KeySLAM: Robust RGB-D Camera Tracking Using Adaptive VO and Optimal Key-Frame Selection

Kyung Min Han and Young J. Kim

***Abstract*—We propose a novel RGB-D camera tracking system that robustly reconstructs hand-held RGB-D camera sequences. The robustness of our system is achieved by two independent features of our method: adaptive visual odometry (VO) and integer programming-based key-frame selection. Our VO method adaptively interpolates the camera motion results of the direct VO (DVO) and the iterative closed point (ICP) to yield more optimal results than existing methods such as Elastic-Fusion. Moreover, our key-frame selection method locates globally optimum key-frames using a comprehensive objective function in a deterministic manner rather than heuristic or experience-based rules that prior methods mostly rely on. As a result, our method can complete reconstruction even if the camera fails to be tracked due to discontinuous camera motions, such as kidnap events, when conventional systems need to backtrack the scene.**

***Index Terms*—Mapping, RGB-D Perception, and SLAM.**

I. Introduction

CAMERA Tracking is a challenging research problem in robotics and computer vision communities. The emergence of the RGB-D camera made vision-based camera tracking problems even more prevalent. The depth sensor was originally invented for gaming devices; however, its application is now more diverse, and no longer limited to gaming devices. Because of the accurate depth information, in particular, 3D reconstruction results using depth sensors are much more robust and reliable than RGB imaging.

Numerous studies have been performed on RGB-D camera tracking [1]–[5]. The main structure of these systems is based on real-time VSLAM where camera poses and maps are constructed in response to the real-time camera motion, followed by pose optimization for loop-closing events. A popular approach to the VSLAM problem is the pose-graph formulation where an edge between two vertices represent a pose constraint including an odometry constraint between two ordinary frames and a loop closing constraint between two key-frames. Therefore, selecting good key-frames and a good initial camera pose impacts the final tracking result significantly.

Our goal in this research is proposing a robust RGB-D camera tracking system by addressing the following two questions:

- How to robustly perform RGB-D visual odometry (VO)?
- How to select good key-frames from a sequence of camera frames?

Our answer to the first question is revisiting the hybrid VO method, which takes advantage of both structure and texture information in RGB-D scenes, and making it adaptive to the tracking environment. That is, our method autonomously determines the weighting factors to combine depth map and image texture information in an RGB-D image without human intervention.

In order to answer the second question, a novel integer programming based set covering scheme is proposed to robustly identify an optimal subset of frames to cover the entire image frames. Specifically, we construct an affinity matrix for the input frames by establishing *semi-dense* putative matches among them, where full key-point matches are not required. Then, the affinity relationship is fed back to the set covering scheme to find globally optimal key-frames.

Note that many existing camera tracking systems rely on heuristic criteria or a locally optimum decision function to determine key-frames, but the accuracy and reliability of the reconstructed map is rather questionable.

In contrast, our system provides more reliable and accurate tracking results while sacrificing some performance loss, and thus it is more suitable for offline robotic applications, for instance, map building for robotic navigation, robust 3D object reconstruction, and automatic visual inspection.

To measure the robustness of our tracking system, we employed mean relative pose error (RPE) and mean absolute trajectory error (ATE) results over the 25 TUM benchmark sequences. As shown in Section VI, our method exceeds the baseline methods in terms of these two metrics. In summary, the main contributions of our paper are:

- Adaptive VO method using a novel pose estimation formulation
- Novel integer programming (IP) based formulation for optimal key-frame selection
- More robust and accurate results compared to the state of the art VO and SLAM systems using public benchmark datasets

Manuscript received March 10, 2020; accepted July 9, 2020. Date of publication September 25, 2020; date of current version October 7, 2020. This letter was recommended for publication by Associate Editor L. Paull and Editor S. Behnke upon evaluation of the Reviewers' comments. This work was supported in part by the ITRC/IITP Program (IITP-2020-0-01460), and in part by the NRF (2018R1A6A3A11049832 and 2017R1A2B3012701) in South Korea. *(Corresponding author: Young J. Kim.)*

The authors are with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, South Korea (e-mail: hankm@ewha.ac.kr; kimy@ewha.ac.kr).

This letter has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3026964

II. Related Work

An ordinary camera tracking system consists of front-end and back-end modules that are independent of each other. The front-end is responsible for estimating the camera trajectory by establishing the pixel correspondences between two consecutive frames, often referred to as visual odometry (VO). In parallel, the back-end of the system runs to correct the drift error possibly accumulated by the front-end of the system.

A. The Front-End

VO dictates the overall performance of a camera tracking system because a robust and accurate VO minimizes the drift error accumulation which can impact on the outcome of the entire system.

VO is often categorized into two methods: direct methods and feature-based methods. The direct methods [2] establish dense correspondences by imposing a brightness consistency constraint. The camera pose is directly recovered by minimizing the pixel intensity error. Conversely, the feature-based methods [4], [6] locate salient pixels in the *image scale-space* followed by a matching process, such as a normalized cross correlation (NCC) measurement, the sum of squared differences (SSD), or descriptor distance. Imposing a camera epipolar constraint can further eliminate outlier matches in order to achieve more accurate VO estimation.

When the depth information is available as prior knowledge, we do not need to be confined in 2D-to-2D image alignment. Motion estimation from 3D-to-3D correspondence is feasible [7]. Furthermore, 3D point clouds can be directly registered using ICP-like methods [8]. Besides, there are RGB-D SLAM methods that rely on ICP registrations [9], [10].

B. The Back-End Optimizer

The importance of back-end module cannot be overstated because VO may accumulate the pose error for long-term operations. One way of correcting the pose error is frequently running non-linear optimization of cost functions built by relative camera pose constraints or by pixel reprojection error. The former refers to pose-graph-optimization [11], and the latter refers to bundle adjustment [12].

C. Global Relocalization

Camera tracking systems are equipped with an additional step for the global relocalization task. VO may lose tracking of the camera pose owing to numerous factors, such as insufficient feature matching or motion blur. Besides, the camera sensor could be kidnapped, and then released at a long base-lined location w.r.t. the kidnapped location. Furthermore, many loop closing constraints may exist among the key-frames with poses that must be corrected by a back-end process. While a bag of words (BoW) [13]-based approach is a well-known efficient method for the global camera localization, recent approaches [14], [15] are capable of coping with drastic appearance changes in scenes.

III. Overview

We use a sequential pipeline for our tracking system consisting of two consecutive phases: tracking phase (the front-end) and optimization phase (the back-end). The tracking phase estimates the primitive poses of input frames and their similarities while the optimization phase identifies a set of key-frames and then refines their poses.

Fig. 1 illustrates the proposed pipeline.

The next few sections elaborate on some crucial steps in the pipeline. Sections IV-A and IV-B explain VO to construct the affinity matrix and our feature matching method in the pipeline, respectively. Section V-A explains how key-frames are selected, and Section V-B describes our final pose optimization procedure.

Before delving into details, we first explain some conventions and assumptions that will be used throughout the paper. The pose of a camera or a rigid body transformation is represented by a matrix $\mathbf{C} \in SE(3)$

$$\mathbf{C} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{1}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ represent the orientation and position of the pose, respectively.

$\mathbf{X} \in \mathbb{R}^3$ represents a point in the 3D space, and $\mathbf{x} \in \mathbb{R}^2$ represents the projected point of $\mathbf{X}$ in 2D pixel space. The process of 3D to 2D projection $\mathbf{X} \rightarrow \mathbf{x}$ is governed by a camera pinhole projection model $\mathbf{x} = \boldsymbol{\pi}(\mathbf{C}, \mathbf{X})$, where $\boldsymbol{\pi}(\cdot)$ projects a 3D point in space onto a pixel point of a camera $\mathbf{C}$. Conversely, $\mathbf{X} = \pi^{-1}(\mathbf{x}, \mathcal{Z}(\mathbf{x}))$ represents an inverse projection, where $\mathcal{Z}(\mathbf{x})$ is the depth of $\mathbf{x}$.

IV. Tracking Phase

A. Visual Odometry

1) Cost Functions: The point to plane ICP residual ρ for the kth point $\mathbf{X}_k$ is defined as

$$\rho_k = (\mathbf{X}_k - \exp(\hat{\xi})\mathbf{C}\mathbf{X}'_{\mathrm{k}}) \cdot \mathbf{n}_{\mathrm{k}} \tag{2}$$

where $\hat{\xi} \in se(3)$ is the twist representing a rigid motion. $\mathbf{X}'_k$ and $\mathbf{n}_k$ are the corresponding 3D point and the normal vector, respectively. Then, the ICP cost function is given as:

$$E_i = \min_{\hat{\xi}} \sum_k \psi(\rho_k) ||\rho_k||^2 \tag{3}$$

where $\psi(\cdot)$ is a robust weight function which minimizes the impact of false correspondences [16]. As for DVO, a pixel intensity residual δ is defined as

$$\delta_k = I_1(\mathbf{x}_k) - I_2(\mathbf{x'}_k) \tag{4}$$

where $\mathbf{x'}_k$ is the pixel correspondence of $\mathbf{x}_k$ computed by a warping operation; that is,

$$\mathbf{x}' = \pi(\mathbf{C}, \pi^{-1}(\mathbf{x}, \mathcal{Z}(\mathbf{x}))). \tag{5}$$

The resulting pixel intensity cost function is given as

$$E_d = \min_{\hat{\xi}} \sum_k \psi(\delta_k) ||\delta_k||^2. \tag{6}$$

Solving (3) or (6) involves iteratively running the re-weighted least square (IRLS) algorithm.

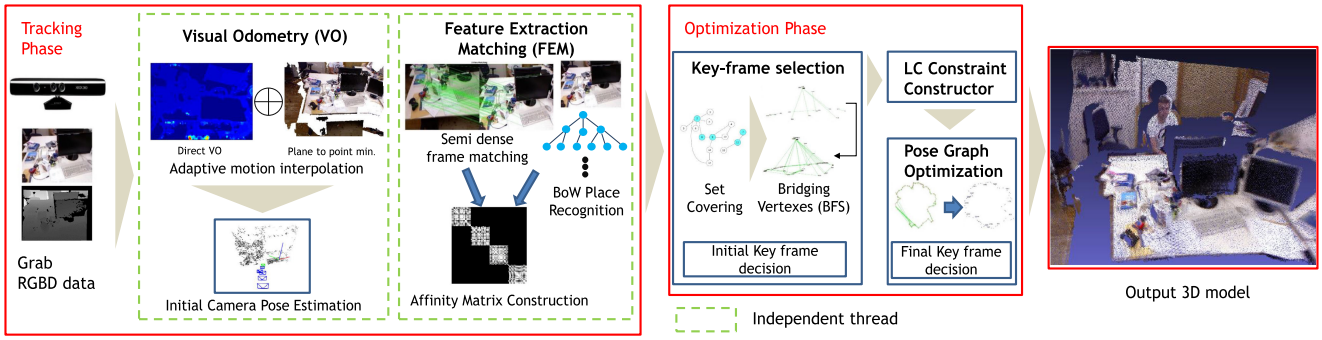


Fig. 1. Pipeline of the proposed tracking system. The tracking phase estimates incremental camera motions and simultaneously establishes similarities among the input frames in an affinity matrix. Then, during optimization phase, key-frames are identified followed by a pose graph optimization.

A method like [17] co-optimizes Eq. 3 and Eq. 6 using a pre-determined weight parameter to estimate the camera pose. Since it is known that ICP and DVO show different characteristic behavior depending on the availability of textures or depth structures in an input scenario [18], our algorithm leverages this fact by adaptively adjusting the weight parameter according to the relative fitness between DVO and ICP at each IRLS iteration. As a result, our VO algorithm can generate more robust and accurate results than [17].

2) Frame-to-Frame Camera Pose Estimation: We downsampled images to four pyramid levels to implement coarse to fine search strategy; that is, beginning from the coarsest pyramid level (80×60 for a TUM sequence,) we iterate the IRLS to find the best camera pose at the current pyramid level.

Then, we assess the quality of the estimated camera motions for n number of correspondences using

$$e_d = \sum_{k}^{n} ||\mathbf{X}_k - \exp(\hat{\xi_d})\mathbf{X'}_k||_2$$

$$e_i = \sum_{k}^{n} ||\mathbf{X}_k - \exp(\hat{\xi_i})\mathbf{X'}_k||_2 \tag{7}$$

where $\mathbf{X}_k$ and $\mathbf{X'}_k$ are the corresponding pair of two 3D points, $\hat{\xi_d}$ and $\hat{\xi_i}$ are the estimated camera twists from the IRLS of DVO and ICP, respectively. Here, the subscript d refers to DVO while i refers to ICP. Then, we introduce a gain parameter λ_d

$$\lambda_d = \begin{cases} \frac{e_i}{e_i + e_d} & \text{if } n \geq \nu \\ 0 & \text{if } n < \nu \end{cases} \tag{8}$$

where ν is the number of meaningful Sobel filter responses used for DVO process - we set ν to 2% of the image resolution, which was found empirically. The purpose of using ν is to trust ICP and discard DVO when the input image has a low amount of textures. Given λ_d, $\lambda_i = 1 - \lambda_d$, we interpolate the two camera motions by

$$\mathbf{C}_{avg} = \mathcal{I}(\exp(\hat{\xi_i}), \exp(\hat{\xi_d}), \lambda_i, \lambda_d) \tag{9}$$

where $\mathcal{I}(\cdot)$ is a weighted interpolation function consisting of two consecutive steps: 1) orientation interpolation using quaternion Slerp [19] followed by 2) position interpolation in Euclidean space. The interpolated camera motion is used as an initial guess for the next pyramid level. A pseudo-code for our pose estimation procedure is also given in Algorithm 1.

Algorithm 1: Camera Pose Estimation Procedure.

```
Function RGBDODOM()
    initialize the camera motion C;
    for each pyramid level do
        for each IRLS iteration do
            solve Eq. 3 for ξ_d;
            solve Eq. 6 for ξ_i;
            recompute ψ(ρ) and ψ(δ)
        end
        compute λ_i and λ_d using Eq. 8 ;
        estimate C_avg using Eq. 9 ;
        C ← C_avg
    end
```

B. Feature Extraction and Matching (FEM)

Semi-Dense Frame Matching: We build an affinity matrix by executing multiple wide baseline matchings for each incoming frame. We observed that this part of the system is a very time-consuming task because the incoming k^{th} frame is exhaustively matched up to $(k-\eta)^{th}$ frame. Of course, it would be beneficial for the algorithm if we can set up full matching. However, such types of set up would hinder the efficiency of the system. Besides, we found that the full matching is unnecessary because the BoW match can be performed before the putative matching to locate possible image pairs existing beyond the k^{th} frame. We used CudaSIFT [20] to implement this part of our system in order to speed up the system. Moreover, we run the feature matching task as an independent thread in parallel to our VO thread to alleviate the time-consuming problem.

The result of η image matching is incorporated into the key-frame decision module in order to select optimal key-frames. The detailed explanation of the key-frame decision step is presented in Section V-A.

V. Optimization Phase

A. Key-Frame Selection

As a trivial solution, all the input frames may be selected as key-frames at the cost of high computational time. Conversely, selecting too few key-frames could create a brittle map, including disconnected sub-graphs owing to potential weak links among the key-frames. Therefore, a small number of key-frame set, maintaining an adequate number of feature matches is preferred.

In pose-graph SLAM, the unknown camera poses are considered a graph $G = (V, E)$ where a set of vertices V expresses camera poses created from its trajectory, and a set of edges E represents visual associations between the vertices in V. $v_i \sim v_j$ denotes that $v_i \in V$ and $v_j \in V$ are connected by an edge e_{ij}. In our work, we measured the number of feature matches between v_i, v_j to determine if they are connected. We empirically found that the number of minimum matching threshold needs to be 36 for TUM sequences.

Set Cover Formulation: Our goal is to select a minimal set of vertices V^* such that the selected vertices can cover the entire vertex set V. The selected key-vertices would cover their neighbors as well as themselves. A vertex v_k and its connected vertices form a sub-graph S_k of G. Therefore, multiple selections of key-vertices or sub-graphs would eventually cover the universe - i.e. the entire set of frames $\mathbb{U}$.

$$\bigcup_{V_i \in V^*} V_i = \mathbb{U}. \tag{10}$$

The formulation above is a set covering problem [21]. There are two main objectives for this formulation. First, we want to cover all vertices by key vertices. Second, a selected key vertex $v_i \in V^*$ should be connected to another key vertex $v_j \in V^*$, $i \neq j$ for pose graph optimization (PGO) step.

The weighted selection of the vertices should be optimized by minimizing a cost function:

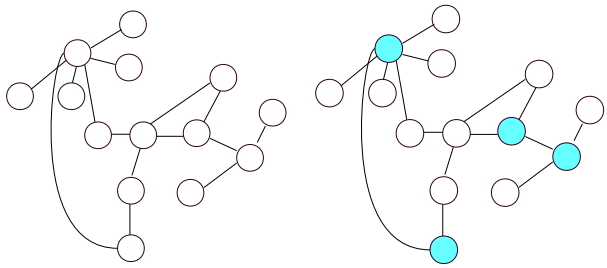
$$\min_x \quad \sum_{x_j \in V} w_j \cdot x_j \tag{11a}$$

$$\text{s.t.} \quad \sum_{j}^{N} a_{ij} x_j \geq 1 \quad i, j = 1...N \tag{11b}$$

$$\sum_{k} x_k - x_i \geq 0 \quad k \neq i, V_k \in S_i \tag{11c}$$

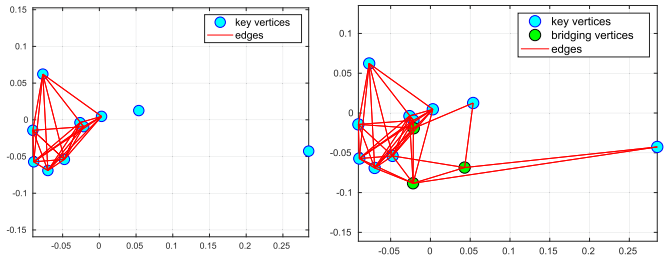
$$\sum_{k} x_k \geq c, \tag{11d}$$

where the cost $w_j = 1/|S_j|$ is the inverse cardinality of a sub-group belonging to v_j. The decision variable x_j is set to one if v_j is selected; otherwise zero. The binary coefficient a_{ij} is set to one iff j^{th} vertex appears in the i^{th} group. Eq. 11b is a set-covering constraint which ensures the constraint in Eq. 10, and Eq. 11c avoids selecting a single isolated vertex. Eq. 11d is an optional third constraint that guarantees to select at least c number of vertices in order to avoid the case where a single key-vertex covers the entire vertices because we cannot proceed to PGO with a single key-frame. We utilized the Gurobi LP solver [22] to solve the the set-covering problem. Fig. 2 illustrates the proposed vertex cover process for a simple example.



(a) A graph (b) Key-vertices

Fig. 2. The nodes in cyan color are the key-vertices found by our algorithm. Note that the key-vertices and their direct neighboring vertices cover the universe set.



(a) Without bridging frames (b) With bridging frames

Fig. 3. The key-frame selection result for TUM *fr1 xyz* sequence: (a) The set-covering procedure initially finds key-vertices (cyan). Two of them are disconnected. (b) The BFS procedure locates three bridging vertices (green) to form a single connected component. The key-vertices and the bridging vertices constitute a set of key-frames.

1) Locating Bridging Vertices: Although the key-vertices selected from Eq. 11 cover the entire vertices, we observed a case where our algorithm finds multiple disconnected sets. In order to guarantee a single connected component of G, it is necessary to locate additional bridging vertices. To this end, we employed breadth first search (BFS) to locate the bridging vertices. Note that the set of selected vertices from Eq. 11 together with the identified bridging vertices correspond to *key-frames*. Fig. 3 shows a set of key-frames formed by key-vertices and bridging vertices. Note that all key-frames are formed as a single connected component.

B. Pose Optimization

1) Initializing and Optimizing Loop Closing Constraints: Thanks to the affinity matrix constructed in the tracking phase, we can locate candidates of loop-closing view-pairs. We use Arun's [7] 3D to 3D registration method with a special treatment for the reflection case suggested by [23]. Given a 3D point pair, we initially estimate the relative pose constraint from a standard RANSAC protocol. In order to estimate accurate pose constraints, we further optimize the initial estimations by executing motion-only bundle adjustment followed by a median absolute deviation (MAD) based outlier elimination step. We

first compute σ of MAD by

$$\sigma = 1.4826 r_i, \tag{12}$$

where r_i is the absolute deviation of the reprojection error $\mathbf{v}$ from its median residual - i.e. $r_i = |\|\mathbf{v}\|_2 - \text{median}(\|\mathbf{v}\|_2)|$. Subsequently the inlier matches are computed by the constraint:

$$r_i < T\sigma, \tag{13}$$

where T is a scale parameter which is set to 2.5 in our implementation. Subsequently, we estimate the camera pose of j^{th} frame by

$$\min_{\mathbf{C}_j} \sum_i ||\mathbf{x}_{ji} - \boldsymbol{\pi}(\mathbf{C}_j, \mathbf{X}_{j-1,i})|| \tag{14}$$

where $\mathbf{x}_{ji}$ is i^{th} key-point in j^{th} image frame, and $\mathbf{X}_{j-1,i}$ is the corresponding 3D point measured in the previous frame. After setting up the relative pose constraints, we proceed to a pose graph optimization (PGO) using Ceres-solver [24].

VI. Experiments and Results

We validated our method by carrying out several experiments on the TUM benchmark RGB-D data set [25]. This data set consists of RGB-D image sequences collected with the ground truth camera poses and many useful tools to assess the performance of RGB-D tracking systems. While the data set includes many different kinds of scenarios including dynamic objects, in this paper, we focused on hand-held sequences in a static environment where the captured objects are within the depth range of the Kinect sensor – i.e., 6 meters. We found 25 of such sequences that could be categorized into the description above.

1) Ablation Study: To justify the effectiveness of our VO and key-frame selection, we measured how each component affects the final tracking performance. In this study, we used [17], which is a joint optimization type of VO, as a VO alternative. Besides, we employed the key-frames selected by ORBSLAM2 as a substitute for our key-frame selection module. To recap, we created four test cases from two different VO methods and two different key-frame selection methods. Each test case was evaluated through the TUM sequences, which are reported in Table I. Overall, the proposed pipeline was 43%, 60%, and 160% more accurate than the 2nd, 3rd, and 4th test cases, respectively.

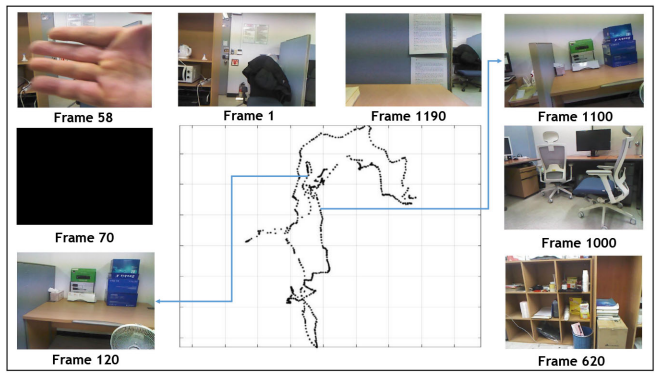
2) Robustness to Discontinuous Sequence: Fig. 4 highlights the advantage of using globally optimal key-frames. We collected a new RGB-D sequence that includes a kidnap event. In this sequence, ORBSLAM2 encounters a tracking failure in the beginning frame of the sequence, caused by the kidnapping event, thus invoking a recovery mode. Subsequently, ORBSLAM2 failed to reconstruct a full map in this scenario while our method completed a full tracking.

3) Comparison to Other VO Methods: To evaluate our VO method, we collected five different states of the art methods: 1) ICP, 2) Fovis [6], 3) DVO [26], 4) Park *et al.* [27] and 5) Whelan *et al.* [17]. Then, we tested their performance on TUM benchmark sequences. Table II reports the performance of the target methods in terms of the relative pose error (RPE) metric. Our method outperforms the target methods in terms of mean RPE errors. For example, our method was 5.8% more accurate (or produce less RPE error) than ICP in terms of mean relative translation error. Compared to image based approaches, our method marked 78.8% more accurate than DVO and 59.6% more than FOVIS. Compared to hybrid type VOs, our method achieved approximately 11.5% more than [17] and 34.6% more than [27].

4) Overall Comparison to Other Systems: To evaluate the overall accuracy performance of our system, we quantitatively

TABLE I
Comparisons Among Different Test Cases: The Bold-Faced Numbers Represent Having min ATEs Among the Four Variations. We Omitted fr3-cabinet, fr3-large-cabinet, fr3-nostr-notxt-far, and fr3-nostrt-notxt-near-loop in This Study Due to the Lack of Both Textures And Structures

Sequences	RMSE of absolute trajectory error (m)			
	Our VO + Our KF	Whelan et al. [17] + Our KF	Our VO + Heur KF [4]	Whelan et al. [17] + Heur KF [4]
fr1-360	0.0947	0.1129	0.0966	**0.0817**
fr1-floor	0.2697	0.2846	**0.2330**	0.2388
fr1-desk	0.1379	0.0954	0.1007	**0.0944**
fr1-desk2	**0.0503**	0.1015	0.0716	0.0943
fr1-room	0.0779	0.1475	**0.0759**	0.1816
fr2-360-kidnap	**0.7162**	1.4302	0.7348	1.4330
fr2-desk	0.1044	**0.0960**	0.1280	0.0971
fr3-long-office	**0.0374**	0.1173	0.1272	1.6711
fr1-xyz	**0.0190**	0.0211	0.0208	0.0201
fr2-xyz	**0.0164**	0.0353	0.0806	0.0394
fr1-rpy	**0.0255**	0.0260	0.0310	0.0279
fr2-rpy	**0.0151**	0.0175	0.0206	0.0188
fr1-plant	**0.0984**	0.1234	0.1303	0.1754
fr1-teddy	**0.0763**	0.1044	0.1601	0.2538
fr3-teddy	0.2872	**0.2346**	0.3357	0.3500
fr3-str-notxt-far	0.1538	0.1625	0.1717	**0.1534**
fr3-nostr-txt-near-loop	**0.0634**	0.0991	1.3000	1.5336
fr3-str-notxt-far	0.1127	0.1127	**0.0891**	0.0891
fr3-str-notxt-near	0.1379	0.1379	**0.1293**	0.1293
fr3-str-txt-far	0.0214	0.0220	0.0211	**0.0206**
fr3-str-txt-near	0.0243	0.0954	**0.0236**	0.0748
Mean ATE	**0.1209**	0.1704	0.1944	0.3228



(a) A kidnapping sequence: The middle image is a camera trajectory estimated by our method while the other images are sample scenes of this sequence. The kidnap event begins at the 58th frame and ends at the 120th frame.

(b) ORBSLAM2 (c) Our Method

Fig. 4. The tracking result comparison between our method and ORBSLAM2 for a late kidnap recovery sequence. ORBSLAM2 finished with a partial reconstruction while our method completed a full reconstruction.

TABLE II
RELATIVE POSE ERROR OF VO METHODS ON TUM BENCHMARK SEQUENCES: EACH DATA INDEX CORRESPONDS TO 1) FR1-360, 2) FR1-FLOOR, 3) FR1-DESK, 4) FR1-DESK2, 5) FR1-ROOM, 6) FR2-360-KIDNAP, 7) FR2-DESK, 8) FR3-LONG-OFFICE, 9) FR1-XYZ, 10) FR2-XYZ, 11) FR1-RPY, 12) FR2-RPY, 13) FR1-PLANT, 14) FR1-TEDDY, 15) FR3-TEDDY, 16) FR3-CABINET, 17) FR3-LARGE-CABINET, 18) FR3-NOSTR-NOTXT-FAR, 19) FR3-NOSTRT-NOTXT-NEAR-LOOP, 20) FR3-NOSTRT-TXT-FAR, 21) FR3-NOSTRT-TXT-NEAR-LOOP, 22) FR3-STRUCT-NOTXT-FAR, 23) FR3-STRUCT-NOTXT-NEAR, 24) FR3-STRUCT-TXT-FAR, AND 25) FR3-STRUCT-TXT-NEAR, RESPECTIVELY. NOTE THAT FR2-360-KIDNAP IS NOT USED FOR TESTING THE VO ALGORITHMS. THE BOLD-FACED NUMBERS REPRESENT HAVING MIN RMSE AMONG THE SIX METHODS

Sequence	RMSE of relative rotation error (°/s)						RMSE of relative translation error (m/s)					
	ICP	DVO [26]	Whelan et al. [17]	FOVIS [6]	Park et al. [27]	Our method	ICP	DVO [26]	Whelan et al. [17]	FOVIS [6]	Park et al. [27]	Our method
1	0.4503	0.3797	0.3966	**0.3680**	0.4099	0.3792	0.0057	0.0075	0.0059	0.0076	0.0068	**0.0051**
2	0.2881	0.2598	**0.2284**	0.2734	0.2379	0.2293	0.0061	0.0040	**0.0031**	0.0042	0.0031	0.0038
3	0.7107	**0.4593**	0.6430	0.4813	0.5270	0.5453	0.0054	0.0068	0.0066	0.0069	0.0079	**0.0051**
4	0.7692	0.5156	0.6522	**0.4965**	0.5943	0.5904	0.0060	0.0075	0.0066	0.0073	0.0084	**0.0059**
5	0.4094	**0.3387**	0.3543	0.3424	0.3494	0.3396	0.0038	0.0054	0.0041	0.0059	0.0047	**0.0036**
7	0.2609	**0.2067**	0.2803	0.2150	0.2322	0.2671	0.0021	0.0022	0.0025	0.0027	0.0026	**0.0021**
8	0.3159	**0.2098**	0.2655	0.2227	0.2710	0.1997	0.0026	0.0040	0.0029	0.0044	0.0042	**0.0024**
9	0.4161	**0.3232**	0.3702	0.3450	0.3374	0.3369	**0.0027**	0.0046	0.0036	0.0052	0.0044	0.0029
10	0.1976	0.1826	0.1772	0.1812	0.1924	**0.1798**	**0.0011**	0.0022	0.0016	0.0020	0.0021	0.0015
11	0.7320	**0.4836**	0.6575	0.6842	0.6284	0.5692	0.0041	0.0058	0.0044	0.0068	0.0075	**0.0037**
12	0.1529	0.1338	0.1262	0.1287	0.1396	**0.1188**	**0.0010**	0.0020	0.0014	0.0018	0.0018	0.0014
13	0.5021	0.3139	0.4116	**0.3052**	0.3722	0.3649	0.0042	0.0056	0.0042	0.0053	0.0044	**0.0037**
14	0.6956	**0.3952**	0.5588	0.4108	0.5962	0.5001	0.0064	0.0089	0.0068	0.0092	0.0092	**0.0058**
15	0.7705	**0.4454**	0.7014	0.4116	1.2075	0.5031	0.0066	0.0052	0.0059	**0.0047**	0.0114	0.0055
16	0.5187	**0.3617**	0.5032	0.4415	0.5763	0.4913	0.0055	0.0075	**0.0050**	0.0090	0.0070	0.0054
17	0.4104	0.5979	0.3929	**0.2855**	0.4545	0.3342	0.0078	0.0239	**0.0078**	0.0138	0.0093	0.0092
18	0.6379	0.8606	0.6459	**0.5983**	0.6749	0.6463	**0.0113**	0.0250	0.0116	0.0138	0.0132	0.0116
19	0.8784	1.2136	0.8723	**0.8242**	0.8810	0.8822	0.0145	0.0248	0.0145	**0.0126**	0.0139	0.0145
20	0.5753	0.5553	0.5628	0.6103	0.6180	**0.5169**	**0.0139**	0.0220	0.0182	0.0240	0.0191	0.0146
21	0.5722	**0.4542**	0.5209	0.4787	0.4621	0.4798	0.0094	0.0077	0.0097	0.0083	0.0087	**0.0071**
22	0.2312	0.3080	**0.2273**	0.3664	0.2292	0.2336	0.0031	0.0103	**0.0026**	0.0129	0.0030	0.0031
23	0.3416	0.6519	0.3395	0.7025	**0.3380**	0.3418	0.0022	0.0125	**0.0020**	0.0127	0.0022	0.0022
24	0.4319	0.3094	0.3833	0.3162	**0.2876**	0.2990	**0.0039**	0.0100	0.0046	0.0101	0.0065	0.0041
25	0.5266	0.3654	0.5131	**0.3732**	0.4593	0.4196	**0.0033**	0.0073	0.0043	0.0072	0.0064	0.0038
Mean RMSE	0.4915	0.4302	0.4494	0.4110	0.4615	**0.4070**	0.0055	0.0093	0.0058	0.0083	0.0070	**0.0053**

measured the pose errors w.r.t. the ground truth poses using the absolute trajectory error (ATE) metric. ATE measures absolute location differences between the ground truth trajectory and the estimated one by aligning the two coordinate systems on top of each other [25]. We compared our full pipeline with three different state-of-the-art SLAM systems: ORBSLAM2 in RGB-D mode, Elastic-Fusion, and DVOSLAM. In the tests, if the number of tracked camera poses are below 50% of the full frame, we counted the case as a failure.

Table III quantitatively shows the performance of our method with reference to the state-of-the-art works. Note that our method outperforms Elastic-Fusion and DVOSLAM. ORBSLAM2 is better than our method on texture-rich sequences but not so much as on texture-free sequences (e.g. *fr3-structure-notexture-far*, *fr3-structure-notexture-near*, *fr3-cabinet*). ORBSLAM2 was not able to initialize a map or failed to complete full tracking in such scenarios. Fig. 5 shows the reconstructed 3D point cloud models, which is a byproduct of using our method. Fig. 6 shows comparisons between the estimated camera trajectories and their ground truths along with time vs. error plots.

5) Timing Test: Fig. 7 reports the timing experiments we have carried out. As it is shown, the speed of our system is highly correlated with BoW matching score β and the window size for dense matching η. In fact, β has much more impact on the tracking results than η because β finds global loop closing candidates while η is only for neighboring frames. Decreasing β causes the system to inspect a larger number of loop closing candidates. Subsequently, the system becomes more robust while loses its efficiency. However, as Fig. 7(c) shows, after a certain point, lowering β does not affect the quality of camera tracking anymore while the computational cost increases. We chose $\beta = 0.1$ and $\eta = 48$ based on these experiments.

TABLE III
COMPARISONS OF DIFFERENT RECONSTRUCTION SYSTEMS

Sequence	RMSE of absolute trajectory error (m)			
	ORBSLAM2 [4]	DVOSLAM [28]	Elastic-Fusion [29]	Our Method
1	0.2125	0.1014	0.2648	**0.0947**
2	**0.0133**	0.1747	0.4455	0.2697
3	0.0337	0.0284	**0.0221**	0.1379
4	**0.0212**	0.0379	0.0667	0.0503
5	**0.0394**	0.0851	0.2232	0.0779
6	**0.0754**	9.3614	1.9631	0.7162
7	**0.0077**	0.0637	0.0820	0.1044
8	**0.0089**	0.0232	0.0187	0.0374
9	0.0146	0.0144	**0.0096**	0.0190
10	0.0148	0.0152	**0.0129**	0.0164
11	**0.0243**	0.0289	0.0341	0.0255
12	0.0104	0.0164	**0.0096**	0.0151
13	**0.0206**	0.0342	0.0430	0.0984
14	0.0633	**0.0432**	0.0988	0.0763
15	**0.0112**	0.1839	0.2012	0.2872
16	N/A	**0.1811**	1.0056	0.4741
17	**0.0906**	0.1738	0.4977	0.4820
18	N/A	**0.8056**	1.7340	0.8136
19	N/A	1.5970	**1.4705**	1.6032
20	**0.0318**	0.0700	0.3009	0.1538
21	**0.0234**	1.8931	0.0328	0.0634
22	N/A	0.0319	**0.0252**	0.1127
23	N/A	**0.0752**	0.8268	0.1379
24	**0.0110**	0.0150	0.0112	0.0214
25	**0.0123**	0.0291	0.0315	0.0243
Mean ATE of texture rich seq	**0.0350**	0.1596	0.1266	0.1386
Mean ATE of all seq	N/A	0.6034	0.3773	**0.2365**

We compared our full tracking pipeline with ORBSLAM2 and DVOSLAM on *fr1 360* and *fr3 long office*.

(a) fr3 long office household

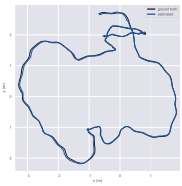
(b) fr3-structure-texture-far

(c) fr1-360

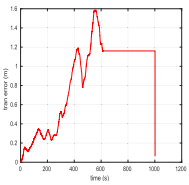
(d) fr3-no-structure-texture-withloop

Fig. 5. The final reconstructed 3D models of TUM benchmark sequences.

(a) fr3-long-office

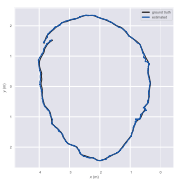


(b) cam trajectories

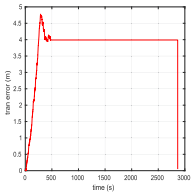


(c) tran error vs. time

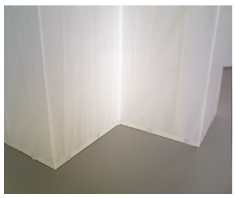
(d) fr3-nostr-texture-near-withloop



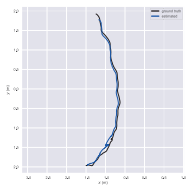
(e) cam trajectories



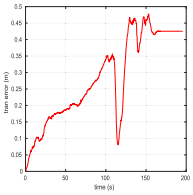
(f) tran error vs. time



(g) fr3-str-notexture-far

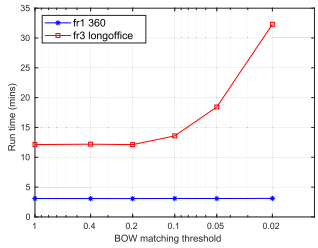


(h) cam trajectories

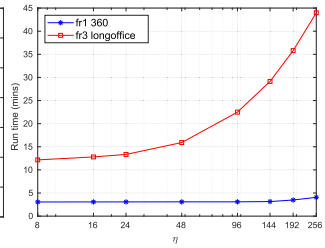


(i) tran error vs. time

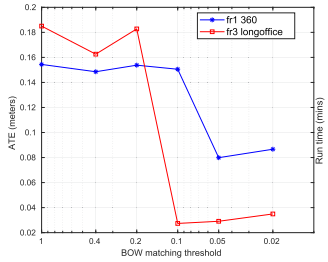
Fig. 6. Each row shows an input image, the estimated camera trajectory and the time vs. error analysis plots. (i) diverged because there is no loop closing event in this sequence. Here, we used [30] to draw camera trajectories.



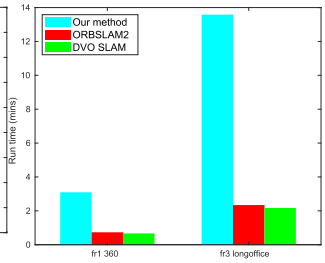
(a) Runtime vs β



(b) Runtime vs η



(c) ATE vs β



(d) Runtime comparisons

Fig. 7. Timing experiments: (a) η is fixed to 48 (b) β is fixed to 1.0.

Our system is approximately 4.8 times slower than these two methods due to two reasons. First, our VO method uses multiple ICP iterations while our competitive methods do not. This process might be considerably accelerated using GPUs just

like Kinect-Fusion [31] and Elastic-Fusion. Also, our method executes semi-dense matching for finding optimum key-frames while other methods use less expensive heuristic or locally optimal decision rules.

VII. Conclusion

In this letter, a new RGB-D camera tracking method is presented. As opposed to many existing VSLAM systems, we formulated an algorithmic approach to determine a set of key-frames. Loop closing constraints are identified using a set of key-frames then further optimized by robust MAD estimator. Our VO method interpolates DVO and ICP results to estimate a more optimal camera motion while conventional methods solve a joint optimization equation based on a pre-determined weight. The proposed method was validated by performing an extensive experiment using TUM RGB-D hand-held sequences. As it is reported in the paper, our method achieved reliable and robust tracking, and the tracking accuracy is better than the state-of-the-art methods without employing a full bundle adjustment.

As a current limitation of our system, we observed that constructing an affinity matrix consumes a large amount of computational resources proportional to the growth of SIFT matching queue, which explains relative slower performance compared to existing methods. In the future, we would like to optimize our system using a method like the GPU based cascaded hashing matching [32].

References

[1] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D slam system," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 1691–1696.

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

[3] A. Concha and J. Civera, "RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6756–6763.

[4] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[5] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*. vol. 36, pp. 416–446, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[6] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. Int. Symp. Robot. Res.*, 2011, pp. 235–252.

[7] 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.

[8] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[9] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 2305–2310.

[10] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proc. Robot.: Sci. Syst.*, Jun. 2009, pp. 435–452.

[11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.

[12] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *Proc. 11th Eur. Conf. Comput. Vision: Part II*, 2010, pp. 29–42.

[13] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[14] O. Vysotska and C. Stachniss, "Effective Visual Place Recognition Using Multi-Sequence Maps," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 4, no. 2, pp. 1730–1736, Apr. 2019.

[15] Z. Chen *et al.*, "Deep learning features at scale for visual place recognition," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 3223–3230.

[16] P. Huber, J. Wiley, and W. InterScience, *Robust Statistics*. Hoboken, NJ, USA: Wiley New York, 1981.

[17] 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.*, May 2013, pp. 5724–5731.

[18] V. Angladon *et al.*, "An evaluation of real-time RGB-D visual odometry algorithms on mobile devices," *J. Real-Time Image Process.*, vol. 18, pp. 1–18, 2017.

[19] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. 12th Annu. Conf. Comput. Graph. Interactive Techn.*, 1955, pp. 245–254.

[20] M. Björkman, N. Bergström, and D. Kragic, "Detecting, segmenting and tracking unknown objects using multi-label mrf inference," *Comput. Vis. Image Understanding*, vol. 118, pp. 111–127, 2014.

[21] M. J. Atallah and M. Blanton, Eds., *Algorithms and Theory of Computation Handbook: General Concepts and Techniques*, 2nd ed. London, U.K.: Chapman & Hall/CRC, 2010.

[22] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com

[23] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-13, no. 4, pp. 376–380, Apr. 1991.

[24] S. Agarwal, K. Mierle, and Others, "Ceres solver," 2010. [Online]. Available: http://ceres-solver.org

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D slam systems," in *Proc. Int. Conf. Intell. Robot Syst.*, Oct. 2012, pp. 573–580.

[26] D. C. Frank Steinbrucker and Jurgen Sturm, "Real-time visual odometry from dense RGB-D images," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 1678–1685.

[27] J. Park, Q. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 143–152.

[28] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, 2013, pp. 2100–2106, doi: 10.1109/IROS.2013.6696650.

[29] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger,, "ElasticFusion: Real-time dense SLAM and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[30] M. Grupp, "evo: Python package for the evaluation of odometry and slam," 2017. [Online]. Available: https://github.com/MichaelGrupp/evo

[31] 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.

[32] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu, "Fast and accurate image matching with cascade hashing for 3d reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1–8.