2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

RESLAM: A real-time robust edge-based SLAM system

Fabian Schenk[1] and Friedrich Fraundorfer[1]

***Abstract*— Simultaneous Localization and Mapping is a key requirement for many practical applications in robotics. In this work, we present RESLAM, a novel edge-based SLAM system for RGBD sensors. Due to their sparse representation, larger convergence basin and stability under illumination changes, edges are a promising alternative to feature-based or other direct approaches. We build a complete SLAM pipeline with camera pose estimation, sliding window optimization, loop closure and relocalisation capabilities that utilizes edges throughout all steps. In our system, we additionally refine the initial depth from the sensor, the camera poses and the camera intrinsics in a sliding window to increase accuracy. Further, we introduce an edge-based verification for loop closures that can also be applied for relocalisation. We evaluate RESLAM on wide variety of benchmark datasets that include difficult scenes and camera motions and also present qualitative results. We show that this novel edge-based SLAM system performs comparable to state-of-the-art methods, while running in real-time on a CPU. RESLAM is available as open-source software[1].**

I. Introduction

Simultaneous Localization and Mapping (SLAM) [1] has been a very active research area due its many applications such as autonomous driving, navigation, 3D reconstruction, augmented and virtual reality. SLAM is the task of creating a map of an unknown environment while simultaneously estimating the egomotion of an agent within this map. In contrast, visual odometry (VO) methods that estimate only the egomotion of an agent with respect to N previous keyframes without keeping a global map, typically suffer from drift, i.e. the difference between the estimated and the real trajectory. A global map offers the capabilities to correct drift by detecting previously visited places and performing loop closure.

Monocular SLAM systems [2], [3] suffer from various limitations since depth is not observable from just one camera. In such systems, the scale of map and trajectory is unknown, scale drift can occur, initialization to get an initial depth estimate is difficult and camera motion estimation fails under pure rotations. Many of these limitations can be addressed by RGBD sensors such as the Microsoft Kinect, the Orbbec Astra Pro or Intel's RealSense that can simultaneously record a scene's texture as RGB image and its geometry as depth map.

RGBD SLAM systems can be divided into two main categories. (i) Feature-based methods that extract and match features, thereby discarding most of the image information [4], [5]. (ii) Direct methods that do not rely on correspondence computation but are based on photometric and/or geometric error [6], [7], [8], [9]. Recently, a new class of direct VO methods that rely on edge alignment has emerged [10], [11], [12]. While these edge-based VO systems show promising results, no full SLAM system that includes loop closure and relocalisation exists.

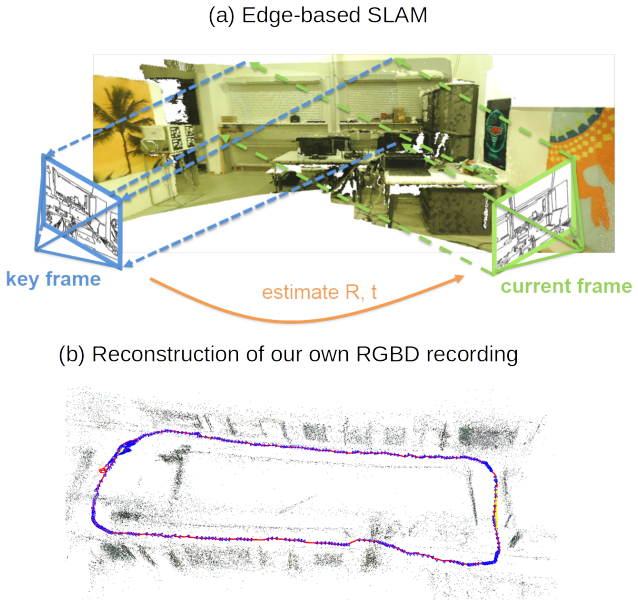

Fig. 1. RESLAM is an edge-based SLAM system for RGBD sensors that comprises a sliding window optimization, loop closure and offers relocalisation capabilities.

In this work, we present RESLAM, a complete SLAM pipeline that utilizes edges throughout the whole process, while running in real-time on a CPU. The main contributions of this work are the following:

- An edge-based SLAM system that comprises trajectory estimation, loop closure and relocalisation.
- A local sliding window optimization over several keyframes to refine the depth of each edge, the camera calibration parameters and the camera poses.
- We extensively evaluate on a wide variety of benchmark datasets and also show qualitative results.
- A fast SLAM system that runs in real-time on a CPU.
- RESLAM will be released as open-source[1].

II. Related Work

In this section, we focus on the most important publications in the fields of RGBD VO and SLAM and refer the reader to [13] for a more thorough overview.

Endres et al. introduced the feature-based RGBD-SLAM [5] that computes the relative camera motion between

[1]Fabian Schenk and Friedrich Fraundorfer are with the Institute of Computer Graphics and Vision , Graz University of Technology {schenk,fraundorfer}@icg.tugraz.at

[1]Code is available: https://github.com/fabianschenk/RESLAM

frames from feature matches and verifies the results by iterative closest point (ICP). The current state-of-the-art feature-based system ORB-SLAM2 [4] utilizes ORB features for tracking, mapping and loop closing and also supports stereo and monocular setups. Known limitations of feature-based methods are the error-prone matching step that typically requires a subsequent filter to remove outliers and the often uneven spatial distribution of the features.

In contrast, direct methods do not require correspondences but estimate camera motion directly. KinectFusion [14] uses the depth map to estimate the relative camera motion and a dense 3D model using ICP but is restricted to very small workspaces due to the high memory demands. Kähler et al. [15] proposed a scalable ICP-based SLAM system that builds a global map from multiple submaps and performs loop closure and relocalisation. Purely ICP-based system are error-prone in scenes with little geometric structure and therefore several systems incorporate an additional photometric error term into their camera motion estimation [6], [9], [16]. Kintinuous [16] addresses several limitations of [14], while the surfel-based ElasticFusion [6] achieves global consistency through non-rigid deformations of the map instead of a pose graph optimization. The recent BundleFusion [9] system performs sparse-then-dense global pose optimization, where the authors first use SIFT features for coarse alignment, and subsequently refine the estimate with a dense alignment. The authors always search through the whole RGBD history, thereby implicitly closing loops and also solving the relocalisation challenge. The capabilities of these systems comes at a high cost as they require a strong GPU [6], [14], [15], [16] or even two GPUs [9] to run in real-time.

Since GPUs are expensive, power hungry and difficult to utilize in mobile robots or UAVs, SLAM systems that run on a CPU are very important. Kerl et al. [7] developed DVO-SLAM, which runs in real-time on a CPU and estimates camera motion by minimizing a dense ICP-based and a photometric error. It inserts keyframes on the basis of an entropy measure, includes loop closure and shows very promising results. In order to make it real-time capable, the authors do not process the highest resolution of typically 640×480, thereby sacrificing accuracy. The RGBDTAM [8] SLAM system optimizes a geometric and photometric error and achieves state-of-the-art performance while running on a CPU. Methods based on photometric error are sensitive to illumination changes since the optimization works directly on raw intensity values and are limited to small inter-frame motions as shown in [10], [12].

While photometric and ICP-based systems are well established, direct edge-based methods that estimate camera motion by aligning edges instead, have received a lot of attention in recent years. Edges have some favorable qualities such as a larger convergence basis and more robustness against illumination changes. Tarrio and Pedre [11] work with a monocular camera and try to align edges by searching for the closest edge along the normal direction, which is computationally expensive and error prone. To avoid this search, Kuse and Shen [12] instead pre-compute the distance to the closest edge at each pixel position with a distance transform (DT) [17] and optimize with a subgradient method. In [10], the authors study the influence of various state-of-the-art edge detectors and demonstrate how to efficiently remove outliers to increase accuracy and robustness. The recent Canny-VO [18] proposes alternatives to the DT and incorporates also the edge orientation into their optimization. Also combinations of edge-based methods with photometric error [19] or ICP-based error [20] have been proposed to increase robustness and accuracy.

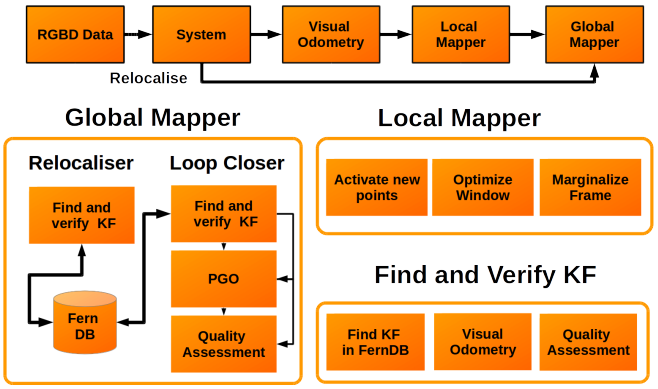

Fig. 2. RESLAM comprises three main components: (i) the VO modul estimates relative camera motion, (ii) the Local Mapper manages keyframes and optimizes a local window and (iii) the global mapper stores a global map and performs loop closure and relocalisation.

To the best of our knowledge no RGBD edge-based SLAM system exists and we are the first to build a complete SLAM pipeline that utilizes edges throughout all stages.

III. Edge-based SLAM (RESLAM)

In this work, we present RESLAM, the first edge-based SLAM system that optimizes the depth of edges, the intrinsic camera parameters, the camera poses within a local window, while also performing loop closure on a global map and offering relocalisation capabilites.

A. System Architecture

Figure 2 shows the main components of RESLAM that run in four different threads: (i) System, (ii) Visual Odometry, (iii) Local Mapper, (iv) Global Mapper. We receive RGBD data and pass it to the system, where we detect edges in a pre-processing step. Then the VO module estimates the relative camera motion between the current frame and a keyframe (KF). We propagate the estimated pose to the local mapper, to decide if we should create a new KF. If a new KF is created we add it to the local window of $\mathcal{K}$ KFs and refine the depth of the edges, the camera poses and the camera intrinsics. After this refinement, the loop closer searches for a loop candidate in the existing database. If a suitable candidate is found, we try to close the loop, otherwise we add the frame to the database. If at some point the system is paused or the camera pose estimation in the VO module failed, the system switches to relocalisation mode and every new RGBD frame is passed to the relocaliser instead of the VO until relocalisation is successful. In the remaining section, we

will explain the camera model and then describe the main components in more detail.

B. Camera Model

In this work, we focus on RGBD sensors from which we receive an RGBD frame at each time step t, which comprises an image I_t and a depth image Z_t. We assume I_t and Z_t to be aligned and synchronized such that at for each pixel position $p = (x, y)$ in I_t the corresponding depth Z is given as $Z = Z_t(p)$.

We define an inverse projection function π^{-1} that computes a 3D point P from a pixel position p and its corresponding depth in the respective camera frame:

$$P = \pi^{-1}(p, Z(p)) = \left(\frac{x - c_x}{f_x} Z, \frac{y - c_y}{f_y} Z, Z \right). \quad (1)$$

Similarly, π is a projection function that maps P to p:

$$p = \pi(P) = \left(\frac{x f_x}{Z} + c_x, \frac{y f_y}{Z} + c_y \right). \quad (2)$$

We further define a relative rigid body motion as transformation matrix T comprising an orthogonal rotation matrix $R_{3\times3} \in SO(3)$ and a translation vector $t_{3\times1} \in \mathbb{R}^3$. A rigid body motion only has 6 degrees of freedom and we utilize the minimal representation as twist coordinates defined by the Lie algebra $se(3)$ associated with the group $SE(3)$ and denote it ξ.

Finally, we define a full warping function τ that computes the reprojection of p_i in frame $\mathcal{F}_i$ to p' frame $\mathcal{F}_j$ under the transformation ξ_{ji}:

$$p' = \tau(\xi_{ji}, p_i, Z_i(p)) = \pi(T_{ji}\pi^{-1}(p_i, Z_i(p_i))). \quad (3)$$

IV. Edge-based Visual Odometry (EBVO)

In each frame, we detect Canny edges [21] and compute the relative motion ξ_{ji} between a KF $\mathcal{F}_i$ and a current frame $\mathcal{F}_j$ by aligning their respective edge detections. To align the edges, we reproject the set of edge pixels with a valid depth $\mathcal{E}_i$ from $\mathcal{F}_i$ to $\mathcal{F}_j$ and try to align with the closest edge detected in $\mathcal{F}_j$. Searching for the closest edge in each iteration is typically very slow and therefore not feasible in practice. Since the edges are detected once and do not change afterwards, we can precompute the Euclidean distance to the closest edge at each pixel position using the distance transform (DT) [17]. We define the reprojected edge distance error E_{p_i}, i.e. the residual, for an edge p_i reprojected from $\mathcal{F}_i$ to $\mathcal{F}_j$ under the transformation ξ_{ji} as:

$$E_{p_i} = \mathcal{D}_j(\tau(\xi_{ji}, p_i, Z_i(p_i))), \quad (4)$$

where $\mathcal{D}_j$ denotes the DT of $\mathcal{F}_j$ and p_i the pixel position of an edge in $\mathcal{F}_i$. We estimate the relative camera motion ξ^*_{ji} that aligns the edge detections of $\mathcal{F}_i$ and $\mathcal{F}_j$ by minimizing (4) for the set of all edges with valid depth $\mathcal{E}_i$.

$$\xi^*_{ji} = \underset{\xi_{ji}}{\operatorname{argmin}} \sum_{p_i \in \mathcal{E}_i} \delta_H E_{p_i}, \quad (5)$$

where δ_H is a Huber weight function that reduces the influence of large residuals. Since edge detections often differ between frames, we drop a potential outlier if $E_{p_i} > \Theta_e$.

We minimize (5) in a coarse-to-fine-scheme with an iteratively re-weighted Levenberg-Marquardt method similar to [2], [10]. Most edge-based methods [10], [12] perform the costly edge detection and DT computation on each pyramid level. This introduces a problem regarding robustness because textures are often smoothed over at lower scale levels. [20] tackles this challenge with an edge transfer method that copies detections from higher levels to the lower ones.

We propose a different method that reduces computational cost while also addressing the robustness issue studied in [20]. Instead of detecting edges on each pyramid level separately, we only detect edges and explicitly compute the DT on the highest level. The edges are then reprojected to lower levels through camera intrinsics, i.e. to go from highest to second highest, we divide the intrinsics by a factor of 2 in the projection function π. We also avoid a costly explicit re-computation of the DT on each pyramid level D_N by downscaling it from a higher resolution level D_{N-1}. We compute $\mathcal{D}_N$ as mean over a 4 pixel patch of $\mathcal{D}_{N-1}$ and additionally divide by 2 since the pixel distances halve between levels:

$$\mathcal{D}_N(x, y) = 0.5 \frac{1}{N_p} \sum_{i=0}^{1} \sum_{j=0}^{1} \mathcal{D}_{N-1}(2x + i, 2y + j), \quad (6)$$

where N_p is the size of the patch at the higher resolution level. The strength of our method to only detect edges on one level becomes apparent, when computationally more expensive machine-learned edge detectors [10] are used.

a) Motion Assumptions: For the optimization of (5), the initialization of ξ_{ji} is a crucial point. Especially when the cameras are farther apart, simply starting with identity can result in slow convergence speeds, convergence to a local minimum or low accuracy. To avoid such problems, we evaluate five different initializations for ξ^0_{ji}: no motion (i) from the last KF $\mathcal{F}_i$ or (ii) from the last frame $\mathcal{F}_j$, or (iii) constant, (iv) double or (v) half motion based on the motion from $\mathcal{F}_{j-1}$ to $\mathcal{F}_{j-2}$ and choose the one with the lowest cost.

b) Alignment Failures: If we cannot align a new frame successfully, which can occur due to a partly or fully covered sensor, very aggressive motions or reflective or sunlit surfaces, where no depth can be estimated, we switch the system to relocalisation mode. We evaluate two criteria to detect alignment failures: (i) if we have less than Θ_{NE} valid edges or (ii) the average reprojection error of inlier edges is above Θ_{reproj}.

V. Local Mapping

In contrast to previous edge-based systems, where only the relative motion to the last KF is estimated [10], [12] or only the most recent frame is optimized with respect to several other KFs [20], we optimize over a window $\mathcal{K}$ of previous KFs. Within $\mathcal{K}$, we jointly refine the depths of all the active edges, the camera poses and the camera intrinsics. Note that we rely on an inverse depth parametrization [22], [23] throughout this work and only refine the depth of points with sufficient baseline and keep it fixed otherwise. The

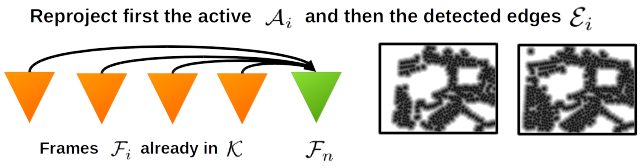

Fig. 3. We activate edges in a two step process: (1) project the set of active residuals $\mathcal{A}_i$ to the new KF (green) and (2) project detected edges with valid depth $\mathcal{E}_i$ to the new KF to active new edges. Each step fills up the distance map $\mathcal{M}_n$.

overall error over the sliding window $\mathcal{K}$ is similar to (5):

$$E_{\mathcal{K}} = \sum_{i \in \mathcal{K}} \sum_{e \in \mathcal{A}_i} \sum_{j \in \mathcal{K}, j \neq i} \delta_H r, \tag{7}$$

where δ_H is again a Huber weight function. We evaluate the residual r with respect to the current state estimates, which unlike (4) also contains the optimized camera intrinsics $\mathcal{C}$ and the inverse depth ρ_i:

$$r = \mathcal{D}_j(p_i'(\xi_i, \xi_j, \rho_i, \mathcal{C})), \tag{8}$$

where p_i' is the reprojection of p_i into $\mathcal{F}_j$ with the refined world poses ξ_i and ξ_j of $\mathcal{F}_i$ and $\mathcal{F}_j$.

We minimize (7) with a Gauss-Newton optimization algorithm similar to [23], [24]:

$$H = J^T W J \qquad b = J^T W r, \tag{9}$$

where $W \in \mathbb{R}^{n \times n}$ is a diagonal weight matrix, $r \in \mathbb{R}^n$ are the stacked residuals and $J \in \mathbb{R}^{n \times d}$ is the Jacobian of r. Since the optimization over a window is computationally expensive, we have to work with and manage a reduced set of active edges.

A. Edge Management

In each KF in the window, we have a set of around 10k - 20k detected edge pixels with valid depth $\mathcal{E}$. Due to the depth initialization provided by the RGBD sensor, this set is significantly larger than the number of points in monocular approaches [23], where only points with already estimated depth are of interest. However, optimizing over all edges from each KF in $\mathcal{K}$ is not possible in real-time on a CPU. Therefore, we follow a strategy inspired by [23], where we maintain a set of active edges $\mathcal{A}_i$ for each KF that we optimize for $\mathcal{K}$. In contrast to [23], we do not limit the size of the active edges but only the number of KFs in $\mathcal{K}$.

Thus, when we add a new KF $\mathcal{F}_n$ to the window, we have to drop an old one. In order to keep the edges well-distributed, we maintain a distance map $\mathcal{M}_n$ for $\mathcal{F}_n$. We first reproject all active edges $\mathcal{A}_i$ from each KF in $\mathcal{K}$ into $\mathcal{F}_n$ (see Fig. 3) and activate an edge if its reprojection p' fulfills the following condtions: $\mathcal{D}_n(p') < \Theta_A$ and $\mathcal{M}_n(p') > \Theta_M$, i.e. p' is close to an edge detection in $\mathcal{F}_n$ and not close to an already activated edge. If an edge is activated, we insert it into the distance map $\mathcal{M}_n$. After reprojecting all active edges $\mathcal{A}_i$, we try to activate new edges detected in previous KFs in $\mathcal{K}$ or $\mathcal{F}_n$. We again reproject and apply the same procedure as before. Figure 3 shows the two-step activation procedure and the progress of $\mathcal{M}_n$.

Seq.	Number of Frames	Created KF	KF After Culling
fr1/desk	573	223	90
fr1/desk2	620	299	69

TABLE I
WE INITIALLY CREATE MANY KFS AND CULL THEM LATER TO REDUCE THE NUMBER OF KFS FOR PGO.

B. Keyframe Management

Maintaining a tractable number of KFs for the global map that are well-distributed is very important for performance and accuracy of the whole system.

1) Keyframe Creation: We follow a strategy, where we initially create many KFs (5 - 15 per second) and cull them later inspired by [4], [23]. Table I shows the number of created KFs and number of KFs after culling, which is around a factor of 2 - 4 depending on the scene. This greatly speeds up later computations over the global map such as loop closure.

We compute three metrics during the optimization process: (i) the mean square optical flow C_{fov} that measures changes in the field of view, (ii) the mean flow without rotation C_{occ} that measures occlusions and (iii) the number of edge reprojections p' below N_{in} and above N_{out} a distance threshold θ_e:

$$C_{fov} = \sqrt{\frac{1}{n} \sum ||p - p'||_2^2} \quad , \quad C_{occ} = \sqrt{\frac{1}{n} \sum ||p - p^t||_2^2}. \tag{10}$$

Based on these metrics, we create a new KF if:

$$C_{fov} + C_{occ} > 1 \quad \text{or} \quad N_{in} < 2N_{out}. \tag{11}$$

2) Keyframe Marginalization: Since the size of $\mathcal{K}$ is limited, we remove and marginalize a KF before adding a new one. If a KF has less than θ_{vis} active points, we marginalize it but if none of the KFs fulfills this constraint, we keep the newest two KFs and compute a distance score s_i over rest to assure that they are well-distributed in 3D space:

$$s_i = \sqrt{d_{i,1}} \sum_{j \in [3,n] \setminus \{i\}} \frac{1}{d_{i,j} + \epsilon}, \tag{12}$$

where $d_{i,j}$ is the Euclidean distance between $\mathcal{F}_i$ and $\mathcal{F}_j$. When we marginalize a KF, we store the relative transformations between it and all the other KFs in $\mathcal{K}$ since these relative transformations will later represent an edge in the pose graph generated during loop closure (see Sec. VI-A). In order to prevent a practically intractable set of active variables, we use the the Schur complement to marginalize old variables. Following [23], [24], we drop any terms that would influence the sparsity pattern of H. Whenever we marginalize a KF, we first marginalize all its active edges, then all the edges that have not been observed in the last two KFs and finally remove all its observed edges completely from the system. We compute the part of the energy E_M that contains all residuals that depend on state variables that should be marginalized and add it to the full edge error $E_{\mathcal{K}}$ in all following optimization and marginalization operations.

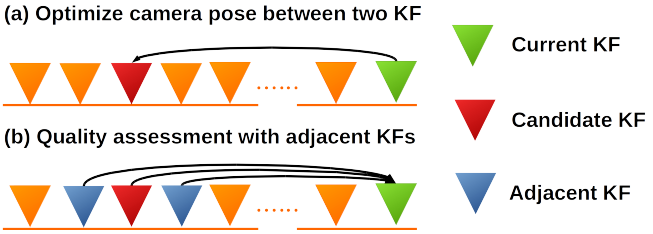

Fig. 4. Whenever we find a candidate KF in the Fern database, (a) we estimate the relative camera motion between the current (green) and the candidate KF (red). (b) Before and after PGO, we assess the quality by reprojecting the edges from the candidate KF and its adjacent KFs (blue) to the current KF.

VI. Global Mapper

Our global mapper keeps a global map of the scene and stores a Fern database of KFs to perform loop closure and relocalisation (see Fig. 2).

A. Loop Closure

Loop closure is a long-standing problem and poses the following challenges: (i) finding loop closure candidates, (ii) estimating the corrected poses and (iii) verifying if the loop closure is correct. We address all three of these challenges in RESLAM and propose an edge-based algorithm to verify if a loop closure is valid. In order to find loop closure candidates, we follow the random-fern-based bag of words approach from Glocker et al. [25], where we compute a Fern descriptor at $N_f = 500$ randomly sampled positions in a down-sampled 40x30 RGB image. [25] is easy to implement and very fast since it is purely based on thresholds of RGB values and does not require a costly feature extraction step like [4]. Note that the positions of the sample points are randomly chosen in the first KF and kept constant throughout the sequence. We then compute similarity scores between the current Fern descriptor and the already stored KFs with the Hamming distance. If we do not find a KF that is similar enough, i.e. its score is above threshold $\Theta_{Hamming}$, we add it to the database and continue. Otherwise we have a candidate KF $\mathcal{F}_{cand}$ and start the loop closing process (see Fig. 2). As depicted in Fig. 4 (a), we start by estimating the relative motion between $\mathcal{F}_{cand}$ (red) and our current KF $\mathcal{F}_{curr}$ (green) with our VO module (see Sec. IV). We then set up a pose graph optimization (PGO) problem, where the measured world poses of the KFs are the nodes and the edges are the estimated relative constraints of the current window $\mathcal{K}$ computed in the Local Mapper also including the ones from marginalized KFs (see V). We define the error for PGO as:

$$e_{ij} = T_{ij}\hat{T}_{wj}^{-1}\hat{T}_{wi}, \tag{13}$$

where T_{ij} is the relative transformation between KF i and j and $\hat{T}_{wi}$ and $\hat{T}_{wj}$ are the corrected world poses. We use the Ceres solver [26] to optimize this pose graph problem and parametrize the rotations as quaternions.

When closing loops, it is of utmost importance that all the detected loops are correct, since even one wrong loop closure can seriously degrade the overall result. We propose an edge-based procedure to quickly verify the validity of the loop closure, which is inspired by the tracking quality measure proposed in [10]. Figure 4 (b) shows the core idea of this procedure, where we reproject a set of verification frames $\mathcal{V}$, which are $\mathcal{F}_{cand}$ (red) and its $N-1$ adjacent frames (blue), to a counting map C in $\mathcal{F}_{curr}$, where we count the reprojections that overlap with an edge detection. To avoid counting coinciding reprojections multiple times, we generate separate counting maps $C_{0,\dots,N-1}$ for each KF:

$$C_i(\tau(\xi_{ci}, p_i)) = 1, \quad \forall p_i \in \mathcal{E}_i, \quad \mathcal{F}_i \in \mathcal{V} \tag{14}$$

where $\mathcal{E}_i$ is the set of valid edge pixels and ξ_{ci} the transformation from $\mathcal{F}_i$ to $\mathcal{F}_{curr}$. We compute the final map C as the element-wise sum of each C_i. Since each $\mathcal{F}_i \in \mathcal{V}$ only contributes one edge overlap, we generate a histogram H of size $N+1$ to count the number of edge overlaps. A candidate is positively verified, if the weighted sum of edge overlaps and is lower than the number of non-overlaps:

$$\sum_{i=1}^{N} w_i H(i) \leq w_0 H(0), \tag{15}$$

where w_i is a weighting factor. In total, we apply this verification procedure twice, once to verify the initial relative motion estimation and once more after PGO to verify the overall result.

B. Relocalisation

In cases, where large parts of a scene do not have sufficient texture for edge detection or depth information is missing, e.g. due to sunlight, or the sensor is fully or partially covered, it is possible that tracking losses occur. Such scenarios require a relocalisation step, where we want to continue from previously seen position. Relocalisation and loop closure are closely related in terms of finding a previously seen scene parts. As depicted in Figure 2, both access the same Fern database. We find and verify a relocalisation the same way as we verify a loop closure. After a successful relocalisation, we remove all the KFs from the current window and restart the system from the relocalised position.

VII. Implementation

We implemented our system in C++ using OpenCV for edge detection, distance transform computation and image in- and output. RESLAM runs in real-time at 30-35 Hz on an Intel i7-4790 desktop computer with 32 GB of RAM. We optimize the relative camera poses in a 3 level coarse-to-fine scheme with maximum resolution of 640×480 px. We set our Huber threshold $\theta_H = 0.3$ and remove potential edge outliers on each respective pyramid level with a distance greater than $\theta_E = 10, 20, 30$ px. Tracking is considered lost if less than $\theta_{NE} = 100$ edges are good or the average reprojection error is above $\Theta_{reproj} = 2.5px$ We marginalize a KF, if less than $\Theta_{vis} = 0.05\%$ of its points are visible in the current KF. A loop closure candidate must be below $\Theta_{Hamming} = 0.25$ and the weights for the counting maps are $\omega_i = [1, 1, 1.25, 1.5]$.

Comparison of the Absolute Trajectory Error $[cm]$

	BF [9]	RGBDTAM [8]	DVO-SLAM [7]	EF [6]	ORB2-SLAM [4]	RGBDSLAM [5]	REVO [10]	Edge+ICP [20]	Our Methods			
Seq.	Direct				Features		Edges		VO	LM	LC	All
fr1/xyz	-	1.0	1.1	1.1	**0.8**	1.3	6.8	1.6	7.4	2.4	2.2	1.1
fr1/desk2	-	**4.2**	4.6	4.8	2.2	4.2	8.2	6.0	7.1	6.3	5.4	4.8
fr2/desk	-	2.7	1.7	7.1	**0.9**	5.7	8.9	9.5	3.5	3.4	3.6	1.9
fr2/xyz	**0.4**	0.7	1.8	1.1	0.8	0.8	1.0	-	0.9	0.5	0.8	0.5
fr3/office	2.2	2.7	3.5	1.7	**1.0**	3.2	11.0	-	13	7.8	4.2	3.5
icl/lr-kt0	**0.6**	-	10.4	0.9	0.8	2.6	24	5.4	6.5	3.2	2.2	2.1
icl/lr-kt1	**0.4**	-	2.9	0.9	1.6	0.8	2.3	0.9	1.3	1.9	2.5	1.7

TABLE II

THE RMSE OF ATE IN $[cm]$ FOR BUNDLEFUSION (BF) [9], RGBDTAM [8], DVO-SLAM [7], ELASTICFUSION (EF) [6],ORB-SLAM2 [4], RGBD-SLAM [5], REVO [10] AND EDGE+ICP [20] COMPARED TO OUR METHOD WITH VO, LOCAL MAPPING (LM), LOOP CLOSURE (LC) AND ALL (VO+LM+LC) ON THE TUM-RGBD [27] AND ICL-NUIM [28] DATASETS.

VIII. RESULTS AND DISCUSSION

We demonstrate the quantitative performance of our method on two standard RGBD benchmarks datasets TUM RGBD [27] and ICL-NUIM [28] covering a large variety of camera motions and scenes. TUM RGBD comprises many sequences recorded with a MS Kinect at 30 Hz with 100 Hz ground truth poses from a motion capture system. ICL-NUIM is a synthetic dataset that is completely noise-free and offers perfect ground truth. From the ICL-NUIM we choose two typical indoor sequences that also contain poorly textured walls. For both datasets we evaluate the root mean squared error (RMSE) of the translational component of the absolute trajectory error (ATE) at time step i [27]:

$$ATE_i = Q_i^{-1} S P_i, \tag{16}$$

where S is a rigid body transformation that aligns the ground truth Q and with the estimated trajectory P.

A. Benchmark Datasets

We compare our system to 6 SLAM approaches: (i) Bundle Fusion [9] that combines indirect and direct principles, three direct methods (ii) RGBDTAM [8], (iii) DVO SLAM [7], (iv) ElasticFusion [6], two feature-based systems (v) ORB-SLAM2 [4] and (vi) RGBDSLAM [5] and finally two edge-based VO systems: REVO [10] and the edge- and ICP-based [20]. Please note, that as suggested in [4] we accounted for the depth bias in the freiburg2 dataset. To demonstrate the influence of each component, we present four different evaluations of our system: (i) VO, which is similar to REVO [10], (ii) VO + Local Mapping (LM) (iii) VO + Loop Closure (LC) and finally (iv) all the modules combined (VO + LM + LC).

The results of the ATE on the benchmark datasets in Table II clearly show that all the state-of-the-art systems are very close together in terms of accuracy. Since the ATE is given in cm, the differences are often in the mm-range, which is probably below the groundtruth accuracy. RESLAM clearly shows improvements over other direct edge-based VO systems [10], [20] on nearly all the datasets. The addition of a sliding window optimization already improves accuracy compared to methods only taking the last KF into account. Interestingly, also the combined method using edges and ICP terms is outperformed by a purely edge-based method.

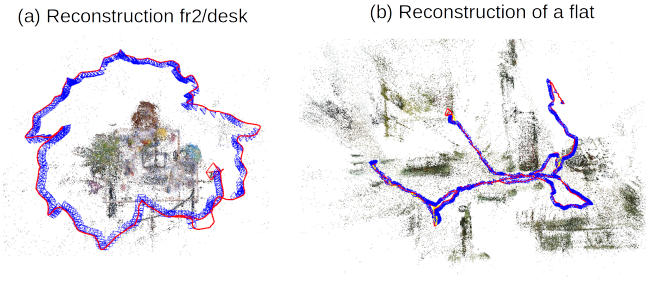

Fig. 5. (a) Reconstruction and trajectory of the fr2/desk dataset. (b) Reconstruction and trajectory of a flat with over 5000 images.

Compared to BundleFusion [9] or ElasticFusion [6], our method does not require a GPU but runs in real-time on a CPU, while achieving competitive results.

B. Qualitative results

We additionally want to show some qualitative examples to demonstrate the performance of our method. In Figure 5(a), we show the trajectory and sparse reconstruction of the fr2/desk sequence. The trajectory is very accurately estimated and also the pointcloud is consistent. Figure 1(b) depicts an example trajectory recorded with our own Orbbec Astra Pro RGBD sensor and the sparse reconstructed pointcloud. Another of our own RGBD recordings is shown in Figure 5(b), where we record a whole flat with over 5000 frames.

IX. CONCLUSIONS

We presented RESLAM, a novel edge-based SLAM system that utilizes edges for VO, local mapping and loop closure/relocalisation verification. Edge-based algorithms are very interesting due to their favorable properties such as larger convergence basin and fast optimization speed. We also demonstrate that we can compete with many state-of-the-art methods that require a strong GPU. In contrast, our method runs in real-time on a CPU, which is essential for mobile robotics applications and navigation tasks. RESLAM is available as open-source[1] to encourage further research in the area of edge-based methods.

ACKNOWLEDGMENTS

This work was funded by the EU Horizon 2020 grant No 730294 *SLIM* and by the EIT Raw Materials grant No 18004 *HoloMine*.

REFERENCES

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[2] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849.

[3] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics (T-RO)*, vol. 30, no. 1, pp. 177–187, 2014.

[6] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1697–1716, 2016.

[7] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[8] A. Concha and J. Civera, "Rgbdtam: A cost-effective and accurate rgb-d tracking and mapping system," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6756–6763.

[9] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics (TOG)*, 2017.

[10] F. Schenk and F. Fraundorfer, "Robust edge-based visual odometry using machine learned edges," in *International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1297 – 1304.

[11] J. Jose Tarrio and S. Pedre, "Realtime edge-based visual odometry for a monocular camera," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 702–710.

[12] M. P. Kuse and S. Shen, "Robust camera motion estimation using direct edge alignment and sub-gradient method," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics (T-RO)*, vol. 32, no. 6, pp. 1309–1332, 2016.

[14] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[15] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 500–516.

[16] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research (IJRR)*, vol. 34, no. 4-5, pp. 598–626, 2015.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, pp. 415–428, 2012.

[18] Y. Zhou, H. Li, and L. Kneip, "Canny-vo: Visual odometry with rgb-d cameras based on geometric 3-d–2-d edge alignment," *Transactions on Robotics (T-RO)*, vol. 35, no. 1, pp. 184–199, 2019.

[19] X. Wang, D. Wei, M. Zhou, R. Li, H. Zha, and C. Beijing, "Edge enhanced direct visual odometry," in *British Machine Vision Conference (BMVC)*, 2016.

[20] F. Schenk and F. Fraundorfer, "Combining edge images and depth maps for robust visual odometry," in *British Machine Vision Conference (BMVC)*, 2017.

[21] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, no. 6, pp. 679–698, 1986.

[22] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE Transactions on Robotics (T-RO)*, vol. 24, no. 5, pp. 932–945, 2008.

[23] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 3, pp. 611–625, 2018.

[24] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research (IJRR)*, vol. 34, no. 3, pp. 314–334, 2015.

[25] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 21, no. 5, pp. 571–583, 2015.

[26] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[28] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1524–1531.