Lidar-Monocular Visual Odometry using Point and Line Features

Abstract— **We introduce a novel lidar-monocular visual odometry approach using point and line features. Compared to previous point-only based lidar-visual odometry, our approach leverages more environment structure information by introducing both point and line features into pose estimation. We provide a robust method for point and line depth extraction, and formulate the extracted depth as prior factors for point-line bundle adjustment , which greatly reduces the features' 3D ambiguity and thus improves the pose estimation accuracy. Besides, we also provide a purely visual motion tracking and a novel scale correction scheme , leading to an efficient lidar-monocular visual odometry system with high accuracy. The evaluations on the public KITTI odometry benchmark show that our technique achieves more accurate pose estimation than the state-of-the-art approaches, and is sometimes even better than those leveraging semantic information.**

I. Introduction

Lidar-visual odometry has been an active research topic due to its wide applications such as robotics, virtual reality, and autonomous driving etc. The combination of visual sensors and lidar sensors as lidar-visual odometry can achieve the benefits of both types of sensors ([1], [2], [3]), and thus has been gaining more and more research interests in the computer vision, computer graphics and robotics communities nowadays.

Due to the sparse depth information and limited field of view (FoV) coming from the lidar, a lot of pixels in resulting images do not have their corresponding depth required for the fusion. This makes the lidar-visual odometry challenging to achieve the ambitious goals of accurate motion estimation meanwhile ensuring the scale consistency and global consistency during the lifelong SLAM (Simultaneous Localization and Mapping). The problem is even more serious for the monocular case due to the absence of scale information.

Recently, tightly coupled fusion algorithms like V-LOAM [1] show impressive performance for visual-enhanced lidar odometry. Those algorithms follow the odometry framework, which does not have a back end with an optimization for bundle adjustment. However, it is clear that using SLAM techniques such as bundle adjustment and loop closure can achieve much better motion estimation accuracy for a lifelong SLAM system, as shown by ORB-SLAM2 [4].

Point-based bundle adjustment has been effectively used in lidar-monocular visual odometry systems such as LIMO [3] and DVL-SLAM [5], [6]. Although DVL-SLAM [5] follows the *direct-based* visual SLAM (as like DSO [7]) in camera tracking without extracting 2D point features in the image plane, it still leverages salient points from lidar data for both pose estimation and bundle adjustment to ensure the accuracy. However, direct-based visual SLAM methods suffer from the problem of lighting, which limits their robustness and further accuracy improvement for DVL-SLAM [5] in lifelong scenarios. DEMO [2] and LIMO [3] follow the *feature-based* visual SLAM in camera tracking like ORB-SLAM2 [4] to avoid the light changing problem. However, the accuracy of those point-only systems is still not very satisfactory and they like LIMO [3] sometimes require extra semantic information as input, which, however, is computationally expensive.

The recent techniques for 3D reconstruction using structure from motion ([8], [9]) and accurate motion estimation of visual SLAM systems ([10], [11]) show that utilizing more structural information from real environments, such as line features, leads to more accurate camera pose estimation. Besides, line feature has conditional benefits that it is less sensitive with problems such as noise [12], wide range of view angle [13] and motion blur [14], which are the main drawbacks for the point-only system such as LIMO [3]. This motivates us to combine point and line features together for an accurate lidar-visual odometry system. However, there is an open issue for line-based visual SLAM systems since line-based 3D triangulation can be sensitive during camera tracking [15], thus causing an unstable visual SLAM system without satisfactory pose estimation accuracy. Although line features might richly exist in various scene environments (especially in urban environments), it is nontrivial to directly adopt line features for the lidar-visual odometry.

In this paper, we provide a robust and efficient lidar-monocular visual odometry method combining both point and line features in a purely geometric way, which extracts more structural information from scene environments than point-only systems. More specifically, our system fuses the point and line features as landmarks during camera tracking, and formulates the point-based and line-based landmarks' reprojection errors as factors for bundle adjustment in the back end. During sensor fusion, we provide a robust method to extract the depth of the points and lines from the lidar data, and use the depth prior to guide camera tracking. In this way, we avoid the creation of 3D landmarks solely based on the possibly ambiguous 3D triangulation, especially for the 3D lines. The depth prior is also formulated as prior factors in the point-line bundle adjustment to further improve the pose estimation accuracy.

Besides, we perform a purely visual frame-to-frame odometry during camera tracking, and extract the depth prior only in keyframes for efficiency. To overcome the scale drift in the frame-to-frame odometry, we recover the scale in each keyframe using a scale correction optimization before the bundle adjustment. In this way, we achieve an efficient but

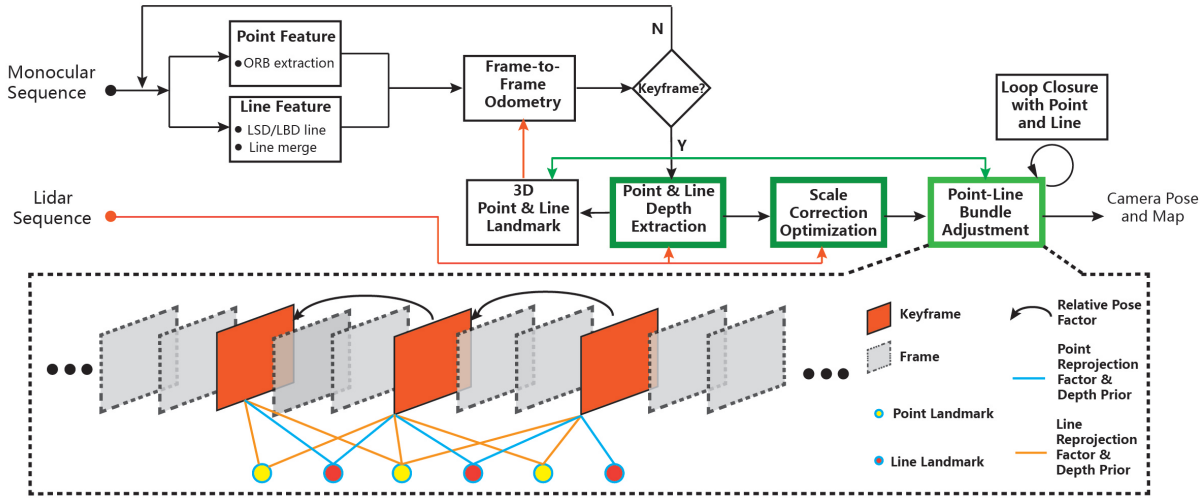

Fig. 1. The system framework of our approach. Given the input monocular image sequence and lidar sequence, we extract the point and line features for each image, and track image frames using our frame-to-frame odometry with scale corrected by our scale correction optimization. For each keyframe, the depth priors for point and line landmarks are extracted and fed to the point-line bundle adjustment. Loop closure with the point and line features is used for further pose estimation correction. The graph structure for bundle adjustment is illustrated in the dashed box with different factors. The components highlighted in the figure are our main contribution in this work.

driftless lidar-visual odometry system. Benefited from a rich set of structural 3D landmarks, we achieve much higher pose estimation accuracy compared with the other purely geometric approaches (such as DEMO [2] and DVL-SLAM [5]) evaluated on the KITTI odometry benchmark [16], and comparable accuracy with LIMO [3] but without using extra semantic information. To the best of our knowledge, we give the first efficient lidar-monocular odometry approach using the point and line features together in a purely geometric way.

II. Methodology

Preprocessing. Given a monocular image sequence and a lidar sequence, we assume that the image data and the lidar data have been temporally aligned, similar to the KITTI odometry benchmark [16]. We also assume that the monocular camera is calibrated with known intrinsic parameters, and the extrinsic transformation between the camera coordinate system and the lidar coordinate system is already known. We use the camera's local coordinate system as the two sensors' local coordinate system, and set the world coordinate system as the camera's local coordinate system at the beginning of the sequences.

Overview. Fig. 1 shows the system framework of our approach. We follow a classic visual SLAM framework like ORB-SLAM2 [4], which has three running threads: a motion tracking thread as front-end, a graph based optimization thread as back-end, and a loop closure thread. In the front-end, we first extract the point and line features for each image (Section II-A), and then estimate the camera poses following our purely visual frame-to-frame odometry (Section II-C). The point and line feature's depth prior is extracted in a robust way (Section II-B) in every keyframe. Before a keyframe is set to the back end, a scale correction optimization (Section II-D) is performed to correct the scale drift after the frame-to-frame odometry.

In the back-end, we perform point-line bundle adjustment (Section II-E) with several constraint factors including those derived from the point-line features. To further adjust the keyframe poses, we perform a bag-of-words based loop closure [17] using the point and line features as described in Section II-F.

A. Feature Extraction

Point Feature. Various point features, like SIFT, SURF, ORB etc, can be used to serve as tracking features. To maintain the accuracy and efficiency, we adopt the Oriented fast and Rotated BRIEF (ORB) feature as the point feature, as done in ORB-SLAM2 [4]. During detection, the ORB features are required to be evenly distributed in the image as much as possible.

Line Feature. For each image, we use the popular line feature detector, Line Segment Detector (LSD) [18], to detect line segments and the Line Band Descriptor (LBD) [19] to compute the descriptors for each line segments.

The LSD algorithm provides sub-pixel accuracy for line segment detection and works on diverse scene images without heavy tuning of parameters. However, it often breaks a long line segment into several short ones, as shown in Fig. 2 (Left). Additionally some detected line segments may be quite near, such as the border edges of a thin area as shown in Fig. 2 (Right). The existence of such detected line segments often makes the subsequent line matching task complex, thus increasing the uncertainty of 3D line triangulation. To address these issues we propose to improve the results of the LSD algorithm by merging such "bad" line segments, as shown in Fig. 2. More specifically, for a line segment pair (l_i, l_j), if only one endpoint of l_i is near to l_j's endpoint, we

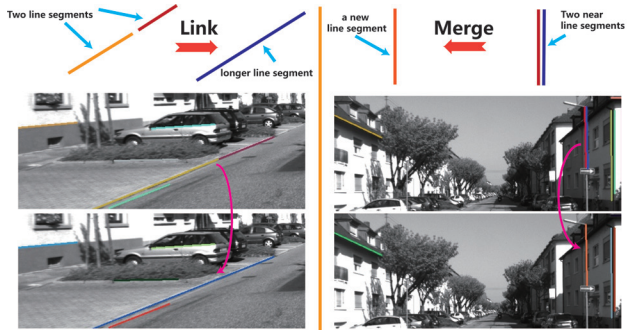

Fig. 2. We chain short line segments into a longer one (Left), or merge near line segments into a new one (Right) to enhance the quality of lines returned LSD. The images come from KITTI dataset sequence 00.

link l_i and l_j to form a longer line segment. If both endpoints of l_i are near to those of l_j and the distance between l_i and l_j is under a given threshold, we merge l_i and l_j as a new line segment. The LBD line descriptor is also updated for the linked or newly merged line segment.

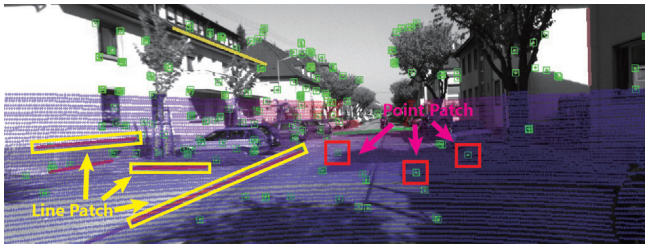

Fig. 3. A simple illustration for point and line depth extraction. After the sparse lidar data is aligned to the image plane, the point depth and line depth are extracted in the point neighbor patch and line neighbor patch separately.

B. Point and Line Depth Extraction

In this section we describe a method to extract the point and line depth from the lidar data. Here the depth for a 2D point feature means the depth of its corresponding 3D point, and for a 2D line feature means the depth of the two endpoints of its corresponding 3D line landmark.

As shown in Fig. 3, the sparse lidar points are aligned to the image plane according the calibration matrix. For each 2D point or line feature detected , a neighbor patch with a given size is searched to estimate the depth prior separately. More specifically, the lidar points belonging to a point patch are first segmented into foreground points and background points using the method introduced by Tateno et al. [20]. Then a plane is fitted using the foreground points , and the depth of the feature point is that of the intersection point between the point ray and the plane.

For the lidar points $P = \{p_i\}$ belonging to a line patch of an image line feature l_j, we first estimate the corresponding 3D line L_j following an optimization, in which the total distance between the lidar points and the optimized 3D line is optimized while the reprojection error between the optimized 3D line and image line feature is minimized. More specifically, we estimate the optimized 3D line L_j^* by minimizing the following energy:

$$E(L_j^*) = \sum_i d(p_i, L_j^*) + e(L_j^*, l_j),$$

with $d(p_i, L_j^*)$ is the Euclidean distance between lidar point p_i and L_j^*, and $e(L_j^*, l_j)$ is the line reprojection error as decribed in Section II-E. After L_j^* is estimated, the two end points priors (as shown in Fig. 5(left)) can be obtained using endpoint trimming as described in ([9], [10]). The extracted depth of the point and line features is served as 3D landmarks in the frame-to-frame odometry in Section II-C and also formulated as the depth prior in the bundle adjustment in Section II-E.

We adopt the Plücker coordinates [8] to represent a 3D line during the motion tracking, and the orthonormal representation [8] as the minimum parameters. This optimization can be efficiently solved by Levenberg-Marquardt algorithm[21].

C. Frame-to-Frame Odometry

We use a pure visual Frame-to-Frame Odometry to estimate very frames' camera pose, which is more efficient than other ICP-based odometry like V-LOAM[1]. For a new frame F_i, we estimate its camera pose $T_{F_i} \in SE(3)$ using the already estimated frames' camera poses and 3D (point and line) landmarks optimized by the bundle adjustment (Section II-E). The detected 2D point features (ORB) and line features (LBD) in F_i are matched with the previous frame F_{i-1}'s tracking features by performing 2D-2D point matching and line matching, respectively. The estimated 3D point landmarks and 3D line landmarks seen in F_{i-1} are extracted to estimate F_i's camera pose by solving a perspective-n-point/line problem [22].

Since the descriptor distance might not be so descriptive for 2D-2D line matching, in practice we use several geometric criteria to filter out candidate line matching pairs with poor quality to improve the matching accuracy. Specifically, a good candidate line matching pair (l_i, l_j) needs to satisfy : (1) the angular difference is smaller than a given threshold; (2) the length difference is also below a given threshold; (3) the distance between the middle points of l_i and l_j should be small enough. After solving the perspective-n-point/line problem, the current frame F_i's camera pose T_{F_i} is updated. The 3D point landmarks and line landmarks are re-projected to frame F_i's image plane to check whether the 3D-2D matching is outlier or not. Matching outliers are removed once detected. As for the details of line camera projection, endpoints trimming, and line triangulation, please refer to [9], [10].

D. Scale Correction Optimization

Since the scale estimated by our frame-to-frame odometry (pure visual odometry) might drift from the real physical scale, we correct the scale by adjusting the estimated frame's camera pose and 3D landmarks to ensure the local scale consistency. Our key idea is to fuse the relative camera poses computed from the ICP point cloud alignment step to adjust

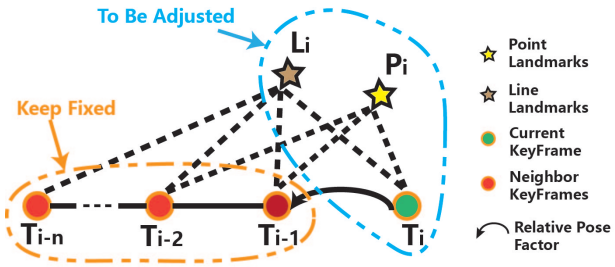

Fig. 4. The graph structure for scale correction optimization.

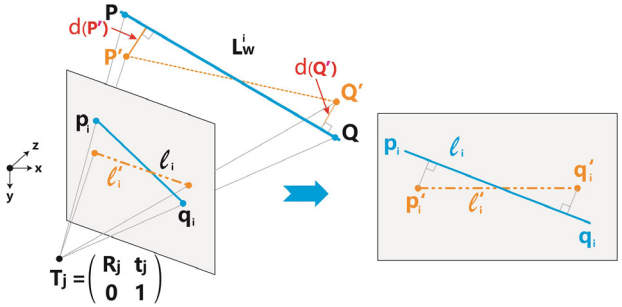

Fig. 5. A 3D line landmark L_w^i is re-projected onto the image plane yielding a 2D line l_i matched with a line segments l'_i(Left), with P' and Q' are the priors computed from the depth prior extracted in Section II-B. The error between the re-projected line l_i and the matched line segment l'_i is defined by the distances from its two endpoints to the line l_i (Right).

the newly estimated keyframe's camera pose and the related 3D landmarks via a scale correction optimization.

This problem is formulated as a graph based optimization as illustrated in Fig. 4. For a newly selected keyframe K_i with the estimated camera pose $T_i \in SE(3)$, we first extract a set of 3D point landmarks P_i and a set of 3D line landmarks L_i, which are visible in K_i. We also extract the past keyframe set $\{K_{i-1}, K_{i-2}, ..., K_{i-n}\}$ with their corresponding camera poses $\hat{\mathcal{T}}_i = \{T_{i-1}, T_{i-2}, ..., T_{i-n}\}$, through which P_i and L_i are visible. Thus we obtain a local keyframe window $\mathcal{T}_i = \{T_i, T_{i-1}, ..., T_{i-n}\}$. Our goal in the optimization is to adjust K_i's camera pose T_i, 3D point landmarks P_i, and 3D line landmarks L_i while keeping fixed for the previous keyframes' camera poses $\hat{\mathcal{T}}_i$. Specifically, this is achieved by minimizing the following error function:

$$E(T_i, P_i, L_i) = e^{icp}_{i,i-1} + \sum_{l \in \mathcal{T}_i, j \in P_i} e^P_{l,j} + \sum_{l \in \mathcal{T}_i, k \in L_i} e^L_{l,k},$$

which e^{icp}, e^P and e^L are the relative pose factor error, point re-projection factor error, and line re-projection factor error as described in Section II-E, respectively. This optimization can be solved efficiently using the *g2o* library [23].

In this way, the scale drift caused in the frame-to-frame odometry is corrected while keeping the local consistency. Note that when performing ICP alignment between $\{K_i, K_{i-1}\}$, the relative camera pose computed by the frame-to-frame odometry can be used as the initial guess to accelerate the ICP alignment. For efficiency, the time-consuming ICP alignment is only performed between keyframes.

E. Point-Line Bundle Adjustment

Unlike point-only bundle adjustment as LIMO [3], we add line landmarks into the optimization leading to point-line bundle adjustment. However, the 3D point and line landmarks estimated using the straightforward point-line bundle adjustment would have large drift since the landmarks are far from the cameras, especially in the wide outdoor scenes as those in the KITTI dataset. To make bundle adjustment more accurate, we use the relative camera poses of two adjacent keyframes estimated using ICP alignment (Section II-D) to constraint the keyframe's camera poses.

Specificially, when a new keyframe K_i arrives, bundle adjustment is performed in its neighboring keyframes set $\mathcal{K}_i$, with 3D point landmark set $\mathcal{P}_i$ and 3D line landmark $\mathcal{L}_i$. As illustrated in Fig. 1, we formulate bundle adjustment as a graph based optimization with the following three factors, aiming at adjusting the keyframes' camera poses $\mathcal{T}_i$, 3D point landmark position $\mathcal{P}_i$, and 3D line landmark position $\mathcal{L}_i$.

Relative Pose Factor. For a pair of adjacent keyframes $\{K_i, K_j\}$ with the estimated camera poses $\{T_i, T_j\}$, and the relative camera pose δT_{ij} computed from the point cloud ICP alignment step, let's formulate the difference vector ξ_{ij} between the relative camera pose $\Delta T_{ij} = T_i * T_j^{-1}$ (from the tracking estimation) and δT_{ij} as:

$$\xi_{ij} = Log(\Delta T_{ij} * \delta T_{ij}{}^{-1}) \in \mathbb{R}^6,$$

where $Log(\cdot)$ is the logarithmic map of $SE(3)$ [24]. The relative pose error e^{icp}_{ij} is computed as $e^{icp}_{ij} = \xi_{ij}{}^T \Sigma^{icp}_{ij} \xi_{ij}$, where Σ^{icp}_{ij} is a 6×6 information matrix given below.

ICP alignment would fail under certain conditions leading to "bad" relative camera poses. Once the relative camera pose δT_{ij} becomes "bad", the estimated camera poses $\{T_i, T_j\}$ would also be "bad" if we minimize the relative pose error directly. To penalize the "bad" ICP alignment, we compute the information matrix $\Sigma^{icp}_{ij}{}_{6\times6}$ adaptively according to the ICP alignment quality. Letting $\hat{e}_{ij}$ be the ICP error for setting the relative camera pose as ΔT_{ij} and $\bar{e}_{ij}$ for δT_{ij}, we compute a scale factor $\zeta_{ij} = e^{(-2.0*(\frac{\bar{e}_{ij}}{\hat{e}_{ij}})^2)}$ and set $\Sigma^{icp}_{ij} = \zeta_{ij} I_{6\times6}$.

Point Re-projection Factor. Assume a 3D point landmark ${}^wP_i \in \mathbb{R}^3$ in the world frame is matched to a 2D point feature positioned p_i in the keyframe K_j with camera pose $T_j(\xi_j)$. By re-projecting ${}^wP_i$ onto the image plane of keyframe K_j, we can obtain a re-projected position for ${}^wP_i$ as $p_i{}' = \pi({}^wP_i, T_j(\xi_j), \mathbb{K})$, where $\pi(\cdot)$ is the projection function and $\mathbb{K}$ is the intrinsic parameter matrix of keyframe K_j. We then compute the re-projection error as $\rho_{ij} = (p_i - p_i{}') \in \mathbb{R}^2$. If there exists depth prior for this point feature, we formulate ρ_{ij} as stereo-reprojection error as described in ORB-SLAM2 [4] by including the depth prior. Finally, the point re-projection error is defined as $e^P_{ij} = \rho_{ij}{}^T \Sigma^P_{ij} \rho_{ij}$, where Σ^P is the covariance matrix.

Line Re-projection Factor. A 3D line landmark $L_w^i(\phi_i), \phi_i \in \mathbb{R}^4$ [9] in the world frame can also be re-projected to a keyframe K_j with camera pose $T_j(\xi_j)$, obtaining a line segment $l_i = [l_1^i \ l_2^i \ l_3^i]^T \in \mathbb{R}^3$ in the 2D image plane as described in Section II-A. Suppose a line segment l_i' with two endpoints p_i' and q_i' is matched to L_w^i, as illustrated in Fig. 5. We can formulate the line re-projection error vector as: $\gamma_{ij} = [\frac{l_i{}^T p_i'}{\sqrt{l_1^{i\,2}+l_2^{i\,2}}} \quad \frac{l_i{}^T q_i'}{\sqrt{l_1^{i\,2}+l_2^{i\,2}}}]^T \in \mathbb{R}^2$. If there exists depth prior for this line feature, we compute the distance between end points priors P', Q' to the 3D line landmark as $d(P')$,$d(Q')$ separately as shown in Fig. 5(left). Then the line re-projection error vector is computed as: $\gamma_{ij} = [\frac{l_i{}^T p_i'}{\sqrt{l_1^{i\,2}+l_2^{i\,2}}} \quad \frac{l_i{}^T q_i'}{\sqrt{l_1^{i\,2}+l_2^{i\,2}}} \quad \frac{d(P_i')+d(Q_i')}{2}]^T \in \mathbb{R}^3$. Finally, the line re-projection error can be computed as $e_{ij}^L = {\gamma_{ij}}^T \Sigma_{ij}^L \gamma_{ij}$, where Σ^L is the covariance matrix.

In summary, our keyframe based bundle adjustment optimization seeks the minimization of the following cost function,

$$E = \sum_{i,j} e_{ij}^{icp} + \sum_{i,j} e_{ij}^P + \sum_{i,j} e_{ij}^L.$$

In solving the non-linear optimization, we adopt a 6D vector $\xi \in \mathbb{R}^6$ as the minimum parameters for a camera pose $T(\xi) \in \mathcal{T}_i$, a 3D vector ${}^wP \in \mathbb{R}^3$ for a point landmark, and a 4D vector $\phi \in \mathbb{R}^4$ for a line landmark $L(\phi) \in L_i$, to make the system computationally efficient and numerically stable. We solve this point-line bundle adjustment efficiently using the *g2o* library [23].

F. Loop Closure

Loop closure involves loop detection and loop correction based on the keyframes during motion estimation. For loop detection, we first use the DBoW [17] algorithm to train the vocabulary for the point feature (ORB descriptor) and line feature (LBD descriptor) respectively. Then every keyframe K_i is converted to a point BoW vector v_i^P and line BoW vector v_i^L. When evaluating the similarity between keyframes K_i and K_j, we define the similarity score as:

$$s(i,j) = e^{(1-s_p(v_i^P, v_j^P))^2} e^{(1-s_l(v_i^L, v_j^L))^2}$$

with $s_p(v_i^P, v_j^P)$ is the similarity score of the point BoW vector and $s_l(v_i^L, v_j^L)$ for the line BoW vector. Once the similar keyframe candidates for a newly selected keyframe K_i are found, the loop correction is performed using a global bundle adjustment algorithm [4] .

III. Experiments

Our system is implemented based on ORB-SLAM2 [4] with three threads, namely frame-to-frame odometry, keyframe based bundle adjustment, and loop closure. When performing ICP alignment, we use the Normal Distributions Transform (NDT) [25] implemented in the PCL library [26] to compute the relative camera pose between two adjacent point clouds. We test our approach on the training dataset of the public KITTI benchmark, which contains sequences 00-10 with ground truth trajectories, and analyze the accuracy and time efficiency of our approach on a computer with Intel Core i7-2600 @ 2.6GHz and 8G memory in a 64-bit Linux operation system.

Fig. 6. The motion tracking accuracy comparison between the point-only system (Left) and our system (Right), at the glance of frame 1380 of KITTI dataset sequence 00. The results by the point-only system have obvious drifts, while our results have drift free motion tracking.

Parameters. For each image, we extract 1000 ORB features. Line segments with length below 50 pixels are discarded. When performing line matching in Section II-C, we set the angle threshold as $2°$, length difference ratio as 0.1, and distance threshold as 5 pixels. In the bundle adjustment step, the covariance matrix is set $\Sigma^P = 0.1 * I$ for the point re-projection factor and $\Sigma^L = 0.2 * I$ for the line re-projection factor.

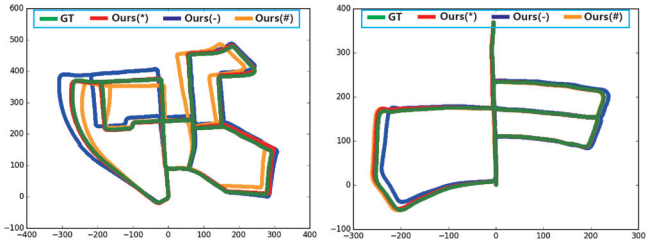

Fig. 7. Our motion tracking trajectories on KITTI training dataset sequences 00 (Left) and 05 (Right), with results by our full system (Ours(*)), the system without depth prior module (Ours(-)) and system without scale correction module (Ours(#)) comparing with the ground truth (GT) trajectory separately.

Point-only versus Point-line. We first evaluate our approach on the performance of tracking features. We implemented a baseline version of our approach using point-only feature, and compared it with our full system. As shown in Fig. 6, our system achieves less drift motion tracking with higher accuracy while the point-only approach has obvious drifts. We conduct a quantitative evaluation on the whole training dateset and summarize the average rotation and translation errors obtained by dividing the segment of each trajectory by 100m, 200m, ..., 800m as done by DVL-SLAM [5], [6]. As shown in Table I, using both the point and line features achieves significantly lower translation error (0.94%) than the point-only baseline (2.16%).

Evaluation on Depth Priors and Scale Correction. We also perform evaluation on the depth priors module (Section II-B) and scale correction module (Section II-D) by comparing the accuracy with our full system. As shown in Table I, we build systems without depth priors module (Ours(-)) or scale correction module (Ours(#)) separately, can

compare with our full system (Ours(*)) by the translation error on the whole KITTI training dataset. The comparison results show that the depth priors module reduces the translation error to a great extent (from 3.64% down to 0.94%). The scale correction module also plays an important role to largely reduce the translation error, with about 50% tanslation error reduced (from 1.95% down to 0.94%). We can conclude that the depth prior module and scale correction module (especially the depth prior module) can largely reduce the pose estimation error in our system. There are several representative motion tracking trajectories by our system with different modules on some KITTI training dataset sequences shown in Fig. 7. Please see more details result in our video attachment.

Comparison with Existing Methods. We compare our approach with DEMO [2], DVL-SLAM [5], [6] and LIMO [3], all of which also have an optimization back-end like ours. Other approaches such as V-LOAM [1] are only visual odometry systems without optimization back-end , and are thus not chosen for comparison. Note that LIMO [3] needs extra semantic label information to identify moving objects such as cars and people. To obtain such semantic label information, it often needs huge computational resources such as Nvidia TitanX Pascal. In contrast, our approach, DEMO and DVL-SLAM do not require such semantic label information and can efficiently run in a CPU-only platform. Our evaluation thus focuses on the comparison between our method, DEMO, and DVL-SLAM. The average translation errors by the compared methods are shown in Table I. Since the semantic label data required by LIMO is available for sequences 00,01,04, we show the results by LIMO for these sequences only.

TABLE I

TRANSLATION ERROR [%] ON THE WHOLE KITTI TRAINING DATASET

Seq.	DEMO	DVL-SLAM	LIMO	Ours (*)	Ours (-)	Ours (#)	Point-Only
0	1.05	**0.93**	1.12	0.99	3.23	4.14	2.81
1	1.87	1.47	**0.91**	1.87	9.40	5.03	3.21
2	**0.93**	1.11	-	1.38	4.96	3.81	2.10
3	0.99	0.92	-	**0.65**	1.06	0.67	1.67
4	1.23	0.67	0.53	**0.42**	0.89	0.66	1.78
5	1.04	0.82	-	**0.72**	2.94	0.88	1.52
6	0.96	0.92	-	**0.61**	3.28	0.85	1.30
7	1.16	1.26	-	**0.56**	3.53	0.74	2.07
8	**1.24**	1.32	-	1.27	3.71	1.83	3.61
9	1.17	**0.66**	-	1.06	4.41	1.82	1.73
10	1.14	**0.70**	-	0.83	2.63	0.99	1.92
avg	1.16	0.98	0.93	0.94	3.64	1.95	2.16

In the whole KITTI training dataset, we achieve the average translation error 0.94% and rotation error $0.0036\frac{deg}{m}$, which are lower than DEMO (translation error 1.14%, rotation error $0.0049\frac{deg}{m}$ in the paper) and DVL-SLAM (translation error 0.98%, rotation error is about $0.0040\frac{deg}{m}$ in the paper). According to the original LIMO paper [3], their reported average translation error is 0.93% and rotation error is $0.0026\frac{deg}{m}$. Though our translation error is slightly higher than LIMO's, we achieve lower translation error in sequences 00 and 04 as shown in Table I. It shows that our approach as a purely geometric approach can achieve comparable (and sometimes even better) accuracy to LIMO, which requires extra semantic label information.

We perform error analysis according to the path length and vehicle speed as did in DVL-SLAM, the comparison curves between DEMO, DVL-SLAM, LIMO and ours is shown in Fig. 8. When the vehicle speed is faster, our system leads to the lower accuracy. This explains why our method performs relatively worse on sequence 01, which is a very challenging case with very fast vehicle speed. Please see more detail results of Ours' method on the whole KITTI training dataset in the video attachment.

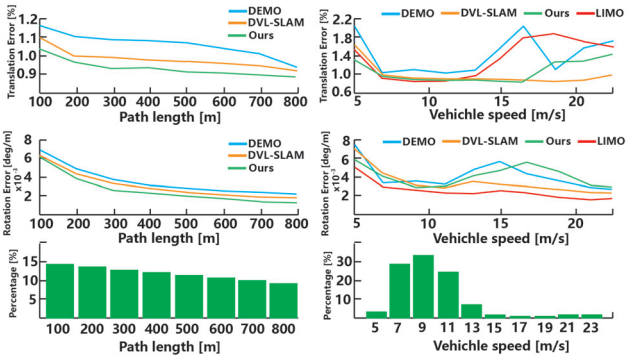

Fig. 8. The error analysis performed on the KITTI training dataset between DEMO, DVL-SLAM, LIMO and Ours. Note that LIMO only provided error curves according to vehicle speed (Right), so we only collect DEMO, DVL-SLAM and Ours error curves in the path length analysis (Left).

Time and Complexity. The feature extraction costs about 80ms, and the scale correction optimization takes about 30ms. The most time consuming part lies in the ICP alignment step, which costs about 300ms each time. Since the ICP alignment is only performed between keyframes, and one keyframe is selected for every 3-4 frames, in average our system achieves about 5Hz processing rate in CPU-only platform, the same as LIMO which is achieved with the essential GPU platform.

IV. CONCLUSION

In this paper we presented an accurate and efficient lidar-monocular visual odometry approach using both the point and line features. We show that our approach is more accurate than the state-of-the-art purely geometric techniques, and achieve comparable accuracy with systems using extra semantic information such as LIMO. One of the drawbacks of our approach lies in the tracking robustness. When the camera moves in large rotations or in fast speed, our tracking system might get failed, leading to less accurate motion estimation. Another drawback of our approach lies in the moving objects, which is beyond the power of purely geometric methods like ours. In the future we would like to incorporate semantic information as LIMO did for a more accurate lidar-visual odometry approach.

REFERENCES

[1] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May,2015*, 2015, pp. 2174–2181. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139486

[2] J. Zhang, M. Kaess, and S. Singh, "A real-time method for depth enhanced visual odometry," *Auton. Robots*, vol. 41, no. 1, pp. 31–43, 2017. [Online]. Available: https://doi.org/10.1007/s10514-015-9525-1

[3] J. Gräter, A. Wilczynski, and M. Lauer, "LIMO: lidar-monocular visual odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, 2018, pp. 7872–7879. [Online]. Available: https://doi.org/10.1109/IROS.2018.8594394

[4] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. [Online]. Available: https://doi.org/10.1109/TRO.2017.2705103

[5] Y.-S. Shin, Y. S. Park, and A. Kim, "Dvl-slam: sparse depth enhanced direct visual-lidar slam," *Autonomous Robots*, Aug 2019. [Online]. Available: https://doi.org/10.1007/s10514-019-09881-0

[6] Y. Shin, Y. S. Park, and A. Kim, "Direct visual SLAM using sparse depth for camera-lidar system," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8461102

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018. [Online]. Available: https://doi.org/10.1109/TPAMI.2017.2658577

[8] A. Bartoli and P. F. Sturm, "The 3d line motion matrix and alignment of line reconstructions," *International Journal of Computer Vision*, vol. 57, no. 3, pp. 159–178, 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013092.07433.82

[9] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[10] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo SLAM," *IEEE Trans. Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015. [Online]. Available: https://doi.org/10.1109/TRO.2015.2489498

[11] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual SLAM with point and line features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, 2017, pp. 1775–1782. [Online]. Available: https://doi.org/10.1109/IROS.2017.8205991

[12] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular SLAM with straight lines," in *Proceedings of the British Machine Vision Conference 2006, Edinburgh, UK, September 4-7, 2006*, 2006, pp. 17–26. [Online]. Available: https://doi.org/10.5244/C.20.3

[13] A. P. Gee and W. W. Mayol-Cuevas, "Real-time model-based SLAM using line segments," in *Advances in Visual Computing, Second International Symposium, ISVC 2006 Lake Tahoe, NV, USA, November 6-8, 2006. Proceedings, Part II*, 2006, pp. 354–363. [Online]. Available: https://doi.org/10.1007/11919629_37

[14] G. Klein and D. W. Murray, "Improving the agility of keyframe-based SLAM," in *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II*, 2008, pp. 802–815. [Online]. Available: https://doi.org/10.1007/978-3-540-88688-4_59

[15] Y. Zhao and P. A. Vela, "Good line cutting: Towards accurate pose tracking of line-assisted VO/VSLAM," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, 2018, pp. 527–543. [Online]. Available: https://doi.org/10.1007/978-3-030-01216-8_32

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[18] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, 2010. [Online]. Available: https://doi.org/10.1109/TPAMI.2008.300

[19] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *J. Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013. [Online]. Available: https://doi.org/10.1016/j.jvcir.2013.05.006

[20] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense SLAM," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, 2015, pp. 4465–4472. [Online]. Available: https://doi.org/10.1109/IROS.2015.7354011

[21] W. M. Häußler, "A local convergence analysis for the gauss-newton and levenberg-morrison-marquardt algorithms," *Computing*, vol. 31, no. 3, pp. 231–244, 1983. [Online]. Available: https://doi.org/10.1007/BF02263433

[22] A. Vakhitov, J. Funke, and F. Moreno-Noguer, "Accurate and linear time pose estimation from points and lines," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, 2016, pp. 583–599. [Online]. Available: https://doi.org/10.1007/978-3-319-46478-7_36

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011, pp. 3607–3613. [Online]. Available: https://doi.org/10.1109/ICRA.2011.5979949

[24] H. Strasdat, "Local accuracy and global consistency for efficient slam," 2012.

[25] P. Biber and W. Straßer, "The normal distributions transform: a new approach to laser scan matching," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, USA, October 27 - November 1, 2003*, 2003, pp. 2743–2748. [Online]. Available: https://doi.org/10.1109/IROS.2003.1249285

[26] P. org., "Pcl library," http://pointclouds.org/.