Accurate and Robust Scale Recovery for Monocular Visual Odometry Based on Plane Geometry

Rui Tian[1], Yunzhou Zhang[1*], Delong Zhu[2*], Shiwen Liang[1], Sonya Coleman[3], Dermot Kerr[3]

Abstract— Scale ambiguity is a fundamental problem in monocular visual odometry. Typical solutions include loop closure detection and environment information mining. For applications like self-driving cars, loop closure is not always available, hence mining prior knowledge from the environment becomes a more promising approach. In this paper, with the assumption of a constant height of the camera above the ground, we develop a light-weight scale recovery framework leveraging an accurate and robust estimation of the ground plane. The framework includes a ground point extraction algorithm for selecting high-quality points on the ground plane, and a ground point aggregation algorithm for joining the extracted ground points in a local sliding window. Based on the aggregated data, the scale is finally recovered by solving a least-squares problem using a RANSAC-based optimizer. Sufficient data and robust optimizer enable a highly accurate scale recovery. Experiments on the KITTI dataset show that the proposed framework can achieve state-of-the-art accuracy in terms of translation errors, while maintaining competitive performance on the rotation error. Due to the light-weight design, our framework also demonstrates a high frequency of 20 Hz on the dataset.

I. INTRODUCTION

Monocular Visual Odometry (MVO) is a popular method for camera pose estimation, but due to the scale ambiguity [1]–[5], the MVO system cannot provide real odometry data. Therefore, an accurate and robust scale recovery algorithm is of great significance in the application of MVO [6]. The key of scale recovery is to integrate absolute reference information, such as the gravity orientation from IMU [7] or the depth measured by Lidar [8], [9]. The baseline of stereo cameras can also serve as such a reference [10]. However, these sensors are not always available in real-world applications. Moreover, a complicated sensor calibration and fusion process is needed to align the MVO system with other sensors.

Another frequently used method for removing scale ambiguity is to take as reference the height of a mounted

camera above the ground plane, which remains a stable signal during the navigation of vehicles. The idea is to estimate a ratio between the real camera height and the relative one calculated from image features. The ratio then can be used to recover the real scale. The advantages of this method are significant since it does not depend on other sensors and is with high feasibility. The method is also regarded as one of the most promising solutions in this research area.

Prior work like [1], [2], [11], [12] typically leverages the results of feature matching to calculate the homography matrix and then decompose the matrix to estimate the parameters of the ground plane, based on which the relative camera height can be obtained. The major deficiency of this method is that the decomposition is very sensitive to noises and multiple solutions exist, which requires additional operations to eliminate the ambiguity. Some other work like [12], [13] chooses to directly fit the ground plane using feature points that lie on the ground, e.g., the center regions of a road. By removing low-quality image features outside the target region, the robustness is improved. However, the target region may be occluded sometimes, which interferes with the detection of image features, thus degrading the accuracy of scale recovery.

In recent work [14]–[18], deep learning based MVO algorithms are proposed, in which the camera pose with a real scale is directly predicted by the neural network in an end-to-end manner. Such methods have received much attention in recent years, but their generalization ability across different scenarios is very limited [18]. Some other deep learning based methods take scale recovery as an independent problem. For instance, DNet [16] is proposed to perform ground segmentation and depth regression simultaneously. Based on the predicted depth points within the ground region, a dense geometrical constraint is then formulated to help recover the scale. In [17], a scale-recovery loss is developed based on the idea of enforcing the consistency between the known camera height and the predicted one. Constrained by this loss, the neural network can predict more accurate ego poses. Nonetheless, these methods usually require a large-scale training process, and the computational cost is prohibitively expensive.

In this paper, we propose a light-weight method for accurate and robust scale recovery based on plane geometry. The method includes an efficient Ground Point Extraction (GPE) algorithm based on Delaunay triangulation [19] and a Ground Points Aggregation (GPA) algorithm for aggregating ground points from consecutive frames. Based on these two algorithms, a large number of high-quality ground points are

*The corresponding author of this paper.

[1]Rui Tian, Yunzhou Zhang and Shiwen Liang are with College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (Email: `zhangyunzhou@mail.neu.edu.cn`).

[2]Delong Zhu is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China (Email: `zhudelong@link.cuhk.edu.hk`).

[3]Sonya Coleman and Dermot Kerr are with School of Computing and Intelligent Systems,Ulster University, N. Ireland, UK.
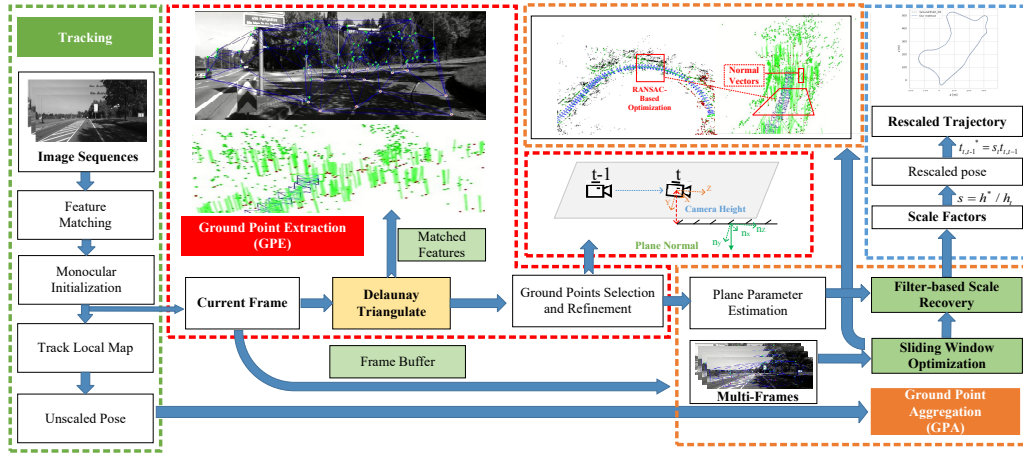
Fig. 1. Overview of the proposed system. There are two parallel threads in the system: 1) The MVO thread takes image frames as input and estimates the current camera pose, 2) The GPE-GPA thread fetches image features from the MVO thread and selects high-quality points for ground plane estimation.

selected. For scale recovery, we first formulate a least-square problem to fit the ground plane and then estimate the relative camera height to calculate the real scale. By leveraging the high-quality points and a RANSAC-based optimizer, the scale can be estimated accurately and robustly. Benefiting from the light-weight design of the algorithms, our method can achieve a 20Hz running frequency on the benchmark dataset.

The main contributions of this work are as follows:

- We propose a GPE algorithm based on Delaunay triangulation, which can accurately extract ground points.
- We propose a GPA algorithm that can effectively aggregate local ground points and perform robust optimization of the ground plane parameters.
- Based on the proposed algorithms, we implement a real-time MVO system with accurate and robust scale recovery, aiming to reduce scale drift and provide accurate odometry in long-distance navigations without loop closure.

II. System Overview

The notations used in this paper are as follows:

- $T_t \in R^{4 \times 4}$ - The camera pose of image frame I_t in the global frame, which is composed of a camera orientation $R_t \in R^{3 \times 3}$ and a translation $\boldsymbol{t}_t \in R^{3 \times 1}$.
- $T_{t,t-1}$ - The relative pose of frame I_{t-1} w.r.t. frame I_t.
- K - The intrinsic matrix of a pinhole camera model.
- $\mathbf{x}_i, \mathbf{u}_i$ - The 3D map point in camera frame and its corresponded 2D point on image plane after projection.
- $\mathbf{n}_t, h_t$ - The plane parameters, i.e., $\mathbf{n}_t \cdot \mathbf{x}_i - h_t = \mathbf{0}$, where $\mathbf{n}_t$ is the normal vector and h_t is the distance to the plane.
- $h, h^\dagger, h^*$ – The calculated camera height form image features, the estimated camera height after scale recovery, and the real camera height.

A. Problem Definition

Given consecutive image frames from a calibrated monocular camera, our goal is to estimate the absolute scale of

camera poses and then recover the real camera trajectory by making use of the prior known camera height h^*. Under scale ambiguity, the camera height h calculated from image features maintains a ratio with the real one, i.e., $s = h^*/h$. Therefore, scale recovery is essentially to compute s, and the key lies in the accurate estimation of the ground plane.

B. System Architecture

The proposed system in this work is shown in Fig. 1. There are two parallel threads in the system: The first one is the MVO thread, which takes consecutive images as input and estimates the current camera pose, e.g., the ORB-SLAM2 framework. The second thread is used to run the GPE and GPA algorithms for scale recovery. The proposed system is based on such an assumption that the ground is locally flat and can be approximated by a plane with a surface normal. The workflow of the second thread is as follows.

As shown in the red block in Fig.1, for each image frame from the MVO thread, the Delaunay triangulation is first applied to segment the matched feature points into a set of triangles. Each triangle is then back-projected into the camera frame, and the associated plane parameters are also estimated. After that, several geometrical constraints are leveraged to select and then refine ground points.

Note that selected ground points are not enough for an accurate estimation of the plane parameters. We thus propose the GPA algorithm to aggregate ground points from multiple frames using a sliding windows method, as shown in the orange block of Fig.1. Based on the aggregated local points, a robust parameter estimation procedure is then performed to fit the ground plane. Accordingly, the relative camera height of each frame can be estimated, and the absolute camera trajectory is recovered, shown in the blue block of Fig.1.

III. Ground Plane Estimation

A. Ground Point Extraction

For a given set of matched feature points $\mathbf{u}_i^t$, $i \in \{1, 2, \ldots, N\}$, in the current image frame I_t, the Delaunay

triangulation uses each of the feature points as a triangle vertex. We back-project the triangles from the image plane into the current camera frame and denote them by $\Delta_i^t, i \in \{1, 2, \ldots M\}$ associated with a set of vertices $\mathbf{x}_{ij}^t, j \in \{1, 2, 3\}$. The normal vector $\mathbf{n}_i^t = (n_{i,x}^t, n_{i,y}^t, n_{i,z}^t)$ of each triangle can be obtained by the cross product,

$$\mathbf{n}_i^t = \frac{(\mathbf{x}_{i1}^t - \mathbf{x}_{i2}^t) \times (\mathbf{x}_{i1}^t - \mathbf{x}_{i3}^t)}{||(\mathbf{x}_{i1}^t - \mathbf{x}_{i2}^t) \times (\mathbf{x}_{i1}^t - \mathbf{x}_{i3}^t)||_2}, \quad (1)$$

where $\mathbf{n}_i^t$ has an unit length. For each vertex of the triangle, the following geometrical constraint then holds:

$$\mathbf{n}_i^t \cdot \mathbf{x}_{ij}^t - h_i^t = 0. \quad (2)$$

Therefore, h_i^t can also be estimated. Here, we also add two additional constraints, i.e., $n_{i,y}^t > 0$ and $h_i^t > 0$, based on the fact that the camera is mounted on the top of the vehicle and is above the ground plane, as shown in Fig.1.

Note that the triangles are scattered in the whole image plane, hence we need to identify the ones located on the ground plane, named *ground triangles*, for estimating the plane parameters. Based on the fact that the normal of a ground triangles is orthogonal to camera translation $\boldsymbol{t}_{t,t-1}$, and that the pitch angle of the camera is zero, the ground triangles can be identified by testing with the following constraints,

$$\arccos(\mathbf{n}_i^t, \boldsymbol{t}_{t,t-1}) = 0,$$
$$|\arctan(-\frac{R_{32}}{R_{33}})| = 0, R_{33} \neq 0. \quad (3)$$

In practice, the equality condition cannot be strictly satisfied. We thus set a tolerance value of $5°$ in the actural implementation.

For ground triangles that satisfy the above constraints, their vertices are categorized into a new point set $\tilde{\mathbf{x}}_{ij}^t, i \in \{1, 2, \ldots K\}, j \in \{1, 2, 3\}, K < M$. Since the same vertex point may be shared by multiple triangles, we also need to remove the repetitive ones from the point set. This will ensure the same contribution of each point to the ground plane estimation.

The ground points are now initially segmented out, denoted by $\tilde{\mathbf{x}}_k^t \in \mathcal{G}$, but there may still exist some outliers introduced by moving objects and some remote points. To further improve the quality of $\mathcal{G}$, a RANSAC-based method is leveraged to optimize $\tilde{\mathbf{x}}_k^t$, which minimizes a plane-distance error as follows,

$$\min_{\tilde{\mathbf{x}}_g^t \in \mathcal{G}} \sum_{g=1}^{|\mathcal{G}|} ||\mathbf{n}^t \cdot \tilde{\mathbf{x}}_g^t - h^t||_2. \quad (4)$$

In the implementation, we randomly sample three points to estimate a new plane with eq. (1)-(2), and then we calculate the total distance of the remaining points to the estimated plane. Such a process repeats Z times, and the plane that induces the minimum total distance error is reserved. The points with a distance larger than 0.01m to the reserved plane are then removed. This is actually a more strict criterion for ground point selection. After this process, only high-quality

---

Algorithm 1: Ground Point Extraction (GPE)

Input: $\mathbf{u}_i^t$, $R_{t,t-1}$, $\boldsymbol{t}_{t,t-1}$
Output: $\{\tilde{\mathbf{x}}_g^t\}$

1 $\{\Delta_i^t\} \leftarrow$ DELAUNAYTRIANGULATION($\mathbf{u}_i^t$)
2 triangles points set $\{\tilde{\mathbf{x}}_{ij}^t\} \leftarrow \emptyset$
3 segmented points set $\{\tilde{\mathbf{x}}_k^t\} \leftarrow \emptyset$
4 ground points set $\mathcal{G}_{best} \leftarrow \emptyset$
5 temp ground points set $\mathcal{G}_k \leftarrow \emptyset$
6 **for** *each* $\mathbf{u}_{ij}^t \in \{\Delta_i^t\}, j = \{1, 2, 3\}$ **do**
7   back-project $\mathbf{x}_{ij}^t = K^{-1}\mathbf{u}_{ij}^t$
8   calculate $\mathbf{n}_i^t$ by Eq.(1)
9   calculate h_i^t by Eq.(2)
10   **if** $|\arctan(-\frac{R_{32}}{R_{33}})| < \theta_1$ & $h_i^t > 0$ **then**
11   **if** $\arccos(\mathbf{n}_i^t, \boldsymbol{t}_{t,t-1}) < \theta_2$ **then**
12             $\{\tilde{\mathbf{x}}_{ij}^t\} \cup \mathbf{x}_{ij}^t$
13   **end**
14   **end**
15 **end**
16 delete repeat vertices and get $\{\tilde{\mathbf{x}}_k^t\} \subset \{\tilde{\mathbf{x}}_{ij}^t\}$
17 /* ensure enough points */
18 **while** *size* $(\{\tilde{\mathbf{x}}_k^t\}) > 5$ **do**
19   **for** *iterations* $< Z$ **do**
20   randomly select $\mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t \in \{\tilde{\mathbf{x}}_k^t\}$
21   calculate $\mathbf{n}^t$ by Eq.(1)
22   **for** $\tilde{\mathbf{x}}_g^t$ *in* $\{\tilde{\mathbf{x}}_k^t\}$ **do**
23             $D \leftarrow |\mathbf{n}^t \cdot \tilde{\mathbf{x}}_g^t - h^t|$ by Eq.(4)
24   **if** $D < d_{thresh}$ **then**
25                 $\mathcal{G}_k \cup \{\tilde{\mathbf{x}}_g^t\}$
26   **end**
27   **end**
28   **if** *size* $(\mathcal{G}_k) >$ *size* $(\mathcal{G}_{best})$ **then**
29             $\mathcal{G}_{best} \leftarrow \mathcal{G}_k$ // best model select
30   **end**
31   **end**
32     $\{\tilde{\mathbf{x}}_g^t\} \leftarrow \mathcal{G}_{best}$
33   **return** $\{\tilde{\mathbf{x}}_g^t\}$
34 **end**

---

ground points are reserved. In Alg. 1, we present a complete procedure of the GPE algorithm, which gives more details about the proposed implementation.

B. Ground Point Aggregation

Due to the strict criteria by GPE, the inliers are not enough for accurate estimation of the ground plane. Therefore, we propose the GPA algorithm to aggregate ground points from consecutive image frames. As shown in Fig. 2, we leverage the sliding window method to select image frames, and a frame buffer is maintained to store the camera poses and ground points in the current window. At each time step, with the arrival of a new image frame, we update the buffer and then estimate the ground plane by solving a least-squares problem.

From the MVO thread, we can get the pose T_t and the inliers $\mathbf{x}_g^t$ of each frame from the buffer. We then can
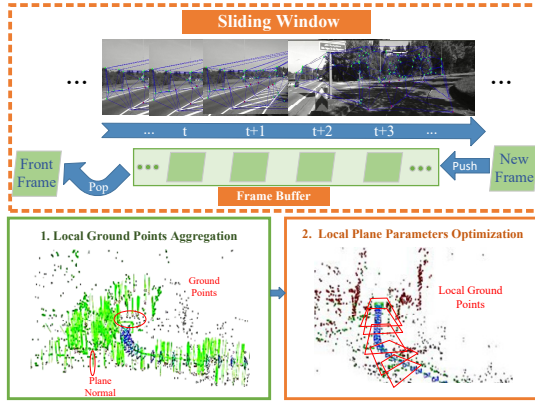
Fig. 2. Illustration of the GPA algorithm. In the bottom-left figure, the red dots indicate ground points, and the green line segment is the normal of each triangle. In the bottom-right figure, the red quadrilateral is the estimated ground plane based on the aggregated ground points in the current window.

transform each of the inliers, denoted by $\mathbf{x}_{gi}^t$, into the global frame,

$$\mathbf{p}_i = T_t\left[\mathbf{x}_{gi}^t, 1\right]^T. \tag{5}$$

Suppose there are N local ground points in the buffer, the least-squares problem that minimizes the plane-distance error of these points is formulated as follows:

$$\min_{\boldsymbol{\mu}} \sum_{i=1}^{N} \|\boldsymbol{\mu}^T \mathbf{p}_i\|_2 = \min_{\boldsymbol{\mu}} \boldsymbol{\mu} P_t P_t^T \boldsymbol{\mu}^T, \tag{6}$$

$$\boldsymbol{\mu} = [\mathbf{n}_t, -h_t]^T,$$

where $P_t = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N] \in R^{4 \times N}$. Equation (6) can be rewritten as follows:

$$\min_{\boldsymbol{\mu}} \boldsymbol{\mu} Q_t \boldsymbol{\mu}^T, \tag{7}$$

$$Q_t = P_t P_t^T \in R^{4 \times 4},$$

which can then be efficiently solved by the SVD method.

To further improve the estimation accuracy of $\boldsymbol{\mu}$, we also introduce a weighting matrix $\Sigma = \sigma_z^{-2}I$, where σ_z is the normalized distance of the point depth to their mean value. As a result, the matrix Q in Eq. (7) becomes,

$$Q_t = P_t \Sigma P_t^T \in R^{4 \times 4}. \tag{8}$$

Another important refinement on the estimated plane parameter is to conduct a RANSAC-based optimization, which shares the same idea as Eq. (4). In each iteration of the optimization, we first estimate $\boldsymbol{\mu}$, and then calculate the distance between $\mathbf{p}_i$ and the estimated plane. Points with a distance larger than 0.01m are removed, and the remaining is then leveraged to estimate a new $\boldsymbol{\mu}$. Such a process continues until convergence. We denote the final plane normal by $\mathbf{n}_t^*$ and the reserved ground points by $\mathbf{p}_k^*, k \in \{1, 2, \cdots, K\}$. The relative camera height then can be calculated by projecting the camera center to the ground plane:

$$h_j^t = \mathbf{n}_t^* \cdot (\mathbf{p}_c - \mathbf{p}_k^*), \tag{9}$$

where $\mathbf{p}_c$ is the camera center of frame I_t in the global frame. It is worth noting that there are K estimated camera

---

Algorithm 2: Ground Points Aggregation (GPA)

Input: I_t, $\{\mathbf{x}_g^t\}$, T_t
Output: $\{h^t\}$
1   buffer $\{queue\} \leftarrow \emptyset$
2   inliers in global frame $\{\mathbf{p}_t\} \leftarrow \emptyset$
3   inliers in current frame $\{\mathbf{x}_g^t\} \leftarrow \emptyset$
4   reserved ground points $\{\mathbf{p}_k^*\} \leftarrow \emptyset$
5   camera heights $\{h_t\} \leftarrow \emptyset$
6   **while** I_t *is not empty* **do**
7     $\{queue\} \cup \{I_t\}$
8   **if** $size(\{queue\}) > 4$ **then**
9   POP_FRONT($\{queue\}$)
      `// fixed number of frames`
10   **for** *each* I_t *in* $\{queue\}$ **do**
11   calculate $\mathbf{p}_i$ by Eq.(5)
12       $\{\mathbf{p}_t\} \cup \{\mathbf{p}_i\}$
13   **end**
14   Matrix P_t
15   calculate $\Sigma = \sigma_z^{-2}I$ `// Weighting Matrix`
16   calculate Q_t by Eq.(8)
17      $\boldsymbol{\mu} \leftarrow$ SVDDECOMPOSITION(Q)
18   **end**
19   **for** *iteration* $< Z$ **do**
20   Randomly select $\mathbf{p}_s \in \{\mathbf{p}_t\}$
21      $D \leftarrow \boldsymbol{\mu}^T \cdot \mathbf{p}_s$
22   **if** $D > d_{thresh}$ **then**
23   Remove $\mathbf{p}_s$ in $\{\mathbf{p}_t\}$
24   **end**
25      $\{\mathbf{p}_k^*\} \leftarrow \{\mathbf{p}_t\}$
26   calsulate Q_s by Eq.(8)
27      $Q \leftarrow Q_s$
28      $\boldsymbol{\mu_s} \leftarrow$ SVDDECOMPOSITION(Q)
29      $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu_s}$ `// update model`
30   **end**
31   **for** $\mathbf{p}^*$ *in* $\{\mathbf{p}_k^*\}$ **do**
32   calculate h_j^t by Eq.(9)
33      $\{h^t\} \cup \{h_j^t\}$
34   **end**
35   **return** $\{h^t\}$
36 **end**

heights, which will be further processed to recover a smooth scale. Details of the GPA algorithm are presented in Alg. 2.
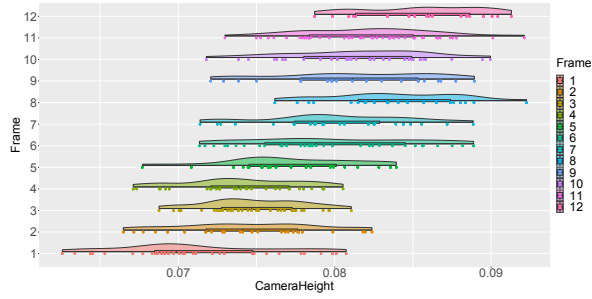
C. Filter-Based Scale Recovery

After we compute the relative camera height h of each frame, the scale factor is then obtained by $s_t = h^*/h$, the motion scale of each frame is recovered by

$$\boldsymbol{t}_{t,t-1}^{\dagger} = s_t \cdot \boldsymbol{t}_{t,t-1}. \tag{10}$$

Corresponding to the multiple h values in Eq. (9), there are also multiple estimated scales.

By plotting the scaled camera heights of each frame in the figure, shown in Fig. 3(a), we find the data do not strictly follow a Gaussian distribution. Therefore, we choose the

median point as the scale of the current frame. In the time domain, a moving average filter is applied, shown in Fig. 3(b), which can give a more smooth result.

(a) The distribution of the scaled camera heights.
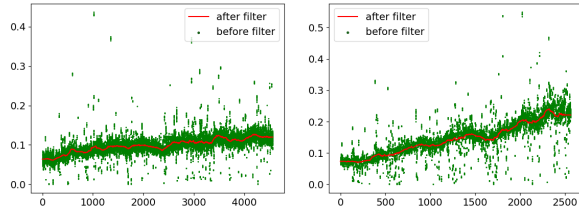
(b) The estimated camera height on sequence-02 and -05.

Fig. 3. Demonstration of the filter-based scale recovery. The green points are the scaled camera heights. The red curve is the smoothed one.

IV. EXPERIMENTS

We conduct experiments to evaluate the performance of our proposed method. The MVO system used in the experiments is implemented based on ORB-SLAM2, and the proposed scale recovery method is integrated as an independent thread. The system architecture is demonstrated in Fig. 1. The KITTI dataset [20] is adopted as the benchmark dataset, in which sequence-01 is not used since it fails most feature-based VO systems. All the experiments are conducted using a laptop with Intel(R) Core(TM) i5-6300HQ CPU @ 2.30 GHz.

A. Qualitative Evaluation

The qualitative evaluation results of the proposed method are visualized in Fig. 4. The trajectories outputted by the system are recovered using the proposed method, which means similarity transformation is not necessary when compared with ground-truth trajectories [21]. The baseline trajectories, indicated by green color in Fig. 4, are generated by ORB-SLAM2 with loop closure detection disabled, denoted by ORB-SLAM2-noLC. We can see that our re-scaled trajectories can eliminate scale drift and form correct loops, which demonstrates the effectiveness of the proposed method.

The comparison of trajectory length between the ground truth and the proposed method is shown in Table I, in which the Relative Length Error (RLE) is computed by $e = |l_{gt} - l_{ours}|/|l_{gt}|$, where l is the length of a trajectory. For sequence-02, -04, -06, -07, and -10, the RLE is less than 1%. The high performance is due to the fact that most roads

TABLE I
COMPARISON OF RE-SCALED TRAJECTORY
LENGTH WITH GROUND TRUTH

Seq	GT (m)	Ours (m)	RLE (%)
00	3645.213	3724.419	2.173
02	5071.151	5067.233	0.757
03	547.774	560.888	0.558
04	391.861	393.645	0.455
05	2175.159	2205.576	1.398
06	1235.982	1232.876	0.251
07	709.065	694.697	0.767
08	3137.398	3222.795	2.722
09	1738.601	1705.051	1.929
10	911.399	919.518	0.890

in these sequences are straight lines and contain rich features. Sequence-00 and -08 are more complicated cases, in which the scenario is composed of a lot of curves and turns. The path lengths are relatively longer than other sequences. The RLE is thus slightly higher, 2.17% and 2.72%, respectively. Nevertheless, the results show that the proposed system can estimate an accurate scale of the entire trajectory.

B. Quantitative Evaluation

The quantitative comparison between our method and the baseline methods, including [1], [10], [11], [16], [17], [22], is presented in Table II. The average translation error and rotation error are adopted as evaluation metrics.

We can see that ORB-SLAM2-noLC and VISO2-M have the worst performance due to the lack of loop closure detection. The scale drift of the two methods induces a large translation error, 18.17% and 14.39% respectively, while the VO systems with scale recovery all maintain a low translation error, < 4%. It can also be seen that a MVO system with scale recovery [1], [11], [16], [17] can exhibit competitive performance with a stereo VO system like VISO2-M [22], which significantly demonstrates the importance of scale recovery for MVO.

In terms of monocular systems, we can see our proposed method achieves the minimum translation error while maintaining a competitive performance on rotation error. The methods proposed by Song et al. [1] and Zhou et al. [11] can not work with sequence-07, because they both rely on a fixed region to extract ground points, whereas occlusions by moving vehicles occur frequently in this sequence. In contrast with [1], [11], the proposed method works well with sequence-07 with a translation error of 1.77%, benefiting from the GPA algorithm.

In [16], a deep neural network, named DNet, is proposed for monocular depth prediction and scale recovery. Compared with this method, our method shows a better accuracy in all the sequences. In [17], a real-time MVO system is implemented based on self-supervised learning techniques. This method can slightly outperform our proposed method in sequence-05 and -09, but has a much lower accuracy in sequence-00, -08, and -10. A similar phenomenon can be observed when comparing with [1]. This indicates a
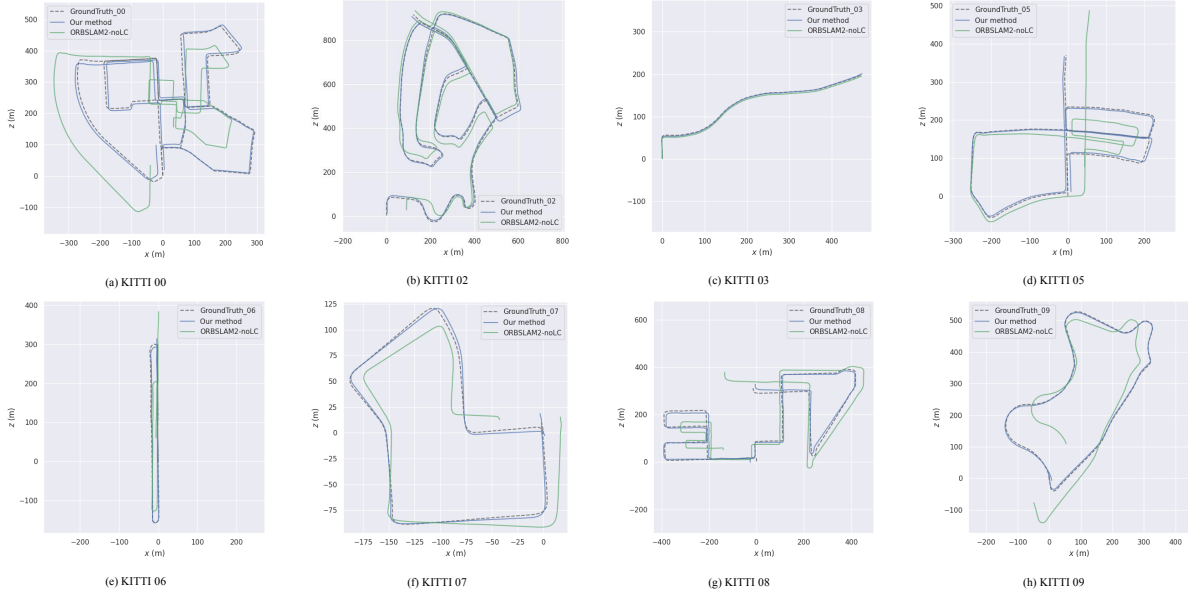
Fig. 4. The re-scaled trajectories on KITTI dataset. The blue and green trajectories are generated by our system and ORB-SLAM2-noLC, respectively.

TABLE II

COMPARISON OF AVERAGE TRANSLATION ERRORS AND ROTATION ERRORS WITH THE LATEST VISUAL ODOMETRY METHODS ON KITTI DATASET

Seq	ORB-SLAM2-noLC [10]		VISO2-M [22]		VISO2-Stereo [22]		Song et.al [1]		Zhou et.al [11]		Brandon et.al [17]		DNet [16]		Ours	
	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)
00	20.8	–	11.91	0.0209	2.32	0.0109	2.04	0.0048	2.17	0.0039	1.86		1.94	–	**1.41**	0.0054
01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
02	9.52	–	3.33	0.0114	**2.01**	0.0074	1.50	0.0035	–	–	2.27	–	3.07	–	2.18	0.0046
03	–	–	10.66	0.0197	2.32	0.0107	3.37	0.0021	2.70	0.0044	–	–	–	–	**1.79**	0.0041
04	–	–	7.40	0.0093	0.99	0.0081	2.19	0.0028	–	–	–	–	–	–	**1.91**	0.0021
05	18.63	–	12.67	0.0328	1.78	0.0098	1.43	0.0038	–	–	**1.50**	–	3.32	–	1.61	0.0064
06	18.98	–	4.74	0.0157	**1.17**	0.0072	2.09	0.0081	–	–	2.05	–	2.74	–	2.03	0.0044
07	13.82	–	–	–	–	–	–	–	–	–	1.78	–	2.74	–	**1.77**	0.0230
08	22.06	–	13.94	0.0203	2.35	0.0104	2.37	0.0044	–	–	2.05	–	2.72	–	**1.51**	0.0076
09	12.74	–	4.04	0.0143	2.36	0.0094	1.76	0.0047	–	–	**1.50**	–	3.70	–	1.77	0.0118
10	4.86	–	25.20	0.0388	1.37	0.0086	2.12	0.0085	2.09	0.0054	3.70		5.09	–	**1.25**	0.0031
Avg	18.17	–	14.39	0.0245	2.32	0.0095	2.03	**0.0045**	2.32	0.045	2.03	–	3.17	–	**1.72**	0.0068

significant variance on the performance of [17]. Actually, this is the limitation of most deep learning based methods, which has been discussed in detail by [18].

The comparative experiments in Table II significantly verify the effectiveness of our method and demonstrates the advantages over the latest methods in the literature.

C. Efficiency Evaluation

Another significant advantage of our method lies in its high efficiency. We evaluate the run-time of our system on all the KITTI sequences mentioned above, and the experiment repeats five times. The median run-time is reported in Fig. 5. In all the experiments, the MVO thread requires 50-55 ms, while the GPE and GPA requires less than 10 ms, which makes the system suitable for real-time applications.

V. CONCLUSIONS

In this work, we present a light-weight MVO system with accurate and robust scale recovery, aiming to reduce scale drift and provide accurate odometry in long-distance navigations without loop closure. We solve the scale ambiguity for MVO by implementing our GPE-GPA algorithm for selecting high-quality points and optimizing them in a local sliding window. Sufficient data and robust optimizer provide accurate metric trajectory leveraging the ratio of the estimated camera height and the real one. Extensive experiments show that our proposed framework can achieve state-of-the-art accuracy and recover a metric trajectory without additional sensors. The system is designed to be a light-weight framework, which can achieve real-time performance with 20 Hz running frequency. Our proposed light-weight MVO system facilitates the localization and navigation of low-cost autonomous vehicles in long-distance scenarios.

Further study into integrating the uncertainty of plane estimation will be considered, which will further improve the accuracy of scale recovery. The light-weight neural network for ground segmentation will also be considered to help constrain the extraction of high-quality ground points.
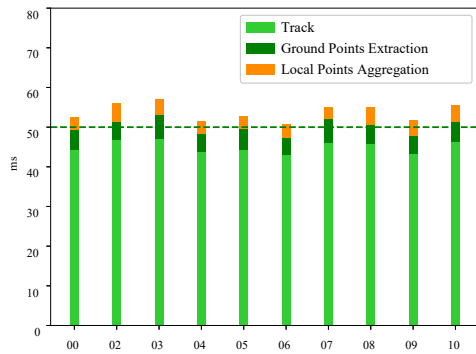
Fig. 5. Run-time of our MVO system on KITTI dataset. The time costs of MVO thread, the GPE algorithm, and the GPA algorithm are reported.

REFERENCES

[1] S. Song, M. Chandraker, and C. C. Guest, "High accuracy monocular sfm and scale correction for autonomous driving," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 730–743, 2015.

[2] D. Zhou, Y. Dai, and H. Li, "Reliable scale estimation and correction for monocular visual odometry," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016, pp. 490–495.

[3] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr, "Eao-slam: Monocular semi-dense object slam based on ensemble data association," *arXiv preprint arXiv:2004.12730*, 2020.

[4] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart, "Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints," in *IEEE International Conference on Computer Vision*, 2009.

[5] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," in *Robotics: Science and Systems VI, Universidad de Zaragoza, Zaragoza, Spain, June 27-30, 2010*, 2010.

[6] D. Zhu, Y. Du, Y. Lin, H. Li, C. Wang, X. Xu, and M. Q.-H. Meng, "Hawkeye: Open source framework for field surveillance," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6083–6090.

[7] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[8] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4973–4980.

[9] T. Li, D. Zhu, and M. Q.-H. Meng, "A hybrid 3dof pose estimation method based on camera and lidar data," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2017, pp. 361–366.

[10] 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.

[11] D. Zhou, Y. Dai, and H. Li, "Ground-plane-based absolute scale estimation for monocular visual odometry," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 791–802, 2019.

[12] Qifa Ke and T. Kanade, "Transforming camera geometry to a virtual downward-looking camera: robust ego-motion estimation and ground-layer detection," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, 2003, pp. I–I.

[13] J. Gräter, T. Schwarze, and M. Lauer, "Robust scale estimation for monocular visual odometry using structure from motion and vanishing points," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 475–480.

[14] X. Yin, X. Wang, X. Du, and Q. Chen, "Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5870–5878.

[15] L. Andraghetti, P. Myriokefalitakis, P. L. Dovesi, B. Luque, M. Poggi, A. Pieropan, and S. Mattoccia, "Enhancing self-supervised monocular depth estimation with traditional visual odometry," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 424–433.

[16] F. Xue, G. Zhuo, Z. Huang, W. Fu, Z. Wu, and M. H. Ang Jr, "Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications," *arXiv preprint arXiv:2004.05560*, 2020.

[17] B. Wagstaff and J. Kelly, "Self-supervised scale recovery for monocular depth and egomotion estimation," *arXiv preprint arXiv:2009.03787*, 2020.

[18] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," *arXiv preprint arXiv:2003.14338*, 2020.

[19] J. R. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," in *Workshop on Applied Computational Geometry*. Springer, 1996, pp. 203–222.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

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

[22] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011.