [23] to reduce the trajectory drift in the front-end. We took instead the approach suggested in [12]: we try to reduce locally the trajectory drift by constructing a pose-graph from m last sensor poses and estimate the motion constraints between the data frame attached to the current pose and the remaining m frames. Then, we apply the graph optimization provided by the back-end to this small pose-graph. This part of the approach, called Windowed Optimization (WO), should not be confused with the windowed bundle adjustment, because it is still purely pose-based and involves no features.

The front-end VO pipeline is proposed in two versions. The difference is in the approach to establish the correspondences between the 2-D features in the RGB images. As we want to estimate the motion between the first image $I_{v(k)}$ and the last image $I_{v(k+n)}$ in a sequence of n images (n is considered to be small), we

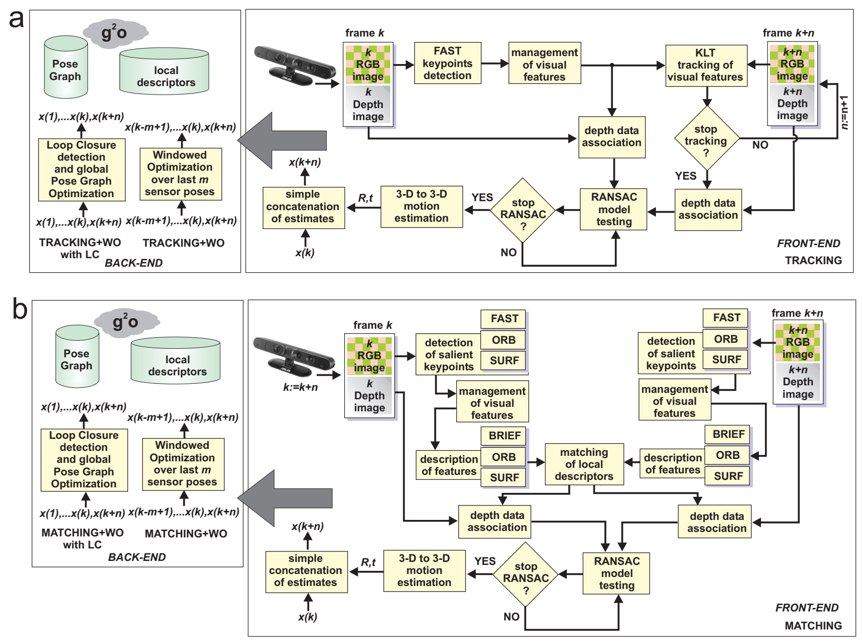

Fig. 1. Block diagram of the RGB-D visual navigation system in various configurations: the tracking-based version with optional Windowed Optimization and Loop Closure detection with pose-graph optimization (back-end) is depicted in (a), the matching-based version front-end with the same optional modules is shown in (b)

can detect the point features only in $I_{v(k)}$, and then track these points through the k images (Fig. 1a), or we can simply match the detected local descriptors of the features in $I_{v(k)}$ and $I_{v(k+n)}$ to obtain the correspondences (Fig. 1b). When the correspondences between $I_{v(k)}$ and $I_{v(k+n)}$ are established, the depth data are associated to the features resulting in two sets of matched 3-D points. The parameters of transformation between these two point patterns are estimated using a least squares estimation method [10, 37]. To make the transformation estimation robust to the outliers resulting from imperfect tracking results or wrong feature matches, the estimation procedure is embedded in the RANSAC scheme [15].

The back-end for pose-graph optimization is based on the open-source g^{2}o software package for least square optimization [20]. This software takes a pose-graph produced by the front-end as input, and performs a minimization of a non-linear error function that is represented by this graph's constraints (see section 4). Hence, the back-end can compute a globally consistent trajectory of the sensor, providing that all the constraints in the pose-graph (i.e. motion estimates) are correct [3]. We employ the g^{2}o back-end in two roles: to optimize small pose-graphs over a moving constant-length window in the Windowed Optimization procedure for local trajectory correction, and to optimize the global pose-graph, representing the whole recovered trajectory. The global optimization occurs whenever a loop closure is discovered. The loop closures are detected on RGB images by matching the frames belonging to poses that are positioned far enough along the trajectory. If a significant (visual) similarity between these frames is discovered on the basis of matching of the local descriptors a transformation is computed between the frames, and added as a constraint to the pose-graph.

3 Visual Odometry: the Front-end

3.1 Extraction and Management of Features

In the front-end we rely on feature-based methods for frame-to-frame motion estimation. Although the dense (appearance-based) methods are potentially more precise [19, 33], as they use more data, this approach is more computation-intensive, and prone to failures due to occlusions and sudden scene changes. Thus, we focus exclusively on the feature-based approach, which is widely considered to be appropriate for real-time robotics applications [29]. The feature-based VO requires to detect a set of keypoints, which should be salient, repeatable, localized precisely in the image, and computed as fast as possible. We employ and test three point feature detectors: FAST [27], ORB [28], and SURF [5]. Using SURF we expected good results, but were concerned about the real-time performance of the system. On the other hand, FAST and ORB are more recent and more computation efficient algorithms, but they are less robust [30]. The SURF and ORB have their own feature descriptors, while the FAST detector is paired in our system with the low-complexity binary BRIEF [7] descriptor. To make the feature detection more robust we use two techniques that were proposed in [24]: unsupervised clustering of the keypoints, and detection of features in subimages. The detection of points in separate, slightly overlapping subimages is a heuristic that helps to distribute the keypoints evenly on the image. However, this heuristic cannot deal with situations where many features are detected on a small area in the image. To solve this problem the DBScan, a fast clustering algorithm [14] is employed. Clusters of features are formed, and then they are represented by maximum two points. This technique provides results similar to the quadtree-based point detection method described in [34], but is much faster.

3.2 Matching, Tracking, and Motion Estimation

The core part of the front-end is motion estimation based on two sets of corresponding point features, whose correspondences are determined either by matching or by tracking. The matching approach relies on the fact, that corresponding 2-D features on two images have similar neighborhood, thus they should have similar local descriptors, such like SURF, ORB, or BRIEF. The similarity of the investigated descriptors is determined using the Euclidean norm for SURF, or the Hamming distance for the binary descriptors BRIEF and ORB. The implementation of matching involves rejecting matches if multiple descriptors from the second image $I_{v(2)}$ correspond to the descriptor of the same feature from the first image $I_{v(1)}$, and accepting correspondences only if the j-th descriptor $I^{j}_{v(2)}$ from the second image is the best match for the i-th descriptor $I^{i}_{v(1)}$ from the first image, and at the same time the descriptor $I^{i}_{v(1)}$ is determined as the best match for descriptor $I^{j}_{v(2)}$.

In comparison to matching, the tracking does not need the description of features, and performs detection only on the subset of images. The idea of tracking is to detect features at the keyframe, and then looking for the position of this feature in the new image by searching locally. In our system, the tracking is performed using a pyramid implementation [6] of the Lucas-Kanade optical flow algorithm [4]. Tracking is initialized with points from the FAST detector, which is more efficient than the usual Shi-Tomasi from [31]. The maximum number of tracked features in our experiments was 500. Tracking is computationally less demanding than matching using classic descriptors, such like SIFT [21] or SURF. The VO front-end tracks features over a number of images of the RGB-D sequence between the two keyframes that are processed with depth images. When the number of successfully tracked features falls below a given threshold or the maximum allowed number of the RGB frames in tracking is reached (max. n=5), the transformation between the keyframes is computed within the RANSAC scheme.

The RANSAC is used to randomly select 3 pairs of points from the set of tracked or matched features and to estimate the 3-D transformation using the Umeyama algorithm [37]. In every iteration, a model transformation is computed and evaluated. The number of RANSAC iterations is estimated using a simple probabilistic model [8] to improve speed. When the RANSAC-based model search is finished, the transformation is re-estimated from all inlier-pairs. Also, if the number of inliers is high, the iterative model correction is applied by rejecting the inliers that are the least probable within the model estimated so far [26].

4 Pose-based Optimization: the Back-end

4.1 Pose-graph Optimization

The back-end of our system is based on g^{2}o – a general framework for graph optimization [20]. We store each measurement between the robot/sensor poses in a graph (Fig. 2). Each vertex v_i in the graph represents a sensor pose. As motivated before, we do not keep point features in the graph structure. The edges in the graph represent measurements between two vertices. Measurement $\mathbf{M}_{ij}$ represents a 3-D transformation (translation and rotation) between poses v_i and v_j. The quality of the measurements is represented by an information matrix $\boldsymbol{\Omega}_{ij}$ (inverse of a covariance matrix), which can be obtained by propagating uncertainty from the measurement model of the RGB-D sensor [25], or set as identity matrix if equal uncertainty of all 3-D transformations is assumed.

The input variables, which are provided for the graph optimization are measurements (edges of the graph). The graph optimization returns poses of the

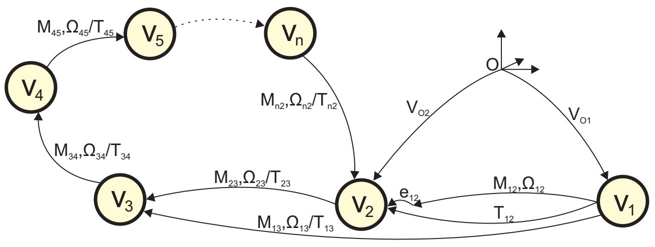

Fig. 2. Graph representation of the pose-based SLAM

sensor (vertices of the graph) which correspond to the trajectory of the sensor. The optimization is possible if at least one vertex has at least two incoming edges. The Windowed Optimization procedure provides additional edges to the graph. With this procedure we can add relations between more distant vertices of the graph (edge M_{13} in Fig. 2). We can also add new measurements whenever the robot returns to previously visited places. Loop closure procedure allows to close the graph and improves the estimation of the sensor pose (edge M_{n2} in Fig. 2). The graph optimization procedure minimizes the global error $\mathbf{E}$:

$$\mathbf{E} = \sum_{ij} \widetilde{\mathbf{e}}_{ij}^T \mathbf{\Omega}_{ij} \widetilde{\mathbf{e}}_{ij}, \tag{1}$$

where $\widetilde{\mathbf{e}}$ is a vector, which determines the discrepancy between the current vertex pose and the measurements. Graphical representation of error computation for each edge is presented in Fig. 2. The error $\mathbf{e}_{ij}$ is computed as:

$$\mathbf{e}_{ij} = \mathbf{M}_{ij}^{-1} \mathbf{T}_{ij}, \tag{2}$$

where $\mathbf{T}_{ij}$ is the estimated transformation between the considered vertices. Defining the poses of the vertices v_i and v_j in the reference coordinate system as $\mathbf{V}_{Oi}$ and $\mathbf{V}_{Oj}$, respectively, we can rewrite (2):

$$\mathbf{e}_{ij} = \mathbf{M}_{ij}^{-1} \mathbf{V}_{Oj}^{-1} \mathbf{V}_{Oi}. \tag{3}$$

Eventually, the homogeneous transformation $\mathbf{e}_{ij}$ is parametrised to the vector $\widetilde{\mathbf{e}}$ and used for optimization.

4.2 Graph pruning

Despite of procedures in the front-end, which remove wrong matches and ensure robust motion estimation, some wrong transformations might be added to the graph. These outlier constraints (edges) influence the optimization results and output trajectory. While the recent results show that the back-end can be made robust to outlier constraints [1], we simply detect such edges and remove them from the pose-graph. For graph pruning we use the error value χ^2 (goodness of fit) provided by g^{2}o for each estimated edge: $\chi^2 = \widetilde{\mathbf{e}}_{ij}^T \mathbf{\Omega}_{ij} \widetilde{\mathbf{e}}_{ij}$.

Algorithm 1: Local Adaptive Pruning procedure

```
Data: praph $G$
Result: pruned graph $G_{pruned}$
continue:=1;
while (continue=1) do
    optimizeGraph(); continue:=0;
    for i:=1:1:n do
        edgeSet:= FindIncomingEdges($v_i$);
        outlier:= FindOutlier(edgeSet);
        if outlier != singleOutgoingEdge then
            RemoveEdge(outlier);
            continue = 1;
        end
    end
end
```

We applied two approaches to remove outlier edges. In the simple approach we use the χ^2 test globally. The graph optimization and graph pruning stages are running alternately. After each optimization cycle we remove edges for which the χ^2 value is greater than a fixed threshold related to the χ^2 distribution (2.0 is used, which corresponds to 0.92 probability). If the vertex contains more than one incoming edge with χ^2 bigger than threshold we remove only the worst one in the single iteration. We repeat the pruning and optimization sequence until all remaining edges have the χ^2 value smaller than the selected threshold. The Local Adaptive Pruning procedure (Algorithm 1) exploits the locality of the pose-graph – we have observed that the incorrect constraints (edges) in the graph are usually the ones that "pull out" the given vertex in another direction than other edges incoming to the given vertex, having therefore a much worse χ^2 value. The adaptive pruning re-runs optimization until all outlier edges are removed. We search over the whole pose-graph. For each vertex v_i we find all incoming edges (FindIncomingEdges procedure). The procedure FindOutlier detects an outlier within the set of edges incoming to the given node. To this end, this procedure computes the median value χ^2_{median} for all the incoming edges of the given node (edgeSet). The edge that has the biggest $\chi^2/\chi^2_{\text{median}}$ value, and its χ^2 is at least p-times worse than the median is considered an outlier. We determined experimentally that p=10 suits best for most of the analysed data sets and all the tested configurations of the system. The procedure RemoveEdge deletes the outlier edge only if its predecessor vertex in the pose-graph has more than one outgoing edge.

5 Experimental Results

5.1 Experiments and Data Sets

The aim of our experiments was to determine the properties of several configurations of the RGB-D visual navigation system. These configurations can be

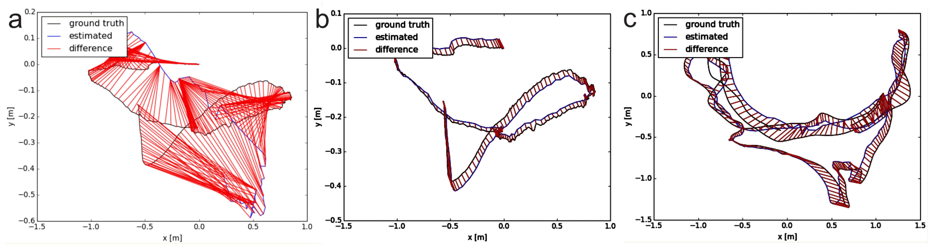

Fig. 3. Example recovered trajectories with the ATE error: trajectories ICL-NUIM `living room/2` estimated using the Matching+WO FAST-BRIEF with LC variant (a), and the Matching+WO SURF with LC variant (b) show a dramatic difference in accuracy, simple Tracking VO performs well for the TUM `fr1/room` sequence (c)

divided in two main groups: the VO systems, either frame-to-frame or using the Windowed Optimization (WO), and pose-based SLAM configurations, with the Loop Closure (LC) discovery and global optimization of the pose-graph. Note that the WO procedure always uses the same detector/descriptor pair as the main VO pipeline. The tracking-based variant was tested with the LC using three different detector/descriptor pairs, as in tracking there is no possibility to re-use the descriptors and they have to be computed for LC.

The experiments were performed using the RGB-D data from two data sets: the TUM RGB-D data set [35], and the recent ICL-NUIM data set [17]. The TUM RGB-D data set, containing data acquired from either the Kinect or Xtion sensor in a scenario of indoor visual navigation, was used to evaluate a RGB-D SLAM system [12]. The ICL-NUIM data set contains RGB-D sequences from a synthetic environment with perfect ground-truth poses of the sensor. The rendered data is free from motion blur and artifacts, hence makes it easier to isolate the causes of failures. The perfect ground-truth is important when testing systems that achieve very small pose errors. The authors of [17] also tested several RGB-D visual navigation solutions on their data set, and their results can be directly compared to the performance of our systems. All experiments shown in this paper were conducted on a standard laptop computer with 2.5GHz CPU and 8GB RAM. We used the evaluation tools provided with the TUM RGB-D data set. The error metric mostly used is the Absolute Trajectory Errors (ATE), as it shows the difference between the recovered trajectory of the sensor and the ground truth trajectory (Fig. 3). The Relative Pose Error (RPE) is used to illustrate the local drift of the VO front-end.

5.2 Performance of the VO front-end

From the results in Tab. 1, it can be observed that matching using the FAST-BRIEF (F-B) detector/descriptor pair definitively yields the biggest errors among the compared methods. When taking into account the RPE results, it is obvious that matching based of FAST-BRIEF performs so poorly not because of a single mismatch, but also due to the low precision of the motion estimation for each frame-to-frame increment.

Table 1. Trajectory estimation results for various configurations of the visual navigation front-end for the TUM `fr1/room` dataset

navigation system config.	Frame-to-Frame				Windowed Optimization			
front-end type,	ATE	RPE	Orient.	FPS	ATE	RPE	Orient.	FPS
detector & descriptor	RMSE	RMSE	RMSE		RMSE	RMSE	RMSE	
	[m]	[m]	[deg.]	[Hz]	[m]	[m]	[deg.]	[Hz]
Tracking	**0.191**	0.026	1.631	33.97	0.923	0.121	6.37	30.68
Matching FAST-BRIEF	1.194	0.052	5.179	50.17	1.042	0.056	4.755	29.57
Matching ORB	0.507	0.031	1.64	47.64	0.366	0.026	1.243	32.25
Matching SURF	0.201	0.017	1.42	20.76	0.206	0.019	0.864	17.16

The lowest ATE error (lowest errors are shown in bold in all tables) for the TUM `fr1/room` dataset (1362 frames, ground-truth trajectory 15.989m) is obtained by pure tracking of FAST features, but this configuration results in bigger error when the WO is enabled. This is believed to be a result of the fact, that in the case of tracking the additional constraints introduced by WO are based on the features positions that have been already used when computing the regular frame-to-frame motion estimates. Therefore, the additional graph edges are redundant to the already existing edges in the simple tracking solution, and may only have a negative impact on the achieved results. The matching using SURF results in less motion drift when compared to matching with ORB, but both methods improve trajectory estimates when Windowed Optimization is used. What is worth noticing, are the framerates (FPS) of each variant The ORB presents similar, fast working speed to the FAST-BRIEF (even 30 Hz with the WO), with better results probably due to the ORB detection being performed on the image pyramid. The tracking has similar working speed (30 Hz) with and without WO. The slowest of all the tested variants is matching based on SURF, which due to multiscale, complicated detection, floating point type descriptors and matching based on the Euclidean norm results in system operating with the maximal frequency of approx. 20 Hz.

Table 2. Trajectory estimation results for various configurations of the visual navigation front-end for the TUM `fr1/desk` dataset

navigation system config.	Frame-to-Frame				Windowed Optimization			
front-end type,	ATE	RPE	Orient.	FPS	ATE	RPE	Orient.	FPS
detector & descriptor	RMSE	RMSE	RMSE		RMSE	RMSE	RMSE	
	[m]	[m]	[deg.]	[Hz]	[m]	[m]	[deg.]	[Hz]
Tracking	0.441	0.052	3.558	34.8	0.447	0.064	4.158	31.57
Matching FAST-BRIEF	0.659	0.131	9.425	49.35	0.693	0.11	8.249	29.14
Matching ORB	0.408	0.033	2.672	48.43	**0.073**	0.023	1.541	35.3
Matching SURF	**0.200**	0.027	1.882	20.38	**0.079**	0.022	1.378	18.46

Experiments evaluating the same variants were also performed for the TUM `fr1/desk` dataset (presented in Tab. 2) and for ICL-NUIM `office/0` dataset (presented in Tab. 3). Similarly to the results presented in Tab. 1, the tracking results in lower error when local optimization is not used. Also, for the data sets in Tab. 2 and Tab. 3, the matching based on FAST-BRIEF pair results in ATE and RPE errors that are much higher than the respective results for other variants. With the maximum frequency of FAST-BRIEF similar or worse than ORB, this proves that the FAST-BRIEF pair is unreliable, and should not be used in RGB-D visual odometry. The Windowed Optimization used with matching-based solutions with SURF or ORB results in slight improvements of the trajectory estimates. What is interesting, both solutions achieve similar ATE, which is below 8 cm for the TUM `fr1/desk` dataset (613 frames, ground-truth trajectory 9.263m) and below 3.3 cm for the ICL-NUIM `office/0` sequence (1510 frames, ground-truth trajectory 6.52m) with the perfect ground-truth. These results compared to the results presented in [12, 13, 17] prove that even without the full SLAM optimization some variants of the proposed navigation system achieve similar or better results than the state-of-the-art solutions.

Table 3. Trajectory estimation results for various configurations of the visual navigation front-end for the ICL-NUIM `office/0` dataset

navigation system config.	Frame-to-Frame				Windowed Optimization			
front-end type, detector & descriptor	ATE RMSE [m]	RPE RMSE [m]	Orient. RMSE [deg.]	FPS [Hz]	ATE RMSE [m]	RPE RMSE [m]	Orient. RMSE [deg.]	FPS [Hz]
Tracking	0.167	0.012	1.904	22.55	0.629	0.076	1.664	20.5
Matching FAST-BRIEF	0.626	0.021	2.136	15.23	0.588	0.022	3.354	16.55
Matching ORB	0.047	0.009	0.469	25.27	0.033	0.007	0.408	24.05
Matching SURF	**0.039**	0.007	0.445	13.08	**0.030**	0.006	0.41	15.84

5.3 Influence of the Loop Closure

The trajectory recovered by the VO system using the frame-to-frame motion estimation has a drift, which can be decreased by the WO procedure, but will still grow with time, as there are no constraints on the trajectory that enforce the global consistency. A possibility to decrease drift estimation arises, whenever the robot/sensor re-visits already explored areas. To detect these situations a simple loop closure technique is used, which operates in the back-end. The results of evaluated versions with additional loop closure are presented in Tab. 4 and Tab. 5. Due to the previously presented poor results for Tracking+WO, the loop closure module was added to the tracking solution without local optimization of the pose-graph. As expected, the addition of LC results in the decreased ATE and RPE. In some cases, the loop closure makes these systems very precise with the ATE errors below 5 cm for the TUM `fr1/desk` and approx 10 cm for the bigger environment in the TUM `fr1/room`. For both of these data sets, the best results are obtained by the system variants based on matching with ORB and

SURF, with the ORB-based version being however almost two times faster than the SURF-based one. A slightly higher error is observed for Tracking with LC, which has similar speed to the ORB-based matching. The variants based on the FAST-BRIEF perform poorly with an exception for Tracking with LC based on FAST-BRIEF for the TUM `fr1/room`, but even in that case the system achieves higher ATE than the similarly configured systems based on ORB or SURF.

Table 4. Trajectory estimation results for various configurations of the visual navigation front-end with the g^{2}o back-end optimization for the selected TUM datasets

navigation system config.	TUM `fr1/room`				TUM `fr1/desk`			
front-end type,	ATE	RPE	Orient.	FPS	ATE	RPE	Orient.	FPS
detector & descriptor	RMSE	RMSE	RMSE		RMSE	RMSE	RMSE	
all with g^{2}o back-end	[m]	[m]	[deg.]	[Hz]	[m]	[m]	[deg.]	[Hz]
Tracking with LC F-B	0.165	0.026	1.596	35.55	8.669	8.654	69.729	35.3
Tracking with LC ORB	**0.113**	0.024	1.383	35.64	**0.079**	0.033	2.54	33.62
Tracking with LC SURF	0.114	0.024	1.431	35.47	0.247	0.037	2.869	35.44
Matching+WO F-B with LC	1.042	0.056	4.755	29.57	5.802	5.027	56.019	29.31
Matching+WO ORB with LC	**0.107**	0.026	1.258	32.35	**0.055**	0.023	1.491	35.19
Matching+WO SURF with LC	**0.103**	0.019	0.837	16.75	**0.049**	0.021	1.356	18.47

On the synthetic ICL-NUIM data sets the lowest errors are for Tracking with LC based on SURF. Due to the fact, that the time allocated to the LC and global pose-graph optimization is constrained by the time of the frame-to-frame matching (the threads have to synchronize at each keyframe), the whole system operates faster (20 Hz), even with the relatively slow SURF detector/descriptor. The best ATE results equal to 2 cm for Tracking with the ORB-based LC for the ICL-NUIM `office/0`, and 2.1 cm for Tracking with SURF-based LC for the ICL-NUIM `living room/2` (882 frames, ground-truth trajectory 8.42m) demonstrate that in the absence of motion blur and image artifacts a simple, but carefully implemented RGB-D visual navigation system can achieve better results than most of the solutions compared in [17].

5.4 Pruning of the Pose-Graph

Unfortunately, there exist situations, where the constrains from loop closure have a negative influence on the performance of the pose-based SLAM. Even a single outlier constraint can have an arbitrary large impact on the graph optimization which is based on the least-squares principle. However, contrary to the situation in the filtration-based SLAM systems [32], such a wrong measurement can be removed, and then the pose-graph can be re-estimated in a correct form. Therefore, the influence of the proposed pose-graph pruning technique is demonstrated in Tab. 6 and Tab. 7.

The back-end optimization (g^{2}o) improves the estimate of the trajectory by reducing the error (1) for the whole graph. Results obtained with one of the

Table 5. Trajectory estimation results for various configurations of the visual navigation front-end with the g^{2}o back-end optimization for the selected ICL-NUIM datasets

navigation system config.	ICL-NUIM `office/0`				ICL-NUIM `living room/2`			
front-end type,	ATE	RPE	Orient.	FPS	ATE	RPE	Orient.	FPS
detector & descriptor	RMSE	RMSE	RMSE		RMSE	RMSE	RMSE	
all with g^{2}o back-end	[m]	[m]	[deg.]	[Hz]	[m]	[m]	[deg.]	[Hz]
Tracking with LC F-B	0.398	0.012	2.473	22.54	0.051	0.007	0.431	21.93
Tracking with LC ORB	**0.020**	0.009	0.671	22.45	0.067	0.013	0.518	20.04
Tracking with LC SURF	0.021	0.01	0.964	22.65	**0.021**	0.006	0.399	21.82
Matching+WO F-B with LC	0.588	0.022	3.354	16.55	0.912	0.035	2.901	15.1
Matching+WO ORB with LC	**0.015**	0.007	0.415	18.65	0.089	0.009	0.341	23.12
Matching+WO SURF with LC	0.062	0.057	1.174	12.12	0.036	0.006	0.303	14.35

Table 6. Pose graph pruning results for various configurations of the visual navigation front-end with or without back-end optimization for the TUM `fr1/room` dataset

navigation system config.	Simple pruning – χ^2 test			Local adaptive pruning		
front-end type,	ATE	RPE	Orient.	ATE	RPE	Orient.
detector & descriptor,	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
back-end type	[m]	[m]	[deg.]	[m]	[m]	[deg.]
Tracking+WO	0.917	0.114	6.17	0.989	0.115	6.17
Matching+WO F-B	1.065	0.056	4.87	1.043	0.055	4.87
Matching+WO ORB	0.365	0.025	1.24	0.341	0.025	1.23
Matching+WO SURF	0.285	0.019	0.86	0.295	0.019	0.84
Tracking with LC F-B	0.164	0.025	1.59	0.165	0.025	1.59
Tracking with LC ORB	0.113	0.024	1.38	0.113	0.024	1.40
Tracking with LC SURF	0.113	0.024	1.43	0.113	0.024	1.43
Matching+WO F-B with LC	0.648	0.099	6.44	0.703	0.106	8.03
Matching+WO ORB with LC	0.055	0.022	1.49	0.048	0.022	1.49
Matching+WO SURF with LC	**0.048**	0.021	1.35	**0.039**	0.021	1.29

best variants of our navigation system (Matching+WO ORB with LC) for the ICL-NUIM `office/0` are presented in Fig. 4. Before optimization the obtained trajectory is slightly distorted. After optimization and pruning the trajectory is smooth and very close to the ground truth trajectory.

In Fig. 5 we present some properties of the pruning method. Fig. 5a presents the optimized trajectory for the ICL-NUIM `office/0` data set, obtained with the Matching+WO SURF with LC variant of the system. (ATE RMSE error is about 6 cm). The same trajectory optimized with pruning is presented in Fig. 5b (ATE error is reduced to 1.5 cm). Incorrect matches introduced by the Loop Closure procedure moved the vertex indicated by the arrow no. 1 in Fig. 5a to a wrong position. The erroneous measurement is detected by the pruning procedure and removed from the graph. This situation corresponds to the initial value of the RPE error presented in Fig. 5c. The RPE error is significantly reduced after pruning. The presented pruning procedure removes also incorrect edges introduced sporadically by the Windowed Optimization procedure (due

Table 7. Pose graph pruning results for various configurations of the visual navigation front-end with or without back-end optimization for the ICL-NUIM `office/0` dataset

navigation system config.	Simple pruning – χ^2 test			Local adaptive pruning		
front-end type,	ATE	RPE	Orient.	ATE	RPE	Orient.
detector & descriptor,	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
back-end type	[m]	[m]	[deg.]	[m]	[m]	[deg.]
Tracking+WO	0.527	0.048	1.26	0.715	0.090	1.80
Matching+WO F-B	0.584	0.021	2.11	0.586	0.020	2.15
Matching+WO ORB	0.032	0.006	0.40	0.028	0.006	0.40
Matching+WO SURF	0.030	0.006	0.41	0.029	0.006	0.40
Tracking with LC F-B	0.398	0.012	2.47	0.398	0.012	2.47
Tracking with LC ORB	0.020	0.009	0.67	0.021	0.008	0.46
Tracking with LC SURF	0.021	0.009	0.96	0.023	0.009	1.07
Matching+WO F-B with LC	0.588	0.022	3.35	0.583	0.020	3.10
Matching+WO ORB with LC	**0.015**	0.007	0.41	**0.014**	0.006	0.41
Matching+WO SURF with LC	0.061	0.057	1.17	**0.014**	0.006	0.41

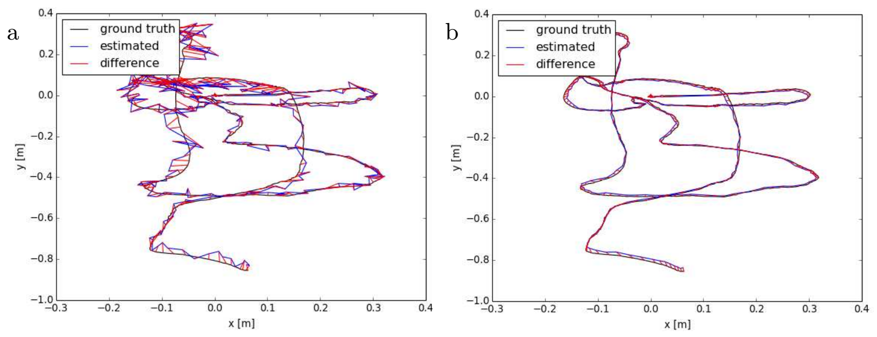

Fig. 4. Trajectory before (a) and after (b) g^{2}o optimization and pose-graph pruning. Matching+WO ORB with LC, ICL-NUIM `office/0` data set

to mismatching descriptors). The part of the trajectory indicated by arrow no. 2 in Fig. 5a is presented as details of the pose-graph in Fig. 5d. The procedure removes edges which do not fit to the obtained path and improves the final estimate of the poses.

6 Conclusions

This paper presents the comparison of several configurations of a relatively simple pose-based visual navigation system using the RGB-D data. The experimental results clearly show that there are significant differences in the performance of the considered variants of the visual navigation system, even though all these variants are based on the same general concept, and they share many critical components. Both structures of the VO, based on tracking and matching, respectively have proven to be suitable for the front-end of RGB-D SLAM system.

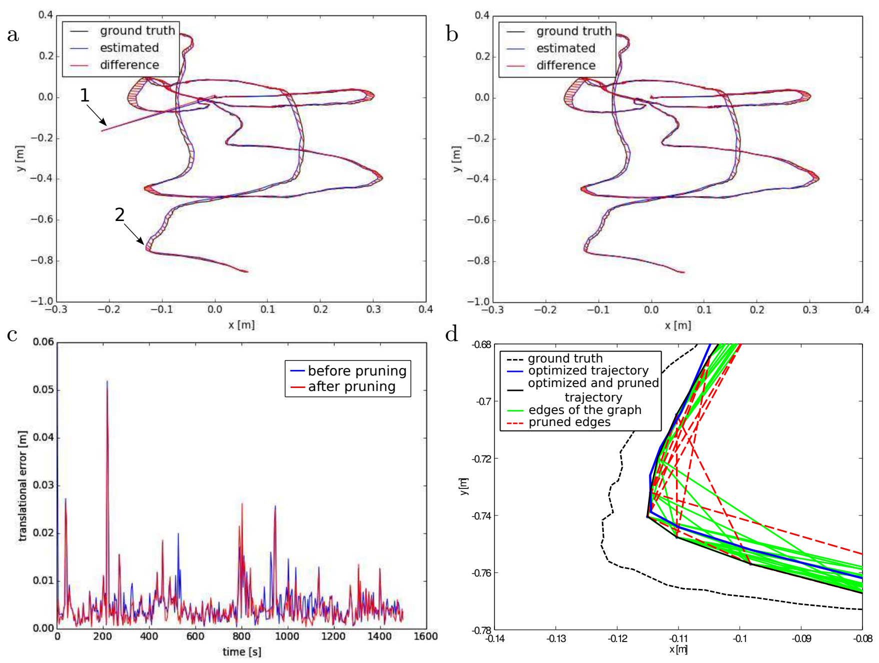

Fig. 5. Pruning properties presented for the Matching+WO SURF with LC on the ICL-NUIM `office/0` data set: trajectory obtained after g^{2}o optimization (a), trajectory obtained after g^{2}o optimization and pruning (b), RPE error before and after pruning (c), details of the pose-graph with removed edges (d)

The tracking-based VO pipeline is simple, fast and precise, but only if it is feed by good quality images at a high frame rate. On the other hand, the performance of the matching-based version critically depends on the used detector-descriptor pair. The performance of the ORB-based version was comparable or even slightly better than the performance of the tracking approach, with regard to both the precision and speed. An advantage of the matching-based version of the front-end is also the possibility to improve the trajectory by local pose-graph optimization, which turned out to be impossible with tracking. However, other variants based on matching did not perform so well, with the SURF-based version being the slowest one, and the FAST-BRIEF variant producing unacceptable trajectory errors. The ability to remove wrong loop closure constraints is very important to the pose-based RGB-D SLAM, as shown by our pose-graph pruning results. In the further research we plan to test robust estimation-based approaches to the outlier removal problem [1], and investigate how to efficiently include the point features in the optimization.

Acknowledgement. This research was financed by the Polish National Science Centre grant funded according to the decision DEC-2013/09/B/ST7/01583.

References

1. Agarwal, P., Grisetti, G., Tipaldi, G., Spinello, L., Burgard, W., Stachniss, C.: Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates. In: Proc. IEEE Int. Conf. on Robotics & Automation, Hong Kong, (2014) 3626–3631
2. Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A., Krainin, M., Maturana, D., Fox, D., Roy, N.: Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. Int. Journal of Robot. Res., 31(11), (2012) 1320–1343
3. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping: Part II. IEEE Robotics & Automation Magazine, 13(3), (2006) 108–117
4. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: A unifying framework. Int. Journal of Computer Vision, 56(3), (2004) 221–255
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110(3), (2008) 346–359
6. Bouguet, J. Y.: Pyramidal implementation of the Lucas-Kanade feature tracker, description of the algorithm. (2000).
7. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a local binary descriptor very fast. IEEE Trans. on Pattern Analysis & Machine Intelligence, 34(7), (2012) 1281–1298
8. Choi, S., Kim, T., Yu, W.: Performance evaluation of RANSAC family. In: Proc. British Machine Vision Conference, London, (2009)
9. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: Part I. IEEE Robotics & Automation Magazine, 13(2), (2006) 99–110
10. Eggert, D. W., Lorusso, A., Fisher, R. B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. Machine Vision and Applications, 9(5–6), (1997) 272-290
11. Engels, C., Stewenius, H., Nistér, D.: Bundle adjustment rules. In: Photogrammetric Computer Vision, September, (2006)
12. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the RGB-D SLAM system. In: Proc. IEEE Int. Conf. on Robotics & Automation, St. Paul, (2012) 1691–1696
13. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D Mapping with an RGB-D camera. IEEE Trans. on Robotics, 30(1), (2014) 177–187
14. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. Int. Conf. on Knowledge Discovery and Data Mining, (1996) 226–231
15. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commununications ACM, 24(6), (1981) 381–395
16. Fraundorfer, F., Scaramuzza, D.: Visual odometry: Part II: Matching, robustness, optimization, and applications. IEEE Robotics & Automation Magazine, 19(2), (2012) 78–90
17. Handa, A., Whelan, T., McDonald, J., Davison, A.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: IEEE Int. Conf. on Robotics & Automation, Hong Kong, (2014)
18. 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), (2012) 647–663

19. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: Proc. IEEE Int. Conf. on Robotics & Automation, Karlsruhe, (2013) 3748–3754
20. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: Proc. IEEE Int. Conf. on Robotics & Automation, Shanghai, (2011) 3607–3613
21. Lowe, D. G.: Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision, 60(2), (2004) 91–110
22. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. on Pattern Analysis & Machine Intelligence, 27(10), (2005) 1615–1630
23. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Generic and real-time structure from motion using local bundle adjustment. Image and Vision Computing, 27, (2009) 1178–1193
24. Nowicki, M., Skrzypczyński, P.: Combining photometric and depth data for lightweight and robust visual odometry. In: European Conference on Mobile Robots, (2013) 125–130
25. Park, J-H., Shin, Y.-D., Bae, J.-H., Baeg, M.-H.: Spatial uncertainty model for visual features using a Kinect sensor. Sensors, 12, (2012), 8640–8662.
26. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.: USAC: A universal framework for random sample consenus. IEEE Trans. on Pattern Analysis & Machine Intelligence, 35(8), (2013) 2022–2038.
27. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision (ECCV), (2006) 430–443
28. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE Int. Conference on Computer Vision (ICCV), (2011) 2564–2571
29. Scaramuzza, D., Fraundorfer, F.: Visual odometry: Part I the first 30 years and fundamentals. IEEE Robotics & Automation Magazine, 18(4), (2011) 80–92
30. Schmidt, A., Kraft, M., Fularz, M., Domagala, Z.: Comparative assessment of point feature detectors and descriptors in the context of robot navigation. Journal of Automation, Mobile Robotics & Intelligent Systems, 7(1) (2013) 11–20
31. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), (1994) 593–600.
32. Skrzypczyński, P.: Simultaneous localization and mapping: a feature-based probabilistic approach. Int. Journal of Applied Mathematics and Computer Science, 19(4), (2009) 575–588
33. F. Steinbrücker, J. Sturm, D. Cremers, Real-time visual odometry from dense RGB-D images, Workshop on Live Dense Reconstruction with Moving Cameras. In: IEEE Int. Conference on Computer Vision (ICCV), Barcelona, (2011)
34. Strasdat, H., Local accuracy and global consistency for efficient visual SLAM. PhD Dissertation, Imperial College, London, (2012)
35. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proc. IEEE/RSJ Int. Conference on Intelligent Robots & Systems, Vilamoura, (2012) 573–580
36. Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: a survey. Foundations and Trends in Computer Graphics and Vision, 3(3), (2008) 177–280
37. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. on Pattern Analysis & Machine Intelligence, 13(4), (1991) 376–380.
38. Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., McDonald, J. B.: Robust real-time visual odometry for dense RGB-D mapping. In: Proc. IEEE Int. Conf. on Robotics & Automation, Karlsruhe, (2013) 5704–5711