Monocular Camera Localization in 3D LiDAR Maps

Yunxiang Lu, Keyue Zhang

TUM School of Computation, Information and Technology - Technische Universität München

Abstract

This work aims to localize the camera in a given 3D LiDAR Map. In contrast to common methods for visual localization that use maps acquired with cameras, our approach tracks the pose of camera with respect to a given 3D LiDAR map. We apply a visual odometry system based on local bundle adjustment to reconstruct a sparse set of 3D local landmarks from image stream. These reconstructed points are continuously matched against the LiDAR Map based on their geometry features, which guarantees the robustness under change of photometric appearance of the environment. We use an ICP scheme to optimize the alignment iteratively and track the camera pose in an online fashion. Our work can eliminate the accumulated drift effectively over long trajectories.

1 Introduction

Accurate localization in a given map is essential for vision-based navigation. For example, accurate information about pose in the environment enables autonomous mobile robots to plan a path to a given goal location. In outdoor situations, GPS can provide accurate position estimates at a global scale, while it suffers from substantial errors due to multipath effects in urban canyons. In indoor situations, since GPS suffers from unaccurate estimations because of complex environment in a small scale, the most common approach is to match the sensor data against a previously acquired map. Many existing methods use the same sensor type like LiDARs or cameras for mapping and localization. LiDARs provide accurate range measurements, but are typically expensive and heavy. Cameras are low-cost, lightweight, and widely available, but do not directly provide range information. In order to exploit the advantages of both sensors, our method uses a LiDAR for mapping and a camera for localization, which allows us to localize the mobile robots in a given 3D LiDAR Map without being equipped with a LiDAR.

The main idea of our method is that we indirectly localize the camera by matching geometry. Compared to common methods that consider matching photometry in form of feature descriptors or intensity values, our method tends to be more robust because the geometry features of the environment tends to be more stable than photometric appearance. In real cases, the acquisition of a map and the time of localization are likely to happen at separate times. The same place might look different due to varying illumination while its geometric structure remains the same. Also, changes in geometry always leads to the changes of photometric appearance.

Our method employs a visual odometry based on local bundle adjustment to reconstruct the camera poses relative to a sparse set of 3D local landmarks. After local reconstruction, we apply the alignment process that aligns this point set with the given LiDAR Map by matching geometry to estimate the 7-DoF similarity transformation from the local reconstruction to LiDAR Map. We formulate the estimation problem as non-linear least squares error minimization and solve it within an iterative closest point (ICP) mechanism.

Our work can be applied on both stereo and monocular cameras and achieves significant improvement on Euroc dataset [3] compared to original visual odometry. During the process, the accumulated errors and especially the drift in scale under monocular situations can be eliminated effectively. Besides that, some attempts have been made on the KITTI dataset [8] using stereo cameras and some results worth analysing are obtained.

2 Related Work

Our method is mainly based on the work of Caselitz, Tim, et al. [4], which is related to approaches developed in the computer vision and robotics community in the last decades. Although the cameras and LiDAR sensors have both successfully been used for solving the Simultaneous Localization and Mapping (SLAM) problem [12, 15], the integration of both are mostly done in back-end of the SLAM process, rather than matching their data directly.

The common visual SLAM [7] refers to the problem of using images as only source of information and typically target real-time performance [6, 18], while approaches using LiDARs typically apply scan matching techniques based on variants of the Iterative Closest Point (ICP) algorithm [2, 16] for estimating incremental movements. Compared to visual SLAM, visual odometry [14, 17] is only concerned with the incremental camera motion and uses local bundle adjustment to reduce drift.

Most research concerning visual localization [5, 11] has focused on matching photometric characteristics of the environment by comparing image feature descriptors like SIFT [9] or SURF [1] or directly operating on image intensity values. Other methods based on LiDAR sensor [19, 13] combine the geometric and photometric data to perform matching and require GPU hardware support.

In contrast, our method based on mono performs matching based on geometric features only. It is robust against changes of environment's photometric appearance and can avoid GPU computations. Also, comparable results in terms of accuracy and frame rate can be achieved.

3 Method

The objective of our method is to localize the camera in a 3D LiDAR map. The inputs are an image stream and a point cloud map, the output is a 6-DoF camera pose estimate at frame rate. Figure 1 shows the pipeline of our visual odometry. The visual odometry is built based on the components from the exercise part of the course that uses local bundle adjustment to reconstruct camera poses and a sparse set of 3D points from image features. After local reconstruction, we indirectly localize the camera by aligning the reconstructed points with LiDAR Map. In order to perform the first alignment step, a coarse estimate for the transformation between the initial local reconstruction and LiDAR map is required.

The Alignment step is performed whenever the local reconstruction is updated, namely a new keyframe is selected and added to the local mapping. The Alignment is consisted of three parts: data association that finds the correspondence set C_k between local reconstruction and map, alignment optimization that estimate the similarity transformation $\mathbf{S}^*$ from local reconstruction to map, update part that apply the $\mathbf{S}^*$ to refine our local landmarks and camera poses. By continuously performing the alignment process, the drift accumulated by the visual odometry can be eliminated.

Since we use a monocular camera, drift occurs in translation, rotation and also scale. So we use a 7-DoF similarity transformation $\mathbf{S}$ to realize the alignment besides the 6-DoF rigid-body transformations $\mathbf{T}$:

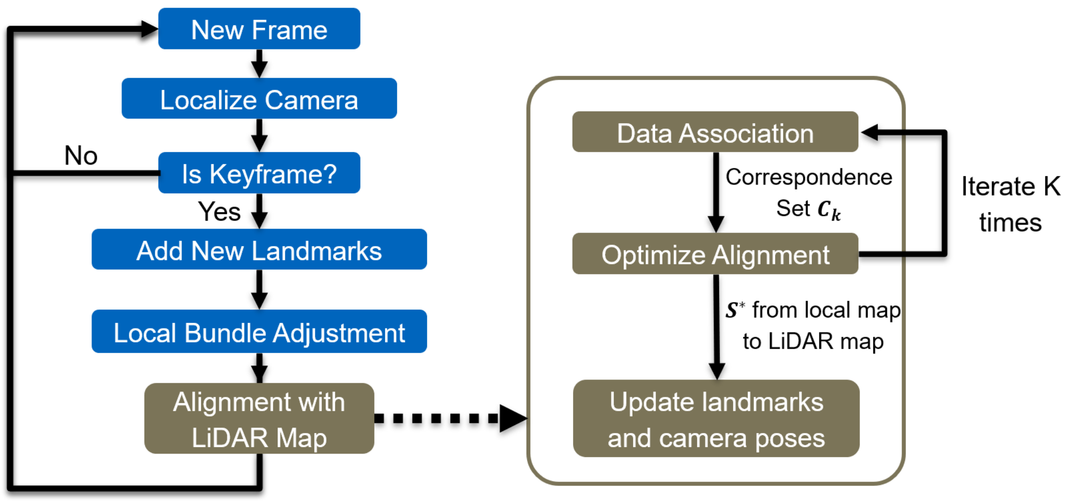

Figure 1: Pipeline of Visual Odometry

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in SE(3), \quad \mathbf{S} = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in Sim(3) \tag{1}$$

with $s \in \mathbb{R}^+$, $\mathbf{R} \in SO(3)$, $\mathbf{t} \in \mathbb{R}^3$. A point $\mathbf{p} \in \mathbb{R}^3$ is denoted $\tilde{\mathbf{p}} = (\mathbf{p}\ 1)^T$ in homogeneous coordinates. To simplify notation we define $\xi(\tilde{\mathbf{p}}) := \mathbf{p}$.

3.1 Local Reconstruction

The visual odometry system reconstructs keyframe poses and 3D points by solving a local bundle adjustment problem. Unlike original paper that uses a covisibility graph to describe the covisible image features between keyframes, our method only considers a maximum number of consequent keyframes in a sliding window during the reconstruction and uses Ceres to solve the bundle adjustment problem following the exercise part.

Since all the camera parameters are given by the dataset, the intrinsic parameters of the camera are frozen during the optimization. In order to conclude the distortion characteristic of the camera, we also rewrite the camera class and use OpenCV functions to realize the undistortion operation.

Following [4], we include the same keyframes and points in our local reconstruction which is defined by the two sets, which ensures the consistency because the local reconstruction is performed right before the alignment process.

$$\mathcal{D} = \{\mathbf{d}_i | \mathbf{d}_i \in \mathbb{R}^3, i = 1, ..., D\}, \tag{2}$$

$$\mathcal{T} = \{\mathbf{T}_i | \mathbf{T}_i \in SE(3), i = 1, ..., T\}. \tag{3}$$

3.2 Data Association

In data association part, we aim to search the correspondences between local reconstructed points in $\mathcal{D}$ and points of the LiDAR map

$$\mathcal{M} = \{\mathbf{m}_i | \mathbf{m}_i \in \mathbb{R}^3, i = 1, ..., M\}. \tag{4}$$

We will perform this within an ICP scheme, namely update the data associations based on the current similarity transformation estimate over $k = 1, ..., K$ iterations. For each local reconstructed point, we search its nearest neighbor in the map. If the distance is smaller than threshold τ_k, this pair is added to the correspondence set

$$C_k = \{(\mathbf{d}_i, \mathbf{m}_j)\ \forall \mathbf{d}_i \in \mathcal{D} \mid \exists \operatorname*{arg\,min}_{\mathbf{m}_j \in \mathcal{M}} ||\mathbf{d}_i - \mathbf{m}_j||_2 < \tau_k\}. \tag{5}$$

Also, we reduce the distance threshold τ_k over the iterations k. We define the linear threshold function

$$\tau_k = -\frac{\tau_{max} - \tau_{min}}{K} k + \tau_{max}, \tag{6}$$

where τ_{max} and τ_{min} are the hyperparameters. As the algorithm converges, point-to-point distance are decreasing and only high-quality associations are remained.

3.3 Local Point Distribution

However, the original correspondence set C_k based on nearest neighbors has a major problem that the set of reconstructed local landmarks typically overlaps only partially with the map because of the LiDAR's limited vertical of view. So points $\mathbf{d}_i$ may have no meaningful correspondence in $\mathcal{M}$ even though they are perfectly reconstructed. These points are likely to be aligned to the boundary points of the LiDAR map and lead to biased transformation estimates. To address this issue, we introduce map's local point distribution to refine the correspondence set C_k.

Since the LiDAR map is given, the computation of local point distribution can be done offline and doesn't effect the real-time performance of the visual odometry.

The point cloud LiDAR map is first voxelized with a resolution Δ depending on its density. Then we perform principle component analysis (PCA) on the covariance matrix $\Sigma(\mathcal{V}) \in \mathbb{R}^{3\times3}$ of the points with the mean $\mu(\mathcal{V}) \in \mathbb{R}^3$ inside each voxel

$$\mathcal{V}_{x,y,z} = \left\{ \mathbf{m}_i \mid \mathbf{m}_i \in \mathcal{M},\ \Delta \begin{pmatrix} x \\ y \\ z \end{pmatrix} \preceq \mathbf{m}_i \prec \Delta \begin{pmatrix} x+1 \\ y+1 \\ z+1 \end{pmatrix} \right\} \tag{7}$$

to determine the main directions along they are distributed. The operator $\preceq$ and $\prec$ are meant component-wise. $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3 \in \mathbb{R}^3$ are the eigenvectors sorted with their corresponding eigenvalues from largest to smallest, which allow us to describe the point distribution $\varepsilon_{x,y,z}$ in a voxel with

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix}^{-1},\ \sigma = (\sqrt{a_{11}}\sqrt{a_{22}}\sqrt{a_{33}})^T,\ N = |\mathcal{V}|, \tag{8}$$

where $\mathbf{R} = (\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3)$ and $\mathbf{t} = \mu(\mathcal{V})$. The transformation T defines the principal components aligned basis of a voxel, the term σ represents the standard deviation of the points along these axes, and N denotes the amount of points inside a voxel.

With these information we are able to determine a refined subset of correspondences $C'_k \subset C_k$. In order to filter out the correspondences associating $\mathbf{d}_i$ that are located in areas not covered by or on the boundaries of the map, we employ the filtering process in three conditions. Firstly, we check if the voxel has sufficient amount of points. Secondly, we verify if the point lies inside a multiple standard deviation along the voxel's principle component axes so that the map boundaries with extremely non-uniform distribution can be avoided. Thirdly, we also allow any neighboring voxel to fulfill these criteria in order to smooth the discretization effects. Our final refined correspondence set can be formulated as

$$C'_k = \left\{ (\mathbf{d}_i, \mathbf{m}_j) \in C_k \mid \Delta \begin{pmatrix} x \\ y \\ z \end{pmatrix} \preceq \mathbf{m}_i \prec \Delta \begin{pmatrix} x+1 \\ y+1 \\ z+1 \end{pmatrix}, \right. \\ \left. \bigvee_{-1 \preceq (x,y,z)^T \preceq +1} \varepsilon_{x,y,z}, N \geq N_{min}, \xi(\mathbf{T}\tilde{\mathbf{d}}_i) \prec N_\sigma \sigma \right\}. \tag{9}$$

where $\bigvee$ means *or* operator used to describe the neighborhood of a voxel.

In practice, we create a lidar map class and use two 3-dimensional vectors to store the information of voxelized LiDAR map. One vector contains the points information of each voxel. We also define a data structure to save the point distribution information of each voxel and store them in another 3-dimensional vector. In pre-processing we first voxelize the map with self-defined resolution Δ and then compute the local point distribution of each voxel. During data association process, for each reconstructed local landmark, the function of the class can search the coordinate of the voxel and then perform the filtering process. If all three conditions are fulfilled and the distance between two points is lower than the threshold τ_k, the pair is added to the refined correspondence set C'_k.

3.4 Alignment Optimization

Given the refined correspondence set C'_k, we are able to estimate a similarity transformation that aligns the local reconstruction with the LiDAR map. Since we perform this within an ICP scheme, we update the set C'_k iteratively. In each iteration k we estimate

$$\mathbf{S}^*_k = \underset{\mathbf{S} \in Sim(3)}{\arg\min}\, F_{Data}(\mathbf{S}, C'_k) \tag{10}$$

by solving a non-linear least squares minimizaztion problem with ceres. The error function is the squared Euclidean distance between corresponding points:

$$F_{Data}(\mathbf{S}, C'_k) = \sum_{C'_k} \rho(\mathbf{e}^T_{Data}\mathbf{e}_{Data}), \tag{11}$$

$$\mathbf{e}_{Data}(\mathbf{S}, \mathbf{d}_i, \mathbf{m}_j) = \xi(\mathbf{S}\tilde{\mathbf{d}}_i) - \mathbf{m}_j. \tag{12}$$

$$\rho(r) = \begin{cases} r^2, & r < r_\Delta \\ 2r_\Delta\,|r| - r^2_\Delta, & \text{otherwise} \end{cases} \tag{13}$$

Follow [4], we use a Huber cost function to be more robust against outliers in the data association, which leads to better convergence properties of the objective function, and set the kernel size $r_\Delta = \tau_{min}$ to retain the quadratic error range over K iterations. After estimating $\mathbf{S}^*_k$ for all iterations, we compute the joint similarity transformation

$$\mathbf{S}^* = \prod_{k=0}^{K-1} \mathbf{S}^*_{K-k}. \tag{14}$$

To align the local reconstruction with the LiDAR map, we use joint similarity transformation $\mathbf{S}^*$ to refine all current local reconstructed points $\mathbf{d}_i$ and keyframe poses $\mathbf{T}_i$ (as defined in 3.1):

$$\mathcal{D}' = \{\mathbf{d}'_i = \xi(\mathbf{S}^*\tilde{\mathbf{d}}_i)\ \forall\ \mathbf{d}_i \in \mathcal{D}\}, \tag{15}$$

$$\mathcal{T}' = \{\mathbf{T}'_i = \mathbf{S}^*\mathbf{T}_i \ \forall \ \mathbf{T}_i \in \mathcal{T}\}. \tag{16}$$

In order to compensate the dimensions of drift better with $\mathbf{S}^*_k$, we also allow estimating the transformation $\mathbf{S}^*_k$ in the reference frame of the current keyframe $\mathbf{T}_c$ rather than in the reference frame of the map. In this way, we transform the refined correspondence set and get

$$C''_k = \left\{(\xi(\mathbf{T}_c\tilde{\mathbf{d}}_i), \xi(\mathbf{T}_c\tilde{\mathbf{m}}_j)) \mid (\mathbf{d}_i, \mathbf{m}_j) \in C'_k\right\}. \tag{17}$$

Also, we need to transform the points $\mathbf{d}_i$ and poses $\mathbf{T}_i$ back into the reference frame of the map before applying the update step:

$$\mathcal{D}' = \{\mathbf{d}'_i = \xi(\mathbf{T}_c^{-1}\mathbf{S}^*\mathbf{T}_c\tilde{\mathbf{d}}_i) \ \forall \ \mathbf{d}_i \in \mathcal{D}\}, \tag{18}$$

$$\mathcal{T}' = \{\mathbf{T}'_i = \mathbf{T}_c^{-1}\mathbf{S}^*\mathbf{T}_c\mathbf{T}_i \ \forall \ \mathbf{T}_i \in \mathcal{T}\}. \tag{19}$$

In practice, we realize two mechanisms of alignment optimization by passing arguments to the program. Since most of our experiments are performed on Euroc dataset [3] that only contains small indoor scenes, these two mechanisms have almost no difference in the final evaluation. Therefore we employ the optimization in the reference frame of the LiDAR map by default to reduce unnecessary transformation operations.

3.5 Implementation Details

We follow the exercise part and [4] to set the hyperparameters. The visual odometry extracts 1500 features per image and keeps the default values for other parameters. The maximum number of keyframes in the local bundle adjustment is 10. The number of iterations used for estimating the alignment is $K = 10$, the parameters for the correspondence distance threshold are $\tau_{max} = 0.5m$ and $\tau_{min} = 0.25m$. The voxel size is $\Delta = 0.25m$, thresholds for the refinement are $N_{min} = 10$ and $N_\sigma = 3$.

Since our method needs a reasonably good initialization, we use the ground truth trajectory to get the initial pose of the first keyframe. In order to obtain the initial scale, we always use the stereo pair to reconstruct the local map at the first keyframe. We save the information of different trajectories in a json file and can thus switch between different scenes and trajectories by passing different scene arguments into the program.

Our program can be applied in both stereo and monocular cases. In stereo situations, we always use stereo image pairs to reconstruct the local points. In monocular situations, we use stereo image pairs for first M keyframes ($M = 1$ by default) and then switch to monocular mechanism. The main difference between stereo and monocular is that when using monocular camera, we use the current and the previous keyframe to triangulate new landmarks.

To improve the stability of alignment, the alignment optimization is only performed when the size of the refined correspondence set bigger than a threshold (100 correspondence pairs by default), which avoids the oscillations at the beginning and end of the trajectory caused by small set of corresponding points.

4 Experiment and Result

In order to evaluate the accuracy of our method, we test it on various sequences of the Euroc dataset and use evo tools to evaluate the results. We set all the hyperparameters following section 3.5 above. All experiments are carried out on an AMD Ryzen 7 5800H CPU without support of GPU hardware.

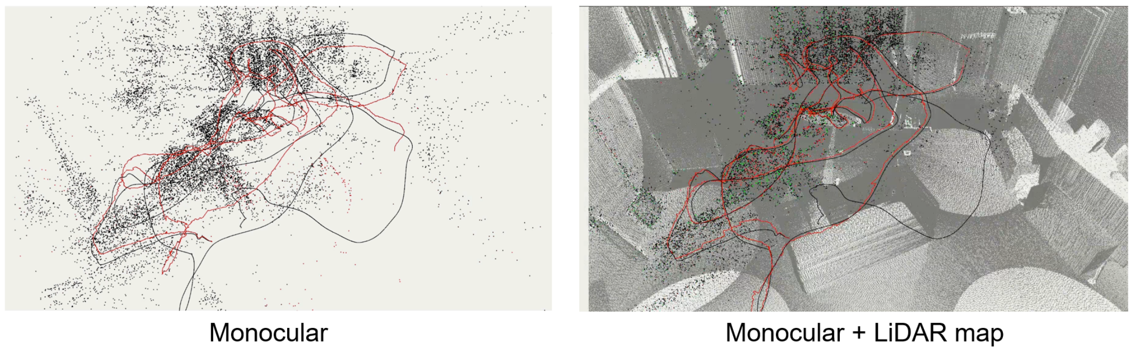

Figure 2: Visualization during the process via Pangolin library. The black trajectory represents the ground truth trajectory and the red trajectory represents the estimated trajectory in real time.

During the process, we use Pangolin library to visualize the estimated trajectory. Like Figure 2 shows, black and red trajectories represent the ground truth and estimated trajectory. Black points are historical local reconstructed landmarks. Red and green points represent all reconstructed points included in the current local bundle adjustment, where the green points indicate all associating local landmarks included in refined correspondence set C_k' that are used to estimate the similarity transformation.

After each alignment process, the estimated similarity transformation $\mathbf{S}^*$ is used to update all current local reconstructed points (red and green points) and the camera poses.

4.1 Qualitative Analysis

Figure 3 provides a visual comparison between the ground truth trajectory and generated trajectory in four setups, namely stereo, stereo with alignment, monocular, monocular with alignment. The first row shows results of our visual odometry in sequence V1-01-easy, using stereo camera alone results in relative large drift in translation, which is well dealt with after we apply the alignment method. At the same time using monocular camera alone results in even larger drift in translation and in addition it causes drift in scale. This situation is also greatly improved after applying our alignment method. The second row shows results in sequence V2-01-easy, the drifting and scale problem are also very well addressed by our method. The results in both sequence V1-01-easy and V2-01-easy show the effectiveness of our alignment method in solving the problem of drift in translation and scale in original visual odometry. One thing to notice is that, the alignment slows down the process of the visual odometry from around 40Hz to around 20Hz. Since the images in Euroc dataset are taken at a rate of 20 frames per second, the visual odometry with alignment currently can still not perfectly meet the requirement of real time operation.

Further experiments have been done in medium and difficult scenes of Euroc dataset and the results are shown in Figure 4 . It is observed that our method doesn't guarantee to find the correct trajectory in these scenes, however good results can still be obtained, such as in V1-02-medium and V2-02-medium. In sequence V1-02-medium the original monocular visual odometry couldn't generate the correct trajectory and failed halfway, but once with our alignment method applied the visual odometry is able to generate trajectory with high accuracy. The same case applies to sequence V2-02-medium using stereo camera. The results again prove the effectiveness of our alignment functionality.

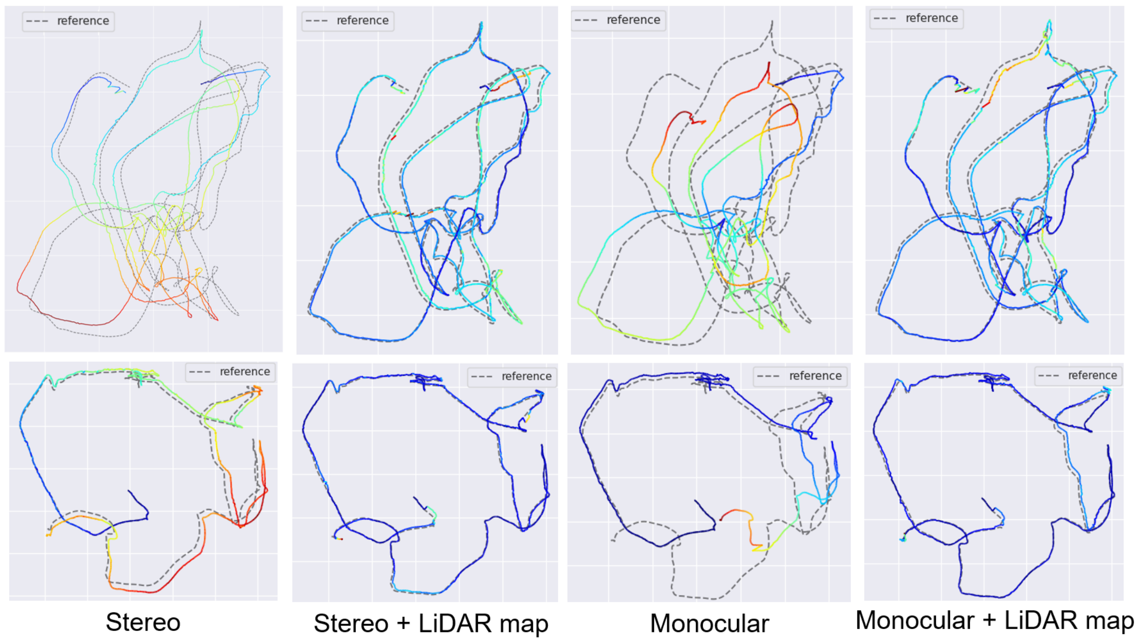

Figure 3: Qualitative results on Euroc dataset. The first row shows the results of sequence V1-01-easy and the second row shows the results of sequence V2-01-easy.

	V1-01	V1-02	V1-03	V2-01	V2-02	V2-03
Stereo	0.263	0.166	0.518	0.126	0.361	-
Stereo + LiDAR Map	**0.054**	**0.048**	**0.117**	**0.038**	**0.044**	-
Monocular	0.362	0.835	-	0.639	-	-
Monocular + LiDAR Map	0.062	0.067	-	0.061	-	-

Table 1: Evaluation results of our method on different sequences of Euroc dataset. We use absolute pose error (APE) w.r.t translational part (in m) as the evaluation metric and choose the best result among 5 tests. V1 and V2 represent two different indoor scenes and 01,02,03 mean easy, medium and difficult. Empty space means that the visual odometry fails at all 5 times.

4.2 Quantitative Analysis

Table 1 shows the quantitative evaluation results of 4 different setups in different scenarios of Euroc dataset. There are two main characteristics we can infer from the table. First, adding the alignment method significantly reduces the error in every situation, no matter it uses stereo or monocular camera, and no matter in which scenario it is. Secondly, in every scenario, the original visual odometry with monocular camera performs worse than that with stereo cameras, while after applying the alignment, both of them can achieve similar high-level performance, which shows the potential of our method to fill in the performance gap between stereo and monocular camera.

Also, the performance of the alignment process is strongly restricted by the original visual odometry. The empty space in Table 1 indicates that though the alignment process has the ability to improve the estimated trajectory, it can hardly reduce the failure rate under some medium and difficult sequences.

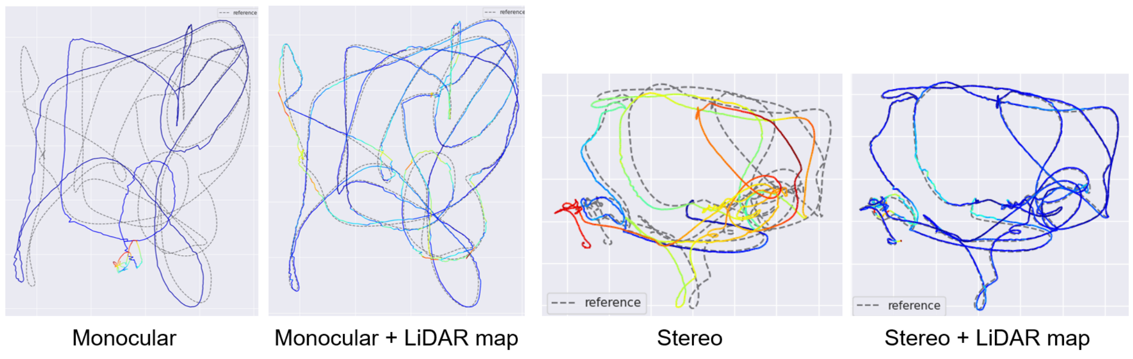

Figure 4: Qualitative results on Euroc dataset. The left 2 results are from the sequence V1-02-medium and the right 2 results are from the sequence V2-02-medium.

Voxel Size Δ	V1-01-easy	V2-01-easy
0.10m	0.063	0.210
0.25m	**0.062**	**0.061**
0.50m	0.61	0.208
1.00m	0.18	0.511

Table 2: Ablation Study 1: Comparison of the results obtained by our method with different **voxel size**. We use absolute pose error (APE) w.r.t translational part (in m) as the evaluation metric and choose the best result among 5 tests.

4.3 Ablation Study

In addition, we did some ablation study to investigate the impact of different hyperparameters on our visual odometry. During the ablation experiments we only adjust the target parameter and keep other hyperparameters fixed by default.

One parameter of interest is the voxel size. Note that when changing the voxel size, some parameters corresponding with voxel size like distance threshold τ should also be adjusted. From the results shown in Table 2, we cannot find particular pattern of how different voxel size choices influence visual odometry performance. However conclusion can be made that the best voxel size is highly scene specific, for euroc V1-01-easy and V2-01-easy dataset, the best voxel size is around 0.25 meter. In addition there is a tradeoff between voxel size and the running time, because the number of voxels increase exponentially as the voxel size decreases.

Another parameter of interest is the frequency of performing alignment and the results are shown in Table 3. In both scenarios, a higher frequency of alignment lead to better outcomes. However, it was also observed that higher-frequency alignment reduces the system's real-time performance. Moreover, when the frequency of alignment is reduced, there is an increased probability of failure to find the correct trajectory in certain scenarios. So when it comes to real world applications, the trade-off between success rate, accuracy and real-time performance need to be taken into consideration.

4.4 KITTI Dataset Attempt

Besides Euroc dataset [3], we also did some attempts on KITTI dataset [8], which indicates some results worth analysing. While Euroc dataset provides a complete 3D point cloud LiDAR map for each scene, KITTI dataset only provides LiDAR data for each frame,

Frequency of Alignment	V1-01-easy	V2-01-easy
No alignment	0.362	0.639
Alignment per 10 kf	0.231	0.275
Alignment per 5 kf	0.106	0.08
Alignment per 1 kf	**0.062**	**0.061**

Table 3: Ablation Study 2: Comparison of the results obtained by our method with different **frequency of applying alignment process**, kf means keyframe. We use absolute pose error (APE) w.r.t translational part (in m) as the evaluation metric and choose the best result among 5 tests.

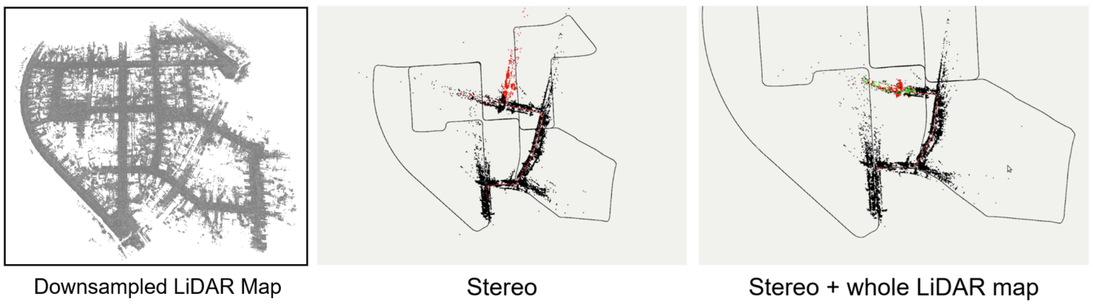

Figure 5: Results of KITTI Dataset. The left one shows the downsampled LiDAR map obtained by Python, the middle one shows the result when using original visual odometry without alignment and the right one shows the result when using visual odometry with given Lidar map.

which means the data is way larger than Euroc dataset.

In order to obtain the complete LiDAR map for one scene, we use Python and Point Cloud Library PCL to merge the point cloud data of all frames and then downsample the map into a resolution of 50 cm (see Figure 5 left). Compared to [4] that uses a LiDAR-based SLAM system to get the ground truth trajectory and builds a map at a resolution of 20 cm, our method lacks consistency in loop closure areas and the accuracy and resolution of our LiDAR map is also lower.

We test our method on sequence 00 of the KITTI odometry dataset with stereo cameras. We employ our downsampled map to track the pose of camera 0 on the vehicle. Since the relative transformation between camera and LiDAR is known by calibration, the initial camera pose can be well inferred from the ground truth. The camera images have a resolution of 1241×376 and are captured at 10 Hz.

The results are shown in the middle and right pictures of the Figure 5. When using original visual odometry, the estimated trajectory has obvious drift in translation and rotation, which is often caused by turns of the car. After using the given LiDAR map, the resulting trajectory shows a small correlation on the scale and translation. However, the deviation between the local reconstruction and the LiDAR map grows with time so that we can not find reliable correspondence set after a few time, like the right picture of Figure 5 showing that only few local landmarks (green points) have corresponding points and the correspondence has great errors. The huge LiDAR map also leads to bad real-time performance (< 5 Hz). To address this issue, a more powerful CPU and a more accurate LiDAR map is essential.

5 Summary

In this work, we present a method to localize the monocular and stereo cameras with respect to a given 3D point cloud LiDAR map. Our method employs visual odometry to track the 6-DoF camera pose and to reconstruct a sparse set of 3D points via bundle adjustment. We align the reconstructed points with the LiDAR map by continuously applying an estimated 7-DoF similarity transformation to indirectly localize the camera. Since the matching mechanism is performed based on time-invariant geometry features, the alignment is robust against illumination changes. In this way, we realize using LiDAR for mapping and cameras for localization, which combines the advantages of both sensors.

Experiments on Euroc dataset show that our method can eliminate the accumulated drift and the drift in scale effectively and can improve the estimated trajectory significantly without any GPU hardware support. However, the performance of the alignment is still strongly restricted by the quality of local reconstruction and the real-time performance also has large improvement space. Further experiments carried out on KITTI dataset indicates that our method has difficulties under large-scale outdoor scenes because of the too large LiDAR map. We suppose to address these issues in the future by improving the visual odometry with more powerful components like ORB-SLAM [10], using a more powerful CPU and applying a more accurate LiDAR map.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: speeded up robust features*, page 404–417. Dec 2005.

[2] Paul J. Besl and Neil D. McKay. Method for registration of 3-d shapes. In *SPIE Proceedings,Sensor Fusion IV: Control Paradigms and Data Structures*, Apr 1992.

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, page 1157–1163, Aug 2016.

[4] Tim Caselitz, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. Monocular camera localization in 3d lidar maps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2016.

[5] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, Nov 2013.

[6] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1052–1067, May 2007.

[7] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, page 55–81, Dec 2014.

[8] 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*, May 2012.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, page 91–110, Oct 2004.

[10] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, page 1147–1163, Sep 2015.

[11] Tayyab Naseer, Michael Ruhnke, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. Robust visual slam across seasons. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Aug 2015.

[12] NüchterAndreas NüchterAndreas, LingemannKai LingemannKai, HertzbergJoachim HertzbergJoachim, and SurmannHartmut SurmannHartmut. 6d slam3d mapping outdoor environments. Jul 2007.

[13] Geoffrey Pascoe, William Maddern, and Paul Newman. Direct visual localisation and calibration for road vehicles in changing city environments. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Nov 2015.

[14] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics amp; Automation Magazine*, page 80–92, Nov 2011.

[15] S. Se, D.G. Lowe, and J.J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, May 2005.

[16] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: Science and Systems V*, Jan 2016.

[17] Hauke Strasdat, J.M.M. Montiel, and AndrewJ. Davison. *Scale Drift-Aware Large Scale Monocular SLAM*. Dec 2010.

[18] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. *Bundle Adjustment - A Modern Synthesis*, page 298–372. Dec 1999.

[19] Ryan W. Wolcott and Ryan M. Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2014.