3D LiDAR Map Compression for Efficient Localization on Resource Constrained Vehicles

Huan Yin, *Member, IEEE*, Yue Wang, *Member, IEEE*, Li Tang, Xiaqing Ding, Shoudong Huang, *Member, IEEE*, and Rong Xiong, *Member, IEEE*

***Abstract*— Large scale 3D maps constructed via LiDAR sensor are widely used on intelligent vehicles for localization in outdoor scenes. However, loading, communication and processing of the original dense maps are time consuming for onboard computing platform, which calls for a more concise representation of maps to reduce the complexity but keep the performance of localization. In this paper, we propose a teacher-student learning paradigm to compress the 3D point cloud map. Specifically, we first find a subset of LiDAR points with high number of observations to preserve the localization performance, which is regarded as the teacher of map compression. An efficient optimization strategy is proposed to deal with the massive data in original map. With the supervision of compressed map, a student model is built by training a random forest model fed with geometric feature descriptors of each point. As a result, the student model is able to compress the map without referring to the expensive numerical optimization. Additionally, by incorporating the features, the innovative student model can be generalized to other new maps while no re-training is required. We conduct thorough experiments on multi-session dataset and KITTI dataset to demonstrate the effectiveness and efficiency of the proposed learning paradigm, and the comparison with other map compression methods. The final results show that the learned student model can achieve efficient map compression with comparable LiDAR based localization performance to the original map at the same time.**

***Index Terms*— LiDAR based localization, map compression, teacher-student learning paradigm.**

I. Introduction

MAP is a critical component for autonomous vehicles in applications. Especially for localization task, high precision maps with millions of points have been constructed via onboard sensors as an assistance. However, due to the limited computing and memory resources of the mobile computing platform, these maps bring significant burden to real-time processing and offline management. Therefore, map compression is one of the desired techniques to save the processing power and storage space, and expected to preserve the accuracy of localization at the same time.

Three-dimensional (3D) light detection and ranging (LiDAR) scanner is widely used for building 3D dense point cloud maps, as well as real-time localization for vehicles. A popular method to compress the point cloud map and accelerate the localization is to define the map with semantic objects, such as trees, light poles and traffic signs, which are lightweight for loading, communication and processing, thus becoming the main trend for representing on-the-road maps [1]–[4]. However, when the technique is applied to off-the-road area, the main concern is that the pre-defined object classes may rarely occur, leading to a very sparse semantic map providing insufficient cues for localization. If the object classes are re-defined to adapt to new environments, large amount of labeled data is required to re-train the classifiers, limiting the generalization of the method.

To enhance the generalization, we consider that the 3D LiDAR map compression should focus on the low level geometric property of point clouds, just as JPEG [5] does for image compression, which focuses on the high frequency components, like edges and corners. The main difference of the map compression is that when the size of the map is reduced, the localization is still required to be reliable. In computer vision community, there are several works proposing visual map compression methods based on observation count [6], [7] through optimization techniques [8], [9]. However, direct extension of these methods to LiDAR map is not possible, since the number of points in the laser map is much larger, causing significantly growing computation complexity. In addition, the observation count is not the low level geometric feature of point clouds, but highly dependents on the motion trajectories of mobile vehicles, which means that for each large map, expensive data preparation phase and optimization process are unavoidable.

It is summarized that a desirable map compression system for LiDAR map should be generalized to new environments with high efficiency, while localization performance on the compressed maps is preserved, at least decreases slightly. To achieve this goal, we set to raise two hypotheses as follows.

Manuscript received March 17, 2019; revised August 29, 2019 and November 27, 2019; accepted December 16, 2019. Date of publication January 3, 2020; date of current version February 2, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1309300 and in part by the National Nature Science Foundation of China under Grant U1609210 and Grant 61903332. The Associate Editor for this article was J. Li. *(Corresponding author: Yue Wang.)*

H. Yin, Y. Wang, L. Tang, X. Ding, and R. Xiong are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou 310058, China, with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310058, China, with the Joint Centre for Robotics Research between Zhejiang University, Hangzhou 310058, China, and the University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: wangyue@iipc.zju.edu.cn).

S. Huang is with the Center for Autonomous Systems (CAS), University of Technology Sydney, Sydney, NSW 2007, Australia, with the Joint Centre for Robotics Research between Zhejiang University, Hangzhou 310058, China, and the University of Technology Sydney, Sydney, NSW 2007, Australia.

Digital Object Identifier 10.1109/TITS.2019.2961120

Firstly, in computer vision community, many works score the map points by observation count to achieve localization oriented map compression [6]–[9], and this indicates that the observation count is correlated with the localization performance. So we propose the first hypothesis: *the observation count is correlated with localization performance*. In this paper, the idea is extended to the LiDAR maps by proposing an optimization strategy to approximate the solution of original programming problem within tractable time. In many other papers, geometric features or semantics can also be extracted for robust localization [3], [10], [11], so we think the geometric properties might also be correlated with the localization performance. Combing with these existing works, the second hypothesis is raised that *the observation count is correlated to the geometric property of point cloud map*, which makes the method possible to adapt to the new environments. To verify the second hypothesis, we propose a teacher-student learning model in this paper. The teacher model generates the compressed map by solving the programming problem as supervision, while the student model learns the compression using the geometric feature descriptor of point clouds. In this context, the student model is encouraged to be innovative as the similar compression result is desired with different input. We conduct thorough experiments to verify the hypotheses and demonstrate the localization on the compressed maps. Overall, the contributions of this paper are as follows:

- A learning paradigm is proposed with teacher-student model for 3D LiDAR map compression, which encourages the student model to learn the compressed results using geometric features, thus achieving fast generalization.
- An optimization strategy is proposed to approximate the full size linear programming problem to achieve the map compression, which is regarded as the teacher model in the learning paradigm.
- From the compressed maps provided by teacher model, we train a random forest model as student model, which incorporates the geometric property of environments into the map compression.
- Evaluation and analyses are performed to verify the hypotheses, and demonstrate the effectiveness and efficiency of the proposed paradigm for 3D LiDAR map compression.

The rest of this paper is organized as follows: Section II reviews the related works of map compression in recent years. The whole teacher-student learning paradigm is introduced in Section III. In Section IV, we present the details of weighted integer linear programming (ILP) for map compression and data annotation. Learning based student model is presented in Section V including feature extraction and random forest. Section VI reports the experimental results using two datasets. We conclude a brief overview on our system and a future outlook in Section VII.

II. Related Work

Essentially, the objective of map compression or summarization in this paper is to select a subset of critical representations in the map database, which is a popular topic in vision community recently. The number of times that a visual landmark has been observed by mobile platforms can be used as a scoring function for selection [7], and landmarks with high score are considered as informative observations for localization. Dymczyk *et al.* [6] presented novel scoring functions and sampling policies for multi-dataset landmark simplification to achieve place recognition, combined with overlapping trajectories. Specifically, map compression for localization task requires that the map points or landmarks cover the most of robot poses or keyframes using the minimum amount, which is a maximum coverage problem or K-over problem and also an NP-complete problem. Li *et al.* [12] proposed a greedy heuristic based approach to select points from large amount of Internet photo collections, which predicts the most reliable points for point-to-feature matching to estimate the camera pose. Cheng *et al.* [9] presented a framework to predict K value by evaluating the matching potential of 3D visual points and also an adaptive weighting scheme for greedy process. These algorithms have no guarantee on solution for map compression problem optimally but approximately solve the problem.

One popular visual map compression method is based on optimization approach, formulating the selection problem as a programming problem. For computer vision tasks, location recognition is similar with the localization, and ILP was used in data database to select the most compact subset of images to achieve recognition [13]. Park *et al.* [8] proposed the quadratic programming (QP) method to solve the visual map compression, which is a baseline for further 3D point cloud reduction. Dymczyk *et al.* [14] split the entire optimization into sub-sections by defining the pose graph, which can accelerate the compression process. They also integrated the compressed map into other works for long autonomy, such as map maintenance [15] and map management [16]. Besides, they summarized the map on the agent-side using a regression model and defined ranking features [17]. The training set is annotated based on the observation counts in multiple datasets covering the same trajectory. Similar work was presented by Merzić *et al.* [18] to predict the quality of visual map for localization and path planning. Van Opdenbosch *et al.* [19] proposed an ILP based method with minimum spanning tree. The tree combined all the features to get the dependency correspondences, which generate an optimal coding order for weighted programming approach.

As for map compression problem in 3D LiDAR, less researches focus on map reduction, but tried to solve the data compression by other measures in the following. In this paper, the proposed lossy map compression aims at reducing the map size by permanently eliminating useless points, which is different from lossless compression methods. Researchers in remote sensing area proposed encoder based representations to compress the LiDAR points [20], [21], thus achieving lossless compression in another perspective. For lossy methods, some researchers downsampled the 3D point clouds by clustering, which is not localization task oriented. Naturally salient regions are extracted from 3D LiDAR data in [22] for scan matching. Similarly, Lee *et al.* [23] introduced rotation invariant descriptors from salient regions in laser map. These methods are mainly based on geometric features and

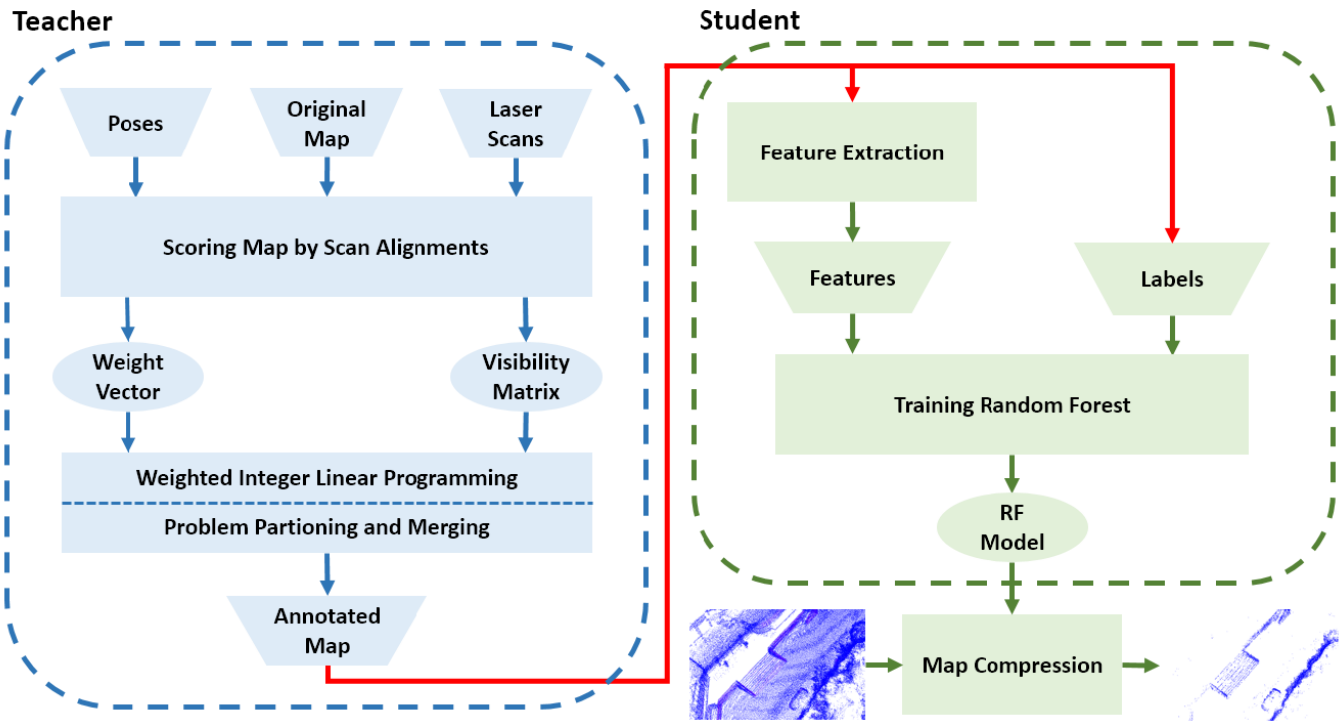


Fig. 1. The framework of the proposed teacher-student learning paradigm for map compression. Blocks are colored to difference teacher and student. Scoring and programming steps aim at annotating the original full map as the supervision. With the extracted features and labels from teacher, we train a random forest model as student for generalization in new environments.

dependent on some pre-defined descriptors heavily. Additionally, in the area of simultaneous localization and mapping (SLAM), if the mobile platforms revisit the same places, the repeated map information or pose graphs can be merged for sparsification. Many related works [24], [25] proposed information theoretic compression of pose graphs, which is also effective for map database reduction, but relies on the graph form of mapping heavily. In this paper, the appearance features of point clouds are used for map summarization without complex organization form of maps, thus simplifying the compression process possibly.

III. Teacher-Student Paradigm

The framework of the proposed teacher-student paradigm is presented in Fig. 1. There are two main components to achieve learning map compression: programming based map compression for annotating map points, which is the teacher in the paradigm; random forest model for map compression in new environments, regarded as the student learning from the optimized results. The programming is time consuming in very large maps, and the learned student, trained model from the teacher, can be generalized for map compression in new environments without much complexity.

In the programming phase of the teacher, we use two original materials from mapping process: vehicle poses $\mathcal{P} = \{p_i\}$ and one-to-one corresponding laser scans $\mathcal{S} = \{s_i\} = \{\{l_i\}\}$, where l_i represents the i_{th} laser point in a current scan s_i. Firstly, original full map $\mathcal{M}_o = \{m_i\}$ can be scored by observation count, where m_i represents one laser point in the map. The score function is defined by the observation counts of map points using sequent scan alignments. From the proposed scoring process, visibility matrix $\mathbf{A}$ and weight vector $\mathbf{q}$ are generated, which demonstrate the visibility and importance of every map point. Then a weighted ILP method is then applied to annotate the original full map. After the programming step, each map point is marked in binary vector $\mathbf{x}$ as whether to be eliminated for summarization. To accelerate the whole process, a problem partitioning and merging strategy is used to solve the complex optimization, and the final result approximates to the global minimum iteratively.

In the learning phase of the student, we extract representative features $\mathcal{F} = \{f_i\}$ from the original full map considering the informative geometric property of 3D point clouds. Then a random forest model is trained using these features $\mathcal{F}$ and compressed results $\mathbf{x}$. With the trained forest model $\mathbf{T}$, other new maps can be fed into the learning model to generate compressed maps. The learning stage or the student can compress the original map easily without complex scoring and optimization steps.

IV. Approximation of Programming for Large Map

In this section, we introduce the map compression method of the teacher model in Fig. 1, which relies on the first proposed hypothesis: the observation count is correlated with localization on the map. The whole process includes two parts: scoring by alignments and the formulation of ILP problem. The scoring process gives different weights and describes the visibility of map points. Thus ILP can achieve the point selection for map compression. We also propose a novel problem partitioning method to simplify the optimization step

and achieve the approximation of global minimum of programming. As a result, the teacher generates the compressed map by programming as the data annotation for the following learning student.

A. Scoring by Alignments

The definition of whether a laser map point is observed in the map comes from iterative closest point (ICP), which is a popular point cloud registration algorithm for its simplicity in the area of computer vision and robotics. The crucial part of ICP [26], [27] is the point-to-point distance d_{l_i,m_i} between the stored map point m_i and current laser point l_i, which is the minimization part of ICP essentially. Based on the matched distance after the pose is estimated or corrected, we set the map point as observed with the following criteria:

$$Observed = \begin{cases} 0 & d_{l_i,m_i} \geq d_{th} \\ 1 & d_{l_i,m_i} < d_{th} \end{cases} \tag{1}$$

where d_{th} is a threshold value in 3D Euclidean space to decide if the map point in one frame is observed or not. This criteria helps to distinguish the dense parts and sparse parts when the map points have uneven point density. Specifically, the dense part of the map points are observed more times than the sparse ones, which indicates the different importance of map points. Given a sequence of poses $\mathcal{P}$ and raw scans $\mathcal{S}$ from LiDAR sensor, we can score each map point by aligning the scans to the map and cumulating the observation counts. This scoring process should fulfill the following strategies:

- a map point can not be observed multiple times repeatedly at one pose: to reduce the sensitivity to high density points in one frame.
- the observation counts can be accumulated in different poses: the larger count is, the more times a landmark is observed, the more useful for localization.

The strategies of our scoring function are presented in Fig. 2. Finally, all map points $\{m_i\}$ can be weighted as $\mathbf{q}_i = \{q_i\}_{i=1,\cdots,N_m}$, where N_m is the number of points in the original full map $\mathcal{M}_o$. For the following programming problem, q_i is set to $(q_{max} - q_i)/(q_{max} - q_{min})$, which maps the original counts to the range of $[0, 1]$. Together, a binary visibility matrix $\mathbf{A}$ is generated to represent the matching results after all the alignments. The size of $\mathbf{A}$ is N_p by N_m, where N_p is the size of $\mathcal{P}$ or $\mathcal{S}$. More precisely, if the i^{th} map point m_i is observed when the platform is at the j^{th} pose p_j, then $\mathbf{A}_{ji} = 1$; conversely $\mathbf{A}_{ji} = 0$. We denote the storage complexity of $\mathbf{A}$ by $O(N_p N_m)$ for clarity, and $O(N_m)$ for $\mathbf{q}$.

B. Integer Linear Programming

Intuitively, the basic requirement for point cloud registration is that the observed map points should be greater than a certain quantity b in one laser frame. Under this constraint, the map compression problem can be formulated as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}}\ & \mathbf{q}^{\mathsf{T}}\mathbf{x} \\ \text{subject to}\ & \mathbf{A}\mathbf{x} \geq b\mathbf{1} \\ & \mathbf{x} \in \{0,1\}^{N_m}. \end{aligned} \tag{2}$$

where $\mathbf{x}$ is a binary vector indicates whether a map point should be eliminated or not; if map point m_i is kept, then $x_i = 1$. $\mathbf{q}$ weights each map point individually in the minimization part. The hard constraint $\mathbf{A}\mathbf{x} \geq b\mathbf{1}$ guarantees the continuous point cloud registration for pose tracking. The larger b is, more map points are kept after programming, and vice versa.

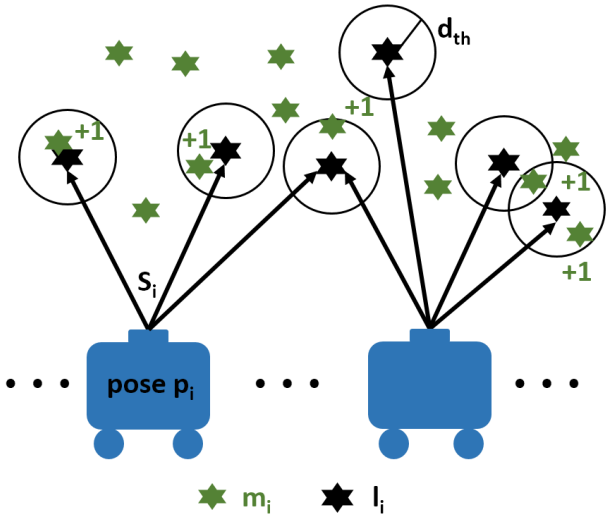


Fig. 2. Graphic illustration of the scoring strategy. Some map points are matched many times after scan alignments, which indicate that they are informative part for localization.

But the programming may fail if the number of matched map points is less than b in some places, we introduce a slack variable $\boldsymbol{\zeta}$ into the inequality to relax the optimization, shows as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}}\ & \mathbf{q}^{\mathsf{T}}\mathbf{x} + \lambda\mathbf{1}^{\mathsf{T}}\boldsymbol{\zeta} \\ \text{subject to}\ & \mathbf{A}\mathbf{x} + \boldsymbol{\zeta} \geq b\mathbf{1} \\ & \mathbf{x} \in \{0,1\}^{N_m} \\ & \boldsymbol{\zeta} \in \{0, \mathbb{Z}_+\}^{N_p}. \end{aligned} \tag{3}$$

where $\boldsymbol{\zeta}$ is a semi-positive integer vector that allows less visible constraints on some laser frames. λ is a tuning parameter that decides the hardness of constraint, when λ goes to zero, the constraint becomes softer, since the minimization part allows large values in $\boldsymbol{\zeta}$ combining the small value of λ.

Some previous methods [8], [14] used quadratic term $\mathbf{Q}$ in the programming. $\mathbf{Q}$ is a N_m by N_m matrix that represents the artificial defined relations between each map points. Such large matrix is a huge burden for optimization, and improves the matching performance not too much [8] [14]. We discard the quadratic term for simplification, same as [19].

After the optimization, we label the results on the original map $\mathcal{M}_o$ to achieve annotation for the following learning phase, and compressed map from programming is also generated:

$$\mathcal{M}_o \xrightarrow[\text{compress}]{\text{programming}} \mathcal{M}_p \tag{4}$$

C. Problem Partitioning and Merging

The optimization process is still a heavy calculation for the millions of map points. It is too hard to solve the problem in short time with thousands of poses or frames

Algorithm 1 Problem Partitioning and Merging Strategy

Input:
The binary vector of map points: $\mathbf{x}$;
The other parameters of ILP: $\mathbf{A}, \mathbf{q}, \lambda, b$;
The number of first partitioned trajectory: N_e;
Output:
while $N_e \geq 1$ **do**
Partition visibility Matrix by rows: $\mathbf{A} = [\mathbf{A}_1; \cdots; \mathbf{A}_{N_e}]$
for $i = 1$ to N_e **do**
Obtain $\mathbf{x}_i$ by solving Eq. (3) with $\mathbf{A}_i$
end for
Merge all the sub-solutions: $\mathbf{x} = \mathbf{x}_1 \vee \cdots \vee \mathbf{x}_{N_e}$
$N_e = N_e/2$;
end while

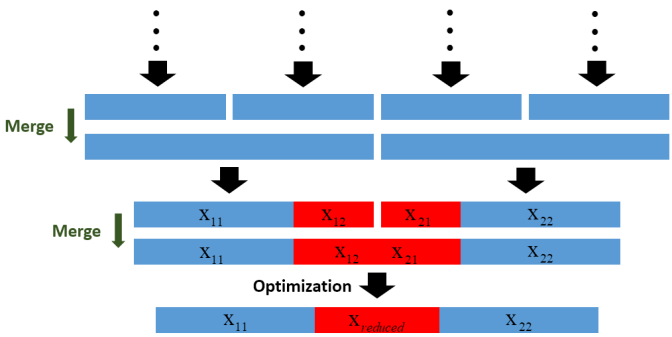


Fig. 3. This figure illustrates the formulation of the proposed strategy. One block represents the relative map points that reserved for one partitioned trajectory. With the proposed partitioning and merging method, the whole map can be compressed to a limited scale iteratively.

(N_p inequalities) and huge number of map points (N_m variables). Theoretically, the time complexity of the binary ILP problem is exponential along with N_m, and is also related to the number of constraints N_p, denoted by $O(f(N_m, N_p))$. For example, binary ILP is solvable in $O(2^{N_m} N_m N_p)$ time by brute-force enumerative algorithm [28]. In this paper, a novel partitioning and merging strategy is proposed to obtain an approximation of the final solution in limited time.

In this paper, first, the trajectory is partitioned into N_e sections evenly:

$$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_{N_e}\} \tag{5}$$

and the visibility matrix is segmented by rows:

$$\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_{N_e}] \tag{6}$$

Note that the total size of map points $\mathbf{x}$ remains invariant before solving all Eq. (3) with each $\mathbf{A}_i$. After optimization, we merge all the sub-results together as the final compressed result using binary OR operations, as follow:

$$\mathbf{x}_{merged} = \mathbf{x}_1 \vee \mathbf{x}_2 \vee \cdots \vee \mathbf{x}_{N_e} \tag{7}$$

Obviously, the new map $\mathbf{x}_{merged}$ is reduced compared to the original $\mathbf{x}$, and still satisfies the constraints for localization. We regard $\mathbf{x}_{merged}$ as the full map, and apply the Eq. (5, 6, 7) as a new iteration. This partitioning and merging operation is repeated iteratively. In each new iteration, the splitting length or the number of divided trajectories is re-doubled from the previous iteration process. Essentially, the proposed split and merge method is a kind of greedy strategy. We first solve a partial ILP problem, which is regarded as the locally optimal solution on one section of whole trajectory. And then all compressed results are merged for the next ILP problems with larger partitioned sections. The iteration will continue until the global programming problem is solved on the whole trajectory finally. An illustration of the partitioning and merging strategy is presented in Algorithm 1 and Fig. 3. For every partitioned ILP during iterations, the storage complexity of $\mathbf{q}$ remains as $O(N_m)$. The complexity of $\mathbf{A}$ is reduced to $O(N_m)$ from $O(N_p N_m)$, since the number of rows is kept at a constant value $C = \frac{N_p}{N_e}$, which is not related to the size of total set N_p essentially. Similarly, the time complexity is also not connected to N_p, and is reduced to $O(f(N_m, C))$, thus the entire problem can be solvable within tractable time.

In [14], both visual features and keyframes in graph are partitioned for faster processing. The experiments were conducted in indoor environments with separated rooms, so it's reasonable that they used graph-cut to partition the problem according to the minimum co-observed landmarks between keyframes. While for the extension to outdoors, where the number of co-observe landmarks between poses are almost the same, we tried minimum graph cut of frames but obtained unacceptable results for map compression (every single pose is split from the pose graph). So we select the uniform partitioning of poses as the strategy in the outdoor scenes, and the whole map points are used for every ILP on sub-piece of trajectory.

Analysis With the proposed partitioning strategy of ILP problem, the optimized result is approaching to the global minimum iteratively. We perform the analysis as follows, and select a case study in Fig. 3.

Suppose that there are two maps and trajectories: $\mathbf{x}_1 = \{\mathbf{x}_{11}, \mathbf{x}_{12}\}$; $\mathbf{x}_2 = \{\mathbf{x}_{21}, \mathbf{x}_{22}\}$, and $\mathcal{P}_1, \mathcal{P}_2$ respectively. $\mathbf{x}_{11}$ represents the part of $\mathbf{x}_1$ that can only be observed by $\mathcal{P}_1$, and $\mathbf{x}_{12}$ is the shared part by $\mathcal{P}_1$ and $\mathcal{P}_2$, which means both $\mathcal{P}_1$ and $\mathcal{P}_2$ can observe the map points in $\mathbf{x}_{12}$. It is same for $\mathbf{x}_{21}$ and $\mathbf{x}_{22}$ in $\mathbf{x}_2$. We denote the reduction result of one map points set $\mathbf{x}$ as $\mathbf{x}^-$. The different observation cases determine the results of the optimization:

- For $\mathbf{x}_{11}$ and $\mathbf{x}_{22}$, no points is eliminated after the merging and optimization. Because some poses in $\mathcal{P}_1$ only observe $\mathbf{x}_{11}$ not $\mathbf{x}_{12}$ or $\mathbf{x}_{21}$, $\mathbf{x}_{11}^-$ will not satisfy the constraints $\mathbf{A}\mathbf{x} + \boldsymbol{\zeta} \geq b\mathbf{1}$. $\mathbf{x}_{22}$ should not be reduced for the same reason.
- Other poses in $\mathcal{P}_1$ or $\mathcal{P}_2$ observe the shared part after merging process. The number of observed laser points of some poses is larger than b, which means that some points in the merged shared part are redundant. We assume that there exist $\mathbf{x}_{new} = \{\mathbf{x}_{12}^-, \mathbf{x}_{21}^-\}$ after the optimization of the whole map part, and the $\mathbf{x}_{new}$ still satisfy the constraints of programming since there must be a summarized $\{\mathbf{x}_{12}^-, \mathbf{x}_{21}^-\}$ that can provide enough observations for localization, same as single $\mathbf{x}_{12}$ or $\mathbf{x}_{21}$ for $\mathcal{P}_1$ and $\mathcal{P}_2$.

Overall, by optimizing the shared part of merging result, the proposed partitioning strategy is able to reduce the amount of map points iteratively.

V. Learning From the Optimized Result

As one can see, for the teacher part in Fig. 1, the programming based approach costs manpower for data preparation and is time consuming in optimization step, which also brings much computational complexity on resource constrained platform. Considering the second hypothesis, the observation count is also correlated to the geometric property of point cloud map, thus we propose to learn the map compression skill. In this section, we select certain representative features from 3D point clouds for Random Forest (RF) training [29]. The trained RF model is regarded as the innovated student model in the proposed teacher-student paradigm, as shown in Fig. 1. The trained forest is able to compress the new original maps, thus achieving learning map compression without complex optimization.

A. Feature Extraction

We first focus on the local geometric features of point cloud map $\mathcal{M}_o$, several descriptors are extracted for each map point m_i as follows:

- The eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ of m_i combined with near points are computed to describe the local surface around the point. Walls, slopes and bushes on both sides of the road are distinguished from each other for the different surface shapes. The ratios of eigenvalues are widely used in various applications in perception, such as loop closing for mapping [30] etc. We extract the original eigenvalues and feed them into RF for training and testing in this section.
- Surface normals $\mathbf{n} = (n_1, n_2, n_3)$ are computed to make difference between similar shapes around the vehicle, walls and road planes e.g. Normal vector is the eigenvector of the smallest eigenvalue λ_3 essentially. We filter the normal vectors of the whole map in one global frame, in order to keep the consistency of orientations of surface normals.
- Density ρ is computed for more descriptions on different shapes of objects. ρ is estimated by a sphere that contains k nearest points around one point, where k is a constant value across whole map. This feature is able to distinguish the sparse branches and dense leaves on trees for example. The uneven density can reflect the different importance of 3D points essentially, which is useful to achieve localization oriented map compression.
- Advanced LiDAR sensors can provide intensity value I of m_i, representing different reflections of surfaces, glasses and walls for example.

Besides the basic descriptors above, considering that mobile vehicle is running on road in maps, the relation information between poses $\mathcal{P}$ and map points $\mathcal{M}_o$ is also important for localization tasks. For each map point m_i, we search the nearest pose p_j in the trajectory. We select p_j to represent the path point that most likely observe m_i. Two main distances in Euclidean space are used as follows:

- The range distance r_{p_j,m_i} from p_j to the m_i is important, which is helpful to keep near walls and eliminate buildings which are too far in the learning stage.
- The elevation h_{p_j,m_i} in local is used, which is helpful to distinguish bushes that lying on the ground and leaves on the trees for example. h_{p_j,m_i} is computed according to the range difference on Z-axis in the global frame.

To accelerate the nearest neighbor search, a K-dimension (KD) tree is built between $\mathcal{M}_o$ and $\mathcal{P}$, and also in $\mathcal{M}_o$ itself for estimations of surface normals and densities. Note that the 3D coordinates of points m_i or poses p_j are in the same global frame.

Finally, the extracted geometric features in the learning phase are as follows:

$$\mathcal{F} = \{f_i\} = \left\{\lambda_1, \lambda_2, \lambda_3, n_1, n_2, n_3, \rho, I, r_{p_j,m_i}, h_{p_j,m_i}\right\} \quad (8)$$

In summary, the feature of every point in point clouds is generated for map compression. Specifically, $\{\lambda_1, \lambda_2, \lambda_3, n_1, n_2, n_3, \rho\}$ are calculated considering the nearby points of one point, and $\left\{I, r_{p_j,m_i}, h_{p_j,m_i}\right\}$ are based on self properties of one point.

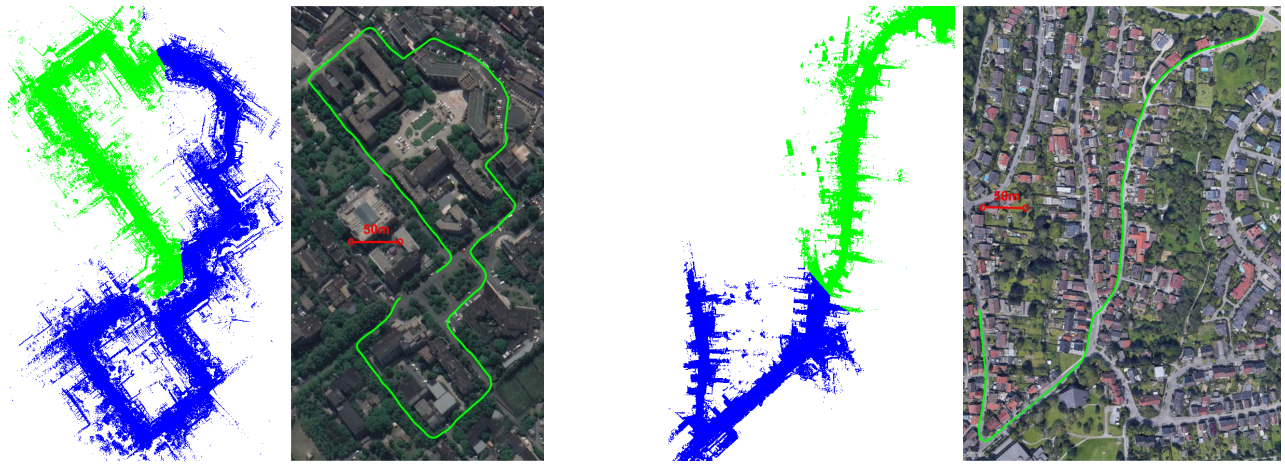
In [17], ranking features are designed artificially on the agent side of camera, such as track length of a visual point, maximum angle between observed rays etc. And there also exist some modeling measurements for laser scans, such as ray casting method [31] or incident angles [32]. But these exhausting feature extractions are expensive and against the proposed concise learning based method. 3D LiDAR scanners can obtain much more accurate range measurements than cameras, so we extract the geometric feature descriptions of laser points in maps, where complex model of ray casting or other observation models are not selected in the learning stage of our algorithm.

B. Building Random Forest

After the feature extraction process from the map $\mathcal{M}_o$, we can train a random forest model $\mathbf{T}$. Random forest is a powerful and popular classifier in machine learning area, and has been used in many applications for its simplicity. There are less parameters in RF model, which can be trained without GPU device. In point clouds, RF model is able to handle the unavoidable noises. For autonomous driving or transportation systems, random forest method is widely used for point cloud segmentation or classification tasks [33]–[35].

In random forest, the trees are independent to each other in training step, but make the final decision with probability together in prediction step. Firstly, trees are built by performing an individual learning algorithm that splits the input features into subsets. The splitting criteria is based on Gini index. Then the predicted class is voted by the trees. The classification result can be presented in probability form:

$$x_i = \begin{cases} 0 & Prob_{m_i} < Prob_{th} \\ 1 & Prob_{m_i} \geq Prob_{th} \end{cases} \quad (9)$$



(a) original full map $\mathcal{M}_o$ and satellite imagery of YQ dataset

(b) original full map $\mathcal{M}_o$ and satellite imagery of KITTI dataset

Fig. 4. (a) Origin full map at south part of Yuquan campus in Zhejiang University, attached with the satellite imagery. The training material provided by the teacher are colored in green, and the blue part are used by the student for evaluation. The green line in satellite imagery is the trajectory of platform motion. (b) The proposed map compression is also evaluated on sequence 10 of KITTI odometry benchmark, which is collected in an rural residential area of Karlsruhe City.

where $Prob_{m_i}$ is the probability of whether a map point m_i should be kept in the learned compressed map $\mathcal{M}_l$.

We can compress the original full map to a certain amount by controlling the threshold $Prob_{th}$ using $\mathbf{T}$, which is trained on $\mathcal{M}_o$ with the same ratio; or we can train a forest with a different compression ratio on $\mathcal{M}_o$, and achieve another ratio $\mathcal{M}_l$ by thresholding $Prob_{th}$ directly. The two different compression methods are compared in the following experimental section.

Overall, after the learning phase of the student, the trained model can also reduce the map size, compared to the teacher (4), as follows:

$$\mathcal{M}_o \xrightarrow{\mathbf{x}=\mathbf{T}(\mathcal{F})} \mathcal{M}_l \tag{10}$$

VI. Experimental Results

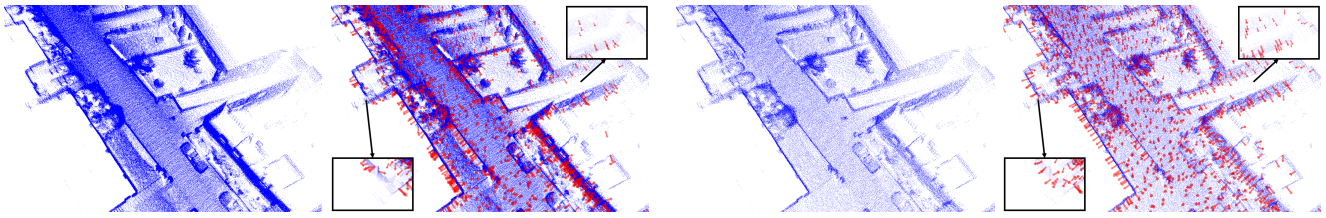
The proposed learning paradigm for map compression in Fig. 1 is evaluated in this section. We validate the prediction effectiveness of the learned map $\mathcal{M}_l$ from programming based compressed map $\mathcal{M}_p$, which is the comparison between student and teacher in the paradigm. And ICP registration is applied on $\mathcal{M}_l$ to validate localization performance compared to other map simplification methods. Computational complexity of localization on different maps are also presented to demonstrate the efficiency of the proposed map compression. Finally, we also present the generalization to new places and datasets without extra pre-training.

A. Dataset and Implementation

In the experiment sections, we first use the multi-session dataset, called YQ, which is self-collected on a mobile robot platform in a university campus during three days [36], in which the LiDAR scans are captured by Velodyne VLP-16. The original full map $\mathcal{M}_o$ is generated by using SLAM, shown in Fig. 4 (a). The point cloud map covers a ground area more than 20000 m^2 with a density nearly 100 points/m^3. We also use a popular dataset KITTI [37] for evaluation, which is a public dataset for autonomous driving. KITTI provides laser scans from Velodyne HDL-64E and ground truth of the poses from GPS data. We select sequence 10 from KITTI odometry benchmark for evaluation, which is collected in a rural residential area of nearly 12000 m^2. The point cloud map generated from KITTI dataset is with a density of 118 points/m^3, shown in Fig. 4 (b). Considering the zero or low speed situation in data collection, we remove some dense scans or poses to generate more uniform keyframes, which avoids large observation count of point clouds at some places. This kind of keyframe based method is widely used in mapping techniques. Specifically, there are nearly 5000 and 1000 keyframes in the YQ and KITTI dataset respectively for the different levels of speeds of mobile platforms. To reduce the huge burden for storage and computation, we use Octree grid filter to generate $\mathcal{M}_o$. The filtered map keeps the original geometric properties and is still suitable for localization, thus is also an appropriate starting point for the following localization oriented map compression. The maps in YQ and KITTI dataset are summarized to 26.73% and 37.06% before map compression, which are acceptable sizes for real operations. The raw data map and summarized $\mathcal{M}_o$ are shown in Fig. 5.

We implement the programming solution using the commercial software Gurobi[1] [38]. The large matrices of ILP are stored in separate files in disk, which can reduce the storage complexity of internal memory on the resource constrained platform. We build KD-tree using libnabo[2] [39] for fast nearest neighbor search between poses and map points. Random forest is trained on the open source code[3] [40]. Feature extraction and registration tests are implemented using libpointmatcher [4] [26]. By using libpointmatcher, the normal vectors and relative eigenvalues are calculated from the covariance matrix of 3D points. The generated surface normals are shown

[1]https://www.gurobi.com
[2]https://github.com/ethz-asl/libnabo
[3]https://github.com/imbs-hl/ranger
[4]https://github.com/ethz-asl/libpointmatcher



(a) raw data map and the surface normals (b) generated $\mathcal{M}_o$ after summarization and the surface normals

Fig. 5. We present part of the raw map in (a), and the filtered $\mathcal{M}_o$ in (b). Compared to the map from raw scans, it is clearly that the summarized map reduces the huge burden on resource constrained platform. In addition, we also present some generated surface normals in the two maps, shown as red arrows in the point clouds. The surface normals vectors are similar in these two maps, which indicates that the geometric properties are kept after the summarization. Therefore, the reduced map with acceptable size is still an appropriate start point for the following localization oriented map compression.

TABLE I
LOCALIZATION ON $\mathcal{M}_p$ WITH DIFFERENT λ ($b = 1000$)

λ	0.01	0.02	0.05	0.1	0.2
size	0.19%	0.98%	1.24%	1.28%	1.30%
error (m)	Fail	Fail	0.096	**0.094**	0.102
error ($^\circ$)	Fail	Fail	0.138	**0.137**	**0.137**

TABLE II
MEAN DECREASE ACCURACY AND MEAN DECREASE GINI

Feature	λ_1	λ_2	λ_3	n_1	n_2	n_3	ρ	I	r_{p_j,m_i}	h_{p_j,m_i}
MDA	53.5	26.5	55.3	62.1	53.0	59.8	247.9	68.8	165.3	257.2
MDG	594.4	535.9	590.6	729.9	664.2	990.0	1591.7	631.1	1373.4	2297.9

in Fig. 5. The common configuration in libpointmatcher is sufficient for localization, so we follow this configuration in the experiments. All maps are stored as PLY[5] format files. To help understanding our implementation, we release the source code online[6].

As for the parameters, the threshold of matched distance is set as $d_{th} = 0.1$m. To solve the proposed ILP, the number of sub-sections is N_e, which is decided by the split length in each iteration of programming. And we set the length equals to 50, 100, ... 3200 in each step (7 times of iterations for YQ, and 5 times for KITTI). For training the forest $\mathbf{T}$, the number of trees is 100 and we set the depth of trees unlimited. About $1/3$ or $1/2$ part of $\mathcal{M}_o$ is used for training the trees and the rest of it is for validation, colored differently in Fig. 4. In the theoretical part in this paper, b and $Prob_{th}$ decide the size of compressed maps in programing and learning methods respectively.

We test different values on λ of Eq. (3), which is used to balance the constraints and computing speed for soft optimization. In YQ dataset, we set $b = 1000$ as a constant value, then the compressed maps $\mathcal{M}_p$ are obtained by the teacher model with different λ. In order to select the most appropriate parameter, we conduct point cloud registration on these maps. The compressed sizes and localization errors are shown in Tab. I. Localization fails on $\lambda = 0.01, 0.02$ for the larger $\boldsymbol{\zeta}$ in Eq. (3), which makes the constraint soft but cause the lack of observations on some poses. Finally, considering the trade off between map size and localization performance, we select $\lambda = 0.1$ for optimization.

As for the random forest model, we evaluate the Mean Decrease Accuracy (MDA) and Mean Decrease Gini (MDG) of every feature. The more the accuracy decreases due to the exclusion of a single feature, the more important that feature is. The Mean Decrease Gini measures how each feature contributes in the random forest. As one can see, in Tab. II, h_{p_j,m_i} and ρ are more critical than the other features for classification. As shown in Fig. 10, most reserved points are mainly dense and close to the road, which also indicates that h_{p_j,m_i} and ρ are helpful for point selection.

B. Approximation of ILP

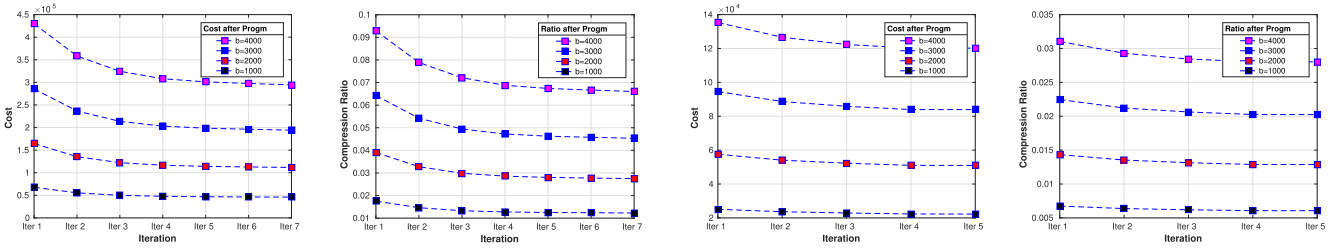
First of all, in the teacher of the learning paradigm, we evaluate the performance of ILP using the proposed partitioning and merging strategy. We tried to optimize the entire ILP problem without partitioning but the final solution cannot be achieved in days. So we use the proposed strategy to compress the map iteratively, and the full map is compressed to a certain ratio. We set $b = 4000, 3000, 2000, 1000$ in YQ dataset, and the compression ratios are 6.60%, 4.53%, 2.75% and 1.23% respectively after optimization; and the ratios are 2.28%, 2.02%, 1.29% and 0.61% in KITTI dataset.

Additionally, we record all the costs after every iterative minimization, and the remained ratio respectively, shown in Fig. 6. As one can see, the results show that the cost and ratio are reduced after the minimization in the iterative programming process. And the bigger b is, the more obvious trend is: ratio is reduced to 6.60% from 9.29% with $b = 4000$, and is also reduced to 1.23% from 1.75% with $b = 1000$ in YQ dataset. Though we do not know the final ratio after the global optimization without partitioning, the whole trend is approximate to the minimum after every iteration of programming. This result validates the effectiveness of the partitioning and merging strategy proposed in this paper.

C. Map Compression Comparison

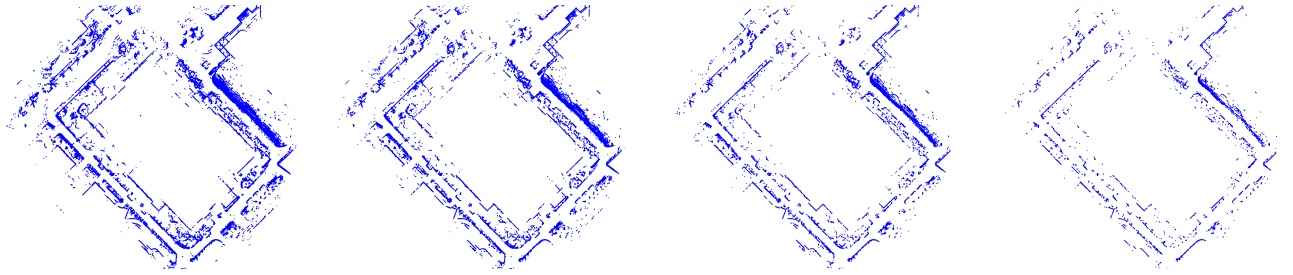
In this section, we compare the similarity between the teacher and the student in the learning paradigm. We first

[5]http://paulbourke.net/dataformats/ply
[6]https://github.com/ZJUYH/map_compression



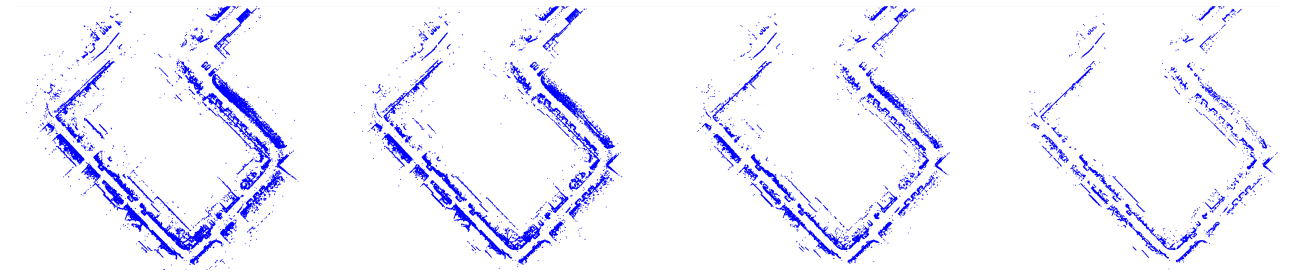
(a) reduction of costs and compression ratios of YQ dataset (b) reduction of costs and compression ratios of KITTI dataset

Fig. 6. The reductions of costs and remained points after the minimization of programming in YQ and KITTI datasets. The trends show that the final result is approximate to the global minimum by using the proposed partitioning and merging strategy.



6.60% $\mathcal{M}_p$ with $b = 4000$ 4.53% $\mathcal{M}_p$ with $b = 4000$ 2.75% $\mathcal{M}_p$ with $b = 2000$ 1.23% $\mathcal{M}_p$ with $b = 1000$

(a) Map compression results $\mathcal{M}_p$ from ILP.



6.60% $\mathcal{M}_l$ with $Prob_{th} = 0.675$ 4.53% $\mathcal{M}_l$ with $Prob_{th} = 0.698$ 2.75% $\mathcal{M}_l$ with $Prob_{th} = 0.716$ 1.23% $\mathcal{M}_l$ with $Prob_{th} = 0.749$

(b) Map compression results $\mathcal{M}_l$ from the trained RF model.

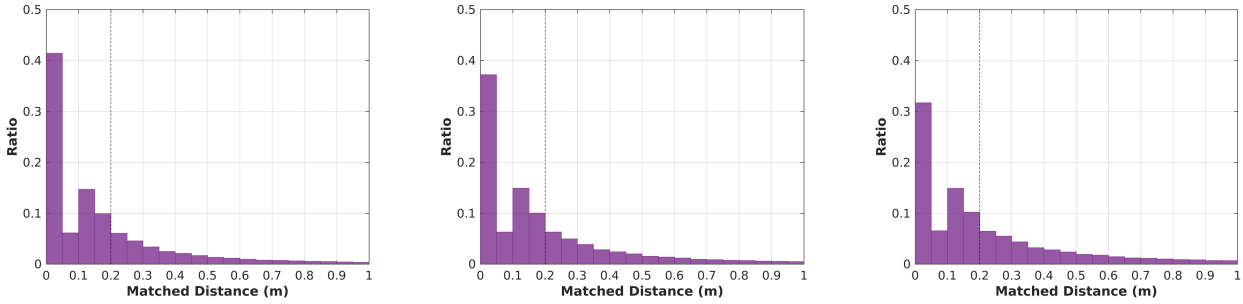
Fig. 7. In YQ dataset, the results from the two methods are quite similar. For the proposed ILP method, the number of points or compression ratio is decided by the value of b. After the RF model is trained, the compression ratio is equal to the corresponding b by thresholding $Prob_{th}$. (a)Map compression results $\mathcal{M}_p$ from ILP. (b) Map compression results $\mathcal{M}_l$ from the trained RF model.

generate different sizes of compressed maps $\mathcal{M}_p$ by setting the parameters $b = 4000, 3000, 2000, 1000$, then same sizes of $\mathcal{M}_l$ are learned by controlling the threshold $Prob_{th}$. With the decrease of b and the increase of $Prob_{th}$, less map points are reserved in $\mathcal{M}_p$ and $\mathcal{M}_l$, shown in Fig. 7. Note that we train different forest models using different annotated maps $\mathcal{M}_o$ for multiple compression ratios respectively. It is the first method in Section V-B.

To compare the similarity between the compressed maps from the teacher and student, we make statistics on the matching distances between $\mathcal{M}_p$ and $\mathcal{M}_l$ via KD-tree in YQ dataset. The nearest neighbor distances are counted as histograms for similarity measure, shown in Fig. 8. When the full map is compressed to 6.60% with $b = 4000$, almost 50% predicted points are very close to the programmed ones and the matched distances are below 0.1m, and above 70% when the distance is 0.2m, shown in Fig. 8 (a). The similarity measurements in Fig. 8 (b), Fig. 8 (c) and Fig. 9 (a) for $b = 3000, 2000, 1000$ also perform acceptable results.

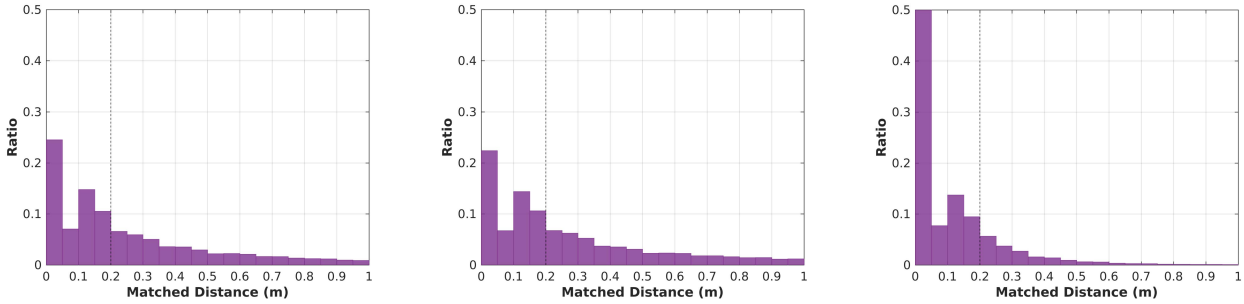
Besides learning from the same ratio compressed by programming, we also propose the second method in Section V-B. By tuning $Prob_{th}$ after the forest is trained, a certain ratio can also be obtained from compressed maps with different ratio. The distance histogram between 1.23% $\mathcal{M}_l$ from $b = 4000$ and $\mathcal{M}_p$ with $b = 1000$ is shown in Fig. 9 (b), which achieves almost same similarity compared to Fig. 9 (a). And the similarity of two different tuning methods is presented in Fig. 9 (c), which validates the consistency of the learning paradigm in this paper. At last, almost all the distances are in the range of 0.2m in Fig. 8 and Fig. 9. The comparison between $\mathcal{M}_p$ and $\mathcal{M}_l$ demonstrates the effectiveness of the proposed teacher-student paradigm for map compression.

We present part of the map compression results in Fig. 10. As one can see, for ILP and learning based methods, the reserved point clouds after map compression are mainly focused on the salient objects with dense points on the sides of roads. Tree trunks, walls, telegraph poles etc., which can be observed frequently for vehicles on roads, are still reserved after compression with high ratio. While for ground, sparse



(a) 6.60%$\mathcal{M}_l$ from $b = 4000$ (b) 4.53%$\mathcal{M}_l$ from $b = 3000$ (c) 2.75%$\mathcal{M}_l$ from $b = 2000$

Fig. 8. Histograms of matched distances between $\mathcal{M}_l$ and $\mathcal{M}_p$ with $b = 4000, 3000, 2000$. Each learned map $\mathcal{M}_l$ is generated from the respective forest model trained by the same ratio of compressed map $\mathcal{M}_p$.



(a) 1.23%$\mathcal{M}_l$ from $b = 1000$ (b) tuning 1.23%$\mathcal{M}_l$ from $b = 4000$ (c) similarity of $\mathcal{M}_l$ in (a)(b)

Fig. 9. (a) Histogram of matched distances between $\mathcal{M}_l$ and $\mathcal{M}_p$ with $b = 1000$. (b) We achieve the same ratio 1.23% $\mathcal{M}_l$ from $\mathcal{M}_p$ with $b = 4000$ directly by tuning $Prob_{th}$. (c) The similarity of these two maps $\mathcal{M}_l$ from different two methods with same compression ratio.

leaves and branches etc., are eliminated for the poor visibility during localization. The map points closer to road are more likely to be reserved for the stronger visibility. In summary, the dense semantic parts of original map are critical components of compressed maps for localization.

We consider that the reserved points will meet the needs of location positioning from two aspects. Firstly, the local feature based maps are widely used for precise registration or localization [3], [10], [11]. Edges, walls or semantic objects are mainly used for robust matching in these papers. Our reserved points are similar to these features, which validates the rationality of the compressed result. Secondly, there is no prior knowledge or semantic information in our system, and the learned result from observation count is still similar with those feature or semantic based maps, thus verifying the proposed hypothesis in qualitative way.

In addition, we consider that there are some reasons limit the applications of the learned student, as follows:

- The limitations of feature extractions and conventional machine learning algorithm can not make the learned results perfect. Some hidden features are hard to describe by geometric descriptors, and this is the main reason.
- Clustered dense points are hard to distinguish each other for almost same geometric features but different compression results.
- Some obscured (hidden feature) surfaces or objects are observed few times on the sides of the road, but they are similar with these kept map points, and can not be classified using geometric property.

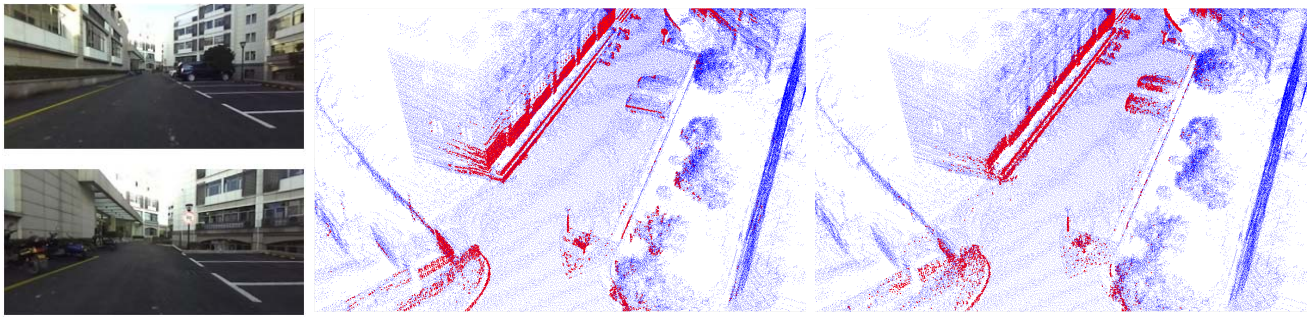
D. Localization Performance

We conduct the localization experiment on a part of YQ and KITTI dataset. YQ dataset is a three-day multi-session dataset, so we build the map using laser data in the first day, and make evaluation in the third day, which validates the localization performance under the temporal semi-static changes of the environments. Since KITTI dataset provides single session for one sequence, we evaluate the localization performance by using the same laser data as the mapping step.

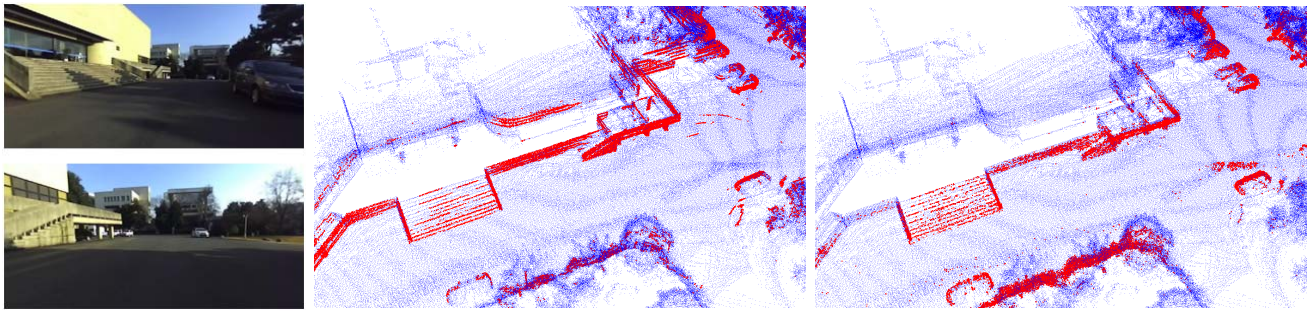
For comparison, we also generate some other compressed maps, all the test maps states as follows:

- $\mathcal{M}_p$ and $\mathcal{M}_l$: the maps compressed by the proposed paradigm in this paper. Programming based method is widely used for map compression as state-of-the-art [8], [14], [16], [19].
- $\mathcal{M}_r$: random sampled map. To eliminate the effect of random factors, we apply localization five times on five different randomly sampled map.
- $\mathcal{M}_v$: voxel grid filtered map. We use PCL[7] [41] to sample the point cloud in 3D space, and the size of leaves decides the compression ratio.
- $\mathcal{M}_c$: segmented map by clustering. We use Euclidean extraction clustering in PCL, and we try to make the high density segments widely distributed in the space.
- $\mathcal{M}_t$: a filtered map by thresholding the weight vector $\mathbf{q}$. The observation count can also be used as a sampling strategy [6], [7], which keeps landmarks that are

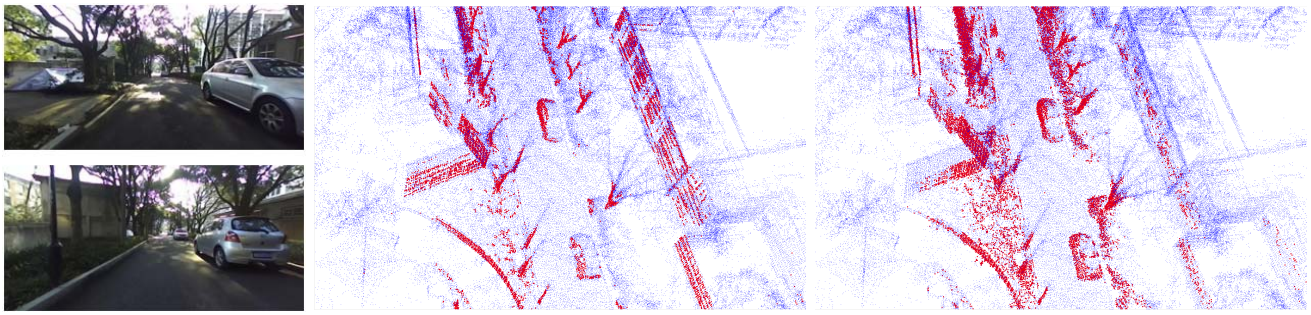
[7]http://pointclouds.org



(a) Reserved points of the proposed methods



(b) Reserved points of the proposed methods



(c) Reserved points of the proposed methods

Fig. 10. After compression of YQ dataset, as shown in (a),(b) and (c), the reserved points are colored in red on the original maps ($b = 4000$, 6.60%). The learned map $\mathcal{M}_l$ is similar with the $\mathcal{M}_p$ from the teacher. Intuitively, tree trunks, walls and some other landmarks are reserved for the higher visibility during localization.

frequently observed, but not guaranteeing the localization performance on the road.

- $\mathcal{M}_s$: a saliency map. We use the geometric based saliency evaluation in [42], [43], and tune the relevant parameters to obtain saliency points from the datasets in this paper. The saliency map represents the interesting or important part from the original map.

Localization performance is evaluated by applying ICP registration continuously on these maps with different ratios. The pose of first frame is given as the fixed start position. ICP is applied for each current laser scan, which uses the previous result as the initial value to achieve pose tracking. Some key variants have effect on the registration process [26], and we use the same set of parameters in different maps to guarantee the fair test. Considering that the most points are in 0.2m in the similarity comparison of Fig. 8 and Fig. 9, we set the distance threshold to 0.2m in YQ dataset. All the variants are listed in Tab. III.

TABLE III
VARIANTS IN ICP REGISTRATION FOR LOCALIZATION IN YQ DATASET

Module	Description
Filtering of map $\mathcal{M}$	None
Filtering of scan $\mathcal{S}$	Random sampling, keep 10%
Data association	KD-tree with k=5
Outlier filtering	Keep matches below 0.2m and maximum normal angles is 0.78 rad.
Error minimization	Point-to-plane
Checking	Iteration count reached 20, minimum error below 0.01m and 0.001 rad

The results of localization tests are shown in Fig. 12. We compare the 6D registration results with the ground truth poses, and translation errors in 3D Euclidean space are presented in Fig. 12 (a) (c). As the heading error is important for vehicles, and the rotation errors of yaw-component are shown

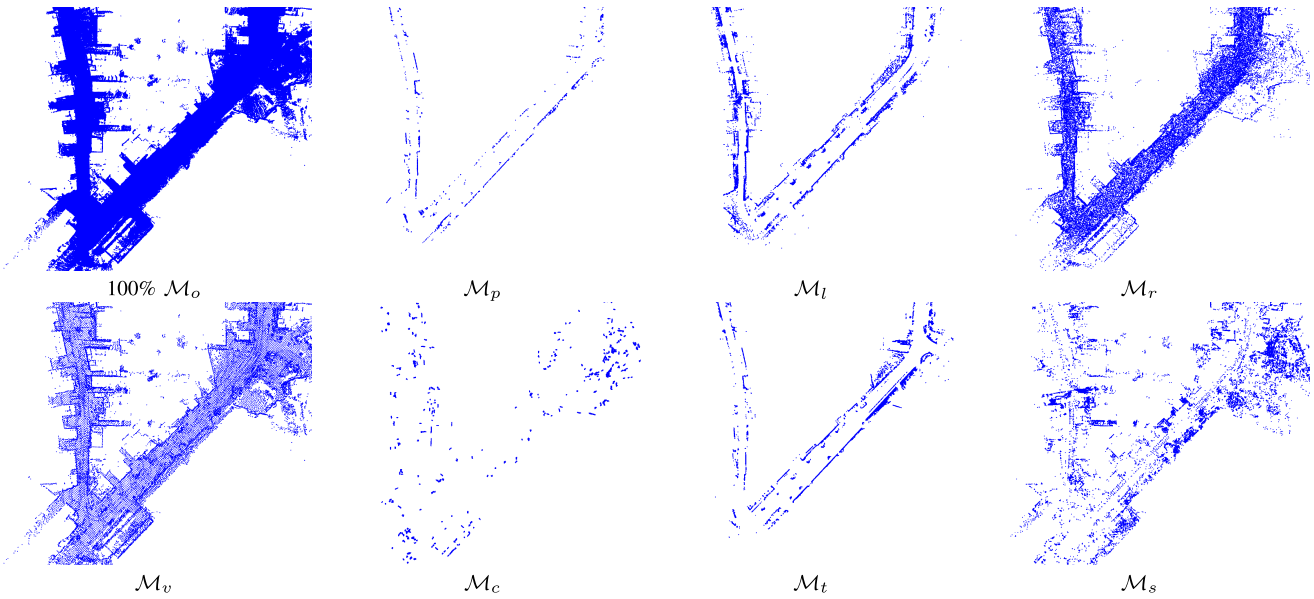
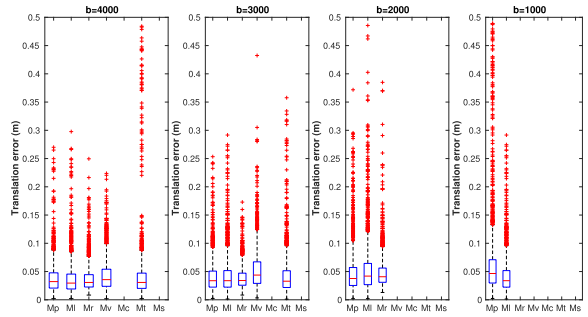
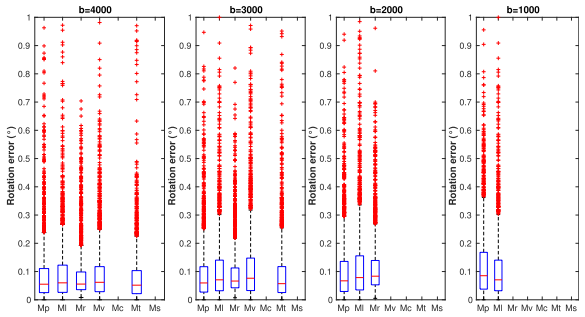


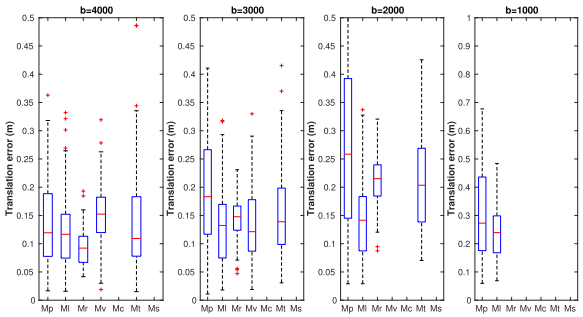
Fig. 11. 100% original map and other 2.28% compressed maps for localization test in KITTI dataset. $\mathcal{M}_p$ and $\mathcal{M}_l$ are obtained by the proposed teacher-student learning model by setting b and tuning $Prob_{th}$. The other compressed maps are generated by using the comparative methods.



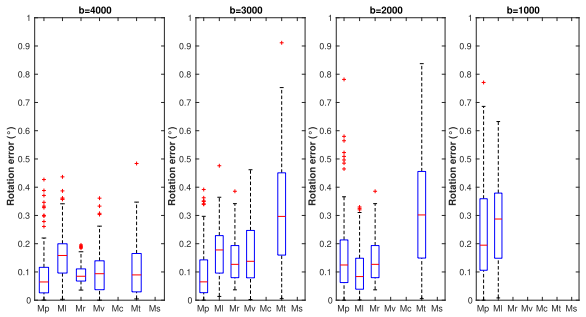
(a) translation errors of YQ dataset



(b) rotation errors of YQ dataset



(c) translation errors of KITTI dataset



(d) rotation errors of KITTI dataset

Fig. 12. Localization errors by applying ICP registration. The maps are compressed to certain ratios, the same as ILP with $b = 4000, 3000, 2000, 1000$.

in Fig. 12 (b) (d). Pose tracking fails on most of 1.23% maps in YQ dataset, while the proposed 1.23% $\mathcal{M}_p$ and $\mathcal{M}_l$ provide informative map points for localization. In KITTI dataset with 0.61% compression ratio, the vehicle is also able to localize itself only on $\mathcal{M}_p$ and $\mathcal{M}_l$. The compressed maps $\mathcal{M}_l$ perform slightly worse than $\mathcal{M}_p$, because the programming based method guarantees the matching points on all poses. Generally, the proposed teacher-student learning paradigm achieves better performance than the other comparative methods, as the other compressed maps are not localization oriented and can not provide sufficient matching points for localization. As for $\mathcal{M}_r$ and $\mathcal{M}_v$, the sparse map can not provide efficient map points for matching. $\mathcal{M}_c$ are dense maps after clustering, but the clusters can not guarantee the reliable localization on the whole road. It is same for $\mathcal{M}_t$ and $\mathcal{M}_s$ for the lack of point visibility at some places.

TABLE IV
TIME COST (MINUTE) TO GENERATE $\mathcal{M}_p$ AND $\mathcal{M}_l$ (TRAIN/TEST)

YQ Maps	6.60%	4.53%	2.75%	1.23%
$\mathcal{M}_p$	34.70	30.72	27.71	27.83
$\mathcal{M}_l$	**10.20/1.09**	**8.94/0.98**	**9.35/0.98**	**7.79/1.04**
KITTI Maps	2.28%	2.02%	1.29%	0.61%
$\mathcal{M}_p$	11.86	10.83	11.87	15.44
$\mathcal{M}_l$	**4,24/1.02**	**3.90/1.00**	**3.69/1.04**	**3.22/1.02**

TABLE V
STORAGE SPACE (MB) AND TIME COST (S) TO LOAD DIFFERENT MAPS

YQ Maps	100%	6.60%	4.53%	2.75%	1.23%
Space (Mb)	729.5	49.4	33.9	20.6	**9.2**
Time (s)	44.95	3.11	2.06	1.30	**0.58**
KITTI Maps	100%	2.28%	2.02%	1.29%	0.61%
Space (Mb)	894.0	25.5	18.5	11.7	**5.5**
Time (s)	55.50	1.62	1.12	0.73	**0.34**

Specifically, as shown in Fig. 10 and 11, our proposed maps $\mathcal{M}_p$ and $\mathcal{M}_l$ contain the most informative points on both sides of the road, which can reduce the sensitivity to the environmental changing possibly, thus keeping the platform localize itself successfully in compressed maps. And the localization results indicate that although $\mathcal{M}_l$ is not totally same as $\mathcal{M}_p$ in the previous validation of prediction, the learned map $\mathcal{M}_l$ from student model is still effective for localization. With the map size reduced down, the localization accuracies become worse. But the errors on $\mathcal{M}_l$ are limited to 0.5 m and 2°, which indicates that the vehicle can achieve successful navigation in the learned compressed maps. Overall, the localization results on $\mathcal{M}_p$ and $\mathcal{M}_l$ show that the observation count and geometric property are correlated with the localization performance, thus verifying the two hypotheses proposed in this paper.

E. Efficiency

Efficiency of learning map compression is validated in this subsection. We first compare the time cost between programming (teacher) and learning (student) methods; then loading and localization time. Two devices are used for optimization and computation on point clouds: we use a workstation to compress the full map using ILP and RF, equipped with Intel E5-2696 2.30GHz and 128G RAM; the computing platform for loading and registration point clouds in this section is a laptop with Intel i5-5200U 2.20GHz and 12G RAM, which is a common configuration for resource constrained platform.

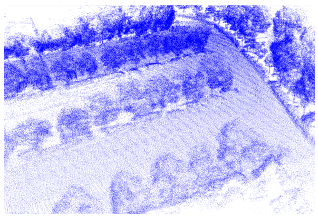
1) Map Compression: The time costs for the programming based method and random forest are evaluated using the workstation. ILP summarizes the original full map $\mathcal{M}_o$ to $\mathcal{M}_p$ from beginning to end. For the RF model, we use $1/3$ and $1/2$ part of $\mathcal{M}_o$ in YQ and KITTI dataset as the training material, and the rest of it is compressed for test. The time cost includes the growing step of trees and the generation of $\mathcal{M}_l$ at the rest of places. The records of time costs to generate maps are listed in Tab. IV.

Obviously, the optimization step of the teacher is more time consuming. And the student can compress new maps on the laptop, while ILP slows down the speed too much on this device. With the trained student model, faster prediction for large maps can be achieved by tuning $Prob_{th}$ directly, compared to the complex optimization by setting b. For ILP, much more time is needed when the map grows larger or the mobile platform travels longer, but it will spend less time for the RF model. In addition, more manpower is spent on scoring process for ILP, while the feature extraction for learning method is much more automatic. In a word, by using RF model, we can generate the informative compressed maps more easily with higher efficiency.

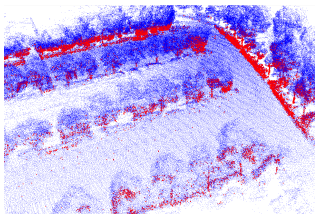
(a) vehicle

(b) onboard sensors

Fig. 13. The vehicle equipped used for data collection.



(a) 100% $\mathcal{M}_o$



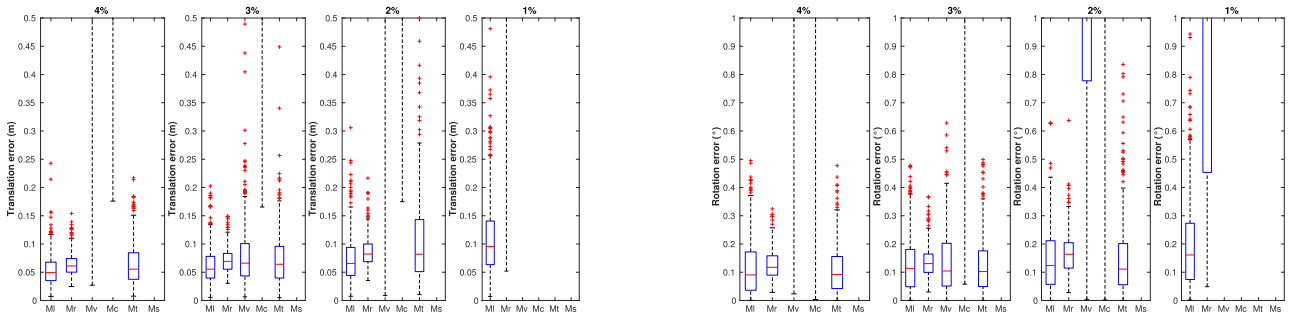
(b) 4% $\mathcal{M}_l$ are colored in red

Fig. 14. The learned model from YQ dataset is also able to compress new map at other places.

2) Loading Map: We compress the 100% $\mathcal{M}_o$ to different sizes of $\mathcal{M}_p$ and $\mathcal{M}_l$ using the trained RF model. The sizes of the maps and loading times on the laptop are listed in Tab. V. Map with 1.23% and 0.61% ratio occupy less space on disk and are efficient for loading. With less size and time, the vehicle is able to achieve fast localization and navigation in real-world applications.

3) Localization: We record the time costs of ICP registration on different ratios of part maps in YQ dataset. ICP registration performs more than 1000 times on each part map from beginning to end, and the average time of all methods in this paper are listed in Tab. VI. We also record the number of iterations in ICP process, which is able to reflect the complexity of this algorithm, shown in Tab. VII.

As shown in the tables, since the map points are compressed to a very small amount, there is little difference in time costs. And when the map is reduced gradually, more iterations of ICP are needed to locate the vehicle. In $\mathcal{M}_r$ and $\mathcal{M}_v$, current laser points are matched with widely distributed map points, so more iterative steps are taken to reduce the distances or plane angles around the vehicle. While for $\mathcal{M}_p$ or $\mathcal{M}_l$, since the map points are semi-clustered, and



(a) translation errors of Park dataset

(b) rotation errors of Park dataset

Fig. 15. Localization errors on the Park dataset. $\mathcal{M}_l$ performs better than the other simplified maps.

TABLE VI
TIME COST (SECOND) FOR ICP REGISTRATION IN YQ DATASET

Maps	6.60%	4.53%	2.75%	1.23%
$\mathcal{M}_p$	0.026	0.025	0.023	0.023
$\mathcal{M}_l$	0.030	0.028	0.029	0.028
$\mathcal{M}_r$	0.033	0.033	0.031	Fail
$\mathcal{M}_v$	0.031	0.029	Fail	Fail
$\mathcal{M}_c$	Fail	Fail	Fail	Fail
$\mathcal{M}_t$	0.026	0.024	Fail	Fail
$\mathcal{M}_s$	Fail	Fail	Fail	Fail

TABLE VII
AVERAGE NUMBER OF ITERATIONS FOR ICP IN YQ DATASET

Maps	6.60%	4.53%	2.75%	1.23%
$\mathcal{M}_p$	6.26	6.62	7.10	8.23
$\mathcal{M}_l$	7.58	8.07	8.86	10.04
$\mathcal{M}_r$	7.68	8.36	9.05	Fail
$\mathcal{M}_v$	8.57	9.28	Fail	Fail
$\mathcal{M}_c$	Fail	Fail	Fail	Fail
$\mathcal{M}_t$	6.37	6.50	Fail	Fail
$\mathcal{M}_s$	Fail	Fail	Fail	Fail

ICP becomes more efficient after using outlier filters. Actually, the use of global maps costs much time on nearest neighbor search, though initial value and KD-tree are given; so less time will be spent if we use keyframes with smaller scale maps.

F. Generalization

Finally, the proposed teacher-student paradigm is generalized to other places out of YQ or KITTI dataset. We use a dataset collected from a park to build a new full map, which is self-collected by another vehicle, shown in Fig. 13. The point cloud map is about 98 points/m^3 and covers the area of 9000 m^2. By using RF model from YQ dataset, we compress the full map to 4%, 3%, 2% and 1% ratios directly without complex ILP. This learned student model is trained in YQ dataset and tested in Park dataset without changing.

As shown in Fig. 14, the final learned maps in park are similar with those in YQ, where trees trunks and bushes near the roads are reserved, and useless map points are removed for clearance. We also conduct the localization experiment on these maps, compared to other map simplification methods. The results are shown in Fig. 15, and the proposed $\mathcal{M}_l$ achieve better performance than the others. This experiment for map compression validates the generalization of the innovated student. Finally, both results shown in Fig. 12 and Fig. 15 show that the localization performance is correlated with the observation count and geometric property of the point cloud map, thus making the proposed hypotheses verified.

VII. CONCLUSION

An efficient learning map compression method is proposed in this paper using 3D LiDAR data and vehicle poses, which is based on the framework of teacher-student paradigm. Specifically, our method is driven by the proposed hypotheses: *observation count is correlated with localization performance, and is also correlated to the geometric property of point cloud map*. We first achieve the map compression by programming with the observation count, which is regarded as the teacher and provides training material. A partitioning and merging strategy is proposed to solve the programming problem within tractable time. The student is the RF model with geometric features of point clouds, which is trained to compress full maps in new environments. The results show that learned maps are effective and efficient for vehicle localization even when the map is reduced to near 1% of the original size. In the future, we intend to achieve map update to remove low dynamics in the environments, thus making the map more reliable for long term localization.

REFERENCES

[1] P. Wang, R. Yang, B. Cao, W. Xu, and Y. Lin, "DeLS-3D: Deep localization and segmentation with a 3D semantic map," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 5860–5869.

[2] C. You, C. Wen, C. Wang, J. Li, and A. Habib, "Joint 2-D–3-D traffic sign landmark data set for geo-localization using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2550–2565, Jul. 2019.

[3] R. Dubé *et al.*, "SegMap: Segment-based mapping and localization using data-driven descriptors," *Int. J. Robot. Res.*, Jul. 2019, doi: 10.1177/0278364919863090.

[4] C. Ye, J. Li, H. Jiang, H. Zhao, L. Ma, and M. Chapman, "Semi-automated generation of road transition lines using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[5] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York, NY, USA: Van Nostrand Reinhold, 1992.

[6] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps—Summarizing experience for lifelong localization," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 2767–2773.

[7] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary maps for lifelong visual localization," *J. Field Robot.*, vol. 33, no. 5, pp. 561–590, 2016.

[8] H. S. Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen, "3D point cloud reduction using mixed-integer quadratic programming," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2013, pp. 229–236.

[9] W. Cheng, W. Lin, X. Zhang, M. Goesele, and M.-T. Sun, "A data-driven point cloud simplification framework for city-scale image-based localization," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 262–275, Jan. 2017.

[10] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. Conf. (RSS)*, vol. 2, Univ. California, Berkeley, Jul. 2014, p. 9.

[11] G. Tinchev, S. Nobili, and M. Fallon, "Seeing the wood for the trees: Reliable localization in urban and natural environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 8239–8246.

[12] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. New York, NY, USA: Springer, 2010, pp. 791–804.

[13] M. Havlena, W. Hartmann, and K. Schindler, "Optimal reduction of large image databases for location recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV)*, Jun. 2013, pp. 676–683.

[14] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep./Oct. 2015, pp. 2536–2542.

[15] T. Schneider *et al.*, "MAPLAB: An open framework for research in visual-inertial mapping and localization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1418–1425, Jul. 2018.

[16] M. Bürki, M. Dymczyk, I. Gilitschenski, C. Cadena, R. Siegwart, and J. Nieto, "Map management for efficient long-term visual localization in outdoor environments," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2018, pp. 682–688.

[17] M. Dymczyk, T. Schneider, I. Gilitschenski, R. Siegwart, and E. Stumm, "Erasing bad memories: Agent-side summarization for long-term mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4572–4579.

[18] H. Merzić, E. Stumm, M. Dymczyk, R. Siegwart, and I. Gilitschenski, "Map quality evaluation for visual localization," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 3200–3206.

[19] D. Van Opdenbosch, T. Aykut, N. Alt, and E. Steinbach, "Efficient map compression for collaborative visual SLAM," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 992–1000.

[20] D. Mongus and B. Žalik, "Efficient method for lossless LIDAR data compression," *Int. J. Remote Sens.*, vol. 32, no. 9, pp. 2507–2518, 2011.

[21] M. Isenburg, "LASzip: Lossless compression of LiDAR data," *Photogramm. Eng. Remote Sens.*, vol. 79, no. 2, pp. 209–217, 2013.

[22] D. M. Cole, A. Harrison, and P. M. Newman, "Using naturally salient regions for SLAM with 3D laser data," in *Proc. IEEE Int. Conf. Robot. Automat. Workshops (ICRAW)*, Apr. 2005, pp. 1–8.

[23] Y.-J. Lee, J.-B. Song, and J.-H. Choi, "Performance improvement of iterative closest point-based outdoor SLAM by rotation invariant descriptors of salient regions," *J. Intell. Robot. Syst.*, vol. 71, nos. 3–4, pp. 349–360, 2013.

[24] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1219–1230, 2012.

[25] Y. Wang, R. Xiong, Q. Li, and S. Huang, "Kullback-Leibler divergence based graph pruning in robotic feature mapping," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, 2013, pp. 32–37.

[26] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[27] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based ICP tuning for robust localization of a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 4721–4728.

[28] K. Genova and V. Guliashki, "Linear integer programming methods and approaches—A survey," *J. Cybern. Inf. Technol.*, vol. 11, no. 1, pp. 1–24, 2011.

[29] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[30] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2009, pp. 23–28.

[31] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[32] J. Lv, Y. Wang, K. Wu, G. Dissanayake, Y. Kobayashi, and R. Xiong, "Planar scan matching using incident angle," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4049–4056.

[33] F. Wu *et al.*, "Rapid localization and extraction of street light poles in mobile LiDAR point clouds: A supervoxel-based approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 2, pp. 292–305, Feb. 2017.

[34] S. Pan, H. Guan, Y. Yu, J. Li, and D. Peng, "A comparative land-cover classification feature study of learning algorithms: DBM, PCA, and RF using multispectral LiDAR data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 4, pp. 1314–1326, Apr. 2019.

[35] Y. Chen *et al.*, "Rapid urban roadside tree inventory using a mobile laser scanning system," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 9, pp. 3690–3700, Sep. 2019.

[36] L. Tang, Y. Wang, X. Ding, H. Yin, R. Xiong, and S. Huang, "Topological local-metric framework for mobile robots navigation: A long term perspective," *Auton. Robot.*, vol. 43, no. 1, pp. 197–211, Jan. 2018.

[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.

[38] Gurobi Optimization. (2015). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com

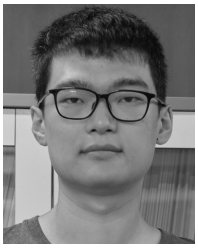
[39] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration," *J. Softw. Eng. Robot.*, vol. 3, no. 1, pp. 2–12, 2012.

[40] M. Wright and A. Ziegler, "Ranger: A fast implementation of random forests for high dimensional data in C++ and R," *J. Stat. Softw.*, vol. 77, no. 1, pp. 1–17, 2017. [Online]. Available: https://www.jstatsoft.org/v077/i01

[41] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2011, pp. 1–4.

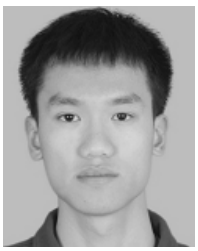
[42] E. Shtrom, G. Leifman, and A. Tal, "Saliency detection in large point sets," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3591–3598.

[43] I. B. Salah, S. Kramm, C. Demonceaux, and P. Vasseur, "Summarizing large scale 3D point cloud for navigation tasks," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Oct. 2017, pp. 1–8.



Huan Yin received the B.S. degree from the College of Biomedical Engineering and Instrument Science, Zhejiang University, Hangzhou, China, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Control Science and Engineering. His current research interests include intelligent vehicles and machine learning.

Yue Wang received the Ph.D. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2016. He is currently a Research Fellow with the Department of Control Science and Engineering, Zhejiang University. His latest research interests include mobile robotics and robot perception.



Li Tang received the B.S. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include vision based localization and autonomous navigation.

Xiaqing Ding received the B.S. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2016, where she is currently pursuing the M.S. degree. Her latest research interests include SLAM and visual based localization.

Shoudong Huang received the bachelor's and master's degrees in mathematics and the Ph.D. degree in automatic control from Northeastern University, China, in 1987, 1990, and 1998, respectively. He is currently an Associate Professor with the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. His research interests include nonlinear control systems and mobile robots simultaneous localization and mapping (SLAM), exploration and navigation.

Rong Xiong received the Ph.D. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2009. She is currently a Professor with the Department of Control Science and Engineering, Zhejiang University. Her latest research interests include motion planning and SLAM.