RGB-D SLAM in Dynamic Environments Using Static Point Weighting

Shile Li and Dongheui Lee

***Abstract*—We propose a real-time depth edge based RGB-D SLAM system for dynamic environment. Our visual odometry method is based on frame-to-keyframe registration, where only depth edge points are used. To reduce the influence of dynamic objects, we propose a static weighting method for edge points in the keyframe. Static weight indicates the likelihood of one point being part of the static environment. This static weight is added into the intensity assisted iterative closest point (IAICP) method to perform the registration task. Furthermore, our method is integrated into a SLAM (Simultaneous Localization and Mapping) system, where an efficient loop closure detection strategy is used. Both our visual odometry method and SLAM system are evaluated with challenging dynamic sequences from the TUM RGB-D dataset. Compared to state-of-the-art methods for dynamic environment, our method reduces the tracking error significantly.**

***Index Terms*—Computer vision for other robotic applications, SLAM (Simultaneous Localization and Mapping), visual tracking.**

I. Introduction

FOR navigation purposes, Simultaneous Localization and Mapping (SLAM) system is required in many robotic applications. In many SLAM systems, visual odometry estimation plays a key role. It estimates the camera's ego-motion by comparing consecutive frames of image. Especially RGB-D based visual odometry is extensively researched in last years due to the emergence of low cost depth camera such as Kinect [1]–[7]. Using RGB-D data, 3D ego-motion of camera can be obtained, whereas conventional encoder based wheel odometry can only provide 2D motion.

To simplify the problem formulation, most state-of-the-art visual odometry methods assume static environment. However, dynamic objects, such as human, exist in many real life environments. While small portion of dynamic objects can be handled by viewing them as noise, large proportion of dynamic objects violate the static environment assumption, thus the usage of many existing visual odometry methods for real applications is limited.

Current RGB-D visual odometry methods can be roughly categorized into two groups. To handle dynamic objects, different strategies are used for these two groups. The first group is dense visual odometry [1], [8]–[10]. These methods formulate the task as an energy minimization problem. The energy function is the sum over pixel-wise intensity/depth difference between the target image and warped source image. Then the camera's 6 DOF motion is iteratively optimized over this energy function. This form of energy function strongly depends on the static environment assumption. In a dynamic environment, even with the correct motion, a dynamic object can cause large intensity/depth difference between the warped source frame and the target frame. Therefore the energy function does not have the minimum at the correct motion. To compensate dynamic object, dynamic objects need to be found and excluded from the optimization process. Wang *et al.* calculate dense optical flow from RGB images, and dynamic objects are found by clustering the image based on point trajectories [11]. The pixels of dynamic objects are then excluded for energy function minimization. Their method improves the robustness against dynamic object effectively, but the optical flow estimation and clustering cannot be performed in real-time. Sun *et al.* [12] use intensity difference image to identify the boundary of dynamic objects. Then dense dynamic points are segmented using the quantized depth image. Their method achieves stable performance for highly dynamic scenes, but segmentation takes half second per frame, which hinders the real-time applicability. Kim *et al.* [13] propose to use depth difference to multiple warped previous frames to calculate a static background model. However, due to the aperture problem, if the dynamic object moves parallel to the image plane, only the boundary of dynamic object can be found effectively using depth difference. Therefore the influence of dynamic object cannot be totally removed.

The second group is correspondence based method [2], [14]–[16]. Correspondences are matched between the source and target frames. Then the camera's ego-motion is estimated using closed-form solution from the correspondences. The correspondence can be found by matching keypoints (such as SIFT, SURF) [15], or for Iterative Closest Point [17] based methods [2], [18], correspondences are densely established using a certain distance metric. Since points from the static environment follow the same motion, RANSAC regression is usually used to filter out dynamic objects [19], [20]. However, if there are more dynamic feature points than static ones, RANSAC may result in a wrong estimate of static feature points.

To compensate dynamic objects, all above mentioned methods require a correspondence matching step, where either dense or sparse correspondences are needed. While accurate dense correspondences matching is time consuming [11], fast approximation [13] suffers from the aperture problem. Accurate matching of 2D keypoints can be performed in real-time [19], [20]. However sparse 2D keypoints can be distributed unevenly in the environment. If a dynamic object has many texture, then the dynamic

Manuscript received February 15, 2017; accepted June 24, 2017. Date of publication July 11, 2017; date of current version July 21, 2017. This paper was recommended for publication by Associate Editor E. M. Mouaddib and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Shile Li.)*

The authors are with the Department of Electrical Engineering and Computer Engineering, Technical University of Munich, Munich 80333, Germany (e-mail: li.shile@mytum.de; dhlee@tum.de).

Digital Object Identifier 10.1109/LRA.2017.2724759

keypoints will outnumber static keypoints, which may results in failure in RANSAC regression. Therefore additional IMU sensor data are often used to compensate this issue [19], [20].

In this paper, we choose to use depth edges to find correspondences. Depth edge contains the structure information of the environment. It was shown that accurate visual odometry can be estimated based on depth edge [21], [22]. Depth edge points are sparsely present, therefore they can be efficiently matched. Furthermore, the amount of depth edge points is more balanced than 2D keypoints. We match edge points between frames by using both geometric and intensity distances [23]. Upon matched edge points, a novel static weighting method is proposed to downweight dynamic points for the visual odometry method. Furthermore, using an efficient loop closure detection procedure, the visual odometry method is fused into a pose graph based SLAM system, resulting in a fast RGB-D SLAM system suitable for dynamic environment.

The main contributions of this paper are:

1) A novel efficient static weighting method is proposed to reduce the influence of dynamic objects on pose estimation. It calculates the likelihood of each keyframe point being part of the static environment.
2) The static weighting terms are integrated into the IAICP method. This leads to a real-time RGB-D visual odometry method for dynamic environment.

The effectiveness of our method against dynamic objects is tested on dynamic sequences from TUM RGB-D dataset [24]. Our proposed RGB-D SLAM using static point weighting outperforms previous methods [1], [13],[12] in most sequences.

II. Preliminaries

Given a 3D point $\mathbf{p} = (x, y, z, 1)^T$ in homogeneous coordinate relative to the camera, the image pixel coordinate $\mathbf{x} = (u, v)^T$ $(u \in [0, height - 1], v \in [0, width - 1])$ of $\mathbf{p}$ is calculated with the camera projection function π:

$$\mathbf{x} = \pi(\mathbf{p}) = \left(\frac{x f_x}{z} + o_x, \frac{y f_y}{z} + o_y\right)^T, \tag{1}$$

where $height$ and $width$ are the pixel number in image's x- and y- direction, f_x, f_y are the camera focal lengths and o_x, o_y are the camera center coordinates.

Due to the ego-motion of the camera, a 3D point $\mathbf{p}$ in the keyframe coordinate is rigidly transformed in the current frame with the transformation matrix $\mathbf{T}_k^t \in SE(3)$. The point's new coordinate in the current camera coordinate frame is then:

$$\mathbf{p}' = \mathbf{T}_k^t \mathbf{p} = \begin{bmatrix} \mathbf{R}_k^t & \mathbf{t}_k^t \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p} \tag{2}$$

At time step t, an intensity image I_t and an organized point cloud P_t with the resolution $width \times height$ are obtained, where $P_t(i)$ indicates the ith point in P_t. Intensity value of pixel $\mathbf{x}$ is a gray scale value $I_t(\mathbf{x}) \in [0, 255]$, which is converted from RGB values (0.299red + 0.587green + 0.114blue). The pixel $\mathbf{x}$'s corresponding 3D point $\mathbf{p}$ is indicated as $P_t(ind(\mathbf{x}))$, where $ind()$ is the mapping from the image coordinate to the point index in the organized point cloud's one-dimensional list:

$$ind(\mathbf{x}) = ind((u, v)^T) = v \times width + u, \tag{3}$$

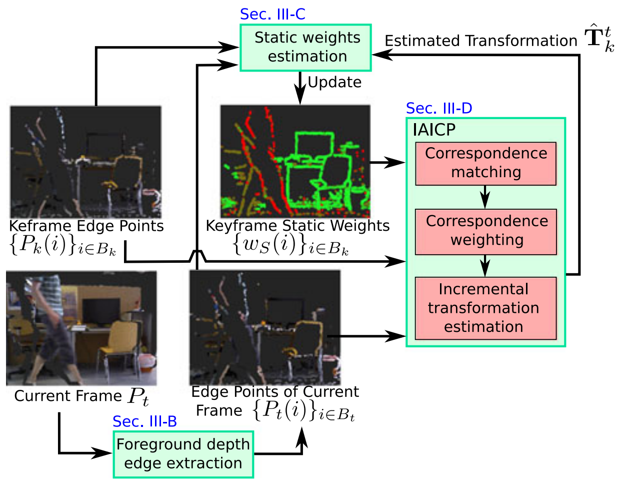
Sec. III-C
Static weights estimation
Estimated Transformation $\hat{\mathbf{T}}_k^t$
Update
Sec. III-D
IAICP
Correspondence matching
Correspondence weighting
Incremental transformation estimation
Keframe Edge Points $\{P_k(i)\}_{i\in B_k}$
Keyframe Static Weights $\{w_S(i)\}_{i\in B_k}$
Current Frame P_t
Edge Points of Current Frame $\{P_t(i)\}_{i\in B_t}$
Sec. III-B
Foreground depth edge extraction

Fig. 1. Overview of our visual odometry system.

To find out the image coordinate $\mathbf{x}$ of a point index i, the inverse mapping is:

$$\mathbf{x} = ind^{-1}(i) = \left(i - \left\lfloor \frac{i}{width} \right\rfloor width, \left\lfloor \frac{i}{width} \right\rfloor\right)^T. \tag{4}$$

III. Foreground Edge Based Visual Odometry

A. Overview

The overview of the proposed visual odometry method is illustrated in Fig 1. For each incoming frame, foreground edge points are first extracted, where only extracted edge points are used for odometry estimation. Every Nth frame is selected as a keyframe[1]. For each keyframe, static weights for edge points are estimated. A static weight indicates how likely one point belongs to the static environment. Then the relative transformation from the keyframe to the current frame is estimated using IAICP algorithm [23], where static weights are combined in order to reduce the effect of dynamic moving objects on the transformation estimation. Finally, the static weights for the keyframe are updated based on the estimated motion.

B. Foreground Depth Edge Extraction

Depth edge points are points that have large depth discontinuity in their neighbourhood. Two types of depth edge points exist: foreground edge points and occluded edge points. Foreground edge points present boundaries of objects, which are in front of other objects. Occluded edge points are from objects behind some other objects, they are caused by occlusion of other objects infront of them. The foreground edge points are stable to a moving camera, because they capture the geometry of the objects. However occluded edge points are sensitive to a moving camera, therefore they need to be excluded for estimating camera trajectory.

Foreground depth edge points play an important role for iterative closest point method. As evidenced in [21], [23], using

[1] Alternatively one can consider to select keyframe based on camera motion. In highly dynamic environment, however, the visual content changes drastically even when the camera does not move. In this case, there might not be enough common visible points to estimate the relative pose, thus it can fail to estimate camera motion.

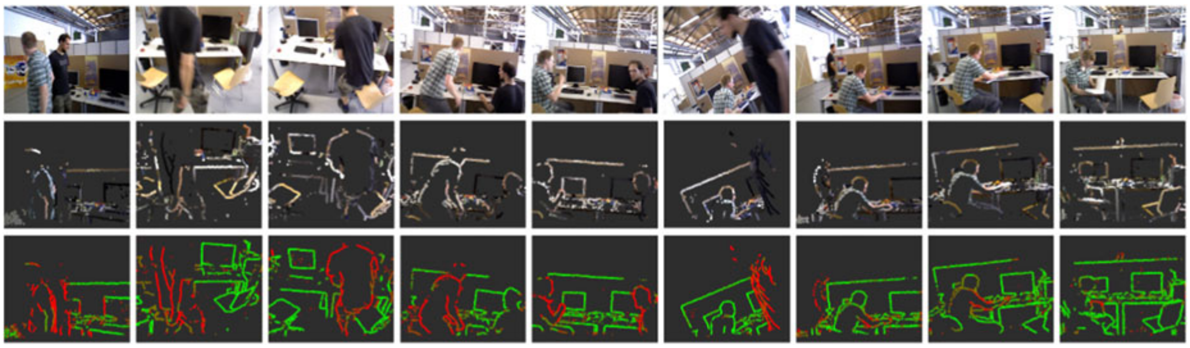

Fig. 2. Foreground depth edge extraction and static weighting examples taken from "fr3/walking" sequences. First row: the original RGB image. Second row: foreground depth edge. Third row: static weighting result, where green indicates static and red indicates dynamic.

foreground depth edge points can improve the accuracy of registration result. Because by using depth edge points, the probability of finding correct correspondence is higher than using uniformly sampled points. Moreover, correct correspondences are also needed for our static weighting process (Section III-C).

Given the point cloud P_t, a set B_t consisting of foreground edge point indices is constructed. Firstly the depth differences $\{h_i\}_{i=1}^4$ between each point and its four neighbours are computed:

$$h_i = e_Z^T P_t(ind(\mathbf{x})) - e_Z^T P_t(ind(\mathbf{x} + \mathbf{o}_i)), \tag{5}$$

where $e_Z = (0,0,1,0)^T$ is used to extract depth value of one point and the four offset vectors $< \mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4 >$ are $< (0,b)^T, (0,-b)^T, (b,0)^T, (-b,0)^T >$. An offset $b > 1$ is used here, because a lot of depth difference between direct neighbours ($b = 1$) cannot be computed due to a lot of NaN (Not a Number) value pixels that exist near the depth discontinous area. On the other hand a larger b value causes more points to be detected as depth edge, resulting in thicker edges. Balancing between the NaN value avoidance effect and the thickness of depth edge, we empirically set $b = 4$. Then a point is considered as foreground edge points and added into the edge point set $B_t = \{B_t, ind^{-1}(\mathbf{x})\}$, if it fulfills the following conditions:

$$\begin{aligned} \max(h_1, h_2, h_3, h_4) &< e_Z^T P_t(ind(\mathbf{x}))\tau_b, \\ \max(abs(h_1 - h_2), abs(h_3 - h_4)) &> e_Z^T P_t(ind(\mathbf{x}))\tau_f. \end{aligned} \tag{6}$$

The first condition rejects occluded edge points, because occluded points have a much larger depth than their neighbours, where $e_Z^T P_t(ind(\mathbf{x}))\tau_b$ is the depth depended threshold. With a very large τ_b, no occluded point can be rejected. If τ_b is too small, a lot of actual foreground edge points can be also rejected due to slightly larger depth than neigbouring pixels. The second condition checks whether a point can be considered as edge point by checking the depth discontinuity with a threshold $e_Z^T P_t(ind(\mathbf{x}))\tau_f$. If τ_f is too large, then no edge points can be detected and if τ_f is too small, almost every point is detected as edge. In our experiments, τ_b and τ_f are set as 0.015 m and 0.04 m respectively. In the second row of Fig. 2, some examples of foreground depth edge extraction are illustrated.

C. Static Weight Estimation

Two types of points exist in the environment: points from static object and points from dynamic moving object. Due to the ego-motion of the camera, observed points are constantly moving in the camera's coordinate. Comparing point clouds from two frames, static points are moved with same rigid transformation, which is the inverse of camera's ego-motion, while dynamic points do not follow the same rigid transformation due to their own movements.

We estimate the static weights for a source point cloud P_{src} by comparing it to a target point cloud P_{tgt}. The static weight is only estimated for the foreground depth edge points $\{P_{src}(i)\}_{i \in B_{src}}$, where B_{src} is the set of edge point indices (Section III-B). The static weight of $P_{src}(i)$ is denoted as $w_i^{src,tgt}$, and it is estimated based on the Euclidean distance between $P_{src}(i)$ and the corresponding point $P_{tgt}(c(i))$ in the target cloud: $d_i = \left\| \hat{\mathbf{T}}_{src}^{tgt} P_{src}(i) - P_{tgt}(c(i)) \right\|$, where $c(i) \in B_{tgt}$ is the found correspondence point index in the target cloud and $\hat{\mathbf{T}}_{src}^{tgt}$ is the estimated transformation that aligns the source cloud to the target cloud. In case that $c(i)$ is not found in the vicinity of $P_{src}(i)$, we set d_i to a large constant value D.

A static point $P_{src}(i)$ after transformation becomes $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$. Assuming that $c(i)$ and $\hat{\mathbf{T}}_{src}^{tgt}$ are both correct, $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$ should be aligned perfectly on its corresponding point $P_{tgt}(c(i))$. Therefore for a static point, d_i should be zero or a small value due to sensor noises. Taking advantage of this characteristic, the static points can be distinguished from the dynamic points based on the statistic over $\{d_i\}_{i \in B_{src}}$. Following [1], the static weight $w_i^{src,tgt}$ is estimated based on the Student's t-distribution

$$w_i^{src,tgt} = \frac{\nu_0 + 1}{\nu_0 + ((d_i - \mu_D)/\sigma_D)^2}, \tag{7}$$

$$\sigma_D = 1.4826\,\mathrm{Median}\{|d_i - \mu_D|\}_{d_i \neq D} \tag{8}$$

ν_0 is the degree of freedom of t-distribution, where a larger ν_0 results in steeper decrease of $w_i^{src,tgt}$ with increasing d_i. Notice that points without a valid correspondence in the close neighbourhood ($P_{src}(i)_{d_i = D}$) are not used for computing σ_D. In our experiments, ν_0 is empirically set to 10. The mean value μ_D is manually set to zero, because smaller distance indicates a more static point. The variance σ_D is estimated using the median absolute deviation.

Fig. 3 shows an example of the histogram of correspondence distance and t-distribution. The chosen distribution fits the actual experimental data nicely. A zero valued d_i indicates highest static likelihood, a small d_i is caused by sensor noises or discrete sampling of the environment, and a large d_i is caused by dynamic movement that differs from the camera motion. The procedure

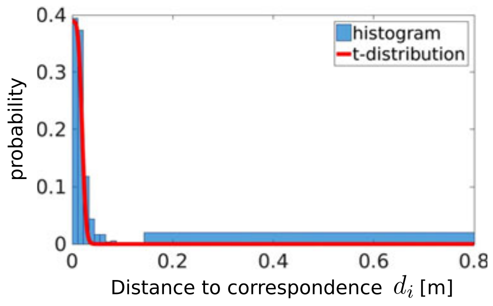

Fig. 3. Histogram of correspondence distance and pdf of t-distribution. The pdf of t-distribution fits the actual data nicely.

to estimate the static weights $\{w_i^{src,tgt}\}_{i \in B_{src}}$ is summarized in Algorithm 1.

Algorithm 1: Static weighting for depth edge points.

```
Input: - a source cloud $P_{src}$ and edge point set $B_{src}$
    - a target cloud $P_{tgt}$
    - corresponding point index for source cloud edge
      points $\{c(i)\}_{i \in B_{src}}$
    - current transformation estimate $\hat{\mathbf{T}}_{src}^{tgt}$
Output: - static weights: $w_i^{src,tgt}$ for $i \in B_{src}$
  for $i \in B_{src}$ do
      Calculate distance $d_i$ between warped source point
      $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$ and its corresponding point $P_{tgt}(c(i))$
  end for
  Calculate variance $\sigma_D$ of $\{d_i\}_{i \in B_{src}}$ (8)
  for $i \in B_{src}$ do
      Estimate static weight $w_i^{src,tgt}$ (7)
  end for
```

In our visual odometry method, the static weights are only estimated for keyframes, where keyframes are always set as source frame. Assuming that the latest keyframe is P_k with k as the index of the keyframe, the static weight of keyframe points are calculated by comparing the keyframe to two other target frames: one is the previous keyframe P_{k-N}, another one is the latest frame at time step t: P_t. The static weight for the point $P_k(i)$ is defined as $w_S(i)$:

$$w_S(i) = \alpha w_i^{k,k-N} + (1-\alpha) w_i^{k,t}, \tag{9}$$

where $w_i^{k,k-N}$ is computed by setting last keyframe P_{k-N} as the target cloud in Algorithm 1, and $w_i^{k,t}$ is computed by setting the current frame P_t as target cloud. In our experiments, α is empirically set as:

$$\alpha = \begin{cases} 1 & if \quad t = k \\ 0.5N/(N+t-k) & otherwise \end{cases} \tag{10}$$

If the keyframe is the current frame, the static weights are initialized with $w_i^{k,k-N}$. With passage of time, more influence from current frame is considered where the initialization term $w_i^{k,k-N}$ and the update term $w_i^{k,t}$ are complementary to each other.

The update term $w_i^{k,t}$ is used for two reasons. i) Previously static object can start moving after the keyframe has been defined, where previously static points can convert to dynamic points. If only the initialization term is used, the new dynamic points can only be detected by the time of the new keyframe $k+N$. This would result in drift problem in the time interval $[k+1, k+N-1]$. ii) Due to occlusion of foreground objects, the visible part of the environment always changes. Newly occluded part from the keyframe cannot find correct correspondences in the new frame P_t, therefore newly occluded part should be avoided in the transformation estimation process. Occluded points usually have large distance to their falsely found correspondences, thus by using $w_i^{k,t}$, they can be efficiently downweighted.

The initialization term is important as well as the update term. It is estimated by comparing the keyframe with last keyframe. If only the update term is used, dynamic objects with small velocity cannot be distinguished effectively. For the time step $k+1$, dynamic points with small velocity is not moved too far within one frame, the distance between correspondences might land in the "small noise" range of static points. On the contrary, the initialization term is calculated with a relative larger time difference N, thus the d_i for small velocity objects is larger and more distinguishable from static points.

In the third row of Fig. 2, some examples of estimated static weights are illustrated. It shows that our method can effectively downweight dynamic points for different cases, including one person moving, two persons moving and part of one person moving.

D. Intensity Assisted Iterative Closest Point

The pair-wise point cloud registration is performed using Intensity Assisted Iterative Closest Point (IAICP) method [23]. Compared to the conventional ICP method, the intensity information of each point is also used for correspondence matching and weighting.

Given a source frame $< P_{src}, I_{src} >$ and a target frame $< P_{tgt}, I_{tgt} >$, IAICP estimates the relative transformation that aligns the source cloud P_{src} to the target cloud P_{tgt}. IAICP is an iterative method, where the transformation matrix $\mathbf{T}^*$ is usually initialized with identity matrix or with a motion prediction. In our experiments, $\mathbf{T}^*$ is initialized with first order motion prediction. Then the optimal transformation value $\mathbf{T}^*$ is searched iteratively, where the kth iteration can be summarized as:

i) Search for each depth edge point $P_{src}(i)_{i \in B_{src}}$ of the source cloud a point in the target cloud as correspondence. For $P_{src}(i)$, the index of the corresponding point index in the target point cloud P_{tgt} is denoted as $c(i) \in B_{tgt}$, where

$$c(i) = \underset{j}{\operatorname{argmin}} \|\mathbf{T}^* P_{src}(i) - P_{tgt}(j)\|. \tag{11}$$

ii) Compute the optimal incremental transformation $\mathbf{T}_k$ that minimizes the sum of weighted Euclidean distance between the established correspondences:

$$\mathbf{T}_k = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i W(i) \|\mathbf{T}\mathbf{T}^* P_{src}(i) - P_{tgt}(c(i))\|, \tag{12}$$

where $W(i)$ is a weighting term that indicates the quality of correspondence. The equation is usually solved with a closed-form solution [25] such as Singular Value Decomposition [26]. In our method, not all edge points are used to compute (12), instead we randomly select 120 depth edge points for each ICP iteration.

iii) Update $\mathbf{T}^*$ as: $\mathbf{T}^* \leftarrow \mathbf{T}_k \mathbf{T}^*$.

In the following, we explain how to compute the weighting term $W(i)$ and how to get the correspondence index $c(i)$.

1) Correspondence Weighting: In practice, not every established correspondence is determined correctly. The outliers badly influences the transformation estimation. To compensate outliers, a weighting term $W(i)$ for corresponding pair $< P_{src}(i), P_{tgt}(c(i)) >$, is estimated. The weighting is based on an intensity term $w_I(i)$, a geometric term $w_G(i)$ and also the static weighting term $w_S(i)$ (Section III-C):

$$W(i) = w_I(i) w_G(i) w_S(i). \tag{13}$$

The static weighting term $w_S(i)$ is estimated as described in Section III-C. The intensity term $w_I(i)$ is calculated based on intensity residual $r_i^{(I)}$:

$$\begin{aligned} r_i^{(I)} &= I_s(ind^{-1}(i)) - I_t(ind^{-1}(c(i))), \\ w_I(i) &= \frac{\nu_1 + 1}{\nu_1 + ((r_i^{(I)} - \mu^{(I)})/\sigma^{(I)})^2}, \end{aligned} \tag{14}$$

where ν_1, degree of freedom of t-distribution, is set to 5, $\mu^{(I)}$ and $\sigma^{(I)}$ are the median and deviation value of all corresponding pairs. We followed [8] to use Student's t-distribution based weighting function[2]. The geometric term $w_G(i)$ is estimated similarly:

$$\begin{aligned} r_i^{(G)} &= \|\mathbf{T}^* P_s(i) - P_t(c(i))\|, \\ w_G(i) &= \frac{\nu_1 + 1}{\nu_1 + ((r_i^{(G)} - \mu^{(G)})/\sigma^{(G)})^2} \end{aligned} \tag{15}$$

Using the proposed weighting terms, outlying correspondences with large intensity difference or having large geometric distance are intuitively downweighted. Furthermore, the static weight is responsible to downweight the influence of dynamic object.

2) Correspondence Matching: Taking advantage of the organized point cloud, the search of matching point is performed in the image coordinate. By warping the source frame point $P_{src}(i)$ with the current estimate of $\mathbf{T}^*$, the image coordinate of the warped point is $\mathbf{x}' = \pi(\mathbf{T}^* P_{src}(i))$. Then target depth edge points $P_{src}(j)_{j \in B_t}$ in the neighbourhood of $\mathbf{x}'$ are considered as candidate of correspondence for $P_{src}(i)$, where neighbourhood $N(\mathbf{x}')$ is defined as a square around image coordinate $\mathbf{x}'$.

Having a query pair $< P_{src}(i), P_{tgt}(j) >$ to check, a score function for this pair is given as:

$$\begin{aligned} s(i,j) = {} & w_I(I_{src}(ind^{-1}(i)) - I_{tgt}(ind^{-1}(j))) \\ & w_G^{(\mu=0)}(\|\mathbf{T}^* P_{src}(i) - P_{tgt}(j)\|). \end{aligned} \tag{16}$$

$w_I(\cdot)$ (see (14)) and $w_G^{(\mu=0)}(\cdot)$ (see (15)) are weighting functions derived from last ICP's iteration, where $w_G^{(\mu=0)}$ sets the mean value to zero, because more closer point is more probable to be true corresponding point.

Then the correspondence is taken as the point that maximizes the score function:

$$c(i) = \underset{j \in (B_{tgt} \cap N(\mathbf{x}'))}{\operatorname{argmax}} \; s(i,j). \tag{17}$$

[2]According to [4], Student estimator achieves better result than Huber estimator and Turkey estimator.

IV. Loop Closure Detection

A pure visual odometry system suffers from drift problem, because current absolute pose is obtained by accumulating previous ego-motion estimates, which also accumulates the estimation errors. To compensate the drift problem, we integrated our method into a pose graph based SLAM system [27]. In the pose graph, consecutive keyframes are connected with a pose constraint, that is from the visual odometry method. In addition to that, if a keyframe detects previously seen environment, new constraints are added to previous keyframes, thus the accumulated drift can be corrected using graph optimization considering all constraints. We refer to [27] for details about pose graph optimization. In the following, our procedure of loop closure detection will be presented.

As a new keyframe P_k is set, we check loop closure of P_k with 10 randomly selected previous keyframe P_r. A loop closure is detected between P_k and P_r, when three conditions are fulfilled.

i) *Geometric proximity*: The two keyframes should not be too far away from each other:

$$\|\mathrm{transl}(\hat{\mathbf{T}}_k^r)\| < \tau_{distance} \tag{18}$$

where $\mathrm{transl}\,(\mathbf{T})$ extracts the translation vector from the transformation matrix $\mathbf{T}$, and the threshold $\tau_{distance}$ is set to 1.5 m in the experiments. This is because distanced keyframes have lower probability of viewing same part of the environment.

ii) *Common visible part*: The two keyframes should have common visible environment in view. A point from P_k, $P_k(i)$ is also possibly visible in P_r, if the warped point is still inside the image border:

$$\pi(\hat{\mathbf{T}}_k^r P_k(i)) \in \{[0, width - 1] \times [0, height - 1]\}. \tag{19}$$

For checking, 100 edge points are randomly selected from P_k. If less than 30% of points are visible, no loop closure between P_k and P_r is defined.

iii) *Forward backward consistency check*: If the previous two conditions are satisfied, the pair-wise registration is then performed using above described IAICP method (Section III-D). The registration is performed twice by setting P_r as source frame in one time and as target frame in the other time. Two registration results $\hat{\mathbf{T}}_k^r$ and $\hat{\mathbf{T}}_r^k$ are compared for consistency. The consistency check can be passed if:

$$\begin{aligned} \|\mathrm{transl}(\hat{\mathbf{T}}_k^r \hat{\mathbf{T}}_r^k))\| &< \tau_{distanceDiff}, \\ \|\mathrm{rotation}(\hat{\mathbf{T}}_k^r \hat{\mathbf{T}}_r^k))\| &< \tau_{angleDiff}, \end{aligned} \tag{20}$$

where thresholds are set as $\tau_{distanceDiff} = 0.02$ m and $\tau_{angleDiff} = 3$ degree.

If the keyframe pair $< P_k, P_r >$ fulfills all three conditions, a new relative pose constraint $\hat{\mathbf{T}}_k^r$ between them is added into the pose graph, which means detection of a new loop closure.

V. Experiment

Our method is tested with TUM RGB-D dataset [24]. Many previous papers [1], [4], [6], [8], [23] evaluated their methods on this dataset and achieved good results, however the sequences containing dynamic objects were not often used for

TABLE I
VISUAL ODOMETRY RESULTS: TRANSLATIONAL DRIFT AND ROTATIONAL DRIFT ON TUM RGB-D DATASET

sequences		RMSE of translational drift [m/s]					RMSE of rotational drift [°/s]				
		DVO [8]	BaMVO [13]	Depth edge + IAICP	Depth edge + RANSAC + IAICP	Our method	DVO [8]	BaMVO [13]	Our method without	Depth edge + RANSAC + IAICP	Our method
static	fr2/desk	0.0296	0.0299	0.0174	**0.0170**	0.0173	1.3920	1.1167	0.7325	**0.7145**	0.7266
	fr3/long-office	0.0231	0.0332	0.0200	0.0193	**0.0168**	1.5689	2.1583	0.9001	1.0683	**0.8012**
low dynamic	fr2/desk-person	0.0354	0.0352	0.0245	0.0189	**0.0173**	1.5368	1.2159	1.0389	0.8310	**0.8213**
	fr3/sitting-static	**0.0157**	0.0248	0.0198	0.0210	0.0231	**0.6084**	0.6977	0.5823	0.6220	0.7228
	fr3/sitting-xyz	0.0453	0.0482	0.0256	0.0254	**0.0219**	1.4980	1.3885	0.9152	0.9791	**0.8466**
	fr3/sitting-rpy	0.1735	0.1872	0.1058	0.1076	**0.0843**	6.0164	5.9834	**5.2157**	10.4392	5.6258
	fr3/sitting-halfsphere	0.1005	0.0589	0.0624	0.0583	**0.0389**	4.6490	2.8804	2.5247	2.7427	**1.8836**
high dynamic	fr3/walking-static	0.3818	0.1339	0.1192	0.0496	**0.0327**	6.3502	2.0833	2.9475	1.3791	**0.8085**
	fr3/walking-xyz	0.4360	0.2326	0.1802	0.1482	**0.0651**	7.6669	4.3911	3.4778	3.8904	**1.6442**
	fr3/walking-rpy	0.4038	0.3584	0.2855	0.3031	**0.2252**	7.0662	6.3398	**5.5704**	11.4640	5.6902
	fr3/walking-halfsphere	0.2628	0.1738	0.2016	0.0799	**0.0527**	5.2179	4.2863	4.5076	4.5912	**2.4048**

evaluation. In these sequences, people move in the environment, while the camera also moves with different patterns (static, xyz, rpy and halfsphere). These sequences are challenging due to the large proportion of dynamic parts in the observation, where in extreme case more than half of the image is occupied with dynamic object. Fig. 2 shows some example frames taken from the "walking" sequences. To handle the high dynamic environment, previous methods require non real-time method to segment dynamic part [11], [12] or suffer from large drift [13].

For our experiments, the "sitting", "walking" sequences from the TUM dataset are used. The "sitting" sequences are considered as low dynamic sequences and the "waling" sequences are considered as high dynamic sequences. To test the performance of our method in a normal static environment, sequences captured in static environment are also used for evaluation.

In the following, our visual odometry method and SLAM system are evaluated and compared with previous methods [8], [12], [13]. All the experiments are performed on a desktop computer with Intel Core i7-4790K CPU (4GHz) and 16 GB RAM. The visual odometry method only uses one CPU core, and for SLAM system, another CPU core is used for loop closure detection and map optimization.

A. Evaluation of Visual Odometry Method

For the evaluation of visual odometry, Relative Pose Error (RPE) metric is used. We firstly investigate the effectiveness of our static weighting strategy and then compare our method with previous other methods.

1) Effect of Static Weighting: To verify the effectiveness of the proposed static weighting strategy, our visual odometry is tested both with and without the static weight term $w_S(I)$ in the IAICP part (13). The comparison is shown in Table I, where "Depth edge + IAICP" means our method without static weighting. In "Depth edge + RANSAC + IAICP", a RANSAC based outlier rejection procedure is used, where the RANSAC procedure uses 100 iterations and has a outlier threshold of 1.5cm. The static weighting term improves the visual odometry result in most of the sequences and works better than the RANSAC based outlier rejection method. The average improvement in terms of translational drift for low-dynamic sequences is 8%, and for high-dynamic sequences, the average improvement is 52%. This verifies that our static weighting strategy effectively reduces the influence of dynamic objects, especially for high-dynamic environments.

2) Effect of Static Weight Initialization: The static weight initialization with previous keyframe is important as explained in Section III-C. To verify the importance of the initialization, experiments are also performed by setting α in (9) to zero. An example case is illustrated in Fig. 4, where a person is walking to the right. In this example, static weights are estimated for the keyframe P_k ($k = 401$). At time step $t = 402$, the person's movement is not that large between consecutive frames. Therefore if the weight initialization is not applied, then some parts of the human body are considered as a static part (green). If the initial value is considered, which are obtained by comparing the keyframe with last keyframe ($t = 396$), then the human body is more distinguished as a dynamic object by leveraging a larger distance during the 5 frames.

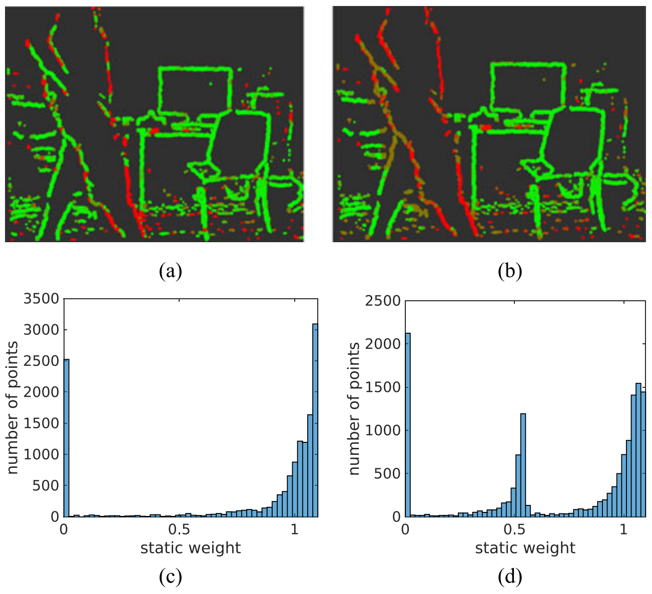

Fig. 4. Static weight of 401th frame (keyframe) in "fr3/walking_xyz" sequence at time step $t = 402$. (a) (b) show the visualization of static weights and (c)(d) show the histogram of static weights. (a) Without the initialization term. (b) With the initialization term. (c) Without the initialization term. (d) With the initialization term.

3) Comparison With Previous Methods: We compared our result with Dense Visual Odometry (DVO) [8] method and model-based dense-visual-odometry (BaMVO) [13] method. DVO is a state-of-the-art RGB-D visual odometry method for static environment, which can only handle small amount dynamic objects. BaMVO is specially designed to handle dynamic environment. The comparison results are shown in Table I, our result outperforms in almost all dynamic sequences. Even in the

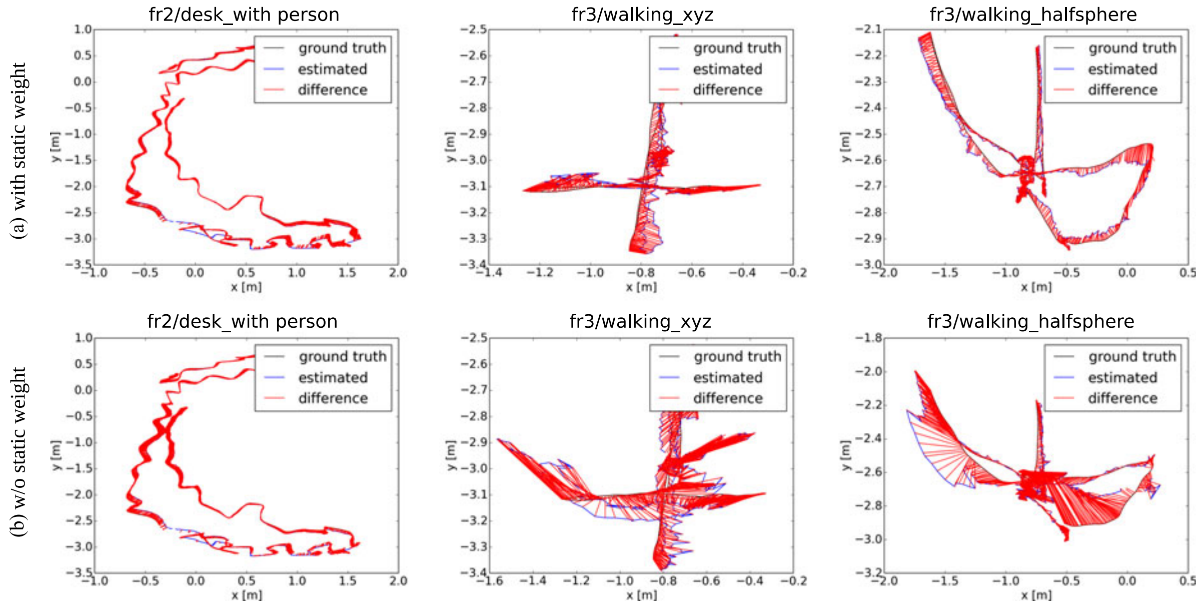

Fig. 5. Examples of estimated trajectories from our SLAM system. (a) Estimated trajectories with the proposed static weighting term. (b) Estimated trajectories without the proposed static weighting term.

static environment sequences, our visual odometry method still outperformed DVO, which takes the static environment assumption. For the highly dynamic sequences, our method outperforms significantly. Our method improves the visual odometry performance by 74.6% compared to DVO, and by 58.2% compared to BaMVO.

The sources of improvements are twofold: firstly by using sparse foreground depth edge points, correct correspondences can be efficiently found, where a higher correct ratio results in a more accurate transformation estimation; secondly, using these correspondences, our static weighting strategy effectively reduces the influence of dynamic objects. Compared to our method, DVO takes the static environment assumption in their problem formulation and cannot perform normally in highly dynamic sequences. In BaMVO, static weights are calculated based on depth difference, where points on the same image coordinate are simply approximated as correspondence. This approximation might cause the aperture problem for parallel dynamic motion to the image plane.

Our visual odometry method is performed on VGA image resolution (640×480) and requires only one CPU thread. The average computation time per frame is 22 ms. Compared to our method, DVO requires 32 ms per frame (320×240 resolution, i7-2600 CPU with 3.40 GHz) and BaMVO requires 42ms per frame (320×240 resolution, Intel i7 CPU with 3.3 GHz). The computation time of our method is less because no dense operation is needed as in DVO and BaMVO, both static weighting and transformation estimation are only performed on sparse depth edge points. The real-time performance makes our method suitable for on-line applications.

B. *Evaluation of SLAM System*

Finally we evaluated our SLAM system that includes loop closure detection and map optimization. For evaluating SLAM system, Absolute Trajectory Error (ATE) [24] metric is used. The estimated trajectories are compared to ground truth, and some examples are shown in Fig. 5. In the first row of Fig. 5, the trajectories are estimated using our proposed weighting term, and in the second row of Fig. 5, the trajectories are estimated without the proposed weighting term. It is notable that for the low-dynamic sequence "fr2desk_with_person" the improvement with static weighting is small, and for high-dynamic sequences the trajectory error is reduced greatly.

TABLE II
SLAM RESULTS: RMSE OF ABSOLUTE TRAJECTORY ERROR [M]

Sequence	Motion Remvoal+ DVO SLAM [12]		Our SLAM system	
	RMSE	Standard deviation	RMSE	Standard deviation
fr3/walking_halfsphere	0.1252	0.0903	**0.0489**	**0.7266**
fr3/walking_rpy	**0.1333**	**0.0839**	0.1791	0.1161
fr3/walking_static	0.0656	0.0536	**0.0261**	**0.0122**
fr3/walking_xyz	0.0932	0.0534	**0.0601**	**0.0330**
fr3/sitting_halfsphere	0.0470	0.0249	**0.0432**	**0.0246**
fr3/sitting_xyz	0.0482	0.0282	**0.0397**	**0.0206**
fr2/desk_with_person	0.0596	0.0239	**0.0484**	**0.0237**

Our SLAM system is compared to a non real-time method [12], which is a recent state-of-the-art RGB-D SLAM method for dynamic environment. In [12], dense dynamic object segmentation is performed for each frame, which takes half second per frame. The authors segment dynamic objects from each frame, and directly use the segmented frames as input for DVO-SLAM system [1]. The comparison is shown in Table II. The first column shows the sequence name from the TUM Dataset, where both low-dynamic "sitting" sequences and high-dynamic "walking" sequences are used for comparison. Our SLAM system works better in most of the sequences. The improvement for low-dynamic sequences is 15.2%, and the improvement for high-dynamic sequences is more notable with 24.7%.

The average computation time for our SLAM system takes ca. 45 ms per frame, including visual odometry estimation, loop closure detection and pose graph optimization. Compared to this, the method from [12] cannot be applied for real-time application, since their segmentation procedure alone already takes half second per frame.

VI. Conclusion

We proposed a real-time RGB-D visual odometry method that can handle highly dynamic environment such as the "walking" sequences from TUM Dataset [24]. The method uses foreground depth edge point to compute pair-wise point cloud registration. A robust static weighting strategy is proposed based on depth edge correspondences distance. Fusing the static weighting strategy into the intensity assisted ICP [23], our visual odometry system handles dynamic environment robustly. Furthermore, loop closure detection and map optimization are integrated, resulting a real-time SLAM system suitable for dynamic environment. Our method is evaluated on the dynamic sequences from TUM Dataset [24]. Compared to state-of-the-art real-time method [13], in terms of translational drift per second, our method improves the visual odometry accuracy by 58% in challenging "walking" sequences. The performance of our SLAM system is also proven using the TUM Dataset, which shows better performance than recent non real-time method [12]. In our method, the static weighting is only applied to foreground depth edges. Therefore our visual odometry method requires geometry rich environments, where a lot of depth edges exist. In future work, we want to investigate how to efficiently propagate the sparsely estimated static weights to the entire image, such that denser information can be used for registration.

References

[1] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 2100–2106.

[2] R. A. Newcombe *et al.*, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[3] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1498–1505.

[4] D. Gutierrez-Gomez, W. Mayol-Cuevas, and J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 83–89.

[5] S. Klose, P. Heise, and A. Knoll, "Efficient compositional approaches for real-time robust direct visual odometry from RGB-D data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1100–1106.

[6] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5724–5731.

[7] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking," *J. Visual Commun. Image Representation*, vol. 25, no. 1, pp. 137–147, 2014.

[8] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3748–3754.

[9] T. Tykkälä *et al.*, "Direct iterative closest point for real-time visual odometry," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 2050–2056.

[10] M. Meilland *et al.*, "A spherical robot-centered representation for urban navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 5196–5201.

[11] Y. Wang and S. Huang, "Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis.*, 2014, pp. 1841–1846.

[12] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auton. Syst.*, vol. 89, pp. 110–122, 2017.

[13] D.-H. Kim and J.-H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1565–1573, Dec. 2016.

[14] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Experimental robotics*. New York, NY, USA: Springer, 2014, pp. 477–491.

[15] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 647–663, 2012.

[16] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.

[17] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Robotics-DL Tentative*. Bellingham, WA, USA: International Society for Optics and Photonics, 1992, pp. 586–606.

[18] S. Izadi *et al.*, "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 559–568.

[19] D.-H. Kim, S.-B. Han, and J.-H. Kim, "Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment," in *Robot Intelligence Technology and Applications 3*. New York, NY, USA: Springer, 2015, pp. 11–26.

[20] T.-S. Leung and G. Medioni, "Visual navigation aid for the blind in dynamic environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2014, pp. 565–572.

[21] L. Bose and A. Richards, "Fast depth edge detection and edge based RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1323–1330.

[22] C. Choi, A. J. Trevor, and H. I. Christensen, "RGB-D edge detection and edge-based registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1568–1575.

[23] S. Li and D. Lee, "Fast Visual Odometry using intensity-assisted iterative closest point," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 992–999, Jul. 2016.

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[25] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: a comparison of four major algorithms," *Mach. Vis. Appl.*, vol. 9, no. 5–6, pp. 272–290, 1997.

[26] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[27] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, Winter 2010x.