SkyLoc: Cross-modal Global Localization with a Sky-looking Fish-eye Camera and OpenStreetMap

Weixin Ma, Shoudong Huang, and Yuxiang Sun

***Abstract*—Global localization can estimate geo-referenced locations (e.g., longitude and latitude), which is a fundamental capability for autonomous vehicles. Most existing solutions rely on the Global Navigation Satellite Systems (GNSS). Their accuracy could be degraded by the multi-path effects or occlusions of GNSS signals in urban environments. Some GNSS-free methods could achieve global localization by comparing the current on-line sensory data with pre-built databases/maps. However, they require tedious human efforts to drive a vehicle to collect and maintain the databases/maps. Moreover, most of these methods use front-looking cameras or LiDARs, so the captured data could be easily contaminated by dynamic objects (e.g., moving vehicles and pedestrians). To provide a solution to these problems, this paper proposes a novel global localization method by comparing an image from a sky-looking fish-eye camera with the publicly available OpenStreetMap (OSM), and using particle filter to achieve real-time metric localization in dynamic traffic environments. To evaluate our method, we extend a public dataset with OSM data, which are retrieved through the given geo-referenced location information. Experimental results demonstrate the effectiveness and efficiency of our method.**

***Index Terms*—Cross-modal Localization, Sky-looking Camera, OpenStreetMap, GNSS-degraded Environments.**

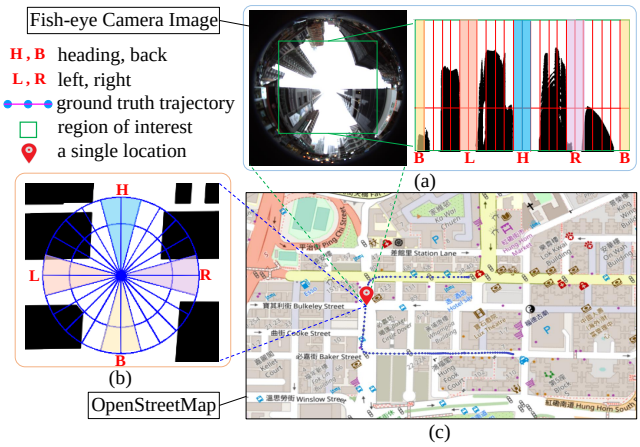

Fig. 1. A hybrid descriptor is used to extract topological and geometric information from both input fish-eye image and OSM at the same location. The ROI of fish-eye image is transformed to the polar coordinate as shown in (a), while operation for the ROI on OSM is based on the cylinder coordinate (b). (c) is the OSM with the trajectory of the vehicle.

I. Introduction

LOCALIZATION is a fundamental capability for autonomous vehicles [1]. It can provide location information for downstream tasks, such as decision making, path planning, and autonomous navigation [2], etc. There are mainly two types of localization for autonomous vehicles: global localization and local localization. Global localization aims to localize a vehicle against a geo-referenced database or map without initial guess [3]. Local localization aims to estimate relative poses with respect to previous poses or a small-size local map. For global localization, most of existing methods rely on the Global Navigation Satellite Systems (GNSS). However, GNSS is not always reliable or even sometimes unavailable due to occlusions or multi-path effects of GNSS signals [4], especially in dense urban environments, such as urban canyons. To improve the GNSS localization accuracy, 3-D city models and digital maps with environmental knowledge (e.g., street layouts and building heights) have been used to identify and use Non-Line-Of-Sight signals in some early works [5], [6].

To relieve the need of GNSS, place recognition and re-localization with visual cameras or LiDARs have been extensively studied in the research community [7]–[13]. These methods generally consist of an off-line stage and an on-line stage. During the off-line stage, a data-collection vehicle usually equipped with expensive global localization equipment is employed to build a geo-referenced image or point-cloud database/map. During the on-line stage, images or point clouds captured from vehicle-mounted sensors are compared with the geo-referenced database/map to determine the current location of the vehicle.

Despite the effectiveness of these methods, they still suffer from several issues. Firstly, building and maintaining the geo-referenced databases/maps is tedious, expensive, and time-consuming. Secondly, dynamic objects (e.g., vehicles and pedestrians) may cause discrepancies between the sensor data captured on-line and those stored in the pre-built databases/maps. Thirdly, in most existing vision-based methods, due to the limited field-of-view, the overlap between the image captured on-line and those stored in the database may not be large enough to accurately determine the locations.

To provide a solution to above issues, this paper proposes a novel cross-modal method using a sky-looking fish-eye camera as on-line observation and OSM as pre-built map, followed by a filter to achieve global localization in GNSS-degraded dynamic and complex urban traffic environments. Due to its

This work was supported by City University of Hong Kong under Grant 9610675. *(Corresponding author: Yuxiang Sun.)*

Weixin Ma is with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: weixin.ma@connect.polyu.hk).

Shoudong Huang is with the Robotics Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, New South Wales, Australia (e-mail: shoudong.huang@uts.edu.au)

Yuxiang Sun is with the Department of Mechanical Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong (e-mail: yx.sun@cityu.edu.hk, sun.yuxiang@outlook.com).

free and open nature, OSM has been used in many applications [14], [15]. One novelty of our method is that we innovatively use a sky-looking fish-eye camera for localization, which has the advantage of avoiding disturbances from dynamic objects and can get a large field-of-view. Moreover, we adopt the combination of on-vehicle visual information and OSM to relieve the need for GNSS to achieve global localization. Since we use OSM to build our database, the tedious database building work using a data-collection vehicle can be alleviated.

To the best of our knowledge, this is the first solution using a sky-looking fish-eye camera and OSM to achieve global localization. Our method can be easily integrated into existing localization systems (e.g., LiDAR, radar, or front-looking camera based methods) to enhance their robustness by providing redundancy from different sensory sources (i.e., sky-looking Fish-eye camera). To evaluate our method, we extend an existing public dataset [16] with OSM data retrieved through the given geo-referenced location information. Our code is open-sourced[1]. The contributions of this work are summarized as follows:

1) We propose a novel cross-modal global localization method with a sky-looking fish-eye camera and OSM data for dynamic and complex urban environments.
2) We design a novel topology-geometry based hybrid (TGH) descriptor to represent both the fish-eye image and OSM data to narrow the modality gap.
3) We test our method on different computing platforms including PC and embedded devices to demonstrate the real-time efficiency of our method.

This paper is structured as follows. Section II reviews the related work. Section III describes our descriptor extraction and weighting model in detail. Experiments and results are presented in section IV. Conclusions and future work are drawn in the last section.

II. Related Work

This section reviews two streams of cross-modal global localization methods using public aerial data. The first stream adopts the *Ground-to-Aerial* matching framework. *Ground* means that the on-line data are captured by an on-vehicle sensor (e.g., camera or LiDAR). *Aerial* means that the off-line database is built with geo-referenced aerial or satellite images. The second stream adopts the *Ground-to-OSM* framework. The key problem in cross-modal localization lies in how to bridge the modality gap.

A. Ground-to-Aerial Matching

1) Vision-based Methods: Some early methods use hand-crafted image features, such as key points, lines, and planes, to match images captured by an on-vehicle camera with the images from a geo-referenced aerial-image database [17]–[20]. These methods rely on geometric information and features to build their descriptors for matching. Differently, Noha *et al.* [21] used semantic-level textual information (i.e., shop, restaurant, or street names) to match camera images with a Google Map. Instead of using hand-crafted features, Kim *et al.* [22] used a Siamese network to learn embeddings from a ground-image sequence and satellite images, which are then used to update particle weights during filtering. Similarly, Hu *et al.* [23] also used a Siamese network to measure the similarity between ground and aerial images. The location of the query on-vehicle image is provided by retrieving aerial images stored in a database. Then, the authors extended [23] with a Markov localization framework to ensure the temporal consistency of the matching results [24]. Different from CNN-based methods, TransGeo [25] is a pure transformer-based approach, eliminating the need for aligned image pairs during training. TransGeo exhibits good flexibility and generalization but, as a retrieval-based method, requires additional registration to determine the vehicle's relative pose to the aerial image.

2) Point cloud-based Methods: Range sensors, compared to visual sensors, are more robust to illumination and weather changes [26]. They can also provide accurate depth measurements. However, the modality gap between range sensor data and aerial images is much larger. To narrow such modality gap, Kummerle *et al.* [27] extracted edge points for both a satellite image and point clouds provided by a 2-D laser scanner, which are then used to estimate the position of a ground robot. RSLNet [28] and its extension [29] all use synthetic radar images generated from satellite images as the intermediate modality for matching and network training. Self-supervised learning technique is used in [29] to release the need of ground truth required in [28]. Unlike the above methods that directly use point clouds, some researchers extract high-level representations for matching. Hussein *et al.* [30] used a LiDAR to scan tree stems and matched them with tree crowns captured in an aerial image to localize a ground robot. Miller *et al.* [31] extracted semantic information for both a ground LiDAR point cloud and a satellite image to calculate their similarity, which is then used to update particle weights during filtering.

B. Ground-to-OSM Matching

1) Vision-based Methods: Some early works achieve global localization by comparing past trajectories of odometry with road routes in OSM. For example, Floros *et al.* [32] combined trajectory of an odometry with road information of OSM by using champfer matching, which shows a 5m average localization error in their results. Similarly, Bruaker *et al.* [33] also localized a vehicle by matching the trajectory of an odometry with road topological information from a OSM, while a probabilistic framework is proposed for matching. Instead of directly using road information, Panphattarasap *et al.* [34] proposed a binary semantic descriptor to represent road junctions and gaps between surrounding buildings for both images and OMS. Multiple descriptors of consecutive input images are then combined together to improve the retrieval performance. Based on [34], Noe *et al.* [35] used a triplet loss to learn embeddings for both surrounding images and OSM instead of extracting binary semantic descriptors. Similarly, Zhou *et al.* [36] used the same approach to extract descriptors for both vehicle-captured images and OSM, followed by a particle filter for global localization instead of the retrieval method used in [35]. Recently, Sarlin *et al.* [37] propose OrienterNet, a deep neural network for sub-meter image localization within OSM. It

[1]https://github.com/lab-sun/SkyLoc

requires a coarse GPS prior to build a local map from OSM and may struggle when the query image includes many unregistered elements in OSM, e.g., pedestrians or vehicles, limiting its suitability for some highly dynamic urban scenarios.

2) Point cloud-based Methods: Similar to some vision-based methods, Ruchti *et al.* [38] and Suger *et al.* [39] both match LiDAR odometry trajectories to OSM road information, focusing on urban environments and outer-urban environments, respectively. Unlike [38] and [39] which focus on geometric information, [40] and [41] first extract semantic information of the surroundings. A 4-bit semantic descriptor and a scan-context based OSM descriptor were proposed in [40] and [41] for matching, respectively. Global localization is then achieved using particle filter and data retrieval in [40] and [41], respectively. A localization error at about 20m on KITTI sequence-00 was reported in [40].

C. Difference from Previous Work

Similarly, our method also uses OSM as the reference map. The major difference is that we use a low-cost sky-looking camera for the observation model, which can avoid the interference from dynamic objects on roads, while the other methods ([40] uses a LiDAR, [36] uses four cameras, [25] uses a panoramic camera, [37] uses a front-looking camera) may need to tackle the interference from dynamic objects on the street. In [40], a compact binary semantic descriptor (BSD) that captures the topological information (i.e., junction type) is used to bridge the modality gap between on-board measurement and OSM. Based on this compact representation, we simultaneously use geometric information (i.e., building outline) to further narrow the modality gap between sky-looking fish-eye camera images and OSM. Lastly, different from [25], [31], [36], [37], [40], which may require GPUs to achieve better efficiency, our method can directly run on a GPU-free computing platform.

III. The Proposed Method

Given a sky-looking fish-eye camera image I_F at time t and a street block-sized geo-referenced OSM M_{osm}, our goal is to estimate the vehicle pose $\mathbf{x}_t$ with respect to M_{osm}. Note that M_{osm} is obtained by re-rendering the original OSM file (see Fig. 1(c)) to retain only building areas, which are rendered in black, to facilitate descriptor extraction (see Fig. 4). We assume that the vehicle runs on a flat road, so the vehicle pose consists of three variables: 2-D coordinates (x, y) and orientation θ.

A. Problem Formulation

As aforementioned, we use particle filter to achieve metric localization. The key idea of particle filter is to use a number of particles to estimate the posterior probability of the vehicle pose. The optimal pose estimation can be achieved through the expectation of the posterior probability distribution. Mathematically, given all observations from time 1 to time t, $\mathbf{z}_{1:t}$, and all the motion control inputs, $\mathbf{u}_{1:t}$, the posterior probability $p\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$ can be calculated as:

$$p\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta p\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \int p\left(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}\right) p\left(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}\right) d\mathbf{x}_{t-1}, \tag{1}$$

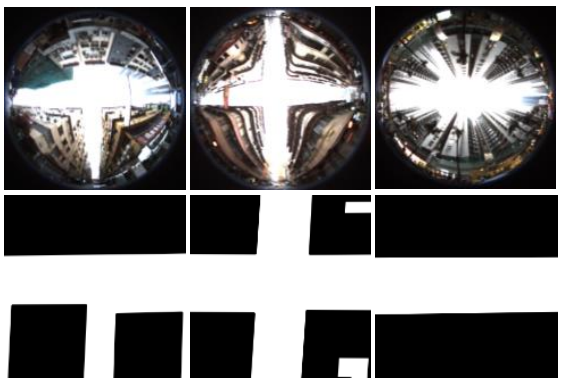

Fig. 2. Fish-eye camera images with typical sky shapes (top row) and the corresponded local area in the re-rendered M_{osm} (bottom row).

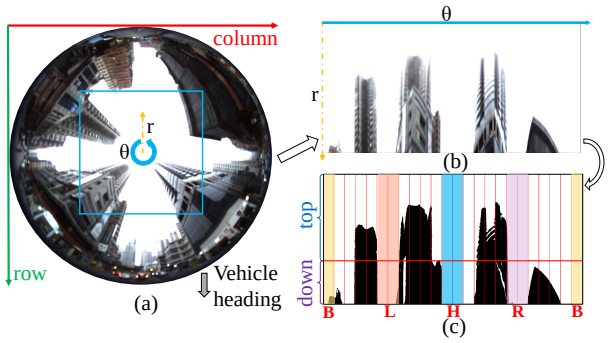

Fig. 3. Descriptor extraction for input fish-eye image I_F. The ROI of I_F is expanded into a polar coordinate and binarized to get the sky-looking mask. The final descriptor is extracted based on the sky-pixel ratios for the divided bins of the sky-looking mask.

where η is a normalization constant, $p\left(\mathbf{z}_t \mid \mathbf{x}_t\right)$ represents the likelihood from the observation at pose $\mathbf{x}_t$, the integral term is the prior probability of the pose [42]. Here we use a LiDAR odometry algorithm, LOAM [43] as our control input $\mathbf{u}_t$. Note that other motion estimation algorithms that can provide absolute scales can also be used as the control input here.

We use the KLD-sampling algorithm [44] to speed up the process of weight calculation for particles by adaptively adjusting the number of resampled particles during filtering. The key idea is using Kullback-Leibler divergence (KLD) to calculate how many particles are needed to approximate the distribution of the current vehicle pose:

$$\begin{aligned} n &= \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \\ &= \frac{k-1}{2\epsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3, \end{aligned} \tag{2}$$

where n is the number of the needed particles, $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard Gaussian distribution, ϵ is the upper error bound in KLD, and k is the number of bins that contain at least one particle. In our case, k represents the number of square-shape ($W_{bin} \times W_{bin}$, where W_{bin} is the length of sides in terms of pixel) bins that contain at least one particle on M_{osm}.

Algorithm 1: TGH Descriptor Extraction

Input: Sky-looking fish-eye image I_F
Output: TGH Descriptor Des_{TGH}

begin
  Get the ROI of I_F and horizontally expand it, noted as I_H
  Classify pixels in I_H as sky or building using binarization operation, noted as I_B
  Divide I_B into 24 bins, $\{\text{Bin}_k\}$, $k \in \{1, 2..., 24\}$
  for *each* Bin_k **do**
    Count sky pixels in top part, get $\gamma(\text{Bin}_k^T)$
    Count sky pixels in down part, get $\gamma(\text{Bin}_k^D)$
  end
  $Des_{\text{geo\_T}} \leftarrow \{\gamma(\text{Bin}_k^T)\}$, $Des_{\text{geo\_D}} \leftarrow \{\gamma(\text{Bin}_k^D)\}$
  $Des_{\text{geo}} \leftarrow \{Des_{\text{geo\_T}}, Des_{\text{geo\_D}}\}$
  for *each* bit_i **do**
    Find two corresponding bins, Bin_{k1} and Bin_{k2}
    if $(\gamma(Bin_{k1}^T) \geq \tau_1^T$ & $\gamma(Bin_{k1}^D) \geq \tau_1^D)$ & $(\gamma(Bin_{k2}^T) \geq \tau_1^T$ & $\gamma(Bin_{k2}^D) \geq \tau_1^D)$ **then**
      $\text{bit}_i = 1$
    end
    else if $(\gamma(Bin_{k1}^T) \geq \tau_2^T$ & $\gamma(Bin_{k1}^D) \geq \tau_2^D) \parallel (\gamma(Bin_{k2}^T) \geq \tau_2^T$ & $\gamma(Bin_{k2}^D) \geq \tau_2^D)$ **then**
      $\text{bit}_i = 1$
    end
    else
      $\text{bit}_i = 0$
    end
  end
  $Des_{\text{topo}} \leftarrow \{\text{bit}_i\}, i \in \{1, 2, 3, 4\}$
  return $Des_{\text{TGH}} \leftarrow \{Des_{\text{geo}}, Des_{\text{topo}}\}$
end

B. The Proposed TGH Descriptors

To associate I_F with M_{osm}, we measure the similarity between the on-line captured I_F and a particle-centered patch in M_{osm}. However, directly comparing I_F and M_{osm} is challenging due to the significant modality gap between fish-eye camera images and OSM. Interestingly, I_F and M_{osm} both encode the structural information about roads and buildings, as shown in Fig. 2. This shifts the problem to: *How can such information be embedded?* For road structures, binary semantic descriptor has shown its efficiency to embed junction types [34], [40]. For buildings, I_F and M_{osm} share similar geometric outlines, but describing such outlines with a vector-based descriptor is nontrivial. Instead, we represent this information implicitly using the ratio of non-building areas, which can be easily extracted from both I_F and M_{osm}. To keep the method intuitive and efficient, we adopt a handcrafted descriptor (i.e., TGH descriptor) rather than a neural network, which usually requires a large amount of paired training data and additional GPU resources for inference.

1) Sky-looking Mask and Descriptor Extraction: Given a I_F, we first extract a square-shaped region-of-interest (ROI) at the center of I_F instead of using all pixels. In this way, pixels from roads, pedestrians, and vehicles can be removed. Only the top parts of buildings and the sky are kept. This could make the structure appearance of I_F more close to M_{osm}. We empirically set the size of the ROI as 900×900 pixels, which can keep enough information while maintain a good efficiency. The ROI is then expanded horizontally to correct the image distortion, followed by grayscale conversion. Otsu's method [45] is then used to binarize the ROI with adaptive threshold, assigning a value of 255 to sky pixels and 0 to building pixels. In this way, we can get a binary sky-looking mask (see Fig. 3(c)).

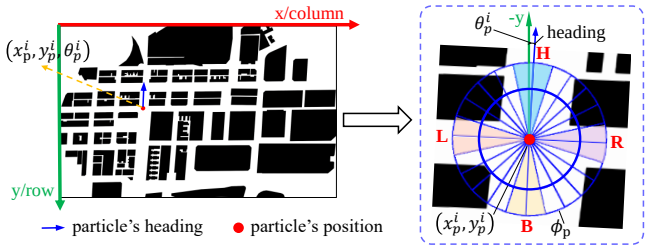

Fig. 4. Example of descriptor extraction for a given particle on OSM. A cylinder coordinate is adapted on the OSM to get the ROI. Then the final descriptor can be obtained as the same way for fish-eye image.

Similar to the binary semantic descriptors in [40], we design our descriptor by first dividing the sky-looking mask into 24 bins along the column direction, in which each bin covers $15°$ out of $360°$. Each bin is then divided into 2 parts, that is, the top part (i.e., the first $2/3$ rows of the mask) and the down part (i.e., the last $1/3$ rows of the mask), see Fig. 3. For geometric information, we count the sky pixels (i.e., non-building areas) in both top and down parts in each bin to respectively calculate the sky-pixel ratios, denoted as $\gamma(\text{Bin}_k^T)$ and $\gamma(\text{Bin}_k^D)$, respectively. Bin_k represents the k-th bin, T is short for the top part, and D is short for the down part. The descriptor vector of the geometric information can be obtained as Des_{geo}=[$\gamma(\text{Bin}_1^T), ..., \gamma(\text{Bin}_{24}^T), \gamma(\text{Bin}_1^D), ..., \gamma(\text{Bin}_{24}^D)$], where Des and geo is short for descriptor, and geometric, respectively. Des_{geo} here implicitly embed the building outliers from I_F to some extent. For topological information, we design a 4-bit binary descriptor Des_{topo} to indicate the presence of a road in the vehicle's heading direction(H), back direction(B), left direction(L), and right direction(R). Each bit covers two bins of the mask, as shown in Fig. 3(c). If the sky pixel ratios of such two bins satisfy conditions listed in Algorithm 1 (see thresholds setting in Tab. I), the corresponding bit of Des_{topo} is set as 1 (i.e., a road exists in this direction), otherwise 0. Let bit_i denotes the i-th bit of Des_{topo}, where $i \in \{1, 2, 3, 4\}$ follows the $heading\text{-}back\text{-}left\text{-}right$ order. The whole process of the TGH descriptor extraction is shown in Algorithm 1.

2) OSM ROI and Descriptor Extraction: Given a particle par^i with pose $(x_p^i, y_p^i, \theta_p^i)$, where i represents i-th particle, we first extract a circle-shaped ROI centered at (x_p^i, y_p^i) with a diameter of ϕ_p (see IV). We then orient the ROI according θ_p^i before the descriptor extraction (see Fig. 4). Similarly, we convert the OSM ROI as grayscale and then binarize it, where non-building pixels approximately correspond to the sky pixels in the sky-looking mask.

Similar to the TGH descriptor for I_F, we divide the OSM ROI into 24 bins based on a cylinder coordinate, where each bin covers $15°$ out of $360°$. Each bin is then further divided

TABLE I
PARAMETERS USED IN TGH DESCRIPTOR AND KLD-SAMPLING

Thresholds for Des^F_{topo} / $Des^{par^i}_{\text{topo}}$				Weights for Des_{TGH}				KLD-sampling		
τ_1^T	τ_1^D	τ_2^T	τ_2^D	λ	ω	α	β	ϵ	δ	W_{bin}
0.7 / 0.7	0.7 / 0.6	0.8 / 0.8	0.8 / 0.8	0.4	0.6	0.6	0.4	0.15	0.1	25

into center part (inner $2/3$ radius) and marginal part (outer $1/3$ radius), as shown in Fig. 4. Here we do not expand the ROI horizontally, since there is not distortion in M_{osm} and circle-shaped ROI is much more close to the raw I_F. Then we count pixels belonged to the non-building area within both center part and marginal part in each bin to calculate the "sky"-pixel ratio, denoted as $\gamma(\text{Bin}_k^C)$ and $\gamma(\text{Bin}_k^M)$, respectively. C is short for center and M is short for marginal. The geometry part $Des^{par^i}_{\text{geo}}$ and topology part $Des^{par^i}_{\text{topo}}$ of $Des^{par^i}_{\text{TGH}}$ for particle par^i are extracted by following the same process in Algorithm 1 (i.e., line 5-26). However, different thresholds are chosen, considering the gap between polar and cylinder coordinate. More details can be found in Tab. I.

C. Observation/Weighting Model

The observation, or weighting model in particle filter is used to update particle weights during the filtering process. A higher weight indicates that the live sensor input data is much more likely captured when the robot is under a state as like the given particle. In our case, such similarity/weight is computed by the distance between descriptors extracted from I_F and M_{osm}.

Given a TGH descriptor $Des^F_{\text{TGH}} = \{Des^F_{\text{geo}}, Des^F_{\text{topo}}\}$ of I_F and $Des^{par^i}_{\text{TGH}} = \{Des^{par^i}_{\text{geo}}, Des^{par^i}_{\text{topo}}\}$ of particle par^i at time t, the final weight for par^i is the combination of two parts, including topology similarity $w^{par^i}_{topo}$ and geometry similarity $w^{par^i}_{geo}$. Similar to [40] and [36], we use Hamming distance d_{ham} to calculate the topology similarity as follow:

$$w^{par^i}_{topo} = 1 - 0.2 d_{ham}(Des^F_{\text{topo}}, Des^{par^i}_{\text{topo}}), \tag{3}$$

The reason for choosing 0.2 as the factor is to make sure those particles with higher Hamming distance can be still kept in a certain probability after re-sampling. As for the geometry similarity, we use *cos* distance d_{cos}, which is widely adapted to evaluate the similarity between two distributions. Des^F_{geo} and $Des^{par^i}_{\text{geo}}$ can be regarded as distributions of sky or building pixels. Specifically, we first calculate *cos* distance between the top part $Des^F_{\text{geo\_T}}$ of Des^F_{TGH} and the center part $Des^{par^i}_{\text{geo\_C}}$ of $Des^{par^i}_{\text{TGH}}$. Similarly, we then calculate the *cos* distance between the corresponding down part and marginal part. To reduce the gap caused by the difference of bin size and the bin shape used for extracting Des^F_{TGH} and $Des^{par^i}_{\text{TGH}}$, we combine these two *cos* distances by weighting factors. The final geometry similarity $w^{par^i}_{geo}$ can be obtained as follow:

$$w^{par^i}_{geo} = \lambda d_{cos}(Des^F_{\text{geo\_T}}, Des^{par^i}_{\text{geo\_C}}) + \omega d_{cos}(Des^F_{\text{geo\_D}}, Des^{par^i}_{\text{geo\_M}}), \tag{4}$$

where λ and ω are weights satisfying $\lambda + \omega = 1$. The final weight of particle par^i can be obtained as follow:

$$w^{par^i} = \alpha w^{par^i}_{topo} + \beta w^{par^i}_{geo}, \tag{5}$$

where α and β are weights satisfying $\alpha + \beta = 1$. More details about values of weights are shown in Tab. I.

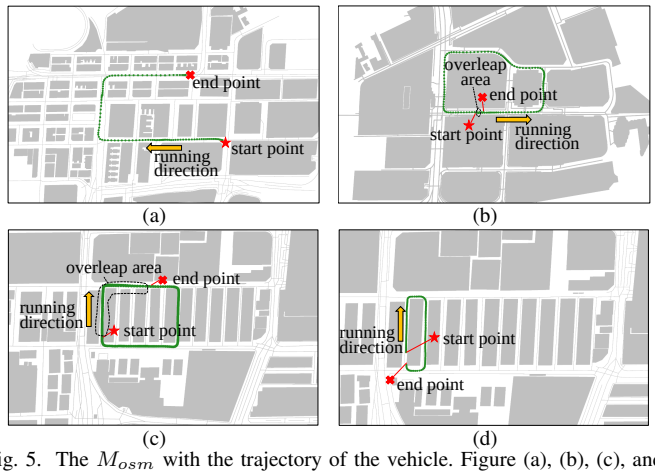

Fig. 5. The M_{osm} with the trajectory of the vehicle. Figure (a), (b), (c), and (d) corresponds to Seq-01, 02, 03, and 04 without expansion, whose length is 562.5m, 723.9m, 657.0m, and 232.5m, respectively.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. The Dataset

We build our dataset based on the UrbanLoco dataset [16] by supplementing the OSM data. The UrbanLoco dataset contains 13 sequences collected in San Francisco and Hong Kong, covering a total length of over 40 km. There are front-looking camera images, LiDAR point clouds, and RTK GNSS poses recorded in each sequence. However, only four sequences have sky-looking fish-eye camera images, including HK-Data20190426-1, HK-Data20190426-2, HK-Data20190316-1, and HK-Data20190316-2. These four sequences are collected in urban canyon areas (i.e., the Whampooa and Ma Tau Kok areas in Hong Kong), as shown in Fig. 5. We make our dataset using the aforementioned four sequences. We sample the fish-eye images at the GNSS collection rate (i.e., 1 Hz) to get the ground truth pose. Since the down-sampled sequences are too short, we reverse the sequence (i.e., seems like reversing the car) at the end of the original sequence. In this way, the sequence is expanded twice long, renamed as Seq-01, 02, 03, and 04, respectively, as shown in Fig. 7. Note that RTK-GNSS trajectories of Seq-03 and 04 are not accurate enough due to severe signal occlusions along the data collection route. So, we refine their ground truth trajectories based on the relative trajectories estimated by LOAM. We understand that the evaluated sequences are limited, and more data will be valuable to better assess the generalizability of SkyLoc.

B. Experimental Setup

We assume that vehicles always run on roads and follow the traffic rules (e.g., cannot make u-turns in the middle of roads). Thus, we evenly distribute $40,000$ particles on the road area, and constrain the particle orientation θ_p within a range with a tolerance φ along the road orientation to reduce the number of particles that have invalid poses (e.g., particles in the middle of the road but with a $90°$ heading direction). So, given a road with $a°$ orientation, the orientations of particles within this road should satisfy the requirement: $\theta_p \in [a - \frac{\varphi}{2}, a + \frac{\varphi}{2}]$ or $[a + 180 - \frac{\varphi}{2}, a + 180 + \frac{\varphi}{2}]$. The a is calculated with the two junctions at the two ends of the

road. In this work, we set φ as 30° and evenly distribute the θ_p with a resolution of 1°. We use a larger ROI of I_F for Seq-02, i.e., 1300×1300 pixels. We set ϕ_p as around 50m for Seq-01 and 02, and around 25m for Seq-03 and 04 because the building density is much higher in these two sequences. Besides, $W_{bin} = 15$ is used for Seq-03 and 04 to better estimate the distribution of sampled particles in narrow street areas. To achieve real-time performance, we pre-extract descriptors from the given M_{osm} to build a database, which takes about 20s for $200,000$ particles on an intel core i7 computer. During filtering processes, $Des^{par^i}_{\text{TGH}}$ can be approximated by the descriptor that is extracted at the closest location to par^i in the database. A KD-search tree is used to accelerate searching. The descriptor for the closest point in the database might have a different orientation from the particle par^i. So, we further align the retrieved descriptor with par^i according to θ^i_p by sifting the column of the retrieved descriptor. The more data points in the database are sampled, the more accurate such approximation would be. Empirically, a larger map M_{osm} usually needs more data points to build such a database.

C. Performance Evaluation

1) Baselines: Since work [36], [40] are not open-sourced, we are unable to directly compare with them. So, we create some baselines for comparison. The work [40] extracts BSD descriptors for both M_{osm} and on-line LiDAR scans to update particle weights during filtering. Similarly, we use road junctions on the heading, back, right, and left of the vehicle to build the BSD descriptor. In the first baseline, LOAM is used to provide pose estimation to update the control input $\mathbf{u}_t$. A perfect BSD-based observation model is used. Specifically, we first find the corresponding coordinates (i.e., x, y, θ) on OSM according to the ground truth poses of the vehicle. Then BSD descriptors extracted at these positions from the OSM are used as on-line observations. So, the BSD descriptors for the on-line observations are exactly the same as those for the particles whose locations are identical to the on-line sensor in the OSM. We name this baseline as the BSD-based method (`BSD`). We use this baseline as the upper limit for the BSD-based method when using a practical motion model. As for the second baseline, we only set particle weights as 0 when particles are outside the road area, namely, the Road Net-based method (`RN`). Motion estimation from LOAM is used to update $\mathbf{u}_t$. This baseline only uses road information in its observation model like [32]. In the third baseline, we replace the original observation model in our method with a perfect observation model similar to the first baseline `BSD`. So, the TGH descriptors for the on-line observations are exactly the same as those for the particles whose locations are identical to the on-line sensor in the OSM. Such baseline is noted as ours with Perfect Observation model (`ours-PO`), which refers to the upper limit for the performance of the TGH-based method when using a practical motion mode.

We run our method and three baselines on Seq-01, 02, 03, 04, and their reversed versions, totally eight sequences. We run ten times of all the methods on each sequence. The filtering process is considered as converged when the standard deviation of all the current particles pose are less than a pre-defined threshold.

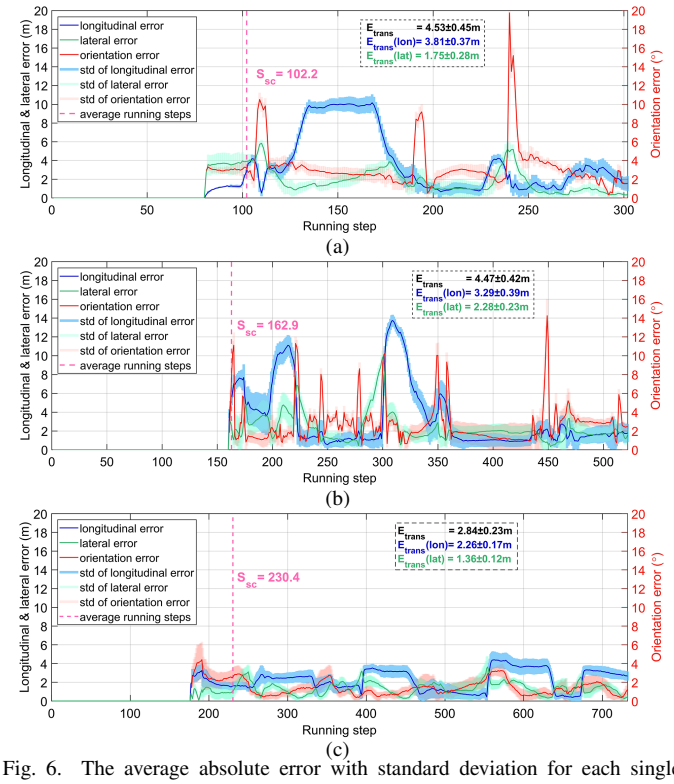

Fig. 6. The average absolute error with standard deviation for each single running step after successful convergence. The first, middle, and bottom row represents results for Seq-01, 02, and 03, respectively.

Here, we set the standard deviation as 40 $pixel$ (around 6m) for both x and y, as well as 10° for θ. Some localization results in one runtime are as shown in Fig. 7.

2) Localization Accuracy and Convergence: We use the successful convergence rate $P_{sc} = |L|/n_{test}$ and average running steps to achieve successful convergence, $S_{sc} = \frac{1}{|L|}\sum_{l\in L} s^l$, to evaluate the performance of convergence. $|L|$ is the cardinality of L, in which $L = \{l\}$ is the set of testing runs that can achieve successful convergence. n_{test} represents total testing times. s^l is the number of running steps for successful convergence in l-th testing run. Once a filtering process is successfully converged, we calculate the average error for translation and orientation as $E_{trans} = \frac{1}{N-s^l}\sum_{j=s^l}^{N}\sqrt{\left(\tilde{x}^l_j - x_j\right)^2 + \left(\tilde{y}^l_j - y_j\right)^2}$ and $E_{ori} = \frac{1}{N-s^l}\sum_{j=s^l}^{N}\left|\overleftarrow{\mathbb{R}}\left(\mathbb{R}(\tilde{\theta}^l_j)^{-1}\mathbb{R}(\theta_j)\right)\right|$. j refers to j-th frame. $(\tilde{x}^l_j, \tilde{y}^l_j, \tilde{\theta}^l_j)$ is the estimation pose of the vehicle at frame j in l-th testing run. (x_j, y_j, θ_j) is the ground-truth pose. N is the frame number of the testing sequence. $\mathbb{R}(\cdot) \in SO(2)$ is the rotation matrix of the given orientation. $\overleftarrow{\mathbb{R}}(\cdot)$ is the orientation of the given 2D rotation matrix. The average E_{trans} and average E_{ori} with standard deviations are then calculated based on all the successful convergence filtering processes, which are used to evaluate the localization accuracy. As shown in Tab. II, our method can achieve similar localization accuracy as `BSD`. The values of average E_{trans} for all the successful converged sequences are less than 4.6m. The values of average E_{ori} are less than 3.3°. When using our method on all sequences except sequence 02-reverse, the average running steps for successful convergence are smaller compared to `BSD`. This indicates that our method can successfully converge faster than `BSD`.

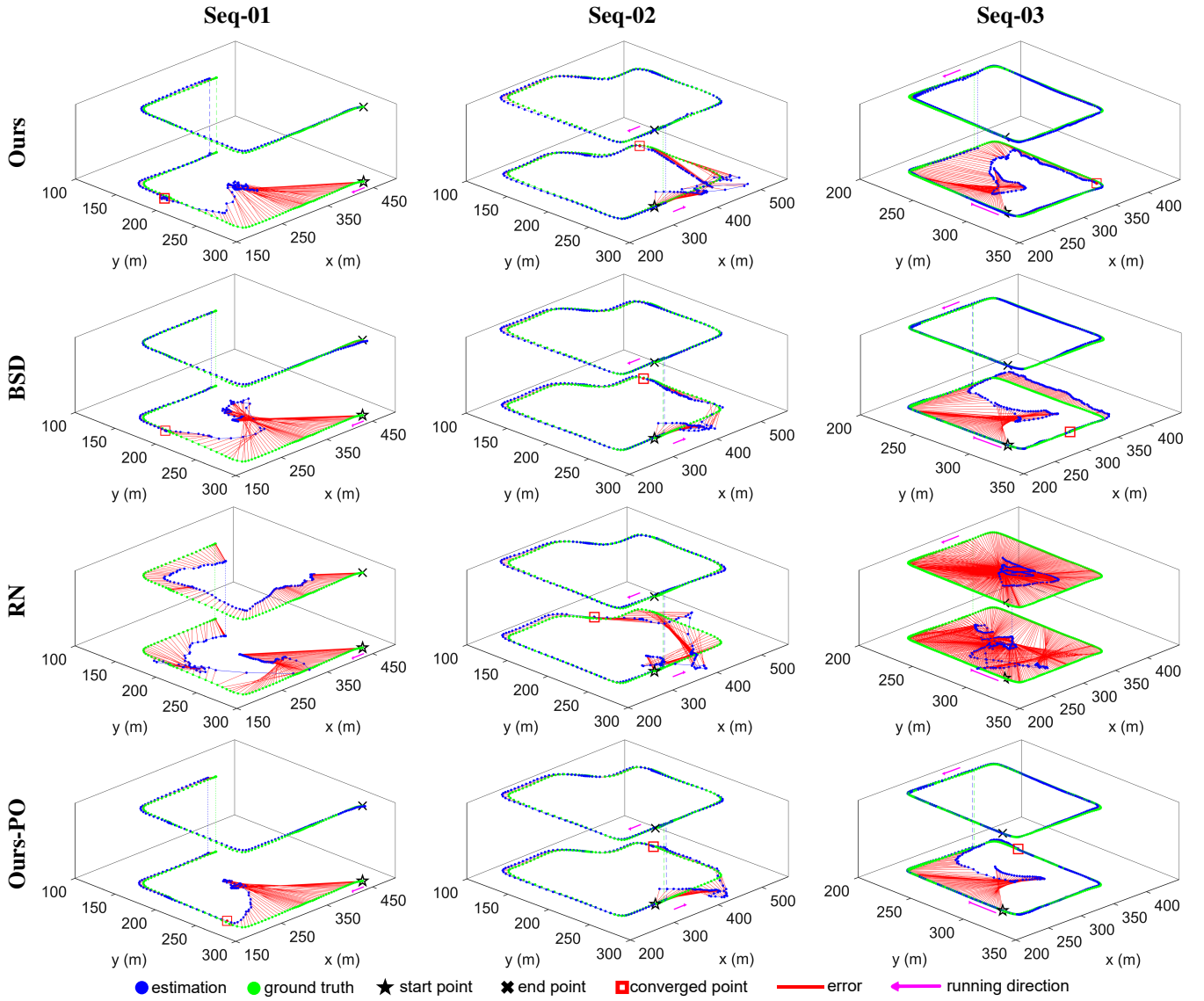

Fig. 7. Examples for qualitative performance of localization. Figures in each row are based on the same method. In each picture, the top trajectory is the expanded part, and the dotted line connects the place where we reverse the sequence (see Part A of section IV).

Our method can also achieve a high successful convergence rate. Indeed, `BSD` has a perfect observation model without noise, which means such results are very difficult or even impossible to be obtained when using practical sensors. It is worth noting that our method outperforms all the other baselines when we using a perfect TGH-based observation model from the perspective of convergence performance with high competitive localization accuracy. This shows that our TGH-based observation model has a higher upper limit than the BSD-based observation model. We interestingly find that `RN` baseline can only localize the vehicle on the Seq-02 and 02-reverse with larger S_{sc}. We consider RN-based method as a kind of "lazy" particle filter, which treats every particle equally. It only drops particles when they are not on roads. The reason why such "lazy" filter can successfully converge could be that the trajectory of the running vehicle is unique in the M_{osm}, while this is not common in urban environments. We also observe that in challenging scenarios, successful convergence can be difficult to achieve, leading to failures. For instance, on the Seq-04, only `ours-PO` manages to converge successfully with a P_{sc} value of 0.2. Moreover, our method and all the other baselines are unable to successfully converge on Seq-04-reverse. We conjecture the primary reason is the lack of enough consecutive and distinguished observations. The junction types, building layouts, structures, and even appearances are highly similar and repeated in Seq-03 and Seq-04. In cases with insufficient observations, such as Seq-04 and Seq-04-reverse, our method fails. While when more consecutive observations are available, as in Seq-03 and 03-reverse, our method still able to successfully converge even in such challenging environments. To enhance the performance in challenging scenarios with limited observations. We believe that incorporating features such as street signs and shop names captured in images and registered in OSM provide promising potentials.

We also visualize the average absolute errors of localization, including $\epsilon_j(\text{trans}) = \frac{1}{|L|}\sum_{l\in L}\sqrt{\left(\tilde{x}_j^l - x_j\right)^2 + \left(\tilde{y}_j^l - y_j\right)^2}$ and $\epsilon_j(\text{ori}) = \frac{1}{|L|}\sum_{l\in L}\left|\mathbb{R}\left(\mathbb{R}(\tilde{\theta}_j^l)^{-1}\mathbb{R}(\theta_j)\right)\right|$ for each running

TABLE II
AVERAGE E_{trans} AND AVERAGE E_{ori} WITH STANDARD DEVIATION, AVERAGE RUNNING STEPS FOR SUCCESSFUL CONVERGENCE S_{sc}, AND SUCCESSFUL CONVERGENCE RATE P_{sc} IN URBANLOCO DATASET.

Metrics	Method	Sequence							
		01	01-reverse	02	02-reverse	03	03-reverse	04	04-reverse
E_{trans}	ours	4.53±0.45	3.98±0.38	4.47±0.42	3.66±0.66	2.84±0.23	2.84±0.26	N/A	N/A
	BSD	3.84±0.25	3.16±0.22	3.17±0.17	2.98±0.42	1.21±0.14	1.47±0.83	N/A	N/A
	RN	N/A	N/A	3.93±0.41	3.01±0.68	N/A	N/A	N/A	N/A
	ours-PO	**3.75±0.31**	**2.98±0.28**	**2.56±0.21**	**2.45±0.32**	**0.73±0.11**	**0.69±0.09**	**0.23±0.05**	N/A
E_{ori}	ours	**3.29±0.32**	**2.87±0.22**	**2.76±0.37**	**2.74±0.32**	1.40±0.19	1.32±0.24	N/A	N/A
	BSD	3.58±0.31	3.05±0.17	3.33±0.34	3.34±0.29	0.83±0.14	0.87±0.10	N/A	N/A
	RN	N/A	N/A	3.49±0.49	3.54±0.53	N/A	N/A	N/A	N/A
	ours-PO	3.37±0.30	3.00±0.23	2.91±0.23	3.03±0.32	**0.68±0.11**	**0.53±0.09**	**0.34±0.14**	N/A
P_{sc}	ours	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.8	0.0	0.0
	BSD	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.8	0.0	0.0
	RN	0.0	0.0	**1.0**	**1.0**	0.0	0.0	0.0	0.0
	ours-PO	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.9**	**0.2**	0.0
S_{sc}	ours	102.2	102.6	162.9	61.3	230.4	**253.6**	N/A	N/A
	BSD	104.8	104.1	163.2	55.9	241.2	263.9	N/A	N/A
	RN	N/A	N/A	207.0	84.3	N/A	N/A	N/A	N/A
	ours-PO	**101.4**	**95.0**	**135.8**	**47.0**	**189.4**	254.6	**287.0**	N/A

E_{trans}(unit: m) and E_{ori}(unit: °) refer to average E_{trans} and average E_{ori} here, respectively. "N/A" represents not-converged or failed-converged cases. The best results for different metrics on each sequence are highlighted in bold font.

step-j once the filtering is converged, as well as their standard deviations. The ϵ_j(trans) is divided into longitudinal (that is, along the head of the vehicle) and lateral errors (i.e., perpendicular to the heading of the vehicle). As shown in Fig. 6, our method can provide a good estimation for both translation and orientation, where the errors are respectively less than 10m and 20° in most of the single running step. We can also observe that the longitudinal errors are generally larger than the lateral errors. We conjecture that the primary reason lies in the road width being significantly smaller than its length. The field of view in the lateral direction of the road is more sensitive than in the longitudinal direction, making fish-eye images captured at different locations across the road width more distinguishable. Therefore, it would be better to set different uncertainties for longitudinal and lateral estimations when using our method in navigation or localization systems.

3) Dead-reckoning Distance Traveled before Successful Localization: Similar to [46], we calculate the probability of travelling a given distance x without successful localization in the target map, $P(x)$. In [46], it follows a key-frame-based place recognition manner, while ours follows a filtering manner. Therefore, successful localization in our case can only occur after the filtering process has converged. To determine whether successful localization has occurred, we calculate Relative Translation Error (RTE) and Relative Rotation Error (RRE) for each step after successful convergence (more details about RTE and RRE can be found in [3]). Localization is considered successful when RTE<7.5m and RRE<10°. For each test, a uniformly distributed random frame is selected as the starting point for the filtering process. We conduct 100 trials per sequence and record the travelled distance at the first instance of successful localization. Results for each scenario (e.g., Scene-01 refers to Seq-01 and Seq-01-reverse combined) are presented in Fig. 8. The vehicle successfully localizes itself within 600m and 700m in 95% of the time for Scene-01 and Scene-02, respectively. In Scene-03, due to more challenging environmental conditions, the vehicle localizes itself within 850m in 95% of the time.

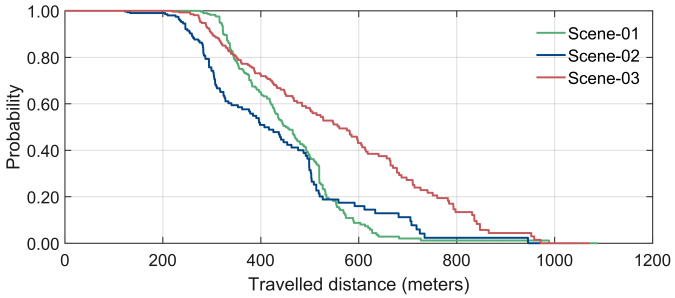

Fig. 8. Probability of travelling a given distance before successful localization.

$$P(x) = \frac{\text{Sum of distance travelled without localization for greater or equal to } x \text{ meters}}{\text{Total distance travelled}} \quad (6)$$

D. Ablation Study

In this part, we investigate how much the topological information and geometric information contribute in TGH. We run our method with Des_{TGH}, Des_{topo}, or Des_{geo} on all the sequences except Seq-04 and 04-reverse for ten times, respectively. Results for P_{sc} and S_{sc} are displayed in Tab. III. We can see that the combination of Des_{geo} and Des_{topo} is able to improve the robustness of global localization. For example, using Des_{TGH} can achieve higher values of P_{sc} on the sequences 02, 03, and 03-reverse compared to using only Des_{geo} or Des_{topo}. Meanwhile, using both topological information and geometric information allows a better convergence speed on the sequences 01-reverse and 03-reverse. In general, only using Des_{topo} and Des_{geo} is not very stable, they sometimes even lead to failures, while the proposed Des_{TGH} allows more robust performance.

E. Parameter Tuning for Re-sampling

To have a better understanding of the influence of KLD-based re-sampling on our method, we investigate the influence

TABLE III
P_{sc} AND S_{sc} FOR USING DESCRIPTOR Des_{TGH}, Des_{TOPO}, OR Des_{GEO}.

Sequence	Des_{TGH}		Des_{topo}		Des_{geo}	
	P_{sc}	S_{sc}	P_{sc}	S_{sc}	P_{sc}	S_{sc}
01	**1.0**	102.2	**1.0**	94.1	0.9	110.4
01-reverse	**1.0**	102.6	**1.0**	114.0	**1.0**	114.5
02	**1.0**	162.9	0.5	162.3	0.0	N/A
02-reverse	**1.0**	61.3	0.9	70.3	**1.0**	51.5
03	**0.9**	230.4	0.6	220.0	0.6	229.5
03-reverse	**0.8**	253.6	0.4	263.3	0.1	365.0

The best result for each row is highlighted. Different formats are used to represent the best results for different metrics (i.e., bold font for P_{sc} and underline for S_{sc}).

TABLE IV
P_{sc} AND S_{sc} FOR DIFFERENT RE-SAMPLING PARAMETERS.

$\epsilon = 0.15,\ \delta = 0.1$			$W_{bin} = 15,\ \delta = 0.1$			$\epsilon = 0.15,\ W_{bin} = 15$		
W_{bin}	P_{sc}	S_{sc}	ϵ	P_{sc}	S_{sc}	δ	P_{sc}	S_{sc}
5	**1.0**	392.8	0.05	**1.0**	329.6	0.01	0.9	262.6
10	**1.0**	293.6	0.10	**1.0**	253.5	0.02	**1.0**	248.4
15	0.9	230.4	0.15	0.9	230.4	0.05	0.9	252.2
20	0.9	234.9	0.20	0.8	213.1	0.10	0.9	230.4
25	0.8	225.3	0.25	0.7	221.0	0.20	**1.0**	222.9
30	0.7	201.4	0.30	0.5	197.4	0.30	0.9	260.2
40	0.3	126.3	0.40	0.3	155.5	0.40	0.8	222.3

of re-sampling parameters (i.e., ϵ, δ, and W_{bin}) on the number of particles, P_{sc}, and S_{sc}. For each parameter, we change its value while keeping the others unchanged, and then we run our method ten times. The experiment is conducted on Seq-03. Examples of the change of particle numbers during the filtering process can be found in Fig. 9, where the y-axis uses the log-scale. It can be easily found that ϵ has the greatest influence on the number of particles. When ϵ becomes larger, the number of particles reduces rapidly. Differently, the relation between the number of particles and W_{bin} can not be easily determined by the partial derivative of (2). When W_{bin} is larger, the value of the statistic result k is generally smaller, leading to the fast reduction of the number of particles (see Fig. 9(a)). As shown in Fig. 9(c), the influence of δ on the number of particles is small during the filtering process. This is because $\sqrt{\frac{2}{9(k-1)}}$ is much smaller than $z_{1-\delta}$ with different values of δ, especially at the early stage of the filtering process. Intuitively, filtering processes usually converge faster when the number of particles decreases faster. As displayed in Tab. IV, we can find that the vehicle can localize itself faster with a larger ϵ or a larger W_{bin}. However, the filtering process becomes not very stable with larger value of ϵ and W_{bin}, resulting in a low successful convergence rate P_{sc}. Empirically, smaller ϵ and W_{bin} could greatly reduce the failure chance of localization. However, it might cause too slow reduction for the number of particles, which usually requires more computational time. It might be better to use smaller values in more challenging environments, where usually have highly similar building structures, high building densities, and narrow streets.

F. Computational Cost

To show the efficiency of our method, we evaluate the computational costs of different parts during the filtering process in Tab. V. They are measured on three different platforms: a PC with i7-11700KF CPU (3.6 GHz) and 32-GB RAM, an Intel NUC Kit with i5-1135G7 CPU (2.4 GHz) and 16-GB RAM, and a NVIDIA Jetson Xavier NX kit with 6-core Careml ARM CPU and 8-GB RAM. We run our method three times on Seq-01 on each platform. We then calculate the average time cost μ for each part of our method during filtering. Λ is the maximum time cost of a single running step during filtering.

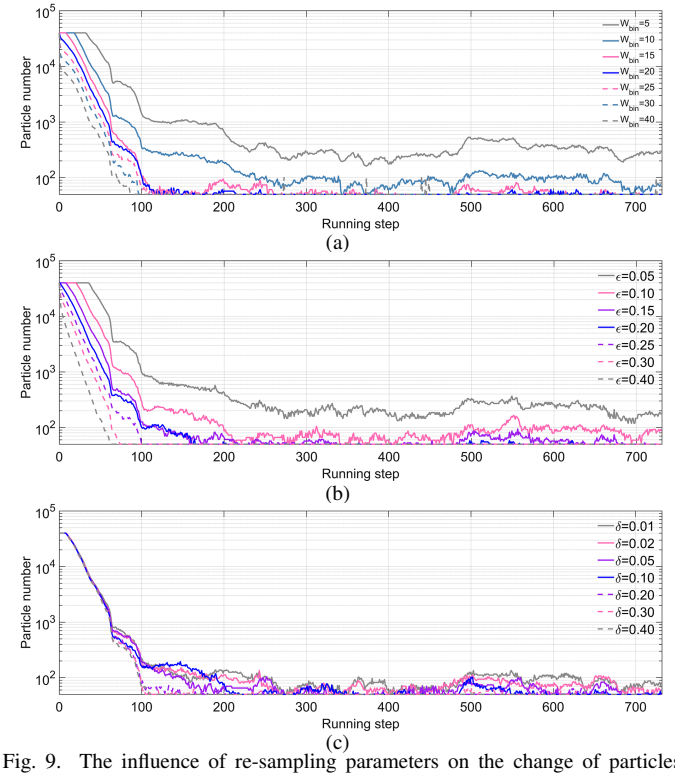

Fig. 9. The influence of re-sampling parameters on the change of particles number. The sub-figures (a), (b), and (c) represent the influences of W_{bin}, ϵ, and δ on particles number, respectively.

TABLE V
COMPUTATIONAL COST OF THE FILTERING PROCESS IMPLEMENTED ON DIFFERENT PLATFORMS.

Platform	i7-PC		Intel NUC		NVIDIA Xavier NX	
	μ	Λ	μ	Λ	μ	Λ
Dataset loading (s)	**3.57**	3.63	4.07	4.08	15.92	16.66
Filter initialization (ms)	**47.92**	48.06	58.40	58.61	212.06	214.78
TGH for I_f (ms)	**5.67**	6.87	7.47	8.38	27.58	40.22
Weight update (ms)	**1.05**	10.41	2.96	23.18	10.72	40.22
Particle number update (ms)	**0.39**	5.26	0.51	6.49	1.37	17.59
Particle re-sample (ms)	**0.40**	4.12	0.58	6.42	1.90	22.25
Total for single step (ms)	**7.51**	26.66	11.52	44.47	41.57	178.87

μ refers to the average time cost. Λ is the maximum time cost.

Specifically, the dataset loading contains loading the M_{osm} descriptor dataset and constructing the KD tree, which takes the longest time. The particle filtering initialization can be finished within 220ms. We consider these two parts as the initial phase, which only needs to be conducted once. As for the other parts (i.e., Des_{TGH} extraction time, time for particle weights update, time for particle number update, and KLD-based particle re-sampling time), the total average time cost is less than 50ms on all the three platforms. Generally, the proposed method can run fast with real-time performance.

V. CONCLUSIONS AND FUTURE WORK

We presented here a novel method to globally localize a vehicle by using a sky-looking fish-eye camera and OSM

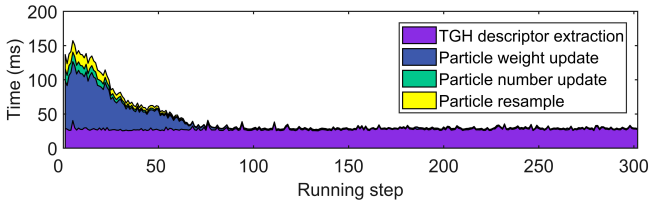

Fig. 10. Computational cost of the filtering process for Seq-01 using a NVIDIA Jeston Xavier NX kit.

in GNSS-degraded environments. We have shown quantitative and qualitative results for global localization performance in challenging scenarios. The proposed method can efficiently run in real-time even on embedding platforms. However, our method still suffers from overexposure caused by sunlight, and affected by various heights of buildings. Our method requires a motion model with an absolute scale to update the state of particles, making it more suitable as a low-cost add-on for a localization system rather than functioning as a standalone system. Moreover, our method is expected to work only in high-rise environments. It might fail when the height and density of buildings are too low, where the TGH descriptor can not be well extracted due to the lack of enough observations for building tops. In the future, we plan to combine IMU with the fish-eye camera as the VIO to release the need of an additional motion model.

REFERENCES

[1] Y. Lu, H. Ma, E. Smart, and H. Yu, "Real-time performance-focused localization techniques for autonomous vehicle: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6082–6100, May. 2022.

[2] P. Cai, H. Wang, Y. Sun, and M. Liu, "DQ-GAT: Towards safe and efficient autonomous driving with deep Q-learning and graph attention networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 21 102–21 112, Nov. 2022.

[3] W. Ma, S. Huang, and Y. Sun, "Triplet-Graph: Global metric localization based on semantic triplet graph for autonomous vehicles," *IEEE Robot. Automat. Lett*, vol. 9, no. 4, pp. 3155–3162, Apr. 2024.

[4] D. Min, M. Kim, J. Lee, M. S. Circiu, M. Meurer, and J. Lee, "DNN-based approach to mitigate multipath errors of differential GNSS reference stations," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25 047–25 053, Sep. 2022.

[5] D. Betaille, F. Peyret, M. Ortiz, S. Miquel, and L. Fontenay, "A new modeling based on urban trenches to improve GNSS positioning quality of service in cities," *IEEE Intell. Transp. Syst. Mag.*, vol. 5, no. 3, pp. 59–70, 2013.

[6] R. Kumar and M. G. Petovello, "A novel GNSS positioning technique for improved accuracy in urban canyon scenarios using 3D city model," in *Proc. 27th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS),*, Sep. 2014, pp. 2139–2148.

[7] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Nov. 2015.

[8] K. A. Tsintotas, L. Bampis, and A. Gasteratos, "The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 19 929–19 953, May. 2022.

[9] A. R. Memon, Z. Liu, and H. Wang, "Invariant loop closure detection using step-wise learning with controlling embeddings of landmarks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20 148–20 159, May. 2022.

[10] M. U. M. Bhutta, Y. Sun, D. Lau, and M. Liu, "Why-So-Deep: Towards boosting previously trained models for visual place recognition," *IEEE Robot. Automat. Lett*, vol. 7, no. 2, pp. 1824–1831, Apr. 2022.

[11] H. Xu, H. Liu, S. Meng, and Y. Sun, "A novel place recognition network using visual sequences and lidar point clouds for autonomous vehicles," in *Proc. IEEE Int. Conf. on Intell. Transp. Syst.*, 2023, pp. 2862–2867.

[12] H. Xu, H. Liu, S. Huang, and Y. Sun, "C2l-pr: Cross-modal camera-to-lidar place recognition via modality alignment and orientation voting," *IEEE Trans. Intell. Veh.*, pp. 1–17, 2024.

[13] W. Ma, H. Yin, P. J. Y. Wong, D. Wang, Y. Sun, and Z. Su, "TripletLoc: One-shot global localization using semantic triplet in urban environments," *IEEE Robot. Automat. Lett*, vol. 10, no. 2, pp. 1569–1576, 2025.

[14] T. Törnros, H. Dorn, S. Hahmann, and A. Zipf, "Uncertainties of completeness measures in OpenStreetMap–a case study for buildings in a medium-sized german city," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 2, pp. 353–357, 2015.

[15] M. Hentschel and B. Wagner, "Autonomous robot navigation based on on OpenStreetMap geodata," in *Proc. IEEE trans. Intell. Transp. Syst.* IEEE, 2010, pp. 1645–1650.

[16] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, "UrbanLoco: A full sensor suite dataset for mapping and localization in urban scenes," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2020, pp. 2310–2316.

[17] O. Pink, "Visual map matching and localization using a global feature map," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. Workshops*, Jun. 2008, pp. 1–7.

[18] K. Y. K. Leung, C. M. Clark, and J. P. Huissoon, "Localization in urban environments by matching ground level video images with an aerial image," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2008, pp. 551–556.

[19] P. Agarwal, W. Burgard, and L. Spinello, "Metric localization using Google Street View," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sept. 2015, pp. 3111–3118.

[20] X. Wang, S. Vozar, and E. Olson, "Flag: Feature-based localization between air and ground," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2017, pp. 3178–3184.

[21] N. Radwan, G. D. Tipaldi, L. Spinello, and W. Burgard, "Do you see the bakery? Leveraging geo-referenced texts for global localization in public maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 4837–4842.

[22] D.-K. Kim and M. R. Walter, "Satellite image-based localization via learned embeddings," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2017, pp. 2073–2080.

[23] S. Hu, M. Feng, R. M. Nguyen, and G. H. Lee, "CVM-Net: Cross-view matching network for image-based ground-to-aerial geo-localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7258–7267.

[24] S. Hu and G. H. Lee, "Image-based geo-localization using satellite imagery," *Int. J. Comput. Vis*, vol. 128, no. 5, pp. 1205–1219, 2020.

[25] S. Zhu, M. Shah, and C. Chen, "Transgeo: Transformer is all you need for cross-view image geo-localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1162–1171.

[26] W. Ma, H. Yin, L. Yao, Y. Sun, and Z. Su, "Evaluation of range sensing-based place recognition for long-term urban localization," *IEEE Trans. Intell. Veh.*, pp. 1–12, 2024.

[27] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based slam using aerial images as prior information," *Auton. Robots*, vol. 30, no. 1, pp. 25–39, Jan. 2011.

[28] T. Y. Tang, D. De Martini, D. Barnes, and P. Newman, "RSL-Net: Localising in satellite images from a radar on the ground," *IEEE Robot. Automat. Lett*, vol. 5, no. 2, pp. 1087–1094, Jan. 2020.

[29] T. Y. Tang, D. De Martini, S. Wu, and P. Newman, "Self-supervised learning for using overhead imagery as maps in outdoor range sensor localization," *Int. J. Robot. Res.*, vol. 40, no. 12-14, pp. 1488–1509, Dec. 2021.

[30] M. Hussein, M. Renner, M. Watanabe, and K. Iagnemma, "Matching of ground-based LiDAR and aerial image data for mobile robot localization in densely forested environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1432–1437.

[31] I. D. Miller, A. Cowley, R. Konkimalla, S. S. Shivakumar, T. Nguyen, T. Smith, C. J. Taylor, and V. Kumar, "Any way you look at it: Semantic crossview localization and mapping with LiDAR," *IEEE Robot. Automat. Lett*, vol. 6, no. 2, pp. 2397–2404, Feb. 2021.

[32] G. Floros, B. Van Der Zander, and B. Leibe, "OpenStreetSLAM: Global vehicle localization using OpenStreetMaps," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2013, pp. 1054–1059.

[33] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 38, no. 4, pp. 652–665, Jul. 2015.

[34] P. Panphattarasap and A. Calway, "Automated map reading: Image based localisation in 2-D maps using binary semantic descriptors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 6341–6348.

[35] N. Samano, M. Zhou, and A. Calway, "You are here: Geolocation by embedding maps and images," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 502–518.
[36] M. Zhou, X. Chen, N. Samano, C. Stachniss, and A. Calway, "Efficient localisation using images and OpenStreetMaps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sept. 2021, pp. 5507–5513.
[37] P.-E. Sarlin, D. DeTone, T.-Y. Yang, A. Avetisyan, J. Straub, T. Malisiewicz, S. R. Bulo, R. Newcombe, P. Kontschieder, and V. Balntas, "Orienternet: Visual localization in 2d public maps with neural matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21 632–21 642.
[38] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on OpenStreetMap data using a 3D laser scanner," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2015, pp. 5260–5265.
[39] B. Suger and W. Burgard, "Global outer-urban navigation with OpenStreetMaps," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2017, pp. 1417–1422.
[40] F. Yan, O. Vysotska, and C. Stachniss, "Global localization on OpenStreetMap using 4-bit semantic descriptors," in *Proc. of the Europ. Conf. on Mobile Robotics*, 2019, pp. 1–7.
[41] Y. Cho, G. Kim, S. Lee, and J.-H. Ryu, "OpenStreetMap-based LiDAR global localization in urban environment without a prior LiDAR map," *IEEE Robot. Automat. Lett*, vol. 7, no. 2, pp. 4999–5006, Feb. 2022.
[42] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.
[43] J. Zhang and S. Singh, "Low-drift and real-time LiDAR odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, Feb. 2017.
[44] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Int. J. Robot. Res.*, vol. 22, no. 12, pp. 985–1003, Dec. 2003.
[45] N. Otsu *et al.*, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.
[46] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Int. Conf. Robot. Automat.* IEEE, 2017, pp. 5266–5272.

Weixin Ma (Student Memeber, IEEE) received the B.S. degree in Mechanical Engineering and the M.S. degree in Mechanical and Electronic Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2018 and 2021,respectively. He is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong.

His current research interests include localization and mapping, robotic perception, and mobile robotic.

Shoudong Huang received the bachelor's and master's degrees in mathematics, and the Ph.D. degree in automatic control from Northeastern University, Shenyang, China, in 1987, 1990, and 1998, respectively.

He is currently a Professor with the School of Mechanical and Mechatronic Engineering, University of Technology Sydney, Sydney, Australia. His research interests include mobile robots, simultaneous localization and mapping (SLAM), and robot path planning and control.

Yuxiang Sun (Member, IEEE) received the Ph.D. degree from The Chinese University of Hong Kong, Shatin, Hong Kong, in 2017, the master's degree from the University of Science and Technology of China, Hefei, China, in 2012, and the bachelor's degree from the Hefei University of Technology, Hefei, China, in 2009.

He is currently an Assistant Professor with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong. His current research interests include robotics and AI, autonomous driving, mobile robots, and autonomous navigation.

Prof. Sun serves as an Associate Editor for IEEE Transactions on Automation Science and Engineering, IEEE Transactions on Intelligent Vehicles, IEEE Robotics and Automation Letters, IEEE International Conference on Robotics and Automation, and IEEE/RSJ International Conference on Intelligent Robots and Systems.