Loosely-Coupled Semi-Direct Monocular SLAM

Seong Hun Lee and Javier Civera

***Abstract*— We propose a novel semi-direct approach for monocular simultaneous localization and mapping (SLAM) that combines the complementary strengths of direct and feature-based methods. The proposed pipeline loosely couples direct odometry and feature-based SLAM to perform three levels of parallel optimizations: (1) photometric bundle adjustment (BA) that jointly optimizes the local structure and motion, (2) geometric BA that refines keyframe poses and associated feature map points, and (3) pose graph optimization to achieve global map consistency in the presence of loop closures. This is achieved in real-time by limiting the feature-based operations to marginalized keyframes from the direct odometry module. Exhaustive evaluation on two benchmark datasets demonstrates that our system outperforms the state-of-the-art monocular odometry and SLAM systems in terms of overall accuracy and robustness.**

I. INTRODUCTION

Real-time visual odometry (VO) and simultaneous localization and mapping (SLAM) play an important role in many emerging technologies, such as autonomous ground/air vehicles [1,2] and virtual/augmented reality [3]. In particular, monocular methods have drawn significant attention due to their minimal hardware constraints.

Traditional algorithms relied heavily on feature extraction and matching to estimate structure and motion [3,4]. In recent years, however, direct methods have gained rapidly increasing popularity [5,6]. In contrast to feature-based ones, direct methods are capable of leveraging raw photometric information from a chosen set of pixels in the image. This removes the need for costly per-frame feature extraction and matching. Also, they are shown to be relatively more robust in low-texture scenes [6].

Although direct methods have their own merits in several aspects, they inevitably miss certain benefits of salient features. For example, feature descriptors such as SIFT [7] or ORB [8] have a high degree of invariance to viewpoint and illumination changes, and they can be matched over wide baselines. Such property is favorable for tracking large inter-frame motions and recognizing revisited places. Recent studies indeed confirm that direct and feature-based methods have their own strengths and weaknesses in respective areas [6,9]. Semi-direct methods such as [10] attempt to take advantage of such complementary characteristics by incorporating ideas from both direct and feature-based methods.

In this paper, we propose a novel semi-direct approach for monocular SLAM that inherits both the robustness of direct VO and the map-reusing capability (e.g., loop closure) of feature-based SLAM. The main contribution of this work is a *loose coupling* between direct and feature-based algorithms such that:

1) Locally, a direct method is used to track the camera pose fast and robustly with respect to a locally accurate, short-term, semi-dense map.
2) Globally, a feature-based method is used to refine the keyframe poses, perform loop closures, and build a globally consistent, long-term, sparse feature map that can be reused.

This strategy allows us to complement the weaknesses of each method without compromising their real-time efficiency and performance. We implement our approach on top of DSO [6] and ORB-SLAM [11], respectively the state-of-the-art in direct and feature-based methods, and demonstrate that our system outperforms both of them on two public benchmark datasets. Fig. 1 shows a snapshot of the estimated camera trajectory and the scene reconstruction using our method. The full reconstruction process is demonstrated in the accompanying video:

`https://youtu.be/j7WnU7ZpZ8c`

We make our implementation publicly available at:

`https://github.com/sunghoon031/LCSD_SLAM`

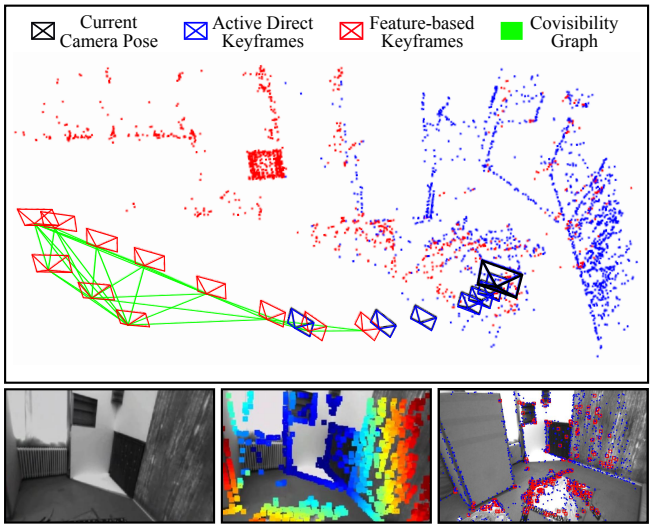

Fig. 1: **Top:** We combine a direct and a feature-based method for monocular SLAM: the former is used for tracking and reconstructing a short-term local map (blue), and the latter for building a reusable global map (red and green). **Bottom:** (from left to right) the current frame, the latest direct keyframe with color-coded depths, and the latest feature-based keyframe with the matched features (red) and the projection of direct map points (blue).

Seong Hun Lee is with I3A, University of Zaragoza, 50018 Zaragoza, Spain (phone: +34 65 463 7956, e-mail: `seonghunlee@unizar.es`)

Javier Civera is with I3A, University of Zaragoza, 50018 Zaragoza, Spain (phone: +34 87 655 5554, e-mail: `jcivera@unizar.es`)

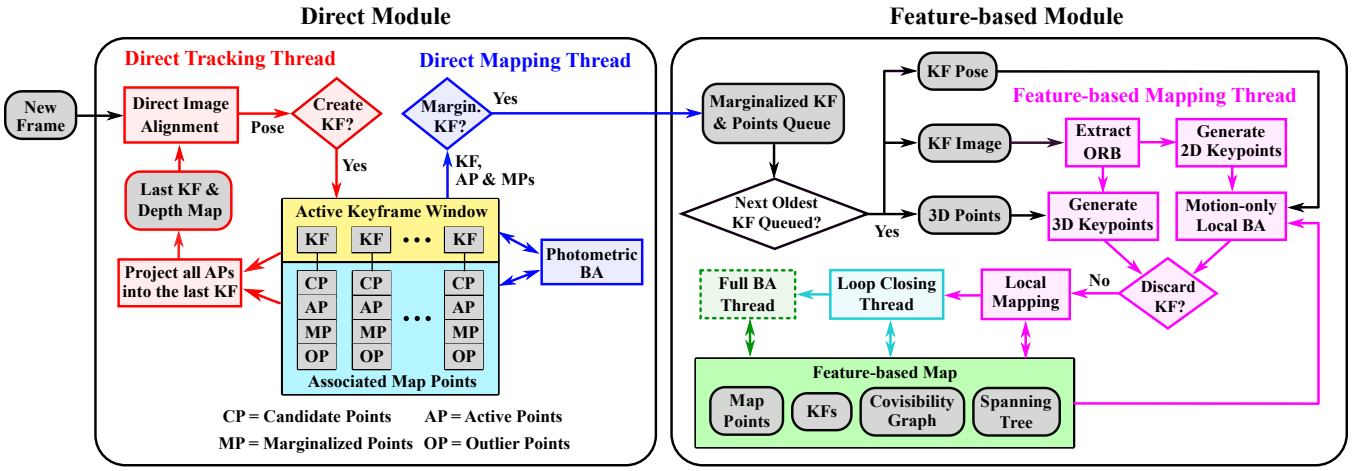

Fig. 2: Our pipeline consists of two modules operating in parallel: One is a direct module module which tracks every new frame with respect to the last keyframe and performs windowed photometric BA. The other is a feature-based module that reconstructs a globally consistent map and keyframe trajectory using the marginalized information from the direct module.

II. RELATED WORK

Modern keyframe-based VO/SLAM systems can be categorized into three classes:

(1) Feature-based: Feature-based (or indirect) methods recover both camera pose and scene structure by matching features and performing *geometric* BA that minimizes the reprojection error. PTAM [3] is the most representative system of this type, where it was first proposed to split tracking and mapping into two parallel threads. At present, ORB-SLAM [11] is arguably the best-performing feature-based system. Based on multiple successful ideas of PTAM and others [12]–[14], ORB-SLAM uses ORB features [8] to perform tracking, mapping, relocalization and loop-closing in a scalable manner. In [15], an extension of ORB-SLAM is proposed to generate a semi-dense reconstruction of the scene. This last method, however, does not use the resulting semi-dense map for tracking.

(2) Direct: Direct methods estimate structure and motion by minimizing the photometric error (i.e., intensity difference) between corresponding pixels in images [16]. Unlike most feature-based methods, direct methods are not limited to a sparse map and can maintain either a sparse [6], semi-dense [5,17] or dense [18,19] map in real-time. Currently, the best-performing odometry system is DSO [6] which performs *photometric* BA to jointly optimize camera intrinsics, extrinsics and inverse depths of sparse (or semi-dense) points in a sliding window fashion. In [20], it was found that, for a small number of map points (e.g., < 1000), such joint optimization tends to be more accurate than alternating tracking and mapping as in, for example, LSD-SLAM [5]. DSO is also the first work to demonstrate the benefits of photometric calibration [21] in direct methods. However, it is subject to drift over time as it is a pure odometry method and does not reuse the map points that leave the field of view (FOV).

(3) Semi-Direct: Semi-direct (or hybrid) methods combine some of the techniques from both feature-based and direct methods. For example, SVO [10] performs direct sparse image alignment to estimate the initial guess of the camera pose and feature correspondences. Afterwards, it performs geometric BA to refine the pose and structure. It was shown in [22] that SVO could also be used for dense mapping. In [23], several improvements to the original SVO are proposed. Although this system is highly efficient, it is shown to be less robust [9,23] than ORB-SLAM [11] and DSO [6]. In [24] and [25], different semi-direct approaches are proposed for a stereo odometry and RGB-D SLAM, respectively.

III. SYSTEM OVERVIEW

Fig. 2 illustrates our proposed semi-direct pipeline. The idea is to combine currently the best-performing feature-based and direct algorithms, namely ORB-SLAM [11] and DSO [6], with some modifications. To achieve real-time performance, we take inspiration from SVO [10] and apply a direct method to quickly track each frame and provide an initial seed for feature-based map optimization. Specifically, we use DSO to achieve real-time tracking and a modified version of ORB-SLAM to build a globally consistent map at a slower rate with marginalized keyframes from DSO. This is shown in Fig. 2 as a direct and a feature-based module, respectively. As they exchange information while running separately and asynchronously, this approach is called *loosely coupled*.

Our system architecture involves three different layers of optimization windows. At the most local level, a sliding window of keyframes and map points are photometrically bundle adjusted to obtain an accurate representation of the surrounding environment. New frames are tracked using direct image alignment [26] with respect to the last keyframe and its corresponding depth map created by projecting active points in the window (see Fig. 1).

When a keyframe is marginalized from the direct module, its image and pose information is sent to the feature-based module, along with the map points within its FOV. The feature-based module extracts ORB descriptors from these

keyframes and refine their poses with respect to the local feature map using motion-only BA. Some of those keyframes and map points are added to the local map and geometrically bundle adjusted for optimal accuracy. This process of feature-based mapping corresponds to the mid-level optimization.

Finally, at the most global level, a pose graph optimization [13] is performed over Sim(3) constraints after each loop closure. Afterwards, a full BA [27] optimizes all keyframes and feature points in the map for global consistency.

The key difference between our approach and SVO [10] is that we maintain two separate maps, each in the direct and the feature-based module. This allows us to utilize a locally accurate semi-dense map for fast and robust tracking, as well as a globally consistent sparse feature map for long-term reuse (e.g., loop detection and closure).

IV. NOTATION

Throughout the paper, we use bold lower- and upper-case letters for vectors ($\mathbf{v}$) and matrices ($\mathbf{M}$), and light lower- and upper-case letters for scalars (s) and scalar functions (F), respectively. The intensity image is denoted by $I : \Omega \mapsto \mathbb{R}$ where $\Omega \subset \mathbb{R}^2$ is the image domain. We denote the camera intrinsic parameters with $\mathbf{c}$, and corresponding camera projection and back-projection functions with $\mathbf{\Pi_c} : \mathbb{R}^3 \mapsto \Omega$ and $\mathbf{\Pi_c}^{-1} : \Omega \times \mathbb{R} \mapsto \mathbb{R}^3$, respectively. Camera poses are represented as either rigid body or similarity transformation matrices $\mathbf{T}_{iw} \in \text{SE(3)}$ or $\mathbf{S}_{iw} \in \text{Sim(3)}$ that transform a point from the world frame to frame i. We use $\mathcal{P}_i$ to denote the set of map points belonging to keyframe i and $\text{obs}(\mathbf{p})$ to denote the set of keyframes in which the point $\mathbf{p}$ is visible. The Euclidean and Huber norms [28] are denoted by $\|\cdot\|_2$ and $\|\cdot\|_\gamma$ respectively. The operator $\boxplus$ is defined as a simple addition for Euclidean parameters and a left multiplication for the pose, i.e., for Lie-algebra $\mathfrak{se}(3)$ elements in twist coordinates $\mathbf{x} \in \mathbb{R}^6$, $\mathbf{x} \boxplus \mathbf{T} := \exp_{\text{SE(3)}}(\mathbf{x})\,\mathbf{T}$. We use the same notation as in [13] for the exponential and logarithmic mapping for SE(3) and Sim(3).

V. DIRECT MODULE

We use the original implementation of DSO [6] as our direct module, which is responsible for initial map bootstrapping, real-time camera tracking and local mapping. In this section, we describe the windowed optimization and marginalization scheme of DSO. The reader is referred to the original work [6] for details on direct tracking and other front-end operations.

A. Windowed Photometric Bundle Adjustment

When a point $\mathbf{p}$ in a reference frame I_i is observed in another frame I_j, the photometric error is defined as the weighted SSD over the 8-point neighborhood pixels $\mathcal{N}_\mathbf{p}$ as proposed in [6]:

$$E_{ij}^{\mathbf{p}} := \sum_{\tilde{\mathbf{p}} \in \mathcal{N}_\mathbf{p}} \omega_{\tilde{\mathbf{p}}} \left\| I_j\left[\tilde{\mathbf{p}}'\right] - b_j - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i\left[\tilde{\mathbf{p}}\right] - b_i\right) \right\|_\gamma \quad (1)$$

$$\text{with} \quad \omega_{\tilde{\mathbf{p}}} := \frac{c^2}{c^2 + \|\nabla I_i(\tilde{\mathbf{p}})\|_2^2}, \quad (2)$$

$$\tilde{\mathbf{p}}' = \mathbf{\Pi_c}\left(\mathbf{T}_{jw}^{-1}\mathbf{T}_{iw}\mathbf{\Pi_c}^{-1}\left(\tilde{\mathbf{p}}, d_\mathbf{p}\right)\right) \quad (3)$$

where t, a and b are exposure time and affine brightness function parameters, $d_\mathbf{p}$ is the inverse depth of $\mathbf{p}$ in the reference frame I_i. The weight $w_{\tilde{\mathbf{p}}}$ down-weights high-gradient pixels with some constant c. The total energy function to be minimized is given by the full photometric error plus a prior pulling the affine brightness parameters to zero:

$$E_{\text{photo}} := \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{ij}^{\mathbf{p}} + \sum_{i \in \mathcal{F}} \left(\lambda_a a_i^2 + \lambda_b b_i^2\right), \quad (4)$$

where $\mathcal{F}$ denotes the set of all frames in the window. When exposure times are known, we set λ_a and λ_b to some constant values. Otherwise, we set $\lambda_a = \lambda_b = 0$ and $t_i = t_j = 1$ in (1). The optimization is performed using an iteratively reweighted Gauss-Newton or Levenberg-Marquardt algorithm in a coarse-to-fine scheme. The update equation is given by

$$\delta\boldsymbol{\xi} = -\left(\mathbf{J}^T\mathbf{W}\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r} \quad \text{and} \quad \boldsymbol{\xi}^{\text{new}} \leftarrow \delta\boldsymbol{\xi} \boxplus \boldsymbol{\xi}, \quad (5)$$

where $\mathbf{r}$ is the stacked vector of residuals, $\mathbf{J}$ is its Jacobian and $\mathbf{W}$ is the diagonal weight matrix. The state variable $\boldsymbol{\xi}$ contains all the active variables in the window, i.e., camera poses, affine brightness parameters, inverse depths and camera intrinsics.

B. Marginalization

The size of the optimization window is kept bounded by marginalizing the least useful keyframes and points using the Schur complement [6,29]. Points are marginalized if they are not observed in the latest two keyframes or their host keyframe is marginalized. Keyframes (other than the latest two) are marginalized if either less than 5% of its points are visible in the latest keyframe, or if it has the highest "distance score" when the window contains more than a certain number of keyframes. We refer to the original work [6] for the computation of this heuristic score.

VI. FEATURE-BASED MODULE

When a keyframe is marginalized from the direct module, the feature-based module receives its image and pose information, as well as the 3D locations of both active and marginalized map points within its FOV. This information is then used for feature-based pose refinement, mapping and loop closing. Note that the marginalization strategy in Section V-B does not necessarily marginalize the oldest keyframe in the window. To avoid temporal inconsistency in such cases, we store the marginalized keyframes and points in a queue and wait until the next oldest keyframe is marginalized. If more than five keyframes are queued and the next oldest keyframe is still active, we take its latest pose and points to proceed further instead of waiting. All optimizations are performed using the original implementation in ORB-SLAM [11], which is based on Levenberg-Marquardt algorithm in g2o [30].

A. Relative Scale and Initial Pose Estimation

In our loosely-coupled approach, the direct and the feature-based modules maintain two separate maps. Due to the scale ambiguity of a monocular system, the scales of these two maps drift over time and do not converge to the same value. Therefore, we continuously compute the relative scale between them using Horn's method [31] by comparing the 30 most recent keyframes in the feature-based module against their counterparts in the direct module.

Once the relative scale s is known, the incremental transformation in the direct module can be scaled appropriately and used as an initial pose guess in the feature-based module: Let i and j denote the previous and current keyframe. Then,

$$T_{jw|\mathrm{F}} = \begin{bmatrix} R_{ji|\mathrm{F}} & t_{ji|\mathrm{F}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} T_{iw|\mathrm{F}} \tag{6}$$

$$\text{with} \quad R_{ji|\mathrm{F}} = R_{ji|\mathrm{D}} = R_{jw|\mathrm{D}}(R_{iw|\mathrm{D}})^{\mathrm{T}}, \tag{7}$$

$$t_{ji|\mathrm{F}} = s\,t_{ji|\mathrm{D}} = s\left(-R_{ji|\mathrm{D}}t_{iw|\mathrm{D}} + t_{jw|\mathrm{D}}\right), \tag{8}$$

where the subscripts D and F indicate the direct and the feature-based module, respectively.

B. 3D Keypoints Generation

The map points from the direct module are used in two ways: (1) for creating an initial set of 3D keypoints to bootstrap the feature-based module, or (2) for adding more local map points to improve the tracking robustness. Given the 2D position $\mathbf{p}$ of an ORB feature in frame i, we generate a 3D keypoint $\mathbf{x}_w$ in the world frame as

$$\mathbf{x}_w = \mathbf{T}_{iw}^{-1}\mathbf{\Pi}_{\mathbf{c}}^{-1}\left(\mathbf{p}, \frac{d_{\mathbf{p}}}{s}\right) \quad \text{with} \quad d_{\mathbf{p}} = \frac{\sum_{k\in\mathcal{P}_{\mathbf{p}}} d_k/\sigma_k^2}{\sum_k 1/\sigma_k^2}, \tag{9}$$

where s is the relative scale between the direct and the feature-based module (Section VI-A), $\mathcal{P}_{\mathbf{p}}$ and $d_{k\in\mathcal{P}_{\mathbf{p}}}$ respectively denote the set of all map points whose projection in frame i is equal to $\mathbf{p}$ and their inverse depths in frame i. We dilate the projection by two pixels to ensure a sufficient number of valid depths for the keypoints. Note that the inverse depth $d_{\mathbf{p}}$ is computed as the inverse-variance weighted average, which is equivalent to the Kalman filter update with multiple measurements [17].

We found that extracting more features during slower camera motions often increases the number of covisibility links [11] between keyframes, thereby improving the mapping accuracy. Therefore, we vary the number of features to extract per image as follows: Let f_{kf} be the keyframe addition frequency in the direct module. Using this as the indicator of the relative camera speed, we set $N_{\text{features}} = 2500$ if $f_{\text{kf}} < 4$Hz and $N_{\text{features}} = 1500$ if $f_{\text{kf}} > 7$Hz. Otherwise, we interpolate between the two bounds based on f_{kf}.

C. Keyframe Pose Refinement and Failure Recovery

Once the direct module provides an initial pose estimate of a new keyframe (Section VI-A), we refine it using motion-only geometric BA with respect to the local feature-based map. The total energy function is composed of the variance-normalized reprojection errors of the local map points:

$$E_{\text{reproj}} = \sum_{i\in\mathcal{F}_{\text{local}}}\sum_{\mathbf{x}\in\mathcal{P}_i}\sum_{j\in\text{obs}(\mathbf{x})}\left\|\frac{\mathbf{p}_{j,\mathbf{x}} - \mathbf{\Pi}_{\mathbf{c}}\left(\mathbf{T}_{iw}\mathbf{x}_w\right)}{\sigma_{\mathbf{x}}^2}\right\|_{\gamma} \tag{10}$$

$$\text{with} \quad \sigma_{\mathbf{x}}^2 := \left(\lambda_{\text{pyr}}\right)^{2L_{\text{pyr},\mathbf{x}}}, \tag{11}$$

where $\mathcal{F}_{\text{local}}$ denotes the set of all local keyframes, i.e., all keyframes sharing map points with the new keyframe and their neighbors in the covisibility graph [11], $\mathbf{p}_{j,\mathbf{x}} \in \mathbb{R}^2$ the match to the keypoint $\mathbf{x}$ in frame j, and $\sigma_{\mathbf{x}}^2$ the variance of the feature location in frame i. This variance depends on the constant scale factor of the image pyramid λ_{pyr} (> 1) and the pyramid level $L_{\text{pyr},\mathbf{x}}$ at which the keypoint was detected.

When the feature-based module fails due to insufficient matches, we reinitialize the module, following the procedure explained in Section VI-B. This is illustrated in Fig. 3. Since our direct module is robust to low-texture scenes, we can rely on its tracking while the feature-based module is lost. Then, as soon as we detect more features, we reinitialize the local map points using the depths from the direct module.

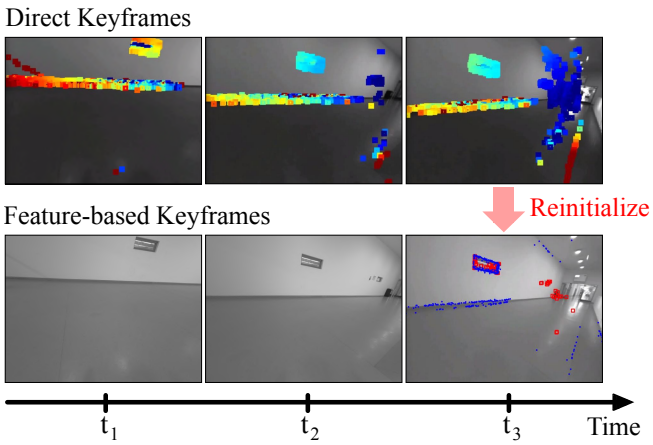

Fig. 3: **[TUM monoVO]** Failure recovery in *sequence 40*: At t_1 and t_2, the feature-based module fails due to the lack of features in the scene, whereas the direct module is able to track the high-gradient pixels. At t_3, the scene now contains a sufficient number of features, and their depths can be initialized with the help of the direct module.

D. Feature-based Local Mapping and Loop Closing

After generating 3D keypoints and refining the keyframe pose, we insert them in the feature-based map if the number of matches falls below 150 or more than three keyframes passed from the last insertion. Once the keyframe and the points are added to the map, they are processed by the local mapping, as outlined in [11]. This includes the local geometric BA that minimizes (10) to jointly optimize the current keyframe, its neighbors in the covisibility graph, and all the map points belonging to those keyframes.

In the loop closing thread, the place recognition module [32] based on DBoW2 [12] detects large loops by querying the keyframe database. Once a loop is detected, the keyframes and map points on each side of the loop are aligned and fused. To correct the scale drift, pose graph

optimization [13] is performed over the essential graph [11], minimizing

$$E_{\text{graph}} = \sum_{(i,j)\in\mathcal{E}_{\text{edge}}} \left\| \log_{\text{Sim(3)}} \left(\mathbf{S}_{ij,0}\ \mathbf{S}_{jw}\ \mathbf{S}_{iw}^{-1} \right) \right\|_2^2, \quad (12)$$

where $\mathcal{E}_{\text{edge}}$ denotes the set of edges in the essential graph, and $\mathbf{S}_{ij,0} = \mathbf{S}_{iw,0}\mathbf{S}_{jw,0}^{-1}$ is the fixed similarity transformation (with the scale 1) between the frame i and j just prior to the pose graph optimization. If the edge is created from a loop closure, this transformation is instead computed using the method of Horn [31]. Finally, a full BA [27] is performed afterwards.

E. Does Feature-based Mapping Always Improve Accuracy?

The answer is no. In general, the feature-based mapping described in the previous sections improves the accuracy if there is a loop closure or when the camera motion is mostly loopy, which enhances the covisibility of features in multiple keyframes and the reuse of map points. However, we found that it actually causes more drift when the camera motion is mostly exploratory without loop closures (a similar finding was also reported in [6]).

We solve this by keeping two versions of the keyframe trajectory: one that is initially given by the direct module and the other modified by the feature-based module. We assume that the latter is more accurate if there is a loop closure or less than a quarter of all past keyframes have *collinear covisibility links* at a given point in time. The covisibility links (i.e., 3D lines connecting the keyframe to its neighbors in the covisibility graph) are considered collinear if none of them form an angle between $30°$ and $150°$. This is illustrated in Fig. 4. We found that this method is especially effective for detecting dominant translational motions towards unexplored areas.

VII. EVALUATION

A. Evaluated Settings, Datasets and Methodology

We implement our method using ROS[1] and compare it against ORB-SLAM [11] and DSO [6]. We evaluate each algorithm in two different settings:

- **ORB-VO** and **ORB-SLAM**: For the VO setting, we disable the loop closing thread. Relocalization is disabled for both settings to evaluate the tracking robustness. We do not apply photometric calibration [21], as we found that it worsens feature extraction and matching (similar findings were also reported in [6,9]).
- **DSO-default** and **DSO-reduced**: We use the default and reduced settings provided in the original DSO implementation. The only difference is that, for the reduced setting, we always resize the input images by half. Photometric calibration [21] is applied when available.
- **Ours-VO** and **Ours-SLAM**: The VO setting uses the combination of DSO-reduced and ORB-VO, whereas the SLAM setting uses DSO-reduced and ORB-SLAM settings. This means that the direct module processes photometrically calibrated images (if available) at half resolution, while the feature-based module processes photometrically non-calibrated images at full resolution. We found that using half resolution in the feature-based module significantly worsens the performance, which is in line with the observation in [21]. For efficiency, we reduce the number of iterations in the local geometric BA by half.

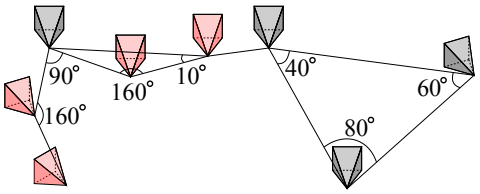

Fig. 4: Illustration of the keyframes with collinear covisibility links (red). None of their covisibility links form an angle between $30°$ and $150°$.

We use two public benchmark datasets for evaluation:

1) **EuRoC MAV dataset** [33], which contains 11 indoor stereo sequences with 752×480 pixel resolution at 20 fps. As in [6], we crop the beginning and end of each sequence to disregard large occlusions due to the ground and aggressive motions meant for IMU initialization. We evaluate the tracking accuracy using the absolute trajectory RMSE (e_{ate}) of keyframe poses after Sim(3) alignment with the ground truth. Photometric calibration and exposure times are not available for this dataset.
2) **TUM monoVO dataset** [21], which contains 50 in- and outdoor monocular sequences recorded at 20–50 fps. We use 640×480 pixel resolution for undistorted images. Since the full ground-truth data is not available for this dataset, the tracking accuracy is evaluated in terms of the alignment error (e_{align}) proposed in [21]. The dataset also provides photometric calibration and exposure times.

To account for non-deterministic behaviour, we run each method 10 times. This amounts to 220 runs for the EuRoC MAV dataset (i.e., 110 runs for each left and right camera) and 500 runs for the TUM monoVO dataset. On the EuRoC MAV dataset, we consider that runs were unsuccessful if more than 20% of the total frames could not be tracked either due to the delayed map initialization or complete tracking failures. On the TUM monoVO dataset, we disable the keyframe culling of ORB-VO/SLAM and our systems within the start- and end-segment of each trajectory where the ground-truth data is available. This prevents the lack of keyframes when computing the alignment error on these segments. All images were preloaded into memory, but not rectified or photometrically calibrated beforehand. All results were obtained by running each system in real-time on a laptop[2].

B. Results

Fig. 9 shows the error values for each sequence of both datasets, and Tab. II and III report the median values.

[1]Robot Operating System, `http://www.ros.org/`.

[2]Intel Core i7-4810MQ, quad-core at 2.8 GHz with 15GB RAM

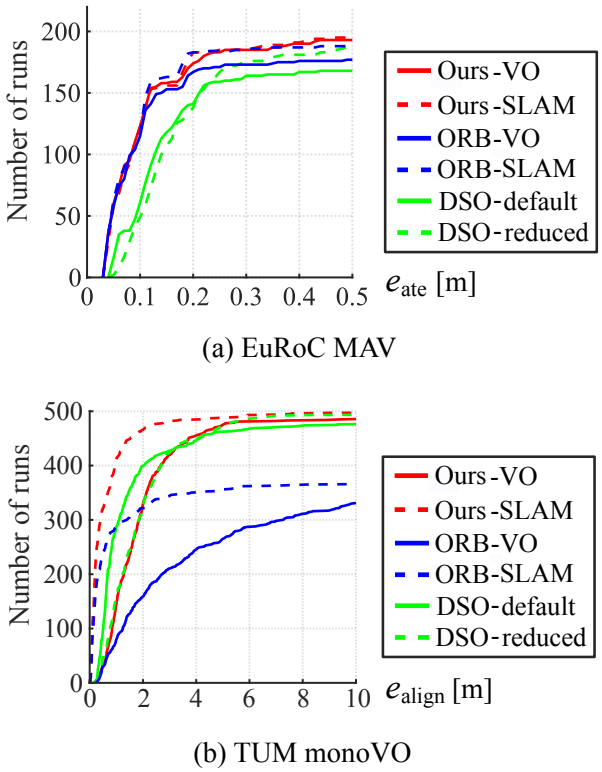

Fig. 5: Cumulative error plot aggregating **(a)** the absolute trajectory errors e_{ate} [m] for the EuRoc MAV dataset and **(b)** alignment errors e_{align} [m] for the TUM monoVO dataset.

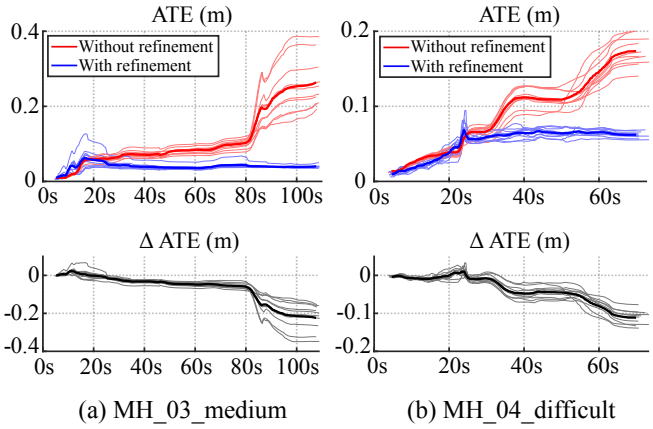

Fig. 6: **[EuRoC MAV]** Time evolution of the ATEs with and without the feature-based refinement (Section VI-C and VI-D). The loop-closure detection is disabled. The average of 10 independent runs is shown in bold.

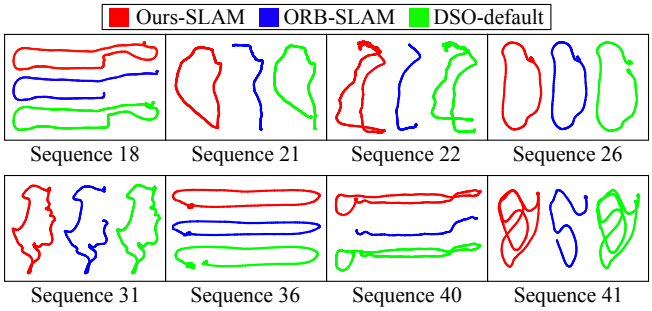

Fig. 7: **[TUM monoVO]** Sample trajectories estimated with the median accuracy. All sequences start and end at the same position, and only our system can track the entire trajectories without consistent drifts or failures.

The aggregated results are given in Fig. 5 in the form of cumulative error plots indicating how many runs have yielded error values below a certain level. On both datasets, we make two common observations: First, DSO-reduced is more robust than DSO-default, which is most likely due to the faster tracking speed, as shown in Tab. I. We observe similar accuracy between the two settings on the EuRoC MAV dataset, but higher accuracy with DSO-default on the TUM monoVO dataset. Second, loop closing in ORB-SLAM improves the performance. It increases accuracy on the TUM monoVO dataset and robustness on both datasets.

On the EuRoC MAV dataset, DSO (both default and reduced) yields the lowest accuracy. It was also reported in [6] that DSO was less accurate than ORB-VO on the same dataset. However, we were unable to reproduce their exact results, in particular those showing that DSO was more robust in real-time. Our systems (both VO and SLAM) and ORB-SLAM show very similar performance on this dataset. The comparison between DSO-reduced and our VO system indicates that the feature-based pose refinement (Section VI-C) and local mapping (Section VI-D) can improve the accuracy even without loop closure. Fig. 6 shows this effect over time on two of the sequences. Note how quickly the drift accumulates if we do not reuse the map points that once leave the FOV.

On the TUM monoVO dataset, DSO (both default and reduced) is significantly more robust than ORB-SLAM/VO, which is similar to the results reported in [6,9]. Our VO system achieves very similar performance to DSO-reduced, as none of the final trajectories are affected by the feature-based module. This is because more than 75% of the total keyframes have collinear covisibility links in all sequences (see Section. VI-E), which is in contrast to the EuRoC MAV dataset where the opposite is true. Fig. 7 compares the estimated trajectories on some of the TUM monoVO sequences. Note how ORB-SLAM fails in the middle of some sequences and DSO accumulates drift, while our SLAM system completes the whole sequences and closes loops. In Fig. 5 and 9, we can clearly see that the loop closure significantly reduced the alignment errors for our SLAM system in the majority of the sequences. Across both datasets, our SLAM system achieves the best overall performance.

Tab. I summarizes the statistics of dropped frames and tracking times on the EuRoC MAV dataset. It can be seen that DSO-reduced and both our systems have lower frame drops and faster tracking speed than the rest. This shows the advantage of direct tracking, which eliminates the need to perform feature extraction and matching for every frame. In Fig. 8, we further show how much percentage of keyframes and map points are reduced in our systems compared to ORB-VO/SLAM. We observe average 27% (up to 46%) keyframes reduction and average 6% (up to 27%) map points reduction for the SLAM system. This suggests that our system is relatively more scalable than ORB-SLAM.

VIII. CONCLUSIONS

In this paper, we proposed a loosely-coupled semi-direct method for real-time monocular SLAM. Our system consists of two modules running in parallel. One module uses a direct method to track new frames fast and robustly with respect to a local semi-dense map. The other module uses the resulting points and pose estimates as prior to build a globally consistent sparse feature-based map. We have shown on two public datasets that our method outperforms the state-of-the-art in terms of tracking accuracy and robustness.

REFERENCES

[1] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[2] S. Lee and G. C. H. E. de Croon, "Stability-based scale estimation for monocular SLAM," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 780–787, 2018.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Eur. Conf. on Computer Vision (ECCV)*, 2014, pp. 834–849.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, no. 99, pp. 1–1, 2017.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[9] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[10] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA)*, 2014, pp. 15–22.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] D. Galvez-Lpez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, 2012.

[13] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," in *Proc. Robotics: Science and Systems*, 2010.

[14] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2352–2359.

[15] R. Mur-Artal and J. D. Tardos, "Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM," in *Proc. Robotics: Science and Systems*, 2015.

[16] M. Irani and P. Anandan, "About direct methods," in *Proc. Workshop Vision Algorithms*, 1999, pp. 267–277.

[17] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2013, pp. 1449–1456.

[18] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[19] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular slam," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 833–840.

[20] L. Platinsky, A. J. Davison, and S. Leutenegger, "Monocular visual odometry: Sparse joint optimisation or dense alternation?" in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5126–5133.

TABLE I: **[EuRoC MAV]** Dropped frames percentage and tracking times. The two smallest values are highlighted in bold. *This is the median of the median results in each sequence.

	Dropped frames (%)			Tracking Times (ms)		
	Med*	Mean	Std	Med*	Mean	Std
ORB-VO	0.95	1.56	1.61	22.09	25.46	8.62
ORB-SLAM	1.54	2.14	1.57	22.74	26.47	9.48
DSO-default	0.81	1.16	**1.07**	7.13	9.66	17.33
DSO-reduced	**0.28**	**0.74**	**1.00**	**2.60**	**4.07**	**7.40**
Ours-VO	**0.31**	1.02	1.48	**4.62**	**6.19**	**8.62**
Ours-SLAM	0.32	**0.97**	1.41	4.69	6.23	9.48

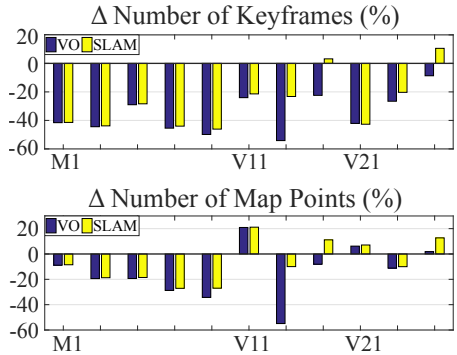

Fig. 8: **[EuRoC MAV]** Difference in the number of total keyframes and map points compared to ORB-VO/SLAM.

[21] J. Engel, V. Koltun, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *CoRR*, vol. abs/1607.02555, 2016.

[22] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 2609–2616.

[23] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017.

[24] N. Krombach, D. Droeschel, and S. Behnke, "Combining feature-based and direct methods for semi-dense real-time stereo visual odometry," in *Int. Conf. on Intelligent Autonomous Systems*, 2016, pp. 855–868.

[25] S. Bu, Y. Zhao, G. Wan, K. Li, G. Cheng, and Z. Liu, "Semi-direct tracking and mapping with RGB-D camera for MAV," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4445–4469, 2017.

[26] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[27] 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. 33, no. 5, pp. 1255–1262, 2017.

[28] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

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

[30] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[31] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[32] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based SLAM," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 846–853.

[33] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The Euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

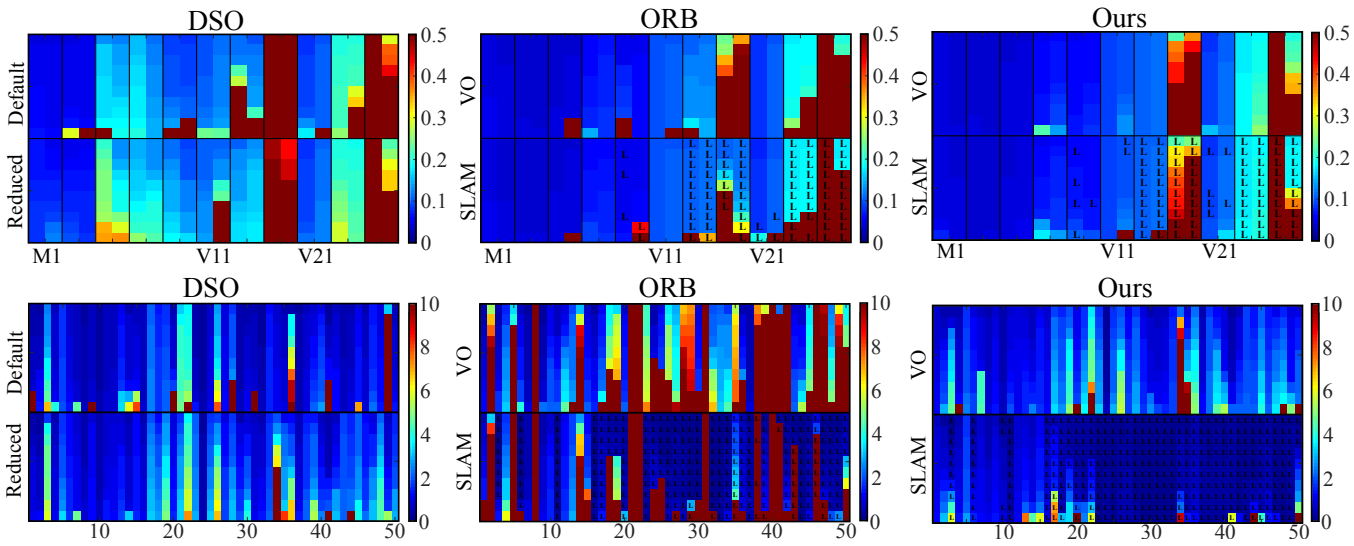

Fig. 9: **Top row:** Absolute trajectory errors e_{ate} [m] on the EuRoC MAV dataset. **Bottom row:** Alignment errors e_{align} [m] on the TUM monoVO dataset. The letter 'L' indicates the presence of one or more loop closure links.

TABLE II: **[EuRoC MAV]** Median absolute trajectory errors e_{ate} [m] over 10 runs on each sequence. The best and the worst results are shown in blue and red, respectively.

	ORB VO	ORB SLAM	DSO default	DSO reduced	Ours VO	Ours SLAM
M1L	0.046	0.046	0.053	**0.076**	**0.044**	0.046
M2L	0.038	**0.037**	0.050	**0.071**	0.038	**0.037**
M3L	**0.039**	**0.039**	0.166	**0.249**	0.040	0.040
M4L	0.064	0.065	**0.172**	0.163	**0.060**	0.064
M5L	0.078	**0.051**	0.101	**0.144**	0.061	0.065
V11L	0.096	0.097	0.104	**0.115**	**0.089**	0.090
V12L	0.127	**0.103**	**0.516**	0.136	0.104	0.105
V13L	0.938	0.423	×	0.757	0.608	**0.412**
V21L	**0.080**	**0.080**	0.089	**0.090**	0.089	0.088
V22L	0.185	**0.178**	0.210	**0.233**	0.181	**0.178**
V23L	×	×	1.429	**1.055**	1.174	1.062

	ORB VO	ORB SLAM	DSO default	DSO reduced	Ours VO	Ours SLAM
M1R	0.040	**0.038**	0.052	**0.076**	**0.038**	0.039
M2R	0.038	0.038	0.052	**0.071**	**0.036**	**0.036**
M3R	0.045	**0.043**	0.149	**0.185**	**0.043**	0.045
M4R	0.074	**0.069**	0.129	**0.174**	0.070	0.074
M5R	0.056	**0.055**	0.095	**0.122**	0.065	0.060
V11R	0.103	0.104	0.119	**0.215**	0.100	**0.099**
V12R	0.125	0.112	0.133	**0.161**	**0.111**	**0.111**
V13R	×	0.130	×	**0.614**	1.277	0.825
V21R	**0.103**	0.104	0.104	**0.116**	0.113	0.114
V22R	0.198	**0.183**	**0.255**	0.209	0.201	0.191
V23R	×	×	0.707	0.432	0.345	**0.238**

TABLE III: **[TUM monoVO]** Median alignment errors e_{align} [m] over 10 runs on each sequence. The best and the worst results are shown in blue and red, respectively.

	ORB VO	ORB SLAM	DSO default	DSO reduced	Ours VO	Ours SLAM
1	0.744	**1.063**	**0.526**	0.799	0.885	0.860
2	**19.08**	**56.03**	0.570	**0.487**	1.412	0.908
3	0.731	**0.510**	3.393	**4.747**	3.283	0.529
4	**2.458**	2.234	0.689	0.770	**0.677**	0.764
5	×	×	**1.718**	2.004	1.921	1.880
6	1.328	0.037	0.808	1.328	**1.358**	**0.522**
7	**0.406**	0.454	0.604	**0.969**	0.926	0.931
8	**596.9**	**425.0**	**0.353**	0.483	0.600	0.625
9	0.534	**0.393**	0.617	**1.069**	1.057	0.994
10	**1.632**	1.312	**0.289**	0.427	0.400	0.564
11	**1.086**	**0.152**	0.592	0.847	0.961	0.172
12	**2.533**	1.646	**0.597**	0.937	1.000	0.812
13	2.232	**2.366**	1.388	1.312	**1.130**	1.216
14	**10.05**	**20.63**	0.718	**0.666**	0.863	0.847
15	1.241	**1.498**	**0.832**	1.310	1.340	1.323
16	0.832	**0.086**	0.514	0.739	**0.931**	0.603
17	1.777	**0.417**	2.300	**2.684**	2.681	1.300
18	5.381	×	1.579	2.135	1.986	**0.411**
19	**6.493**	5.621	1.898	3.534	3.673	**1.232**
20	1.266	**0.273**	0.754	1.095	**1.331**	0.295
21	×	×	4.219	2.707	2.504	**0.230**
22	×	×	3.954	4.785	4.732	**1.131**
23	**5.140**	0.161	0.478	2.344	2.538	**0.061**
24	**5.731**	0.263	0.297	0.361	0.275	**0.156**
25	1.657	0.099	0.821	**2.275**	2.070	**0.081**

	ORB VO	ORB SLAM	DSO default	DSO reduced	Ours VO	Ours SLAM
26	**5.386**	0.323	3.336	4.756	3.705	**0.241**
27	**3.810**	**0.198**	0.965	1.496	1.530	0.321
28	**9.517**	0.433	2.185	2.309	2.186	**0.121**
29	**8.255**	0.325	0.430	1.232	1.311	**0.039**
30	1.278	0.046	0.663	**1.919**	0.821	**0.037**
31	×	×	0.593	0.770	0.738	**0.078**
32	**3.094**	0.077	0.320	0.816	0.876	**0.070**
33	**2.726**	**0.074**	1.449	1.730	1.632	0.078
34	2.687	**0.105**	0.884	**9.434**	**22.54**	0.355
35	**7.046**	0.705	0.578	2.620	2.758	**0.290**
36	1.465	0.590	**5.851**	2.898	3.661	**0.528**
37	0.442	**0.103**	0.379	0.685	**0.709**	0.180
38	×	×	0.768	1.662	1.966	**0.171**
39	**32.58**	0.314	1.293	2.221	2.896	**0.210**
40	×	×	1.805	1.185	1.007	**0.164**
41	×	×	0.897	0.542	0.449	**0.195**
42	×	0.547	0.889	1.293	1.158	**0.084**
43	1.388	0.181	0.458	**1.963**	1.902	**0.102**
44	1.388	**0.065**	0.552	**1.780**	1.713	0.120
45	**4.085**	**0.204**	1.258	2.442	2.693	0.227
46	**12.09**	1.345	0.608	1.415	1.605	**0.403**
47	**12.68**	0.252	1.519	2.019	2.234	**0.122**
48	**3.753**	0.208	1.089	3.314	2.621	**0.060**
49	**11.40**	**0.164**	×	1.023	1.172	0.195
50	**137.2**	4.557	**0.771**	1.830	1.999	1.586