CoFi: Coarse-to-Fine ICP for LiDAR Localization in an Efficient Long-lasting Point Cloud Map

Yecheng Lyu, Xinming Huang and Ziming Zhang

Abstract—LiDAR odometry and localization has attracted increasing research interest in recent years. In the existing works, iterative closest point (ICP) is widely used since it is precise and efficient. Due to its non-convexity and its local iterative strategy, however, ICP-based method easily falls into local optima, which in turn calls for a precise initialization. In this paper, we propose CoFi, a Coarse-to-Fine ICP algorithm for LiDAR localization. Specifically, the proposed algorithm downsamples the input point sets under multiple voxel resolution, and gradually refines the transformation from the coarse point sets to the fine-grained point sets. In addition, we propose a map based LiDAR localization algorithm that extracts semantic feature points from the LiDAR frames and apply CoFi to estimate the pose on an efficient point cloud map. With the help of the Cylinder3D algorithm for LiDAR scan semantic segmentation, the proposed CoFi localization algorithm demonstrates the state-of-the-art performance on the KITTI odometry benchmark, with significant improvement over the literature.

I. INTRODUCTION

Vehicle localization is one of the fundamental problems in automated driving, which supports many modules in the vehicle perception, planning and control systems [1], [2], [3], [4]. Traditional vehicle localization solutions utilize the Global Navigation Satellite System (GNSS) and Inertial Navigation System (INS), however they are limited in precision and frequency. In addition, GNSS based systems fail to localize in urban canyons due to the non-line-of-sight (NLOS) receptions and multi-path effects [5], [6]. Some existing works employ the visual odometry to localize the vehicle where GNSS signal is not available, however they are highly subject to the illumination and the error accumulates rapidly as the vehicle goes. In recent years, LiDAR odometry has attracted increasing research interest as it acquires precise point scanning and it is invariant to the light conditions, which makes it works during night driving where visual odometry methods usually fail.

Iterative closest point (ICP) [7], [8], [9], [10] is a widely applied point cloud registration algorithm that iteratively searches the closest points as correspondences and minimizes the euclidean distance between the paired points. This algorithm is precise and efficient, however, due to its non-convexity and its local iterative strategy, ICP-based method easily falls into local optima, which in turn calls for a precise initialization. To this end, we look for a ICP strategy that gradually refines the transform initialization without falling in to the local optima.

Motivation. Voxelization is a commonly used filter in point cloud processing. In point cloud voxelization, points are sampled by 3D grid cells (voxels) and the center of the points in each voxel is extracted. By setting different voxel size, we can compress a point cloud to certain rate and make it near evenly distributed in the Cartesian coordinate. During experiment, we observe that point cloud voxelized in low resolution has fewer local optima, which in turn to have a larger range of convergence.

This observation inspires us to explore the strategy that optimizes the point cloud transform estimation using subsets voxelized under multiple sizes. Since the coarse subsets have a larger range of convergence while the fine-grained subsets have a better global-optima, there exists an intuitive choice to refine the estimated transform matrix gradually from its coarse subsets to fine-grained subsets.

Approach. In this paper, we propose CoFi, a coarse-to-fine ICP algorithm that refines the estimated transformation between two point clouds under multiple 3D resolutions. This strategy plays a key role in local optima avoiding in the point cloud registration. In addition, we utilize this algorithm to estimate the transformation between two LiDAR point clouds, and between a LiDAR point cloud and a prebuilt point cloud map, in which a map-based LiDAR localization system is implemented.

Our localization system includes two major modules: a point cloud map generation module and a map-based LiDAR localization module. The map generation module selects several frames from a LiDAR scan sequence, filters out the non-long-lasting objects from the LiDAR scans, projects them to the same coordinate frame, and voxelize them to an efficient long-lasting point cloud map. The map-based localization...
performs LiDAR odometry for initial pose estimation and
map-based LiDAR localization for pose refinement.

Performance review. To demonstrate how well our approach
works, we summarize several state-of-the-art performance on
a benchmark data set, KITTI odometry, in Figure 2 and
compare ours with these results under the same metrics.
Clearly our results are significantly better than all the others
with large margins.

![Figure 2](image)

Fig. 2. State-of-the-art comparison on KITTI odometry set.

Contributions. In summary, our key contributions in this
paper are as follows:

- We propose CoFi, a coarse-to-fine ICP algorithm that
  updates the transform estimation from low-resolution
  subsets to high-resolution subsets, which minimizes the
  local optima during ICP.
- We propose a point cloud map generation method that
  creates an memory-efficient long-lasting map from a
  LiDAR sequence.
- Based on the point cloud maps, we propose a novel
  LiDAR localization that estimates the vehicle pose in
  3D.
- We demonstrate the state-of-the-art performance on the
  KITTI odometry dataset, with significant improvement
  over the literature for vehicle odometry estimation and
  localization.

II. RELATED WORK

Map-based localization. Map-based localization is a hot
topic in the research field. In the recent decades, several
kinds of map have been proposed to provide a strong prior
knowledge to the localization algorithms.

Open street map (OSM) [11] is an open-sourced, world-
wide 2D map that contains the shape and position of
buildings, roads and some other infrastructures. This map
labels the objects using polygons with the GNSS position
of the vertices. OSM is widely used in [12], [13], [14],
[15], [16], [17], [18], [19]. However, OSM does not have
3D features, which limits its use in 3D localization. In [20],
road segments in the OSM are parametrically represented as
arc segments and added to a State-Space Model and fused
with LiDAR odometry to perform widely-converging global
vehicle localization.

Satellite image [21], [22], [23] is another world-wide 2D
map for vehicle localization. Similar to the OpenStreetMap,
satellite images do not contain 3D features, which limits its
use in 3D localization. In [22], semantic segmentation is
applied to the satellite images and compared with the LiDAR
scannings to perform global vehicle localization.

Point cloud map [24], [25], [26], [27], [28], [29], [30],
[31], [32] is a novel map source that provides dense and
accurate 3D reference points. Compared with the OSM
and satellite images, a point cloud map well supports the
3D localization, which makes it popular in modern vehicle
localization algorithms. In [33] and [34], planar features
are extracted from camera frames and compared with a
point cloud map for visual localization. In [35], radar scans
are matched to a point cloud map for vehicle localization.
In [36], [37], [38], LiDAR frames are utilized to search
the right pose on the point cloud map to perform vehicle
localization. However, point cloud maps are usually large
in size and part of the points may change over the time.
Several algorithm has been proposed to compress the point
cloud map by compression [39], [40] or parametric representation
[41], [42], [43]. To correct the changes in the point cloud
map, several methods have been proposed to update the map
during driving [36], [44].

In this paper, our proposed method generates a lighted-
weighted map with long-lasting objects only, which over-
comes both drawbacks in traditional point cloud maps.

LiDAR odometry. LiDAR odometry [45], [46] estimates the
transformation from LiDAR frames to frames and multiply
them to get the poses. In map-based LiDAR localization,
LiDAR odometry is usually applied as an initial guess on the
map. According to the kernels, LiDAR odometry methods
follow three major branches: ICP-based, NDT-based, and
network-based. ICP-based methods [47], [7], [8], [48], [9]
utilize the iterative closest point (ICP) algorithm to search for
the point correspondences between two point clouds. ICP-
based methods are efficient however require strong initial
pose. In DCP[49] and DGR [10], a vanilla-ICP method is
applied to refine the registration between point clouds. NDT-
based methods utilize the normal distributions transform
(NDT) algorithm to register a point cloud to another. Com-
pared with ICP-based methods, NDT-based methods have a
larger range of convergence but take longer time. Several
works have been proposed to extract keep points from input
point clouds to speed up the NDT process. LOAM [50],
LEGO-LOAM [51] and LIMO [52] are good examples in
this branch. Network-based methods [10], [53], [54], [55]
take advantage of the fast-growing convolutional neural net-
works (CNNs) to estimate the transformation from one point
cloud to another. Thanks to the power of general-purpose
computing on graphics processing units (GP-GPU), network-
based methods run fast odometry estimation over large
point clouds. However, network-based approaches sometimes
perform not so well when the pattern of input point clouds
are much different from the training samples.

In this paper, we follow the ICP approach and propose
Pipeline: map generation
Semantic Segmentation
LiDAR sequence
Long-lasting points
Feature point extraction
Feature point
Map
poses
Voxelization
Voxelization Projection

In this module, points from each LiDAR frame are first separated based on the semantic categories, where the dynamic objects (vehicles, pedestrians, cyclists, temporal infrastructures and leaves) are filtered out and long-lasting objects (buildings, poles, traffic signs, roads, terrain) are kept for further processing. Secondly, a feature point extraction submodule is applied to the long-lasting points, where high-quality feature points are extracted for the anti-noised feature-point-map localization. At last, both long-lasting points and feature points in each LiDAR frame are projected to the same coordinate frame and voxelized to a long-lasting map and a feature point map.

Semantic Segmentation Submodule. A semantic segmentation submodule is applied to each selected LiDAR frame so that every point is classified for further processing. This submodule is applied to both the map generation module and localization module. In this paper, we adopt the category set from the SemanticKITTI [61] dataset. In implementation, we use ground truth semantic labels for the map generation module while using the semantic segmentation results from Cylinder3D [62] for map-based localization. This is because we assume that extensive effort has been done to achieve accurate semantic segmentation during map generation, while during real-time driving only on-board modules are used.

Feature Point Extraction Submodule. A feature point extraction submodule is used in both the point cloud map generation module and the map-based localization module. By extracting the high-quality feature points from a point cloud, valuable point correspondences can be collected in the iterative closest point (ICP) submodule during localization.

In this paper, we utilize manually defined features for high-valued and fast feature point extraction. In general, we search for vertical-line features because they are quite helpful for horizontal localization while the ego-car moves horizontally in majority. Specifically, we search for two kinds of objects: (1) stick-like objects including poles, traffic lights and traffic signs, and (2) vertical edges of buildings. For the former kind of objects, we easily collect all the points belonging to those semantic categories, while for the latter kind of objects, we propose a novel algorithm that efficiently extracts edge points from building points in each LiDAR scan. The proposed algorithm is described in Algorithm 1.

Algorithm 1: Building edge points extraction

Input: Building Points $p_{ij} \in P_{in}$, $0 \leq i < ROW_{max}$, $0 \leq j < COL_{max}$, angle threshold $thr$

Output: Edge Points $P_{out}$

$P_{out} \leftarrow \emptyset$

For all $p_{ij} \in P_{in}$ do

(search around for available points)

if $Neighbour(p_{ij}) \in P_{in}$ do

if Angle($p_{ij-1}, p_{ij}, p_{ij+1}) < thr$ do

$P_{out} \leftarrow P_{out} \cup \{p_{ij}\}$

end

end

end

Projection Submodule. The projection submodule projects all the selected points to the same coordinate so that a map can be generated. In this paper, we project the LiDAR points in the experiments based on the ground truth poses from the KITTI odometry dataset [63] as we assume we can get precise poses before map generation.

Voxelization Submodule. Voxelization submodule is applied to the map generation module to eliminate the overlapped points from multiple LiDAR scans. Thanks to the voxelization module, the feature map and the long-lasting map are compressed for fast loading during localization.

Ablation study. To evaluate the feasibility, efficiency and robustness of the map generation module, we test the module...
TABLE I
COMPARISON BETWEEN DIFFERENT LiDAR FRAME SAMPLE STRATEGIES.

| Distance  | 5m    | 15m   | 30m   | 100m  |
|-----------|-------|-------|-------|-------|
| Feature extraction | w/ | w/o | w/ | w/ |
| Frames (%) | 18.79 | 6.71 | 3.36 | 0.96 |
| Feature points (%) | 0.04 | 0.15 | 0.02 | 0.09 |
| Long-lasting points (%) | 0.79 | 0.48 | 0.30 | 0.10 |
| $t_{rel}$ (%/100m) | 0.28 | 0.30 | 0.52 | 0.52 |
| $r_{rel}$ (%/100m) | 0.15 | 0.15 | 0.19 | 0.19 |

$t_{rel}$: Average translational RMSE (%) on length of 100m-800m.
$r_{rel}$: Average rotational RMSE (◦/100m) on length of 100m-800m.

on KITTI [63] odometry dataset with help of the localization module in Section IV. Specifically, we arrange to compare the statistics (percentage of used LiDAR frames, used points in the feature point map, used points in the long-lasting map) and odometry metrics ($t_{rel}$ for translation error and $r_{rel}$ for rotation error) under two variables: (1) minimum distance between the collected frames, where we skip any LiDAR frame that within the given distance of used LiDAR frame, and (2) with or without edge point extraction during feature point extraction. Table I presents the statistics and odometry metrics under different variable settings. It is obvious that our method outperforms LOAM, one of the best LiDAR odometry methods, using as low as 3.36% LiDAR frames, and needs as low as 0.01% points for the feature point map and 0.30% points for the long-lasting map. In comparison, the existing point cloud compression methods [64], [40], [65], [66], [67], [68], [69], [70] require at least 2% points. As the distance between selected LiDAR frames enlarges, the map generation module uses fewer LiDAR frames to build the map, resulting in smaller map sizes and worse performance in localization. Edge points extraction in map generation has little impact on localization accuracy, but it significantly lowers down the feature point map size. In addition, this submodule plays a key role in the map-based localization module.

In Figure 4 we compare the feature point map with a long-lasting map and a point cloud using all the raw points in the LiDAR scan. It can be seen that the feature point map focuses on sharp features and contains much fewer points than the other two maps. In the experiments, we take a 5 m rule to select the LiDAR frames for a fair comparison with existing works.

In Figure 5 we compare the feature point map collected with 15 m per frame with the feature point map in 5 m per frame, and its variant without edge point extraction.

IV. MAP BASED LOCALIZATION

In this section, a map based localization method is introduced that estimates the current vehicle pose by comparing the current LiDAR frame and the point cloud maps in Section III. Figure 6 illustrates the pipeline of the localization module. The inputs of this module are estimated pose and speed from previous localization, the current and past LiDAR frames, the corresponding feature point map and the corresponding long-lasting map. After four steps of localization, the localization module outputs the final pose for the current LiDAR frame.

Local map extraction. Each time a LiDAR frame comes, local maps are generated from the feature point map and the long-lasting map to speed up the ICP submodule. In this paper, the local maps are extracted in a determined radius from the location of the last LiDAR frame.

First localization. The first localization step is based on the pose of the last LiDAR frame and the vehicle velocity. This localization is achieved by adding the estimated velocity in the previous frames to the current last frame so that we can estimate the vehicle pose in the current frame. In implementation, we average the velocity in the previous four frames for vehicle position, while leave the orientation same to the last frame, as we assume there is no sharp turn during driving.

Coarse-to-Fine ICP. The second to fourth localization steps utilize the iterative closest point (ICP) algorithm to estimate the current pose. Different from the traditional ICP based methods that search the correspondences from raw point cloud pairs, in this paper, we apply the ICP algorithm from coarse to dense upon point cloud pairs in different voxel resolution. Following this coarse-to-dense strategy, the localization gradually refines the pose estimation, which makes the localization algorithm robust to the initial pose and ambiguous road segments.

Fig. 4. Example of a feature point map and a long-lasting map.

Fig. 5. Comparison between feature maps generated using different sample strategies.
ICP submodule Implementation. In implementation, we adopt the Open3D package to process the point cloud and execute the local map extraction, voxelization and ICP. For the second localization step, the current LiDAR frame is taken as the source point cloud and the previous LiDAR frame is taken as the target point cloud, with long-lasting object extraction in Section III applied to both point clouds. For the third localization step, the submodule takes the current LiDAR frame with long-lasting object extraction and feature point extraction as the source point cloud and the local feature point map as the target map. For the fourth localization step, the submodule takes the current LiDAR frame with long-lasting object extraction as the source point cloud and the local long-lasting map as the target source map.

In default, the voxelization resolutions used in the submodule are 5.0 m, 1.0 m and 0.2 m. To speed up the ICP submodule, we use 1.0 m resolution for the fourth localization step.

Ablation study. In the ablation study we explore the impact of each localization step to the localization accuracy. In Table II different localization step strategies are tested on all the training sequences in the KITTI odometry dataset.

TABLE II

| Localization steps | 1+2 | 3 | 4 | 3+4 | 1+2 | 1+2+4 | 1+2+3+4 |
|--------------------|-----|---|---|-----|-----|-------|---------|
| $r_{rel} (\%/100m)$ | 1.50 | 23.51 | 74.59 | 17.3 | 0.95 | 0.92 | 0.3 |
| $r_{rel} (°/100m)$ | 0.61 | 6.28 | 23.89 | 3.75 | 0.63 | 0.22 | 0.15 |

V. EXPERIMENTS

Dataset. To evaluate the performance of the proposed point cloud map generation and map based localization algorithms, we test them on the KITTI odometry dataset. KITTI odometry dataset is a widely used dataset for vehicle odometry estimation and localization that contains 22 sequences of calibrated camera and LiDAR scans as well as the corresponding vehicle poses. Since the ground truth poses are required from the dataset to build the map, we only evaluate on Seq 00 to Seq 10.

In the KITTI odometry, each LiDAR frame is a point cloud with 2.2 million points 360° around within 120 meters and in a 26.9° vertical field of view. In the dataset, each LiDAR sequence contains at most 585 million points and allocates 9.4 GB in memory, while in total it contains 2 billion points and allocates 45.6 GB in disk. Obviously, the memory allocation exceeds the capacity of most automated
driving systems. Meanwhile, the training set only includes 22 km of road segments that cover only a tiny part of the urban area where the data is collected. Therefore, a light-weighted point cloud map is essential to the point cloud map based localization system.

**Implementation.** In this paper, we implement the proposed modules on a desktop machine with an Intel i5-9100F CPU (3.1GHz) and an NVidia GTX-1060 GPU. The key packages used are Python 3.8, Open3D 0.12 [81], and Pytorch 1.8.1 [82]. We also utilize the SemanticKITTI [61] development kit to manage the KITTI raw data, and Cylinder3D [62] for semantic segmentation during map-based localization.

The feature point map generated from the LiDAR sequences ranges from 0.23 to 4.5 MB in size, with an average compression ratio 0.4%; the long-lasting map generated from the LiDAR sequences ranges from 6.1 to 43.8 MB in size, with an average compression ratio 0.79%.

**Odometry comparison with state-of-the-art methods.** In the KITTI odometry dataset, two metrics are compared to evaluate the performance: $t_{rel}$ and $r_{rel}$. The $t_{rel}$ denotes the average translational RMSE (m) on length of 100m – 800m, and the $r_{rel}$ denotes the average rotational RMSE (°/100m) on length of 100m – 800m. Meanwhile, we also compare the absolute translation and rotation error on each LiDAR frame with other localization works.

Table [V] presents our result of odometry metrics in the KITTI odometry dataset. In Figure [2], we compare our results with existing works. Obviously, our results outperform the existing works by 0.4 in $t_{rel}$ and 0.05 in $r_{rel}$, which clearly shows the advantage of map-based over traditional simultaneous localization and mapping (SLAM) based methods.

**Localization comparison with state-of-the-art methods.** The odometry metrics evaluate the relative offsets in every hundred meters, however, the localization module in an automated driving system concerns more upon the absolute pose errors. Besides the odometry metrics, we also evaluate the absolute translation error (ATE) [83] in meters at each LiDAR frame in the KITTI odometry dataset. In Table [III] the results are presented compared with several existing map-based localization methods. Clearly, our method outperforms existing works in 8 out of 11 sequences. In Figure [9] we plot the trajectories of Seq 00 together with the ground truth.

**VI. CONCLUSION**

In this paper, we address the problem of map-based vehicle localization by coarse-to-fine ICP on an efficient point cloud map. Through semantic segmentation and the proposed feature point extraction submodule, an efficient feature point map and a long-lasting map is generated from the given LiDAR sequence with corresponding poses. In addition, we proposed a map-based online localization method that precisely localizes the vehicle based on the LiDAR scans and the point cloud map. We evaluate our pipeline on the KITTI odometry dataset, achieving much better performance on both odometry metrics and absolute translation error compared with the literature.
