

# Robust Edge-based Visual Odometry using Machine-Learned Edges

Fabian Schenk and Friedrich Fraundorfer

**Abstract**—In this work, we present a real-time robust edge-based visual odometry framework for RGBD sensors (REVO). Even though our method is independent of the edge detection algorithm, we show that the use of state-of-the-art machine-learned edges gives significant improvements in terms of robustness and accuracy compared to standard edge detection methods. In contrast to approaches that heavily rely on the photo-consistency assumption, edges are less influenced by lighting changes and the sparse edge representation offers a larger convergence basin while the pose estimates are also very fast to compute. Further, we introduce a measure for tracking quality, which we use to determine when to insert a new key frame. We show the feasibility of our system on real-world datasets and extensively evaluate on standard benchmark sequences to demonstrate the performance in a wide variety of scenes and camera motions. Our framework runs in real-time on the CPU of a laptop computer and is available online.

## I. INTRODUCTION

One of the most active research areas in computer vision and robotics is camera motion estimation or visual odometry (VO) [1], [2]. This is mostly due to the many practical applications in fields of robotics such as autonomous driving, UAV navigation, augmented and virtual reality as well as 3D reconstruction, where camera motion and the structure of the scene are of great importance.

Traditional monocular cameras cannot record the geometric structure of the scene due to an unknown scale parameter introduced by the projective nature of the system. In contrast, RGBD sensors offer great benefits for VO because they jointly capture a scene’s geometry as a depth image while recording a scene’s texture as an RGB image in real-time. Since the introduction of inexpensive RGBD sensors such as the MS Kinect, Asus Xtion or Orbbec Astra Pro, research in the field of RGBD VO has rapidly evolved.

For a long time, VO research was dominated by feature-based (indirect) methods, which typically extract features, find correspondences and track them through images to estimate the relative motion between them [3], [4], [5]. In the past few years, direct approaches that estimate the camera motion directly from image data, thereby omitting the need for a robust correspondence matching step, have become very popular [6], [7]. The use of the complete image information usually results in better accuracy than rather sparse, feature-based methods but requires a much smaller inter-frame motion.

Edge-based methods can be classified as a crossover between indirect and direct principles, where the edges are the

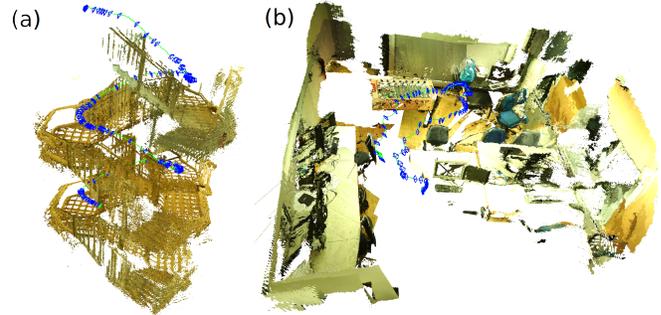


Fig. 1: Two real-world reconstructions performed with REVO, where the trajectory is depicted in green and the respective key frames in blue. (a) shows a staircase over several floors and (b) is a typical university office room.

features but motion estimation works without correspondence computation [8], [9], [10]. Performance of edge-based VO highly depends on the localization accuracy and repeatability of the edges, which has not been studied in any of these previous works.

In this work we present REVO, an edge-based VO framework that can track and reconstruct difficult scenes (see Fig. 1). We introduce several improvements over existing edge-based methods and show that the use of state-of-the-art edge detectors further raises accuracy. We present real-world trajectories and reconstructions as well as an extensive evaluation on the standard TUM RGBD benchmark dataset [11] covering a large variety of scenes and camera motions to show that REVO performs comparably or better than various state-of-the-art methods. The main contributions of this work are:

- An in-depth study of various edge detectors and their influence on accuracy
- A histogram-based tracking quality measure to decide when to insert a new key frame
- Experiments that demonstrate how to further raise the accuracy using edge filters and constant motion assumption
- A very fast implementation that runs in real-time on a CPU while being more accurate than previous edge-based methods<sup>1</sup>

## II. RELATED WORK

In this section, we give a short introduction to edge detection and provide an overview of the most important publications in the field of VO with the focus on RGBD systems.

All authors are with the Institute of Computer Graphics and Vision (ICG), Graz University of Technology, Styria, Austria  
{schenk, fraundorfer}@icg.tugraz.at

<sup>1</sup>Code available: <https://www.tugraz.at/index.php?id=22399>

### A. Edge Detection

While humans can easily find meaningful or natural edges such as object boundaries, this is still a very challenging task for a computer and well-known detectors such as Canny [12] only detect edges based high-intensity gradients. The BSDS500 dataset [13] comprises 500 natural images with manually annotated edges and is heavily used for training of learning-based edge detection techniques [14], [15]. Dollar and Zitnick [14] introduced Structured Edges (SE) that delivered state-of-the-art results in terms of meaningful edge detection while being relatively fast (around 12 fps on a CPU). It comprises (i) a large number of manually designed features, (ii) the fusion of multi-scale responses and (iii) the incorporation of structural information. Recently, Xie and Tu [15] demonstrated the capabilities of CNNs with their Holistically-Nested Edge Detection (HED) that combines (i) holistic image training and prediction and (2) nested multi-scale feature learning performing deep layer supervision to guide early classification results. They proposed an RGB model trained on BSDS500 [13] and an RGBD model with an additional depth input channel trained on NYUv2 [16]. Even though the inference speed of 2.5 fps on a GPU is rather slow, in the next few years specialized CNN hard- and software will most likely drastically increase the speed.

### B. Visual Odometry (VO)

Traditional feature-based (indirect) methods extract features (e.g., SIFT, SURF, ORB), find correspondences between images and track them over a sequence to estimate the camera motion. Such feature-based approaches only use few parts of the image, i.e. discarding most of the image information. Dryanovski et al. [3] proposed a real-time VO approach that used a consistent model updated dynamically with a Kalman filter upon new observations. RGBD-SLAM [4] is a feature-based (indirect) mapping system that uses an environment measurement model to validate the transformations estimated by feature correspondences and the ICP algorithm [17]. It also performs pose graph optimization and loop closure to refine the trajectory estimates. Recently, Mur-Artal and Tardos [5] released ORB-SLAM2, an open-source SLAM system that utilizes ORB features for tracking, mapping and loop closing while running on a single CPU. The system is able to handle monocular (RGB or intensity), RGBD and stereo data.

Another widely used approach is to register 3D point clouds directly instead of aligning images, which is usually done by the iterative closest point (ICP) [17] algorithm. KinectFusion [18] directly uses the depth images from an RGBD sensor to estimate frame-to-model motion and was later extended to a full SLAM system by Whelan et al. [19].

Direct (featureless) approaches estimate the camera motion directly from image data, thereby leaving out feature extraction and robust correspondence matching. However, they are typically limited to small inter-frame motions [20], which can be circumvented only up to certain degree by an image pyramid. Most of these approaches heavily rely on the photo-consistency assumption [6], [20], [7], which

makes them especially prone to changing lighting conditions. Kerl et al. [6] introduced DVO, a probabilistic formulation for direct motion estimation from RGBD data with a robust sensor model based on t-distribution and a special way to deal with frames not containing sufficient texture or structure. They combined the photo-consistency error with a variant of ICP [17]. While this approach is quite robust, it is limited to small inter-frame motions.

Edge-based methods are a crossover between indirect and direct approaches, where the features are the edges but camera motion estimation does not require correspondences. The general idea is to minimize the distance between a frame's edges and the reprojected edges in another frame. Due to difficulties in terms of robust optimization, edge-detection and computational capabilities, edge-based algorithms were not extensively applied for model-free VO in the past. Now, with faster hardware and new optimization strategies, such methods have become feasible again. Tarrío and Pedre [8] proposed an edge-based VO for a monocular camera, where they try to match edges by searching along the normal direction. This matching step can be greatly accelerated by pre-computing the distance transform (DT) [21] in a frame, which gives the distance to the closest edge pixel at each pixel (see Fig. 2). This idea was adapted by Kuse and Shen [9] in their RGBD direct edge-alignment (D-EA), where they estimate camera motion with a sub-gradient based optimization. Wang et al. [10] also compute the DT and extend the standard photometric error by an additional edge term.

In contrast to previous edge-based VO approaches [9], [8], [10] that utilize the well-known Canny [12] edge detector, we study the influence of different edge detectors on accuracy [14], [15]. We also show that accuracy and optimization speed can be increased with an edge filter that removes large outliers and constant motion initialization. Further, we propose a measure to assess tracking quality and to know when to insert a new key frame. Finally, we introduce REVO, a ready-to-use fast edge-based VO framework that runs on the CPU of a laptop computer in real-time.

## III. ROBUST EDGE-BASED VISUAL ODOMETRY (REVO)

In this section, we describe our edge-based VO RGBD method, which we refer to as REVO throughout the paper.

### A. Notations and Definitions

At each time step  $t$  we receive a frame  $F_t$  that consists of an RGB image  $I_t$  and a depth image  $Z_t$ , which are aligned and synchronized. The inverse projection function  $\pi^{-1}$  computes the 3D point  $P = (X, Y, Z)$  in the respective camera frame from the pixel coordinates  $p = (x, y)$  and the corresponding depth value  $Z = Z_t(p)$ :

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

where  $f_x, f_y$  are the focal lengths and  $c_x, c_y$  the optical centers as defined by a standard pinhole camera model. The

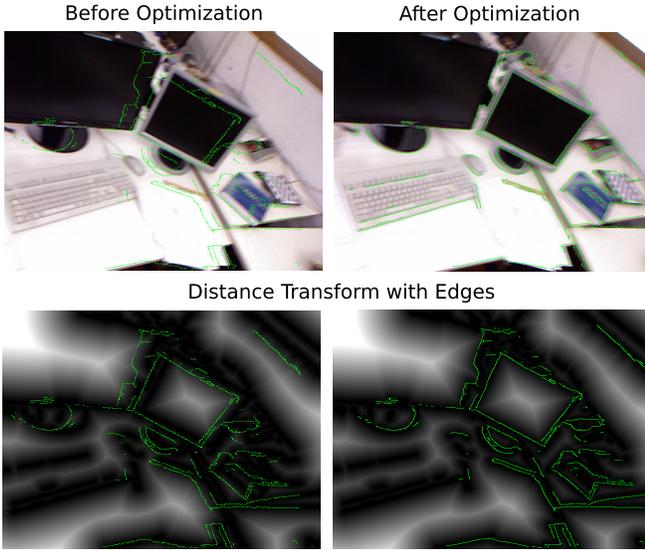


Fig. 2: Reprojected edges in the key frame before and after optimization. The edge residuals are evaluated on the distance transform.

projection function  $\pi$  is given as:

$$p = \pi(P) = \left( \frac{Xf_x}{Z} + c_x, \frac{Yf_y}{Z} + c_y \right). \quad (2)$$

A rigid body motion  $g \in SE(3)$  between two cameras comprises a rotation described by an orthogonal  $3 \times 3$  matrix  $R \in SO(3)$  and a translation described by a  $3 \times 1$  vector  $t \in \mathbb{R}^3$ . The point  $P$  can then be transformed by  $g$  as:

$$T(g, P) = R \cdot P + t. \quad (3)$$

As  $g$  only has 6 degrees of freedom, we use a minimal representation as twist coordinates  $\xi \in \mathfrak{se}(3)$ :

$$\xi = (v_1, v_2, v_3, \omega_1, \omega_2, \omega_3)^T \in \mathbb{R}^6, \quad (4)$$

where  $v_1, v_2, v_3$  is the linear velocity and  $\omega_1, \omega_2, \omega_3$  the angular velocity. The Lie algebra  $\mathfrak{se}(3)$  can be mapped to the Lie group  $SE(3)$  by the exponential map as  $g(\xi) = \exp_{\mathfrak{se}(3)}(\xi)$  with the inverse being  $\xi(g) = \log_{SE(3)}(g)$ . The transformation from a frame  $i$  to frame  $j$  is defined as  $\xi_{ji}$  and the concatenation of two transformations is:

$$\xi_{ij} = \xi_{ik}\xi_{kj} = \log_{SE(3)}(\exp_{\mathfrak{se}(3)}(\xi_{ik}) \cdot \exp_{\mathfrak{se}(3)}(\xi_{kj})) \quad (5)$$

We define the full warping function  $\tau$  that reprojects  $p$  to  $p'$  under the transformation  $\xi$  as:

$$p' = \tau(\xi, p) = \pi(T(g(\xi), \pi^{-1}(P, Z))). \quad (6)$$

### B. Edge-based Camera Motion Estimation

In REVO, we have to detect edges  $E_t$  in each frame  $F_t$  from the intensity  $I_t$  as:

$$E_t = E(I_t), \quad (7)$$

with  $E(\cdot)$  being an arbitrary edge detector. The detected edges are subsequently used to estimate the relative rigid motion  $\xi_{kc}$  from a current frame  $F_c$  to a key frame  $F_k$  by

minimizing the sum over all edge distance errors or edge residuals  $r$  (see Fig. 2):

$$\xi_{kc}^* = \operatorname{argmin}_{\xi_{kc}} \sum \delta_H(r)r^2, \quad (8)$$

where  $\delta_H(r)$  is the Huber weight function given as:

$$\delta_H(r) = \begin{cases} 1 & r \leq \Theta_H \\ \frac{\Theta_H}{r} & r > \Theta_H \end{cases} \quad (9)$$

We reproject the edges of  $F_c$  into  $F_k$  and minimize the Euclidean distance to the closest edge pixel, which completely avoids any type correspondence computation. An exhaustive search for the closest edge pixel in each iteration is typically very slow, thus we pre-compute the Euclidean distance to the closest edge from the edges in the key frame  $E_k$  beforehand using the distance transform (DT) [21]. We then only evaluate the edge distance error  $r$  at the reprojected pixel positions in  $F_k$  in each iteration (see Fig. 2):

$$r = DT_k(\tau(\xi_{kc}, p)), \quad \forall p \in \Omega_{E_c} \quad (10)$$

where  $DT_k$  denotes the DT computed from  $E_k$ ,  $\Omega_{E_c}$  is the set of edges with valid depth in  $F_c$  and  $p$  the corresponding pixel position  $(x, y)$ .

We optimize Eq. (8) using an iteratively re-weighted Levenberg-Marquardt method in a left compositional formulation similar to [7]. Even though only optimizing on full image resolution gives good results on most of the dataset, in practice we found that a coarse-to-fine scheme and initialization according to a constant motion assumption increase convergence basin and speed. Further, we utilize a key frame-based formulation to increase accuracy.

*a) Constant motion assumption:* In VO, a common question is how to initialize camera pose estimation as simply starting with identity is problematic when the frames are farther apart. Kuse and Shen [9] start with the previous estimate and set the transformation to identity when a new key frame is added. In our method, we set the initialization  $\xi_{kc_{init}}$  for the current frame as the previous estimate  $\xi_{kc-1}$  times the relative motion  $\xi_{c-2c-1}$  between the two previous frames  $F_{c-2}$  and  $F_{c-1}$ . The initial transformation between the current frame  $F_c$  and the key frame  $F_k$  is then given as  $\xi_{kc_{init}} = \xi_{kc-1}\xi_{c-2c-1}$ . This initialization typically starts the camera motion estimation very close to the final estimate, which in turn reduces convergence time and in practice avoids converging to a local minimum.

*b) Key frame based VO:* We use a key frame-based setup, where we compute the relative camera motion  $\xi_{kc}$  from the current frame  $F_c$  to the key frame  $F_k$  (see Eq. 8 and Fig. 3 (I)). Optimization on the key frame has the great advantage that the costly DT computation on all pyramid levels has to be performed only when a new key frame is inserted. We follow the policy that when the tracking quality gets poor, we take the last well-tracked frame (the previous one) as key frame and optimize the relative camera motion again (see Fig. 3 (I)-(III)). In contrast, Kuse and Shen [9] reproject edge pixels from the key frame to the current frame and have to compute the DT for every single frame. By

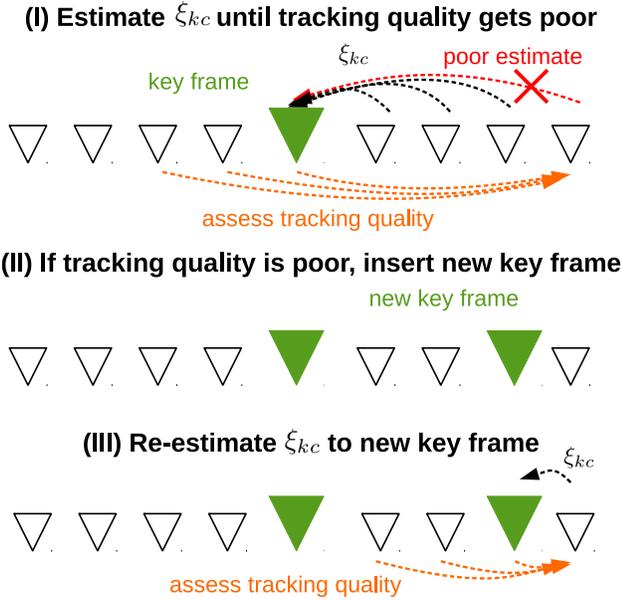


Fig. 3: (I) We estimate the relative camera motion  $\xi_{kc}$  from the current to the key frame until the tracking quality gets poor. (II) If the tracking quality is poor, we set the last frame with good tracking quality (the previous one) as new key frame and (III) re-estimate  $\xi_{kc}$ .

always setting the previous frame as key frame, REVO can also be used in frame-to-frame VO mode, which considerably raises the computational effort as the DT has to be computed every frame. Nevertheless, the system still runs in real-time.

### C. Tracking Quality Measure

A common challenge in VO is to assess tracking quality and to know when to insert a new key frame. Many approaches simply take every  $n$ th frame [9] or insert a new key frame after a particular threshold is reached such as relative angle or distance [7], number of certain features [5] or a certain ratio [6]. Some [5] even follow the policy to insert many key frames and cull them later. In our case, we have to compute the DT for each newly inserted key frame, which is costly (around 15 ms for all pyramid levels) compared to tracking. Hence, we want to use as few key frames as possible.

We propose an edge-based tracking quality measure that is also well suited to determine when to insert a new key frame. We reproject the edges of  $N$  previously tracked frames (see Fig. 3 (I)) into the current one and count the number of reprojections at each pixel position in a counting map  $M$ . To avoid multiple countings of coinciding reprojections due to rounding errors, we generate  $N$  counting maps  $M_0, \dots, M_{N-1}$ :

$$M_i(\tau(\xi_{ci}, p)) = 1 \quad \forall p \in \Omega_{E_i}, \quad i = [0, \dots, N-1], \quad (11)$$

where  $\Omega_{E_i}$  is the set of valid edge pixels. The final map  $M$  is then  $M = M_0 \oplus M_1 \oplus \dots \oplus M_{N-1}$ , where  $\oplus$  is the element-wise sum. The values at each pixel position are in the range of  $[0, 1, \dots, N]$ , whereas a value of  $N$  indicates that edge pixels from all previous frames are reprojected to a particular

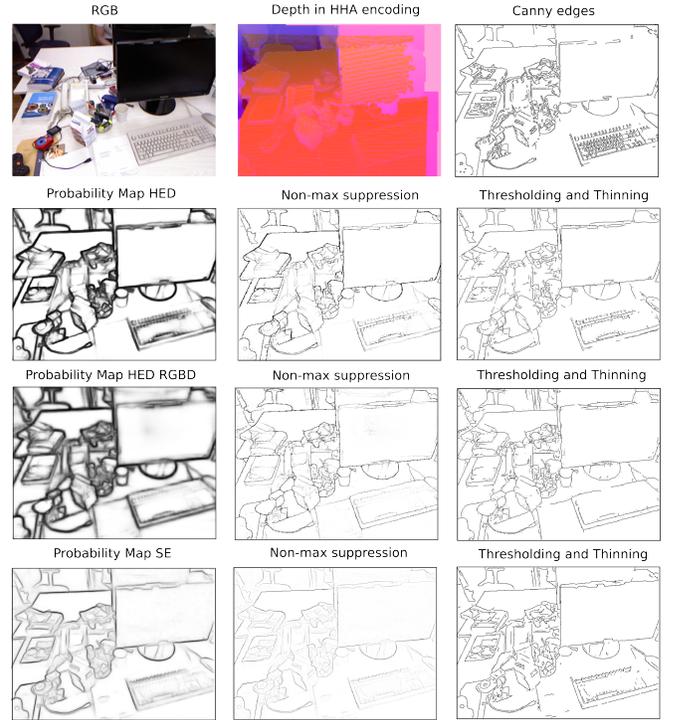


Fig. 4: The preprocessing steps necessary to convert the probability maps given by SE and HED into edge images.

position and 0 means none are reprojected to it. Intuitively, the tracking quality is good when the reprojections strongly overlap with the edges in the current frame. To measure the overlap, we generate an overlap histogram  $H$  of size  $N + 1$  by evaluating  $M$  at the positions of the edge pixels  $p$  in the current frame:

$$H(M(p)) = H(M(p)) + 1, \quad \forall p \in \Omega_{E_c}, \quad (12)$$

where  $\Omega_{E_c}$  is the set of valid edge pixels. Good tracking quality can be assumed if  $H(N-1)$  and  $H(N)$  are high and the number of non-overlaps  $H(0)$  is low. In our method, we insert a key frame when the weighted sum of edge overlaps is lower than the number of non-overlaps:

$$\sum_{i=1}^N w_i H(i) \leq w_0 H(0), \quad (13)$$

where  $w_i$  is a weighting factor. Such a tracking measure also has the benefit that it triggers new key frame insertion when the edges have changed too much, e.g. strong lighting changes or new parts of the scene are seen.

### D. Edge Detection

Up to now, the choice of the edge detector  $E$  in Eq. (7) is still an open question. Parts of a scene that show a high concentration of features or edges such as a keyboard on a white desk or a poster on wall, typically introduce bias in the motion estimates due to uneven spatial distribution. Thus, for feature- or edge-based methods, a good distribution over the image, high repeatability as well as localization accuracy are of great importance. As argued by [15], deep learned

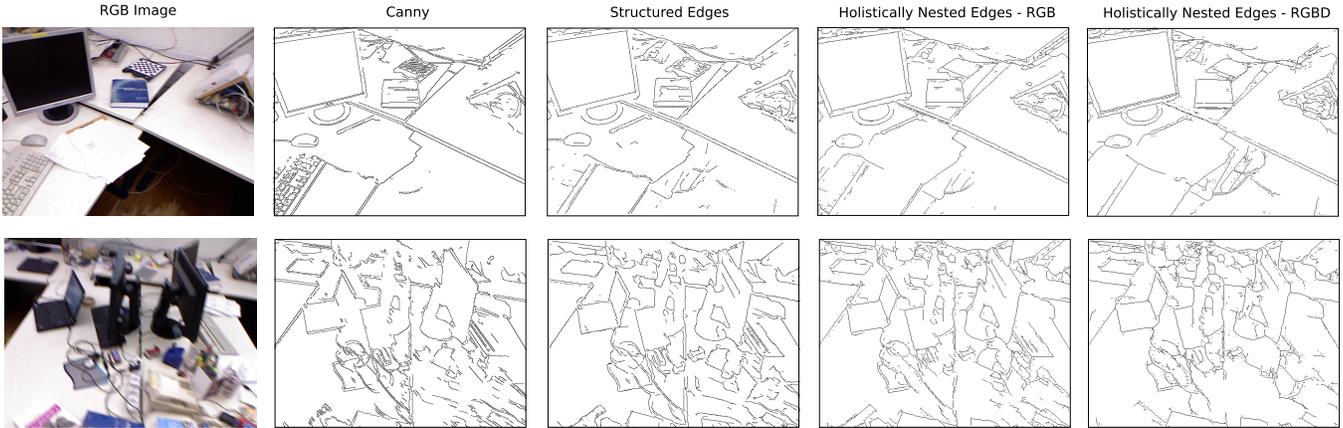


Fig. 5: We show the results of the different edge detection algorithms on two images from *fr1/desk*. The first row is a rather sharp image where all edge detectors give clear edges. In the second row, we see the edges computed from a blurred image, where especially Canny [12] shows many double detections at blurry edges, while the learning-based methods [14], [15] still deliver clear contours.

features favor object boundaries and omit weak edges, in turn reducing potential clutter in a scene.

In this work, we study various edge detection algorithms [12], [14], [15] depicted in Figure 5. The machine-learning methods [14], [15] give a probability map of the edges. Thus, we perform non-max suppression, followed by thresholding at a value  $\Theta$  above which we consider a pixel an edge and finally apply edge-thinning (see Fig. 4). We denote these thresholds as  $\Theta_{SE}$  for [14],  $\Theta_{HED}$  for the RGB and  $\Theta_{HEDD}$  for the RGBD version of [15].

Edge-detections can vary between frames, thus we typically face the challenge of outliers, i.e. edges that were not seen before or can be no longer seen. The Huber weighting used in Eq. (8) reduces the influence of outliers, but very large residuals can result in either longer convergence time or poor estimates. Thus, for edge-based VO the repeatability is of grave importance, i.e. can we detect the same edges in consecutive frames. To determine, how well the various edge detectors work in terms of repeatability, we use the *fr1/xyz* dataset of the TUM RGBD benchmark [11]. We first detect edges in each frame and reproject them to next frame using the provided ground truth information. We then compute residuals by evaluating the DT at the reprojected locations (see Eq. (10)) and filter when the distance to the closest edge is larger than 30 px. Figure 6 depicts the number of filtered residuals for Canny [12], SE [14] and HED [15]. For Canny many edges with a large distance (outliers) are removed, while the machine-learned SE and HED detectors show very constant detections and therefore comparably few outliers. Nevertheless, we apply the edge filter for all detectors and remove large residuals at each optimization step for all pyramid levels.

### E. Implementation

We implemented the complete REVO system to run in real-time on a laptop CPU using C++ and OpenCV (only the optional graphical viewer runs on the GPU). Table I shows the average timings over a common TUM RGBD sequence for a laptop (i7-6500U, 8 Gb RAM, NVidia GTX 940m) and a desktop computer (i7-4790, 32 Gb RAM, NVidia GTX

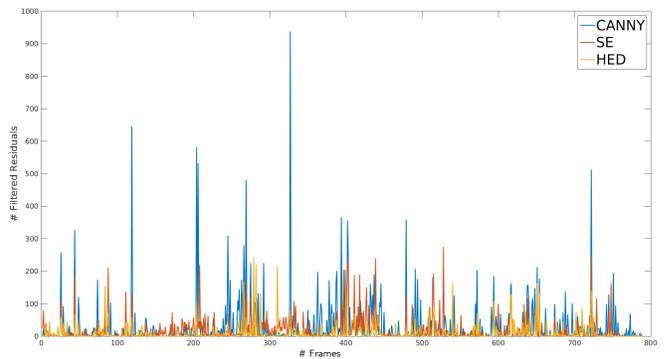


Fig. 6: The number of filtered residuals for Canny [12], SE [14] and HED [15] over the *fr1/xyz* sequence. SE and HED show a much lower number, which indicates a more constant edge detection.

| Timings        | Every Frame<br>Tracking | Key frame only<br>Distance Transform | Average<br>over 573 frames |
|----------------|-------------------------|--------------------------------------|----------------------------|
| <b>Laptop</b>  | 17.5362 ms              | 18.1115 ms                           | 19.3343 ms                 |
| <b>Desktop</b> | 9.06408 ms              | 14.8205 ms                           | 10.4527 ms                 |

TABLE I: Average timings on a laptop and desktop computer for the *fr1/desk* dataset with 573 frames and 57 key frames.

970), where tracking has to be performed each frame and the DT is computed only when a key frame is added. Note that tracking time increases with the number of edge pixels.

Our coarse-to-fine scheme is realized with 3 pyramid levels at a maximum resolution of  $640 \times 480$  px with edge filter thresholds  $\Theta_E = 10, 20$  and  $30$  px and Huber weighting  $\Theta_H = 0.3$ . To get a good estimate for tracking accuracy while retaining real-time capabilities, we empirically found that  $N = 3$  and  $w_i = [1, 1, 1.25, 1.5]$ ,  $i \in [0, N]$  are reasonable choices (see Fig. 3).

For Canny [12], we set an upper threshold of 150, a lower threshold 100 and kernel size of 3. We use the pre-trained models of SE and HED provided online without any additional training. Experimentally we found, that cut-off values for the probability maps of  $\Theta_{SE} = 0.05$ ,  $\Theta_{HED} = 0.2$  and  $\Theta_{HEDD} = 0.25$  give good results.

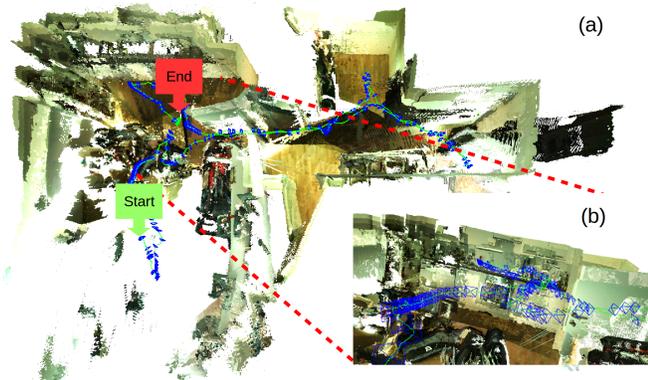


Fig. 7: The estimated trajectory (green) and key frames (blue) through a whole flat in top-down view (a) and in a zoomed version (b). In (b) it can be seen that the drift in height is very low.

#### IV. RESULTS AND DISCUSSION

To demonstrate the capabilities of our method, we perform challenging real-world experiments with our own RGBD sensor and show the trajectories and the respective reconstructions. For quantitative evaluation, we use the TUM RGBD dataset [11], which has become a standard for VO evaluation due to its large variety of scenes and camera motions. It provides RGBD sequences recorded with a Microsoft Kinect at 30 Hz with highly accurate ground truth poses.

##### A. Real-World Experiments

For qualitative evaluation and to prove the feasibility of our method, we show three trajectories and reconstructions of real-world scenes recorded with an Orbbec Astra Pro RGBD sensor. Figure 1 (a) shows the challenging sequence of a staircase with a trajectory over several stories and the corresponding sparse (edge) reconstruction directly from our online viewer. A common university student room is depicted in Figure 1 (b), where we additionally show a dense reconstruction also generated by our viewer. Finally, a long trajectory through a whole flat is shown in Figure 7 (a) in a top-down view with a zoomed version of start and end point of the trajectory depicted in (b). The drift in all sequences is very low and in Figure 7 (b) start and end point are at the same height, implying that there is hardly any drift in z-direction.

##### B. Results on the TUM RGBD Benchmark

We compare REVO in key frame-based and frame-to-frame VO configuration using the various edge detection algorithms introduced in Section III-D to four state-of-the-art approaches that can handle RGBD data, namely DVO [6], ORB-SLAM2 [5], RGBD-SLAM [4] and D-EA [9]. We run DVO in the standard weighted configuration with 4 pyramid levels. For the evaluation of RGBD-SLAM we take the trajectories available on the TUM benchmark [11] website. In ORB-SLAM2 we set `mbOnlyTracking = true` such that it only performs VO instead of the full SLAM pipeline. For D-EA we use the code available online without any modifications.

To measure the local accuracy of visual odometry methods, Sturm et al. [11] proposed the relative pose error (RPE) and the absolute trajectory error (ATE). The RPE measures the drift over a fixed time interval  $\Delta t$  between a set of poses  $Q$  from the ground truth trajectory and a set of poses  $P$  from the estimated trajectory and at time step  $i$  is defined as:

$$RPE_i = (Q_i^{-1}Q_{i+\Delta t})^{-1}(P_i^{-1}P_{i+\Delta t}), \quad (14)$$

where  $\Delta t$  is the time distance between poses. The ATE at a time step  $i$  is given as:

$$ATE_i = Q_i^{-1}SP_i, \quad (15)$$

where  $Q$  and  $P$  are aligned by a rigid body transformation  $S$ . As suggested by Sturm et al. [11], we evaluate the root mean squared error (RMSE) of the translational component of the RPE and ATE.

We present our results in Table II with the ATE in  $[m]$ , the RPE in  $[\frac{m}{f}]$  with an inter-frame distance of  $\Delta t = \frac{1}{30}s = 1f$ , i.e. consecutive frames, and finally the RPE in  $[\frac{m}{s}]$ , i.e. the drift over one second ( $\Delta t = 1s$ ). Please note that in [9],  $\delta$  is given in frames and their  $\delta = 1$  and  $\delta = 20$  correspond to  $\Delta t = \frac{1}{30}s$  and  $\Delta t = \frac{20}{30}s$ . DVO and ORB-SLAM2 perform frame-to-frame VO, thus their results should be compared to  $REVO_{FF}$ , while D-EA and RGBD-SLAM should be compared to  $REVO_{KF}$ .

Even though RGBD-SLAM [4] performs pose graph optimization and loop closure to refine the overall trajectory it does not have the best ATE score on all the datasets. When evaluating the RPE between consecutive frames, it is common that frame-to-frame VO performs better than key frame-based VO because we explicitly optimize the pose between consecutive frames, while it is the other way around for the drift over one second (see Tab. II). On most of the datasets, all the VO approaches perform better than RGBD-SLAM most likely because its pose graph optimization tries to distribute the error caused by drift over the whole trajectory to reduce the overall error. REVO performs extremely well on all datasets and is better or on par with the state-of-the-art approaches [5], [6], [9]. It is very encouraging to see that REVO's performance is similar to recently released indirect approach ORB-SLAM2.

REVO greatly outperforms the other edge-based approach D-EA on all datasets and scores. We attribute this to quality-based insertion of key frames and the good initialization of our optimization. Contrary to D-EA [9], we can also utilize the full resolution of  $640 \times 480$  while still retaining real-time capabilities. Further, REVO variations outperform DVO [6] on all datasets and scores, which we mostly attribute to the edges being more stable under changing lighting conditions.

When analyzing the 21 scores (3 measures of 7 datasets) for each VO mode (key frame-based, frame-to-frame) separately, we found that machine-learned edges performed best in 16 out of 21 cases for both modes. This suggests that machine-learned edges are a good way to further increase robustness and accuracy of edge-based methods, even though they were not specifically trained for the task of VO.

| Comparison of the Absolute Trajectory Error (ATE) [m]    |          |                 |                 |          |                     |                     |                   |                   |                   |                   |                    |                    |
|--|----------|-----------------|-----------------|----------|---------------------|---------------------|-------------------|-------------------|-------------------|-------------------|--------------------|--------------------|
| Seq.   | DVO [6]  | ORB2 [5]        | SLAM [4]        | D-EA [9] | REVO Canny [12]     |                     | REVO SE [14]      |                   | REVO HED [15]     |                   |                    |                    |
|  | ICP+Gray | Features        |                 | Canny    | Canny <sub>KF</sub> | Canny <sub>FF</sub> | RGB <sub>KF</sub> | RGB <sub>FF</sub> | RGB <sub>KF</sub> | RGB <sub>FF</sub> | RGBD <sub>KF</sub> | RGBD <sub>FF</sub> |
| fr1/xyz  | 0.057601 | <b>0.008820</b> | 0.013473        | 0.130058 | 0.067820            | 0.133107            | 0.053749          | 0.090115          | 0.091125          | 0.141664          | 0.068696           | 0.123372           |
| fr1/rpy  | 0.163409 | 0.080904        | <b>0.028738</b> | 0.148215 | 0.049470            | 0.122989            | 0.076841          | 0.089332          | 0.088578          | 0.078708          | 0.094127           | 0.075722           |
| fr1/desk   | 0.182512 | 0.090906        | <b>0.025831</b> | 0.163761 | 0.060936            | 0.105381            | 0.547886          | 0.186484          | 0.436865          | 0.167203          | 0.095860           | 0.126698           |
| fr1/desk2  | 0.188611 | 0.100898        | <b>0.042558</b> | 0.448858 | 0.082223            | 0.129600            | 0.181626          | 0.168655          | 0.092466          | 0.118868          | 0.147212           | 0.151647           |
| fr1/room   | 0.215587 | 0.202820        | <b>0.101165</b> | 0.603607 | 0.297600            | 0.267711            | 0.288973          | 0.305937          | 0.293755          | 0.344048          | 0.310790           | 0.358645           |
| fr1/plant  | 0.122159 | 0.072341        | 0.063884        | 0.569270 | 0.067125            | 0.056688            | 0.056227          | 0.073000          | 0.067376          | <b>0.043552</b>   | 0.049789           | 0.044347           |
| fr2/desk   | 0.467958 | 0.386566        | 0.095053        | 0.945456 | <b>0.088583</b>     | 0.343053            | 0.095900          | 0.329024          | 0.139392          | 0.360144          | 0.197403           | 0.522299           |
| Comparison of the Relative Pose Error (RPE) in [m/frame] |          |                 |                 |          |                     |                     |                   |                   |                   |                   |                    |                    |
| fr1/xyz  | 0.005913 | <b>0.005151</b> | 0.007569        | 0.007046 | 0.008389            | 0.005430            | 0.005743          | 0.005145          | 0.006256          | 0.005613          | 0.005719           | 0.005567           |
| fr1/rpy  | 0.007045 | 0.007096        | 0.017499        | 0.016808 | 0.009336            | 0.007379            | 0.007436          | <b>0.006622</b>   | 0.007587          | 0.007174          | 0.007772           | 0.007249           |
| fr1/desk   | 0.009938 | 0.008164        | 0.010673        | 0.011375 | 0.010358            | 0.008477            | 0.030225          | <b>0.008032</b>   | 0.025997          | 0.008341          | 0.008691           | 0.008152           |
| fr1/desk2  | 0.009267 | 0.009206        | 0.016614        | 0.018339 | 0.010570            | 0.009281            | 0.010960          | 0.009980          | 0.009088          | <b>0.008602</b>   | 0.011722           | 0.010654           |
| fr1/room   | 0.006234 | 0.007017        | 0.011987        | 0.015723 | 0.008779            | 0.006906            | 0.007206          | <b>0.005883</b>   | 0.008031          | 0.006525          | 0.008358           | 0.006637           |
| fr1/plant  | 0.006215 | <b>0.004970</b> | 0.007022        | 0.024109 | 0.006410            | 0.005643            | 0.005514          | 0.005075          | 0.005728          | 0.005309          | 0.005985           | 0.005426           |
| fr2/desk   | 0.002800 | 0.002550        | 0.002683        | 0.008114 | 0.004327            | <b>0.002440</b>     | 0.003476          | 0.002493          | 0.002862          | 0.002540          | 0.003100           | 0.002724           |
| Comparison of the Relative Pose Error (RPE) in [m/s]     |          |                 |                 |          |                     |                     |                   |                   |                   |                   |                    |                    |
| fr1/xyz  | 0.026610 | <b>0.014700</b> | 0.041928        | 0.049424 | 0.030356            | 0.042284            | 0.019570          | 0.032021          | 0.036457          | 0.048075          | 0.026147           | 0.047505           |
| fr1/rpy  | 0.048653 | <b>0.032208</b> | 0.070280        | 0.161495 | 0.035646            | 0.040548            | 0.040370          | 0.035534          | 0.035270          | 0.035199          | 0.034314           | 0.036720           |
| fr1/desk   | 0.044288 | 0.061779        | 0.053456        | 0.106539 | <b>0.033294</b>     | 0.048029            | 0.221955          | 0.077998          | 0.186867          | 0.073144          | 0.047543           | 0.067893           |
| fr1/desk2  | 0.057216 | 0.065347        | 0.069546        | 0.201169 | 0.063328            | 0.074684            | 0.067031          | 0.070562          | <b>0.052318</b>   | 0.061529          | 0.065607           | 0.072351           |
| fr1/room   | 0.064266 | 0.070806        | 0.066657        | 0.216494 | 0.049967            | 0.058260            | <b>0.042716</b>   | 0.048161          | 0.048130          | 0.057825          | 0.060220           | 0.060312           |
| fr1/plant  | 0.043624 | 0.042179        | 0.037893        | 0.340992 | 0.028291            | 0.035783            | <b>0.023805</b>   | 0.030629          | 0.030358          | 0.030770          | 0.032056           | 0.032714           |
| fr2/desk   | 0.032475 | 0.030671        | 0.014002        | 0.099676 | 0.014306            | 0.021706            | 0.014256          | 0.024531          | <b>0.012474</b>   | 0.028131          | 0.013323           | 0.037393           |

TABLE II: Comparison of the ATE in [m], the RPE in [m/frame] and the RPE in [ $\frac{m}{s}$ ] of DVO [6], ORB-SLAM2 [5], RGBD-SLAM [4], D-EA [9] and REVO in key frame (KF) and frame-to-frame (FF) VO on the RGBD TUM datasets [11].

### C. Visual Odometry with Large Relative Motion

Steinbrücker et al. [20] state that a limitation of direct approaches [6], [7], [20] is that they can only achieve good accuracy when the inter-frame motion is small. To study the influence of larger relative motion on the direct method DVO [6] as well as the indirect ORB-SLAM2 [5], we skip one frame (15 fps) and two frames (10 fps) of three sequences from a typical office setup. We again compare DVO [6], ORB-SLAM2 [5], D-EA [9] to several REVO variants in frame-to-frame mode using the same configuration as previously. We present the results of the ATE in [m], the RPE with  $\Delta t = 1f$  in [ $\frac{m}{f}$ ] and with  $\Delta t = 1s$  in [ $\frac{m}{s}$ ] in Table III.

REVO shows by far the best results on the challenging fast motion sequences *fr1/desk* and *fr1/desk2* when only every second frame is taken and performs best on *fr1/desk* even if every third frame is taken. However, all the approaches have problems with *fr1/desk2* when only every third frame is processed, which we attribute to the rapid motion changes not being sufficiently covered at 10 fps. All methods run well on the low-paced *fr2/desk* dataset with REVO showing the best scores. When comparing to the results without frame-skipping (see Tab. II), it can be seen that in contrast to REVO the performance of ORB-SLAM2 [5] and DVO [6] drops severely when increasing inter-frame motion. Through our experiments we found that edge-based methods have a larger convergence basis than direct ones, which is in line with [9]. We also investigated indirect methods and showed that they also suffer a large decline in accuracy when inter-frame motion gets large, which was previously not studied.

| Seq.                                | DVO [6]            |                 | ORB2 [5]        |                 | D-EA [9]        |                 | Canny [12] |  | SE [14] |  | Robust Edge-based VO (REVO) HED [15] |  |
|-------------------------------------|--------------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------|--|---------|--|--------------------------------------|--|
|                                     | Every Second Frame |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| Absolute Trajectory Error (ATE) [m] |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.509550           | 0.248495        | 0.407339        | 0.086730        | 0.178783        | <b>0.083135</b> |            |  |         |  |                                      |  |
| fr1/desk2                           | 0.189339           | 0.537884        | 0.504204        | <b>0.096031</b> | 0.097558        | 0.106747        |            |  |         |  |                                      |  |
| fr2/desk                            | 0.320031           | 0.280787        | 0.557992        | 0.241221        | <b>0.240713</b> | 0.266310        |            |  |         |  |                                      |  |
| Relative Pose Error (RPE) [m/frame] |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.048847           | 0.021777        | 0.034126        | 0.012029        | 0.023664        | <b>0.011999</b> |            |  |         |  |                                      |  |
| fr1/desk2                           | 0.018320           | 0.025209        | 0.039395        | 0.012893        | 0.016158        | <b>0.011564</b> |            |  |         |  |                                      |  |
| fr2/desk                            | 0.004223           | <b>0.003415</b> | 0.010050        | <b>0.003180</b> | 0.003291        | 0.003415        |            |  |         |  |                                      |  |
| Relative Pose Error (RPE) [m/s]     |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.268046           | 0.094782        | 0.201994        | <b>0.039849</b> | 0.099242        | 0.041874        |            |  |         |  |                                      |  |
| fr1/desk2                           | 0.075311           | 0.154527        | 0.251130        | 0.071256        | 0.065778        | <b>0.057623</b> |            |  |         |  |                                      |  |
| fr2/desk                            | 0.020814           | 0.022215        | 0.059885        | <b>0.016071</b> | 0.017999        | 0.020440        |            |  |         |  |                                      |  |
| Every Third Frame                   |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| Absolute Trajectory Error (ATE) [m] |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.611447           | 0.451888        | 0.653550        | <b>0.213626</b> | 0.983956        | 1.646638        |            |  |         |  |                                      |  |
| fr1/desk2                           | 1.733888           | 1.659247        | <b>1.093120</b> | 1.261090        | 1.717328        | 1.294249        |            |  |         |  |                                      |  |
| fr2/desk                            | 0.257220           | 0.231347        | 0.480751        | 0.193598        | <b>0.188689</b> | 0.223433        |            |  |         |  |                                      |  |
| Relative Pose Error (RPE) [m/frame] |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.092544           | <b>0.02845</b>  | 0.042230        | 0.030373        | 0.068388        | 0.105283        |            |  |         |  |                                      |  |
| fr1/desk2                           | 0.152108           | 0.521885        | <b>0.095456</b> | 0.108315        | 0.125948        | 0.101552        |            |  |         |  |                                      |  |
| fr2/desk                            | 0.004830           | 0.003973        | 0.011696        | <b>0.003726</b> | 0.003875        | 0.003981        |            |  |         |  |                                      |  |
| Relative Pose Error [m/s]           |                    |                 |                 |                 |                 |                 |            |  |         |  |                                      |  |
| fr1/desk                            | 0.395817           | 0.128363        | 0.200108        | <b>0.110992</b> | 0.382118        | 0.603190        |            |  |         |  |                                      |  |
| fr1/desk2                           | 0.605486           | 0.965886        | <b>0.445737</b> | 0.633889        | 0.778350        | 0.640105        |            |  |         |  |                                      |  |
| fr2/desk                            | 0.016626           | 0.018080        | 0.053848        | <b>0.013989</b> | 0.015547        | 0.017009        |            |  |         |  |                                      |  |

TABLE III: ATE and RPE of [6], [5], [9] and REVO.

## V. CONCLUSIONS

In this paper, we proposed REVO, a real-time capable robust edge-based RGBD visual odometry method, which utilizes machine-learned edges for relative camera motion estimation. We introduced an edge-based tracking quality measure to know when to insert a new key frame and introduced several ways to further increase robustness and optimization speed such as edge filters and constant motion assumption. REVO greatly outperforms a previous edge-based method [9] and performs better or on par with state-of-the-art methods [6], [5]. We also addressed the question which edge detector should be chosen for edge-based VO and experimentally demonstrated that machine-learned edges give better results for VO. By skipping one or two frames, we evaluated the motion estimation capabilities of REVO under higher inter-frame motion. The results imply a larger convergence basin compared to direct [6] and feature-based methods [5].

Even though in the current evaluation SE performs mostly better than HED, we still think there is a lot of potential in CNN edge detectors. A possible direction for further research is to train a CNN specifically for the task of VO including the post-processing step. With the rapid development in the area of CNNs and new hard- and software, we expect the inference speed to greatly increase in the future.

The REVO framework presented in this work is available online and can be downloaded for research purposes.

## ACKNOWLEDGMENT

This work was financed by the KIRAS program (no 850183, CSISmartScan3D) under supervision of the Austrian Research Promotion Agency (FFG).

## REFERENCES

- [1] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 652–659.
- [2] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.
- [3] I. Dryanovskii, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from rgb-d data," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2305–2310.
- [4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.
- [5] 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*, vol. PP, pp. 1–8, 2017.
- [6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.
- [7] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849.
- [8] J. J. Tarrío and S. Pedre, "Realtime edge-based visual odometry for a monocular camera," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 702–710.
- [9] M. P. Kuse and S. Shen, "Robust camera motion estimation using direct edge alignment and sub-gradient method," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 573–579.
- [10] X. Wang, D. Wei, M. Zhou, R. Li, H. Zha, and C. Beijing, "Edge enhanced direct visual odometry," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.
- [11] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.
- [12] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 8, no. 6, pp. 679–698, 1986.
- [13] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 33, no. 5, pp. 898–916, 2011.
- [14] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 37, no. 8, pp. 1558–1570, 2015.
- [15] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.
- [16] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgb-d images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.
- [17] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 14, no. 2, pp. 239–256, 1992.
- [18] 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 *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.
- [19] 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.
- [20] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2011, pp. 719–722.
- [21] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, pp. 415–428, 2012.