AVP-Loc: Surround View Localization and Relocalization Based on HD Vector Map for Automated Valet Parking

Chi Zhang Hao Liu Zhijun Xie Kuiyuan Yang Kun Guo Zhiwei Li

***Abstract*— Localization is a crucial prerequisite for automated valet parking, in which a vehicle is required to navigate itself in a GPS-denied parking lot. Traditional visual localization methods usually build a feature map and use it for future localizations. However, the feature map is not robust to changes in illumination, appearance, and viewing perspective. To deal with this issue, we need a more stable map. In this paper, we propose to use the parking lot's HD vector map directly for localization. The vector representation is ultimately stable but brings challenges in data association as well. To this end, we present a novel data association method to match the surround-view images with the vector map. In addition, we also propose a closed-form relocalization strategy by exploiting distinctive road mark combinations in the vector map. Experiments show that the proposed method is able to achieve centimeter-level localization accuracy in a multi-floor parking lot.**

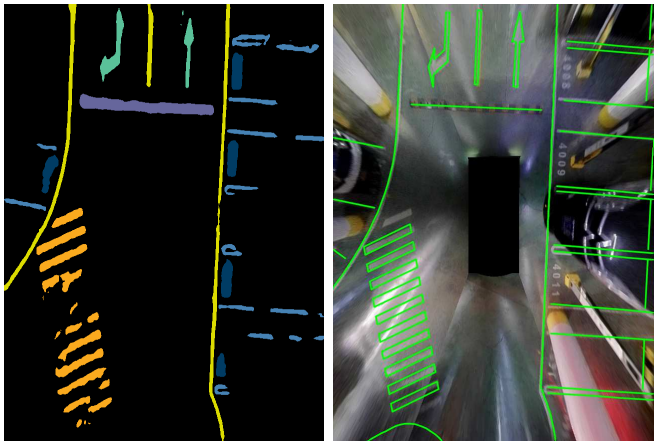
I. Introduction

Automated Vallet Parking (AVP) has become a popular application in recent years due to the rapid growth of the autonomous driving industry. In AVP, a vehicle is required to navigate in GPS-denied parking lots and park itself into available parking spaces. Accurate localization is a prerequisite for the application. In this paper, we focus on visual-based localization solutions in this paper since cameras are cheap and many commercial vehicles have been deployed with camera sensors.

Traditional visual SLAM methods[1], [2] have achieved great success in terms of accuracy. They accumulate a sparse map of local features while simultaneously localize the robot. After accumulation, the feature map can be used to relocalize in future runs. Albeit excellent accuracy, local feature-based approaches suffer from robustness issues. The feature map is not robust to changes in illumination, appearance, and viewing perspective. Appearance changes are common in parking lots, e.g., occupancy statuses of parking slots change, parked cars change, etc.

To deal with this issue, we need a more stable map. In this paper, we propose to use the parking lot's HD vector map directly for localization. The vector map consists of a set of vectorized 3D shapes of the semantic traffic elements, e.g., parking lines, lane lines, arrows, and speed bumps. Although the vector representation is inherently stable, it brings us new challenges in data association. The images and the vector map are heterogeneous data and cannot be matched directly. To this end, we propose a novel method to associate the two modalities. The method is based on the semantic segmentation of the Bird's Eye View (BEV) image stitched from four surround-view images. A class-wise matching strategy is designed to associate the segmentation with the vector map. Besides localization, we also propose a relocalization strategy by exploiting the vector map. The strategy works by searching and matching distinctive road mark combinations in the map and enables a closed-form relocalization.

All authors are with DeepMotion Co., Ltd. Beijing, China.



(a) BEV segmentation (b) HD map projection

Fig. 1: The proposed localizer matches BEV segmentations with the vector map for localization. Some zebra strips in (a) are missed due to the local over-exposure in the BEV image. The HD vector map projection in (b) is computed using the localizer's output pose.

Using a vector map directly for localization provides us with some collateral advantages. First, the map size is much smaller without a separate localization layer. Second, mapping and localization can be done separately. Mapping can be made easier by employing stronger sensors such as LiDARs. The online system can focus solely on localization and relocalization without worrying about implementing heavy logic such as submap maintenance, bundle adjustment, or loop closing in the visual modality. It is also worth noting that creating a vector map from the local feature map is cumbersome and sometimes impossible since the feature map is usually far from complete and is non-intuitive to interact with.

The contribution of this paper is summarized as follows:

- A heterogeneous data association method that matches the surround-view images with the HD vector map based on semantic segmentation.
- A closed-form relocalization strategy by exploting distinctive road mark combinations distributed among the vector map.
- A full surround-view based localization system that achieves centimeter-level accuracy in parking lots without a separate localization layer.

II. Related Work

A large number of visual localization methods have been proposed in the last decades. Here we review the most related and classify them into two classes.

A. Feature-Based Approaches

Campos et al.[1] have demonstrated centimeter-level localization accuracy using ORB feature matching and IMU fusion. Sons et al.[3] present a surround camera localization system but using a different descriptor called DIRD[4], which is more robust to illumination. The method achieves a localization accuracy within 30cm. Under a similar framework to [1], Xuan et al.[5] propose to detect parking slots and use them to constrain the sliding window optimization further. Albeit excellent accuracy, the local feature-based approaches suffer from long-term robustness problems caused by changes of appearance, illumination, and viewing perspective.

Another branch of related work focuses on extracting dense pixel-wise features from IPM images. Rehder and Albrecht[6] maintain an occupancy grid map of road marks, which are divided into submaps and subsequently used to perform loop closing. Road marks are detected by thresholding the result from the Difference of Gaussian (DoG) filter. The authors present the experiments on a toy test track. Since most road marks are repetitive, leading to high matching ambiguity, Jeong et al.[7] propose to train random forest trees to classify only the distinguishable road marks. They present experiments on an over 4.7km long test route and achieves a mean error of 1m. To further improve robustness, Qin et al.[8] propose to use semantic class labels for each IPM pixel. Such features have demonstrated outstanding performance in the parking lot scenarios. Relocalization in these approaches is based on ICP, which requires a reasonably good initial guess. Therefore the relocalization might converge to a local minimum depending on the quality of the initial guess. This problem is not present in our closed-form relocalization method.

B. Vector Map-Based Approaches

Lu et al.[9] treat the vector map as a set of discrete 3d points and project them to the front camera image. The localization is then adjusted to minimize the chamfer distance between the projected point set and the front image's edge detection. The overall localization accuracy is 0.6m. Ranganathan et al.[10] use a map consisting of a set of discrete and special road marks, e.g., arrows, pedestrian crossings, and speed limits. These road marks are detected from the front view IPM image using template matching. The authors then extract corner points to match with the template for pose calculation. The limitation lies in the robustness of the template matching. Since the discrete markings are typically separated far apart, drift may accumulate to a large amount before visiting the next discrete marking. The method achieves an average localization accuracy of 1m. To deal with the robustness issue above, Wu and Ranganathan[11] extend their work to include a shadow boosting preprocessing step.

Schreiber et al.[12] propose the match lane lines in the front image using oriented filters and match the detected points with the closest projected vector in the map. A v-disparity[13] image is computed from the stereo depth to deal with the pitch sensitivity issue during projection. Poggenhans et al.[14] assemble a local stereo point cloud and detect lane lines and road curbs from it. They then match the lines and curbs with the vector map in 3D metric space. Jeong et al.[15] perform semantic segmentation on the stereo images and convert the stereo point cloud into a top-view 8bit image, where each bit represents the presence of a corresponding semantic class. The 8bit image is then matched with an 8bit image database preprocessed from the vector map using a particle filter. The above approaches all rely on the stereo setup, which provides useful depth information for data preprocessing. The stereo requirement could be considered a limitation in our context. Since many indoor parking lot scenes have large textureless areas and reflective grounds (Fig. 1b shows a zebra region overwhelmed by highlights), stereo matching performs poorly in such circumstances.

III. The Localizer

The proposed localizer uses the BEV image's semantic segmentation as the medium to match against the vector map. The matching process adopts a class-wise matching strategy so that the process can be accurately modeled. The following subsections explain the process in detail.

A. Semantic Segmentation on BEV

We construct a BEV image from the four surround fisheye cameras using Inverse Perspective Mapping. A BEV pixel $[u, v]^\top$ and its position in the vehicle coordinates $[X, Y, 0]^\top$ is related by:

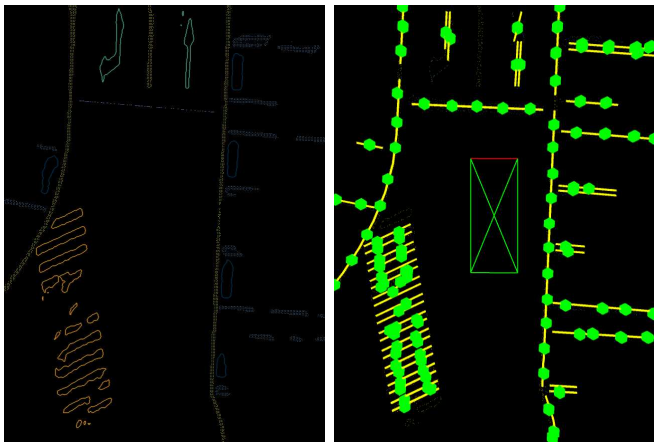
$$\pi([u, v]^\top) = [s(u - \frac{w}{2}), -s(v - \frac{h}{2}), 0]^\top = [X, Y, 0]^\top \quad (1)$$

where w, h are the image width and height. s is the pixel scale. Each pixel in the BEV occupies a $2cm^2$ space in vehicle coordinates. We use a 640×872 image size covering an ROI of $12.8 \times 17.44m^2$ in physical world.

To associate the heterogeneous data between the images and the vector map, we train a convolutional network that converts the BEV images into semantic segmentations. The semantic classes we use here are different types of road marks in the parking lots, e.g., lane lines, parking lines, arrows, text, speed bumps, zebra strips, dash segments. In the label definition, we distinguish horizontal and vertical parking lines w.r.t. the parking slots' entrance direction. The distinguishment gives us better positional constraints in lateral and longitudinal directions. We employ [16] to train the network. 5,000 training samples across ten parking lots were used in our experiments.

B. Vector Map Matching

We now match the BEV image's semantic segmentation with the HD vector map. Unlike existing approaches[12], [8] that match for each feature in the frame to their feature maps, we do the reverse. We match for each vector segment



(a) Processed segmentation (b) Vector map matches

Fig. 2: (a) A BEV segmentation is preprocessed before matching. Thin lines are downsampled; Discrete road marks are converted into contours; Speed bumps are projected to colinear points. (b) Each green dot represents a vector segment to feature point match.

a semantic feature point in the BEV segmentation. This naturally results in an evenly distributed set of matches and reduces the number of matches drastically.

A naive matching strategy is to adopt ICP directly. However, this will result in matching ambiguity. For example, a speed bump could be 40cm wide in BEV, but it is represented as a line segment in the vector map. An arrow is a connected region in BEV but the vector map stores only its boundary. Also, a direct ICP would be slow since we need to index the foreground semantic feature set, which could contain up to 100,000 points. To this end, we propose a class-wise matching strategy:

1) For a discrete road mark (e.g., arrow, zebra strip) represented as a polygon in the map, we match each segment in the polygon against the *contour* pixels of the semantic instance.
2) For a continuous *thin* road mark (e.g., lane line, parking line) represented as a polyline in the map, we down-sample the semantic pixels in these classes by an order of magnitude and match each vector segment against *down-sampled set*.
3) For a continuous *thick* road mark (e.g., speed bump) represented as a polyline or a line segment in the map, we project its contour pixels to the contour set's major PCA axis and match each vector segment against this *projected set*.

Fig. 2a and 2b illustrates the preprocessed road marks and the matched results. The purpose of the varied matching strategies is to provide a better matching model for different semantic types, as well as to speed up the matching process. Algorithm 1 illustrates the full process in detail. Three points are worth noting. First, we divide the vector shapes into sets of one-meter segments and match the middle points of these segments to the pixels so that the space can be covered uniformly. Second, at most one pixel can be matched with a given one-meter segment. Third and obviously, only segments and pixels with the same semantic label can be matched.

Algorithm 1 BEV segmentation and vector map matching

Input: Vector map $\mathcal{M}$; BEV segmentation $\mathcal{S}$; Three class label sets $\mathcal{C}_{\text{Thin}}$, $\mathcal{C}_{\text{Thick}}$, $\mathcal{C}_{\text{Discrete}}$; Current pose $\mathbf{T}$.
Output: The set of point-to-line matches Ω.

$\Omega = \emptyset$; $\mathcal{P}_{\text{Thin}} = \emptyset$; $\mathcal{P}_{\text{Thick}} = \emptyset$; $\mathcal{P}_{\text{Discrete}} = \emptyset$;
for each connected component s in $\mathcal{S}$ **do**
  if s.label $\in \mathcal{C}_{\text{Thin}}$ **then**
    $\mathcal{P}_{\text{Discrete}} = \mathcal{P}_{\text{Discrete}} \cup \text{ExtractContour(s)}$
  else if s.label $\in \mathcal{C}_{\text{Discrete}}$ **then**
    $\mathcal{P}_{\text{Thin}} = \mathcal{P}_{\text{Thin}} \cup \text{DownSample(s)}$
  else if s.label $\in \mathcal{C}_{\text{Thick}}$ **then**
    $c = \text{ExtractContour}(s)$
    for each $p \in c$ **do**
      $\mathcal{P}_{\text{Thick}} = \mathcal{P}_{\text{Thick}} \cup \{\text{Project}(p, c.\text{MajorPcaAxis})\}$
    end for
  end if
end for
$\mathcal{Q}_{\text{Thin}}$ = BuildQuadtree($\mathcal{P}_{\text{Thin}}$)
$\mathcal{Q}_{\text{Thick}}$ = BuildQuadtree($\mathcal{P}_{\text{Thick}}$)
$\mathcal{Q}_{\text{Discrete}}$ = BuildQuadtree($\mathcal{P}_{\text{Discrete}}$)
for each element e in RadiusSearch($\mathcal{M}, \mathbf{T}, 20m$) **do**
  for each one-meter segment $\mathbf{c}_i\mathbf{c}_{i+1}$ in e **do**
    $\mathcal{Q}_* = \{\mathcal{Q}_{\text{Thin}}, \mathcal{Q}_{\text{Thick}}, \mathcal{Q}_{\text{Discrete}}\}[e.\text{label}]$
    p = NearestNeighborSearch($\mathcal{Q}_*, \pi^{-1}\left(\mathbf{T}\frac{\mathbf{c}_i+\mathbf{c}_{i+1}}{2}\right)$)
    if dist$(p, \mathbf{c}_i\mathbf{c}_{i+1}) < \tau$ **then**
      $\Omega = \Omega \cup (p, \mathbf{c}_i\mathbf{c}_{i+1})$
    end if
  end for
end for

C. State Estimation by ESKF

We adopt the Error State Kalman Filter (ESKF)[17] for state estimation. The localizer aims to estimates the true IMU state

$$[\mathbf{p}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \mathbf{b}_a{}^\top, \mathbf{b}_g{}^\top]^\top \tag{2}$$

representing respectively its world position, world velocity, quaternion from body frame to world frame, accelerometer, and gyroscope biases. The ESKF propagation step is a standard procedure and is omitted here. Two sources of information are used for ESKF update.

1) **Vector Map Matching.** Given a point-to-segment match $(p, \mathbf{c}_i\mathbf{c}_{i+1})$ in pixel coordiantes and world coordinates respectively, the vector map observation model penalizes the following point-to-line distance in the world frame

$$\frac{\|(\mathbf{x}^W - \mathbf{c}_i) \times (\mathbf{x}^W - \mathbf{c}_{i+1})\|}{\|\mathbf{c}_i - \mathbf{c}_{i+1}\|} \tag{3}$$

where

$$\mathbf{x}^W = \mathbf{q} \otimes (\mathbf{T}_V^I \mathbf{x}^V) + \mathbf{p} \tag{4}$$

Here, $\mathbf{x}^W$ and $\mathbf{x}^V$ are the matched point expressed in world coordinates and the vehicle coordinates respectively. $\mathbf{T}_V^I$

denotes the transformation from the vehicle frame to the IMU frame. The vehicle coordinates $\mathbf{x}^V$ is computed from the input pixel (p_u, p_v) using Eq. 1. All point-to-segment matches form a residual vector by stacking Eq. 3.

2) **Wheel Speed.** Given a wheel speed $\mathbf{v}^V$ recorded in the vehicle frame, and an IMU gyro reading $\boldsymbol{\omega}$ interpolated to the same timestamp, the vehicle speed observation model requires the speed component in the IMU state to agree with the wheel encoder measurement

$$\mathbf{q}^{-1} \otimes \mathbf{v} + [\boldsymbol{\omega} - \mathbf{b}_g]_\times \mathbf{t}_V^I = \mathbf{R}_V^I \mathbf{v}^V \tag{5}$$

Both sides of the equation denote the speed of the vehicle frame's origin in IMU coordinates. The right-hand side is obtained from the wheel speed reading, while the left-hand side is computed from the IMU state vector with lever-arm compensation imposed. Here $\mathbf{t}_I^V$ shall be interpreted as the location of the vehicle frame's origin in the IMU frame.

State correction then follows the standard ESKF update procedure. Please refer to [17] for derivation details.

IV. Relocalization

We also exploit the HD vector map for the relocalization task. The general idea is to determine a set of distinctive road mark combinations in the vector map so that a road mark combination detected in the BEV segmentation can be uniquely matched with its counterpart in the vector map.

A. Distinctive Landmark Selection

Our first step is to determine the set of distinctive landmarks from the vector map to serve as the relocalization targets. Here, a landmark refers to a specific *combination* of nearby discrete road mark instances. The concept of "combination" has implicitly introduced for each landmark a spatial signature, represented by the relative orientations and distances between the road marks belonging to that landmark. The spatial signature significantly improves the distinctiveness and availability of landmarks.

Formally, we represent each discrete road mark instance by a 3-tuple $(\mathbf{p}_i, \theta_i, c_i)$, which represents its centroid position, orientation computed from Principle Component Analysis (PCA), and the class label respectively. Each discrete road mark is given an index $i \in \mathbb{N}$. We represent a landmark α by a set of road mark indices. A landmark α is said to be distinctive in radius R if there exists no other similar landmarks within a that radius. Two landmarks are called similar if there exists a rigid body transformation that well-aligns the two sets of 3-tuples. However, this definition requires solving for the rigid transformation, which will result in unnecessary computation. To ease similarity testing, we transform the rigid transformation test into a "spatial signature" test. Under this test, a landmark α and a landmark β are similar if and only if

$$\exists \gamma \in P(\beta), \quad \forall i \in \{1, \dots, n\}, \tag{6}$$

$$\|(\mathbf{p}_{\alpha_i} - \mathbf{p}_{\alpha_1}) - (\mathbf{p}_{\gamma_i} - \mathbf{p}_{\gamma_1})\| < \tau_{\mathbf{p}} \tag{7}$$

$$\|(\theta_{\alpha_i} - \theta_{\alpha_1}) - (\theta_{\gamma_i} - \theta_{\gamma_1})\| < \tau_{\theta} \tag{8}$$

$$c_{\alpha_i} = c_{\gamma_i} \tag{9}$$

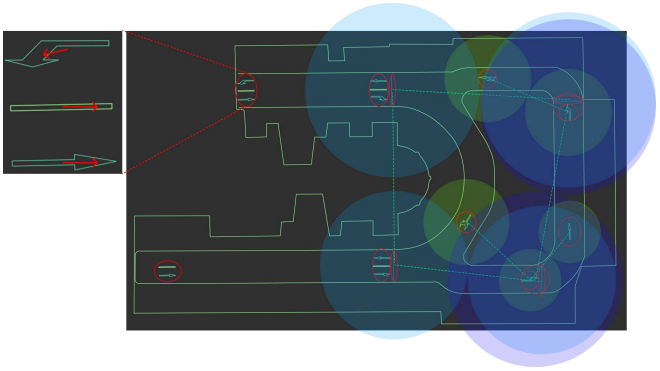


Fig. 3: The distribution of relocalization landmarks (red circles) in our parking lot map. A landmark is a specific combination of one to three nearby discrete road marks. Each landmark is assigned as radius that it is unique in.

where γ is a permutation of β. n is the number of element indices in each landmark. α_i, γ_i represent the ith element index in each landmark respectively. $\tau_{\mathbf{p}} = 0.2m$, $\tau_\theta = 5^\circ$ are the similarity thresholds. Intuitively, Eq. 6 to 9 mean whether the relative distances and orientations (i.e. the spatial signature) within each landmark are similar.

We choose speed bumps, arrows, and dash segments as the elementary discrete road marks for combination. The dash segment cannot be used alone but can participate in augmenting a combination for better distinctiveness. In practice, we assign at least one and at most three discrete marks for a landmark. Each landmark is also associated with a radius it is distinctive in. Fig. 3 shows the distribution of the landmarks we extract from the HD vector map. Note although a similarity test in Eq. 6 to 9 involves permutation, this will not cause any combinatorial explosion problem, because a landmark contains at most three instances, and only the instances with the same label need to be permutated. In practice, at most two instances share the same label in a real-world landmark, which results in only two permutation results.

B. Drift Detection

Relocalization mode is triggered only when drift is detected. The idea of drift detection is based on the following observation. In a correct solution, every semantic feature point extracted from a BEV image will have a matched vector segment from the vector map. However, when there is a drift, many semantic feature points will fail to find their matched vectors. We call these feature points the orphan feature points. We choose the ratio of the orphan feature points as the drift detection metric. If the orphan ratios are lower than a predefined threshold for a consecutive number of frames, we are almost certain that drift has occurred.

Formally, the orphan ratio r for one frame is defined as

$$r = \frac{\sum_{s \in \mathcal{S}} \mathbf{1}\{d(s, \mathcal{L}) > \tau_m\}}{\|\mathcal{S}\|}, \qquad d(s, \mathcal{L}) = \min_{l \in \mathcal{L}} d(s, l) \tag{10}$$

where $\mathcal{S}$ is the set of semantic feature points. $\mathcal{L}$ is the set of line semgents that participate in vector map matching.

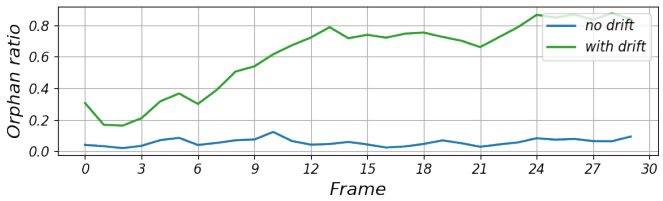


Fig. 4: Illustration of orphan ratio changes.

Algorithm 2 Landmark-based relocalization

Input: BEV segmentation $\mathcal{S}$; Vector map $\mathcal{M}$; Global landmark map L_M;
Output: Relocalized pose $\mathbf{T}$;

L_S = GenerateLocalLandmarks($\mathcal{S}$)
for landmark l_s in landmark candidate set L_S **do**
  $\{L_1, \ldots, L_n\}$ = RetrieveLandmarks(l_s.label_signature, L_M)
  for landmark set $L_i \in \{L_1, \ldots, L_n\}$ with i-th spatial signature **do**
    for landmark $l_j \in L_i$ at j-th localization **do**
      if SpatialSignatureMatches(l_s, l_j) **then**
        if l_j's circle covers l_s's covariance ellipse **then**
          $\mathbf{T}$ = ClosedFormReinitialize(l_s, l_j)
          $\mathbf{T}$ = NonlinearRefine($\mathbf{T}, \mathcal{M}$)
        end if
      end if
    end for
  end for
end for

$d(s, l)$ denotes the point to line segment distance. τ_m is a matching distance threshold. To save computation power, we set $\mathcal{S}$ to the down-sampled set of feature points described in Section III-B. To accelerate the computation of $d(s, \mathcal{L})$, only the line segments that actually found a closest match are kept. We create an octree index for the middle points of these segments so that $\min_{l \in \mathcal{L}} d(s, l)$ can be computed by one nearest neighbor search. Fig. 4 illustrates the orphan ratio change for a simulated drifted trajectory.

C. Closed-form Reinitialization

Relocalization is done by matching local landmark candidates with their counterparts in the vector map.

Local landmark detection. From Section. III-B, we have already obtained the contours of the discrete elements for a given BEV segmentation. Each contour represents an instance of the discrete marks. We then compute each instance's 3-tuple representation $(\mathbf{p}_i, \theta_i, c_i)$ in local coordinates using PCA. After that, we form a landmark candidate set by free combinations of these 3-tuples. Instances of nonstandard sizes are eliminated from candidate generation.

Local-global landmark matching. Now we are given a local landmark candidate set L_s, our goal is to return a correct match between L_s and the global landmark set L_M. Algorithm 2 illustrates the matching process. For a landmark candidate l_s, we ask the map to return the landmarks that share the same "label signature" with l_s. A landmark's label signature is defined as the frequency histogram on the three labels {speed bump, arrow, dash segment}. The returned is a landmark set list $\{L_1, \ldots, L_n\}$, where they all share the same label signature but each set $L_i \in \{L_1, \ldots, L_n\}$ has it own spatial signature. Obviously, at most one set from the list will match the query l_s's spatial signature. Let us denote the matched set as L_* (recall that spatial signature comparison is performed using Eq. 6 to 9). The remaining problem is to determine which landmark in L_* is the real match to l_s. If L_* only contains one landmark, we are done. Otherwise, we resort to the distance information to distinguish identical landmarks.

A naive solution is to choose the landmark that is closest to the current position estimate. However, since the tracking has already lost, we cannot trust the state estimator's output. To deal with this issue, we maintain a separate Kalman filter that tracks the odometry using IMU and wheel encoders only. The filter will be regularly reset when there is no drift so that its covariance stays small and bounded. When drift occurs, the filter provides us an ellipse of uncertainty from its mean and covariance. We use a one-sigma ellipse size in our experiments. To determine which landmark in L_* is the real match to l_s, we return the landmark in L_* whose distinctive circle completely contains the ellipse of uncertainty. If such a match does not exist, we continue to other local landmark proposals. Note that a BEV might contain multiple local landmark proposals, and it is possible that each of them has a legal landmark match in the vector map. In this case, any of these matches can be used for relocalization.

However, the above matching process does not distinguish similar landmarks across parking lot floors. To deal with this problem, we need to maintain a z-estimate to reason about which floor we are currently in. We propose two solutions. The first solution simply uses the z-estimate from the localizer before drift happens. The xy-component of the odometry trajectory since the drift happens can be used to determine whether the vehicle has gone upstairs or downstairs, by matching them roughly with the vector map's lane topology. The second solution uses a barometer, which can give a z-estimate at 1m accuracy.

Closed-form reinitialization. Now we are given two sets of matched 3-tuples, denoted as $\{\mathbf{p}_{\alpha_i}, \theta_{\alpha_i}, c_{\alpha_i}\}$ and $\{\mathbf{p}_{\beta_i}, \theta_{\beta_i}, c_{\beta_i}\}$. We want to compute a 3D rigid body transformation that aligns the two sets. Since only 4DoF need to be estimated, one 3-tuple already suffices to compute the transformation:

$$\mathbf{R} = \text{AngleAxis}(\theta_{\beta_i} - \theta_{\alpha_i}, [0, 0, 1]^\top) \tag{11}$$

$$\mathbf{t} = \mathbf{p}_{\beta_i} - \mathbf{p}_{\alpha_i} \tag{12}$$

For landmark containing multiple instances, we compute a separate $\mathbf{R}, \mathbf{t}$ for each instance, and average them to get the final reinitialized pose. The accuracy already suffices for the following nonlinear refinement.

Note however that there is a sign ambiguity in each instance's principal direction. To deal with this problem, we resort to the separate Kalman filter introduced previously in

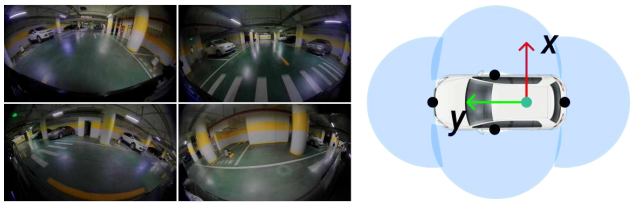


Fig. 5: Illustration of the test platform.

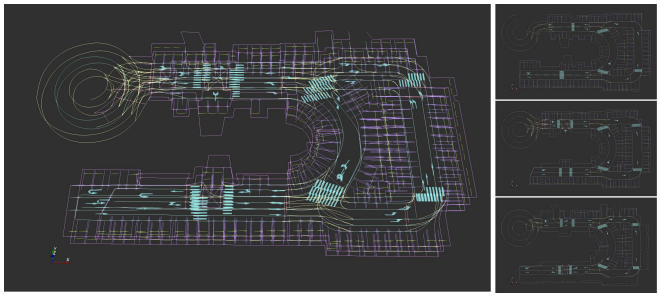


Fig. 6: The parking lot vector map used in our experiments. The parking lot consists of three floors, where each floor is nearly identical.

this section. The solution is based on the assumption that the attitude in the filter has drifted less than $90°$. Under this assumption, we can transform each principal direction to the world frame and flip it if the direction points oppositely to its counterpart in the vector map. In practice, the separate filter is periodically reset so that the assumption will always hold.

Nonlinear refinement. Using the rough reinitialized pose from the last step, we then refine it using the point-to-segment alignment cost in Eq. 3. This is a nonlinear least square problem that can be solved by the Gauss-Newton method. We implement the optimization using Ceres Solve[18].

V. Experiments

A. Test Environments

The test vehicle platform consists of four fisheye cameras, an IMU, and a wheel encoder. Fig. 5 illustrates the sensor setup. The image stream and semantic segmentation both run at 10Hz. IMU and wheel encoder run at 100Hz and 50Hz, respectively. The algorithm runs on an onboard Xavier computer.

We adopt a LiDAR-inertial method[19] to create the point cloud of the parking lots. We use the interactive tool[20] to add loop closures manually if needed. The vector elements are currently annotated manually. How to create the HD vector map efficiently and automatically is currently out of the scope of this paper. Using a LiDAR-based approach to create the HD map is much easier compared to a visual approach. Note that it is ok for the map to contain drift. The map needs only to be locally accurate for relative localization purposes. Fig. 6 shows the parking lot vector map created.

B. Localization Evaluation

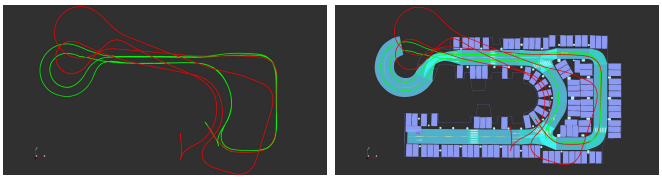
We evaluate the localizer's performance on two indoor parking lot sequences. The parking lot contains three underground floors B2, B3, and B4. The first sequence is a multi-floor sequence (Fig. 7b). It starts from a parking slot in B4, pays a visit to B3, and goes back to starting point in B4. The sequence sequence is a intra-floor sequence (Fig. 7d). It starts from a parking slot in B4, makes two loops, and goes back to the same parking slot. Since the parking lots used in the experiments are GPS-denied, we opt for three alternatives to evaluate the localization accuracy.

First, we co-install a LiDAR on the vehicle platform and compare the BEV trajectory with the trajectory estimated by an NDT[21] style LiDAR-based localizer. We found that the NDT algorithm can produce fairly accurate estimates. Hence we use it as a pseudo ground truth. Fig. 7 provides an quantitative evaluation. Note that the NDT representation and the vector map are created from the same underlying point cloud, making the comparison legitimate. The average translation error is within 10cm. Most of the rotation errors are within $3°$. This experiment shows that our surround-view localizer can achieve a comparable accuracy of a LiDAR localizer while using a much smaller vector map.

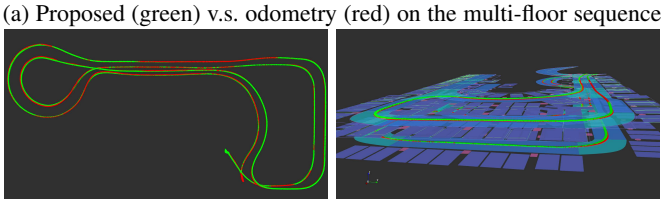
Second, we project the HD vector map onto the BEV image using the estimated pose and evaluate the distances between the correspondences from the map projection and the image content. Fig. 1b shows an example of the vector map projection. Visually distinguishable features such as parking spot corners, arrow corners are chosen as evaluation candidates. When there are no sufficient corner features, we pick arbitrary lane line points and fallback to the point-to-line distance metric. Since the scale difference between the BEV image and the physical world is known, distance measure on the BEV images serves as a reasonable evaluation metric. Fig. 8 visualizes the errors from the distance measures. Since errors in the cameras' extrinsics are magnified as we get close to the BEV border, this experiment tends to output a pessimistically biased evaluation.

Third, we park the vehicle into a parking slot many times and evaluate the distance between the vehicle and the parking lines. We compare the distance output by the localizer and the one measured by tape. The localizer achieves a 2.23cm accuracy in this experiment, as shown in Fig. 9. Since the two parking lines around the vehicle contribute very strong clues for lateral positioning, this experiment tends to output an optimistically biased evaluation.

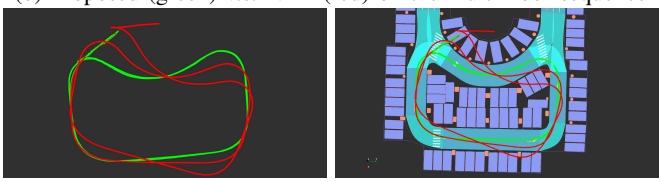
Table I summarizes our localization accuraries using the three evaluation methods above. It also provides a comparison with AVP-SLAM[8] in terms of accuracy and map sizes. Note that [8] uses the third method to evaluate its accuracy. Under the same evaluation method, both approches have similar accuracies. The AVP-SLAM's map size is calculated from the semantic feature map accumulated from the intra-floor sequence's trajectory. The vector map size correspond to all vector elements in the same floor of that sequence, which is an order of magnitude smaller than the feature map.



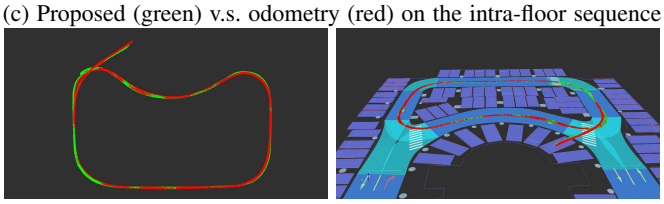
(a) Proposed (green) v.s. odometry (red) on the multi-floor sequence



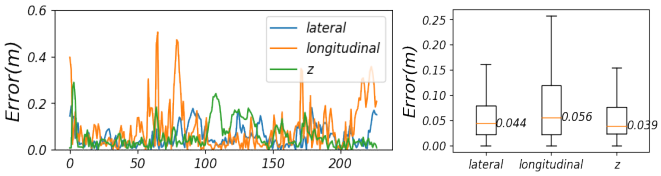
(b) Proposed (green) v.s. NDT (red) on the multi-floor sequence



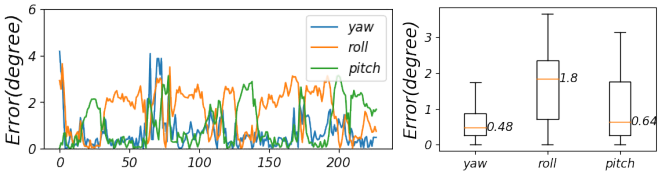
(c) Proposed (green) v.s. odometry (red) on the intra-floor sequence



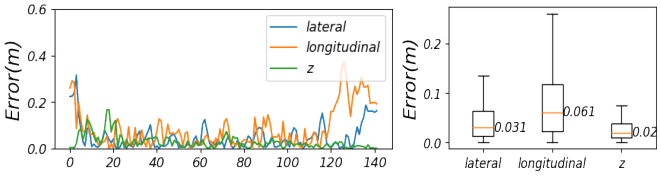
(d) Proposed (green) v.s. NDT (red) on the intra-floor sequence



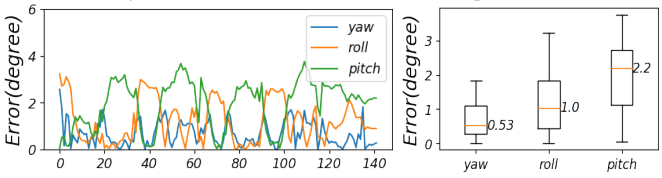
(e) Translation errors (multi-floor sequence)



(f) Rotation errors (multi-floor sequence)

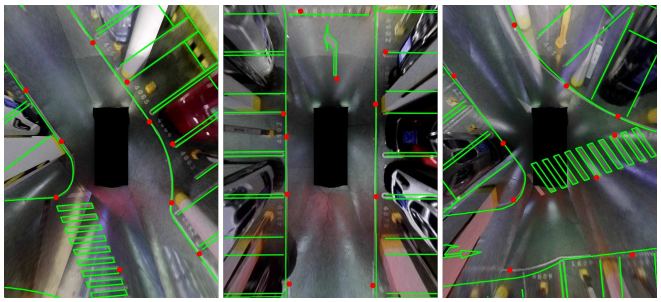


(g) Translation errors (intra-floor sequence)

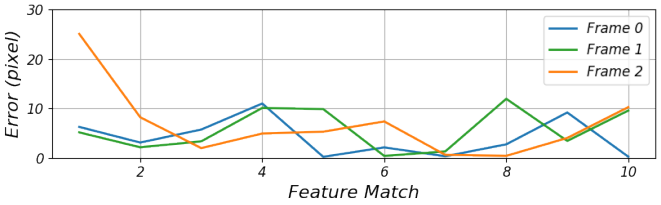


(h) Rotation errors (intra-floor sequence)

Fig. 7: Quantitative evaluation using NDT[21] as pseudo ground truth.



(a) Evaluation positions.



(b) Errors in pixels at evaluation positions.

Fig. 8: Localization errors evaluated from vector map projections.

We also perform an ablation study that compares the full trajectory to the odometry trajectory using IMU and wheel encoder only. The result is shown in Fig. 7a and Fig. 7c. The odometry drifted quickly without the constraints from the vector map. It is worth noting that we have not performed any calibration on the IMU-vehicle extrinsics. Instead, a rough hand-measured value is used. We will expect a better (but still drifted) odometry performance if a better calibration is provided. However, from another perspective, this also shows that our method is not sensitive to the odometry's quality, and is able to turn a rather poor odometry system into a centimeter-accurate localizer.

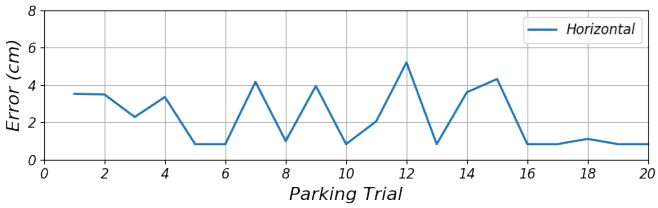


Fig. 9: Lateral localization errors evaluated from 20 parking trials.

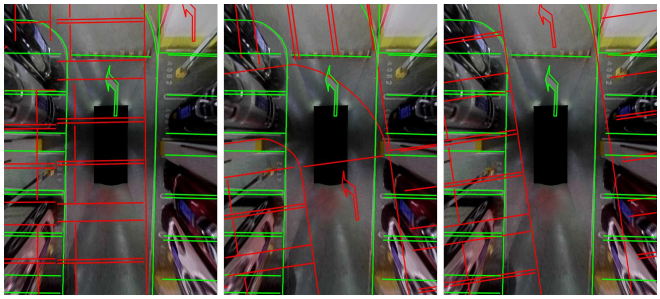
C. Relocalization Evaluation

In our experiments, the proposed localizer is able to navigate the vehicle among the multi-floor parking lots without tracking lost. To evaluate the relocalization performance, we simulate perturbed poses around a good pose, and test how well the perturbed poses can be pulled back to their original pose. Fig. 10a illustrates the relocalization results from three significantly perturbed poses. The relocalization uses the speed bump and arrow combination as the landmark. Because the reinitialization is done in closed-form, the solution will converge to the same minimum regardless of the perturbation noise.

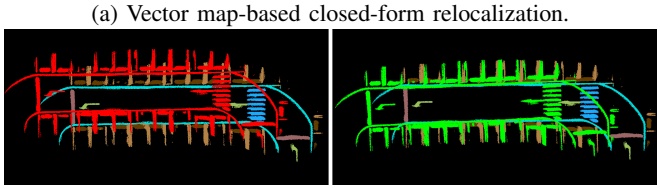
We compare our relocalization performance with AVP-SLAM[8] following the same procedure. AVP-SLAM also

Method	Map size	Eval. method	Mean error(cm)	Max error(cm)
AVP-SLAM[8]	1.24MB	Method-3	2.36	5.23
Ours	0.13MB	Method-1 (lat./lon.)	4.98 / 8.67	31.66 / 37.27
		Method-2	11.06	50.09
		Method-3	2.23	5.20

TABLE I: Comparison of localization accuracies and map sizes. Evaluation methods and map size calculation are detailed in Section V-B.



(a) Vector map-based closed-form relocalization.



(b) Submap and ICP-based relocalization.

Fig. 10: (a) Our landmark-based relocalization strategy is agnostic to the perbured pose (red) and always converges to the true minimum (green), since it is done in closed-form. (b) The submap-based relocalization is sensitive to the initial pose (red), and might converge to a local minimum (green), since it uses ICP.

makes use of semantic segmentation. It performs loop closing (or equivalently, relocalization) by aligning two submaps using ICP. A submap is an accumulation of nearby segmentation results in 3D. Fig. 10b provides some submap examples. Currently, we do not have access to AVP-SLAM's implementation, but it is not difficult to simulate the submaps used for relocalization purposes. To do that, we accumulate the recent BEV segmentations into a semantic point cloud using the (non-drifted) trajectory from our estimator. The submap-based relocalization result is shown in Fig. 10b. Since submap-based relocalization uses ICP, it depends heavily on the proximity of the initial pose. The approach would inevitably converge to local minimums from time to time. In AVP-SLAM, the authors show that AVP-SLAM has a better relocalization success rate than the visual feature approach ORB-SLAM2[1], proving that semantic features have enjoyed better robustness than local features. Hence we do not repeat the experiment here.

VI. Conclusion

We have presented a surround view system that exploits the parking lot's vector map directly for localization as well as for relocalization. At the core, it is a set of novel strategies to associate heterogeneous data in the pixel level, the instance level, and the landmark level. Experiments show the system is able to achieve centimter-level localization accuracy and to relocalize robustly.

References

[1] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[2] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] M. Sons, M. Lauer, C. G. Keller, and C. Stiller, "Mapping and localization using surround view," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1158–1163, IEEE, 2017.

[4] H. Lategahn, J. Beck, and C. Stiller, "Dird is an illumination robust descriptor," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 756–761, IEEE, 2014.

[5] X. Shao, L. Zhang, T. Zhang, Y. Shen, H. Li, and Y. Zhou, "A tightly-coupled semantic slam system with visual, inertial and surround-view sensors for autonomous indoor parking," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2691–2699, 2020.

[6] E. Rehder and A. Albrecht, "Submap-based slam for road markings," in *2014 IEEE intelligent vehicles symposium (IV)*, pp. 1392–1398, IEEE, 2014.

[7] J. Jeong, Y. Cho, and A. Kim, "Road-slam: Road marking based slam with lane-level accuracy," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1736–1473, IEEE, 2017.

[8] T. Qin, T. Chen, Y. Chen, and Q. Su, "Avp-slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot," *arXiv preprint arXiv:2007.01813*, 2020.

[9] Y. Lu, J. Huang, Y.-T. Chen, and B. Heisele, "Monocular localization in urban environments using road markings," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 468–474, IEEE, 2017.

[10] A. Ranganathan, D. Ilstrup, and T. Wu, "Light-weight localization for vehicles using road markings," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 921–927, IEEE, 2013.

[11] T. Wu and A. Ranganathan, "Vehicle localization using road markings," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1184–1190, IEEE, 2012.

[12] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 449–454, IEEE, 2013.

[13] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 646–651, IEEE, 2002.

[14] F. Poggenhans, N. O. Salscheider, and C. Stiller, "Precise localization in high-definition road maps for urban regions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2167–2174, IEEE, 2018.

[15] J. Jeong, Y. Cho, and A. Kim, "Hdmi-loc: Exploiting high definition map image for precise localization via bitwise particle filter," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6310–6317, 2020.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

[17] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[18] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[19] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[20] K. Koide, J. Miura, M. Yokozuka, S. Oishi, and A. Banno, "Interactive 3d graph slam for map correction," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 40–47, 2020.

[21] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.