DT-Loc: Monocular Visual Localization on HD Vector Map Using Distance Transforms of 2D Semantic Detections

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

Abstract— **Localizing a vehicle on a prebuilt HD vector map is a prerequisite for many autonomous driving applications. Existing visual localization approaches usually require a separate local feature layer to function. The separate localization layer suffers from the robustness issue inherited from the local features. Also, it could be difficult to create a feature layer that aligns perfectly with an existing vector map. In this paper, we propose a monocular visual localization method that exploits the vector map directly as the localization layer. The method detects semantic traffic elements from the images and matches them with the vectors in the map. To deal with the harmful problem of false matches, we propose to align the vector map to the distance transforms of the semantic detections, which enables a non-explicit and differentiable data association process. The system is able to achieve centimeter and sub-meter accuracies in lateral and longitudinal directions, respectively.**

I. Introduction

With the rapid development of the autonomous driving industry, many high-level applications nowadays come with a prebuilt HD vector map. A typical HD vector map stores a set of vectorized 3D traffic elements such as lane lines, traffic signs, poles, and the lane topology. A few standards have been proposed to formalize the vector map's specification, e.g., OpenDrive[1], Lanelet2[2]. Localization on the prebuilt vector maps is a crucial prerequisite of these applications.

While GPS can provide centimeter-level accuracy using Real-Time Kinematic (RTK) technique with the assistance of nearby ground stations, it suffers from the multi-path effect and signal blockage in urban canyons, tunnels, or places with an occluded sky. To realize localization without temporary interruptions, additional modalities (e.g., LiDAR, camera) have to be introduced. In this paper, we focus on the camera-based solutions for their low-cost nature and the potential for large-scale commercial applications. Specifically, we focus on the monocular front camera setting since many vehicles with the Advanced Driver-Assistance Systems (ADAS) functionalities already have a front camera installed.

Visual localization has been a long-standing research topic. A large number of visual SLAM researches has been proposed[3], [4], where a robot localizes and simultaneously accumulates a landmark map of visual features during probing. The accumulated map can then be reused to relocalize the robot when the probed regions are revisited from future operations. However, traditional visual approaches suffer from two issues under the context of autonomous driving. First is the robustness problem inherited from the local features. They are not robust in terms of changes of lighting, appearance, and viewing perspectives. Second, it is difficult to create a visual feature map that aligns perfectly with an existing vector map. In general, there will exist non-rigid differences between the feature layer and the vector layer which are produced by different algorithms.

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

To circumvent the above issues, researchers have proposed to match the visual contents directly with the vector map so that a separate visual feature layer is not required. Many of these successful methods[5], [6], [7] are based on a stereo camera setup so that they could convert what the cameras see to 3D before matching with the vector map. Under the monocular setting, the data association process becomes more challenging. Unlike traditional local feature matching, the vector map elements are of similar shapes, nor do they have an easily distinguishable visual descriptor. It is obvious that incorrect 3D-2D matches are impossible to correct afterward.

Motivated by this observation, we propose a monocular visual localization approach, which has the following contributions

- A novel semantic exaction module that regresses traffic lines and poles directly from a neural network.
- A non-explicit and differentiable data association method based on the distance transforms of semantic detections.
- A full 6-DoF monocular visual localization system that achieves centimeter and sub-meter accuracies in lateral and longitudinal directions respectively, using only an HD vector map without a separate localization layer.

II. Related Work

A. Visual Localization Using a Separate Layer

ORB-SLAM3[3] uses ORB descriptor matching for localization. This approach demonstrated excellent accuracy but suffers from long-term robustness issues caused by appearance changes from different illumination conditions and viewing perspectives. To alleviate the long-term robustness problem, Stenborg et al.[8] propose to align the SfM point cloud with semantic class labels to the semantic image segmentations. Experiments showed an increase of robustness at the price of decreased accuracy. Jeong et al.[9] exploit discrete markers extracted from the Inverse Perspective Mapping (IPM) from the front camera for localization and achieves 1m average accuracy. The use of IPM brings could easily introduce large modeling errors when the ground is not level or flat, or when the camera extrinsics are not accurate enough. Wolcott and Eustice[10] proposed to match a LiDAR ground reflectivity map with the raw image intensities using the normalized mutual information. This method's main drawback is its sensitivity to dynamic obstacles, or more generally, when the correlation assumption between the two modalities breaks. Kim et al.[11] instead match the stereo depth map against LiDAR point cloud map. The diversity of the depth gradient is of crucial importance to this method.

B. Visual Localization with Vector Map

Schreiber et al. propose LaneLoc[5]. The authors use oriented filters to detect sample points from lane lines and match them with the corresponding polylines in the vector map. The method relies heavily on the stereo setup for two purposes. First, it uses the stereo point cloud to lift the detected 2D samples to 3D for matching. Second, it uses the stereo depth to compute a V-Disparity[12] image to refine the pitch, whose accuracy is crucial for match fiding. The method tracks a 3DoF pose using the Kalman filter. Ranganathan et al.[13] present a similar idea to LaneLoc[5], but uses a sparser map consisting of the polygon shapes of the discrete road marks, such as arrows. Data association is done through template matching instead of closest match search. Lu et al.[14] proposed to associate the road vector map with edge detection results. However, the detected edges are usually cluttered by many other scene objects, which affects robustness. The authors therefore use epipolar relations from SURF features to further constrain the problem.

Poggenhans et al.[6] detect road markings and curbs from the top view image of the stereo point cloud, and match the detected markings with their counterparts in the vector map under the Unscented Kalman Filter framework. Jeong et al.[7] represent the vector map as 8bit image patches from the top view, where each bit represents the presence of an element class for a given pixel. Localization is done by matching the online 8bit patch to the map patches using a bitwise particle filter. Both approaches require a stereo camera setup. The stereo point cloud enables them to create a top-view image with metric scale, so that map matching can be done in metric space.

Xiao et al.[15] propose to use RANSAC[16] to match vectorized map elements between 3D and 2D. However, this matching method is associated with an inherent limitation. RANSAC works by the assumption that there are enough inliers in the input so that a randomly generated valid correspondence will gain bigger support and eventually wins. However, under the 3D map matching context, the number of semantic features is relatively small (and will be inevitably subject to occlusion, making the candidate set even smaller). The RANSAC assumption might no longer hold. Localization will quickly converge to bad local minimums when the assumption breaks. We use the non-explicit matching scheme in Section III-B to deal with this problem.

III. Approach

Our localizer adopts a factor-graph optimization approach. The graph is a small sliding window of the recent image frames. The system consumes the distance transforms, HD vector map, IMU stream, wheel encoder readings, and outputs optimized poses. Fig. 1 provides an overview of the pipeline.

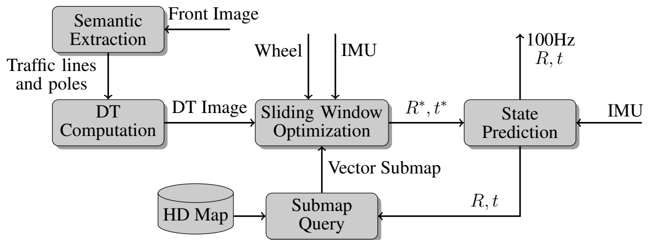


Fig. 1: Overview of the proposed localizer.

A. Semantic Detection

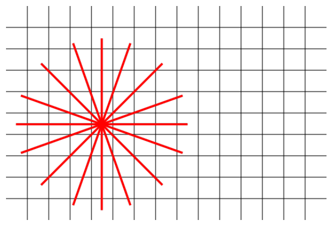
Our goal is to detect traffic lines (e.g., lane lines, stop lines, road curbs) and poles represented respectively as polylines and line segments from the front view images. We fail to find existing methods that detect vectorized traffic lines and poles in a unified framework. Therefore we propose our own. We start with the discussion for traffic line detection and then model pole detection as a special case.

Traffic line detection. The traffic line detection problem has attracted a large amount of research. Among these approaches, some assumes fixed number of lane lines[17], some represents the line as $x = f(y)$ the x coordinate as a function of y coordinate[18], [19]. This representation cannot deal with nearly horizontal lines such as stop lines or even the regular lane lines observed when the vehicle is taking a turn. We propose a novel traffic line detector to deal with both problems. The detector adopts the SSD[20] detection framework. However, instead of regressing bounding boxes, the method regresses polylines.

Traffic line representation. we represent a traffic line as n sample points evenly spread among itself. On one hand, the number of sample points should not be too large to cause difficulty in learning the regressions. On the other hand, the sample count should not be too small to lose the flexibility to model curved shapes. We set $n = 9$ by cross-validation.

Anchor design and prediction. The proposed detector uses the same framework as SSD. But instead of using bounding boxes as anchors, we replace them with oriented straight segments. Specifically, each feature map position is associated with an anchor set, as shown in Fig. 2a. Each anchor is a line segment (θ_i, s_i) representing its orientation and length, whose centroid coincides with the anchor position. For each anchor, we regress C class labels and $2n$ offsets w.r.t. the anchor position. The anchor position plus the $2n$ offsets give the n sample points' final output positions. The anchor set we use includes eight different orientations, where each orientation has three different lengths.

Training loss. Similar to SSD[20], the training objective is a sum of a confidence loss and a localizaiton loss. The condidence loss is a softmax loss over multiple classes, same to that in [20]. Different to SSD, our localization loss is the is defined as the Euclidian distance between the n


(a) Traffic line anchors

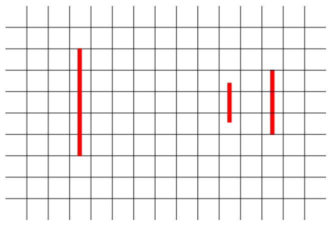

(b) Pole anchors

Fig. 2: Illustration of the anchors used in our semantic detector.

predicated sample points and the ground truth sample points, where each point is stored in a normalized image coordiantes $[0,1] \times [0,1]$. To determine whether an anchor segment matches (θ_i, s_i) with a ground truth traffic line l_j, first we converts the 2-tuple (θ_i, s_i) into the n-point representation, denoted as l_i. We then compute the bidirectional distance between the two polylines l_i, l_j:

$$d(l_i, l_j) = \sum_{p \in l_i} d(p, l_j) + \sum_{q \in l_j} d(q, l_i) \qquad (1)$$

where p, q are the sample points, $d(p, l_j)$ is the point-to-polyline distance, i.e. distance between p and its closest point in l_j. $d(p, l_j)$ has the same meaning. Eq. 1 is also used for non-maximum supression.

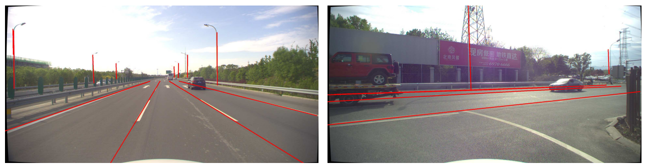
Pole detection. Few literature has directly addressed pole detection from monocular images, perhaps due to its rather specific use case. Here we define the detected poles as 2D line segments. A careful look will reveal that this is a special case of the traffic line detector we just proposed. However, there are three customizations. First, instead of using an anchor set with many orientations, only the vertical direction is needed. Second, instead of regressing $n = 9$ sample points, we only need to regress $n = 2$ sample points corresponding to the top and bottom vertices of the pole. Third, since poles at different depths could appear in a large range of sizes, we introduce more length scales for the pole anchors. Fig. 2b visualizes three scales of the pole anchors on the same feature map.

Training. The two tasks are trained together using a ResNet18[21] backbone network consisting of two heads, one for traffic lines, the other for poles. The traffic line head is trained with the CurveLane-NAS[22] datasets. The pole head is trained with a custom dataset consisting of 10,000 images, collected from the test highway and tunnel environments in Section IV. We use the 512×384 image size for network input. The detector runs at 37Hz and 11Hz on 1080Ti and Xavier respectively. Fig. 3 shows some sample detections.

Discussion. Instead of regressing the traffic lines and poles directly, we also test the alternative approach based on instance segmentation[23]. The method produces a lane line segmentation mask and use mean-shift clustering to obtain the traffic line instances. A polyline is then fitted to each instance. Both [23] and the proposed detector achieve comparable accuracy and recall, but in general the regression approach is faster than the segmentation one, as shown in Table I. Note that the tested alternative[23] only produces lane lines, while ours outputs traffic lines and poles simultaneously.

	1080Ti	Xavier
Proposed	37fps	11fps
Neven et al.[23]	20fps	8fps

TABLE I: Speed comparison between our regression-based method and the segmentation-based approach [23]. [23] contains a mean-shift post-processing step in CPU, whose runtime is not affected by the GPU device. This explains the non-proportional speed ratio on the two devices.


(a) Traffic line and pole detections.

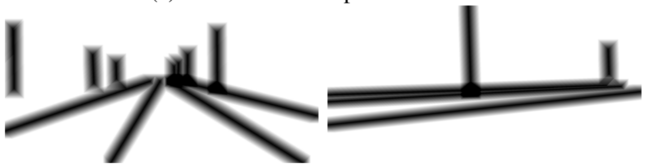

(b) Distance transforms.

Fig. 3: Example semantic detections and distance transforms. (a) Our traffic line representation allows us to detect nearly horizontal lines. (b) In actual implementation, the distance transforms from traffic lines and poles are stored separately to avoid them from interfering with each other.

B. Non-explicit Vector Map Matching

Given a set of detected poles and traffic lines, our goal is to match these semantic elements with their counterparts in the 3D vector map so that constraints could be built for the to-be-estimated state. First, we need to determine the set of candidate 3D vectors for matching. We use the pose $\mathbf{T}$ from the last estimate to retrieve the set of 3D elements within 100m in front of the vehicle.

Now given the candidate set of 3D vectors and 2D vectors, a straightforward way to match them is to project each 3D vector to the image and associate the projection with its closest 2D neighbor. However, missed positives and false alarms occur from time to time. It is obvious that mistakes in the resulted association are impossible to correct. Xiao et al.[15] propose to adopt RANSAC to alleviate the false match issue. Although this might work for common cases, it still fails regularly due to the following reason. At the core of RANSAC algorithm is the assumption that an inlier will get better support (consensus) than an outlier. This assumption often breaks in the vector map matching scenario, as the detected elements could be very sparse. When only a minimal candidate set is available and the set contains false detections, the method is guaranteed to fail because there is no inlier available, but RANSAC is not able to distinguish this situation.

We propose to perform non-explicit data association to deal with the above problem. The non-explicit behavior is realized through the use of distance transforms (DT) of the semantic detections. Each pixel in the DT image represents the distance between the pixel and its closest 2D vector. Fig. 3 illustrates an example. We employ a breadth-first search approach to compute the DT image $\mathcal{D}$. Pixels that overlapped with the detected polylines are seeded with a zero distance. The seeded set then expands one pixel distance at a time until a maximum threshold τ_{DT} is reached. We set τ_{DT} to 5% of the image width in our experiments. After DT computation, an intensity valley will form around each semantic detection. The matching problem now converts to the alignment problem from the 3D vectors to the DT image. Each semantic detection now attracts the projection of its potential match to the basin of its valley. To conduct actual alignment, we generate evenly spaced (i.e., 1m) sample points from the 3D vectors $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, and project them to the image. We seek for a pose-induced projection that minimizes the sum of DT distances of the projected points

$$\sum_{j=1}^{m} \mathcal{D}(\mathbf{x}(\mathbf{X}_j, \mathbf{T})) \tag{2}$$

The sum of distances is naturally differentiable w.r.t. to the pose,

$$\frac{\partial \mathcal{D}}{\partial \mathbf{T}} = \frac{\partial \mathcal{D}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{T}} = \nabla \mathcal{D}(\mathbf{x})^\top \frac{\partial \mathbf{x}}{\partial \mathbf{T}} \tag{3}$$

where $\nabla \mathcal{D}(\mathbf{x})$ is the gradient of the DT image at pixel localization $\mathbf{x}$. The alignment process can then be solved in an optimization framework. The DT-based association also provides a collateral advantage: it enables us to naturally deal with curved lane lines. In explicit matching, even two curved polylines are correctly associated at the instance level, an ambiguity still exists on how the small segments in the polylines correspond to each other.

C. State Estimation

We adaopt a sliding window optimization apparoch for state estimation. The sliding window consists of N states corresponding to the recent N image frames. Each state is represented as

$$\mathbf{x}_i = [\mathbf{p}_i^\top \ \mathbf{v}_i^\top \ \mathbf{q}_i^\top \ \mathbf{b}_{a_i}^\top \ \mathbf{b}_{g_i}^\top]^\top, \ i \in [1, \ldots, N] \tag{4}$$

where the five components correspond respectively to the IMU position in world frame, the IMU velocity in world frame, the quaternion form IMU body frame to world frame, and the accelerometer and gyroscope biases. N ranges from 2 to 5 inpractice. We optimize for the following objective in the sliding window

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_N} \left\{ \sum_{i=1}^{N} \left(\|\mathbf{r}_{\text{DT}}(\mathbf{x}_i)\|^2 + \|\mathbf{r}_{\text{RP}}(\mathbf{x}_i)\|^2 + \|\mathbf{r}_{\text{WS}}(\mathbf{x}_i)\|^2 \right) + \sum_{i=1}^{N-1} \|\mathbf{r}_{\text{IP}}(\mathbf{x}_i, \mathbf{x}_{i+1})\|^2_{\mathbf{\Sigma}} \right\} \tag{5}$$

where the abbreviations DT, WS, RP, and IP stand for Distance-Transform, Road-Plane, Wheel-Speed and IMU-Preintegration respectively.

1) The distance transform factor. This factor seeks to align the 3D vectors to the distace transforms of the 2D detections

$$\mathbf{r}_{\text{DT}}(\mathbf{x}_i) = [\mathbf{r}_{\text{DT}}(\mathbf{x}_i, X_1), \ldots, \mathbf{r}_{\text{DT}}(\mathbf{x}_i, X_m)]^\top \tag{6}$$

where

$$\mathbf{r}_{\text{DT}}(\mathbf{x}_i, X_j) = \mathcal{D}_i(\pi(\mathbf{T}_I^C(\mathbf{q}_i \otimes X_j + \mathbf{p}_i))) \tag{7}$$

Here, $\mathcal{D}_i$ denotes the distance transform of the ith image frame; $\{\mathbf{X}_j\}_{j=1}^m$ is the set of 3D points sampled from the set of the elements retrieved from the map query stage described in section III-B. Each 3D polyline or line segment is typically sampled with a $1m$ step size; The symbol $\otimes$ denotes a quaternion product with a 3D point; π represents the camera projection operator using the pin-hole model. $\mathbf{T}_I^C$ denotes the transformation from the IMU frame to the camera frame.

2) The road plane factor. We define the the vehicle frame's origin to be the rear axle's projection onto the ground plane. The road plane factor wants the vehicle frame's origin to always lie on the ground plane from the HD vector map

$$\mathbf{r}_{\text{RP}}(\mathbf{x}_i) = (\mathbf{q}_i \otimes \mathbf{t}_V^I + \mathbf{p}_i - \mathbf{c}_i)^\top \mathbf{n}_i \tag{8}$$

where $(\mathbf{c}_i, \mathbf{n}_i)$ are respectively the pivot point and the normal of the HD map road plane in world coordinates. $\mathbf{q}_i \otimes \mathbf{t}_V^I + \mathbf{p}_i$ is the vehicle frame's origin in world coordinates, tranferred through the IMU pose and the IMU-vehicle calibration.

3) The wheel speed factor. The wheel speed factor seeks for an agreement on the vehicle frame's origin's velocity, observed respectively from the IMU and the wheel encoders

$$\mathbf{r}_{\text{WS}}(\mathbf{x}_i) = \mathbf{q}_i^{-1} \otimes \mathbf{v}_i + [\omega_i - \mathbf{b}_{g_i}]_\times \mathbf{t}_V^I \tag{9}$$

$$- \mathbf{R}_V^I \mathbf{v}_i^V \tag{10}$$

where $\mathbf{v}_i^V$ is the vehicle origin's velocity in the vehicle frame measured by the wheel encoders. $\mathbf{R}_V^I \mathbf{v}_i^V$ is the same velocity but transferred into the IMU frame. The right-hand side of Eq. 9 also expresses the vehicle origin's velocity in the IMU frame, but the velocity is measured from the IMU instead of the wheel encoders. $\mathbf{q}_i^{-1} \otimes \mathbf{v}_i$ is the IMU origin's velocity in the IMU frame. $[\omega_i - \mathbf{b}_{g_i}]_\times \mathbf{t}_V^I$ is the lever-arm compensation for the vehicle's origin under the rotating IMU frame, where $\omega_i - \mathbf{b}_{g_i}$ denotes the measured angular velocity.

4) The IMU preintegration factor. The IMU preintegration factor wants the two states associated to two adjacent image frames to agree with the IMU measurements accumulated between these two image frames. The factor is expressed as

$$\mathbf{r}_{\text{IP}}(\mathbf{x}_i, \mathbf{x}_j) = \Delta \mathbf{x}_{ij} - \Delta \hat{\mathbf{x}}_{ij} \tag{11}$$

where

$$\Delta \hat{\mathbf{x}}_{ij} \triangleq [\boldsymbol{\alpha}_j^{i\top} \ \boldsymbol{\beta}_j^{i\top} \ \boldsymbol{\gamma}_j^{i\top} \ \mathbf{0}^\top \ \mathbf{0}^\top]^\top \tag{12}$$

is the IMU preintegration[24] between frame i and frame j, integrated from the IMU measurements. The preintegration can be interpretated as the nominal state evaluated at the end of the preintegration process, whose kinematics is defined as

$$\dot{\boldsymbol{\alpha}}_t^i = \boldsymbol{\beta}_t^i \tag{13}$$

$$\dot{\boldsymbol{\beta}}_t^i = \mathbf{R}_t^i(\hat{\mathbf{a}}_t - \mathbf{b}_{at} - \mathbf{n}_a) \tag{14}$$

$$\dot{\boldsymbol{\gamma}}_t^i = \frac{1}{2}\boldsymbol{\gamma}_t^i \otimes (\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{gt} - \mathbf{n}_g) \tag{15}$$

where $\hat{\mathbf{a}}_t, \mathbf{n}_a, \hat{\boldsymbol{\omega}}_t, \mathbf{n}_g$ are respectively the readings and noises from the accelerometer and gyroscope. Here, $\boldsymbol{\alpha}_t^i, \boldsymbol{\beta}_t^i, \boldsymbol{\gamma}_t^i$ are respectively the *pseudo* changes of position, velocity, and rotation between the virtual frame at time t and frame i. The pseudo changes are deliberatedly defined in a way such that the changes do not depend on the initial IMU states and the gravity, so that changes of IMU states during optimization will not cause the expensive integration operation to be recomputed. Note that for the rotational component, the pseudo change and the physical change are identical. The preintegration process starts from a zero state at frame i and ends at frame j. The $\boldsymbol{\Sigma}$ matrix in Eq. 5 is the 15×15 covariance matrix associated to the error state at the end of the preintegration process.

So far, we know $\Delta\hat{\mathbf{x}}_{ij}$ is the preintegration computed from the IMU measurements. Now,

$$\Delta\mathbf{x}_{ij} \triangleq \begin{bmatrix} \mathbf{R}_i^\top(\mathbf{p}_j - \mathbf{p}_i - \frac{1}{2}\mathbf{g}\Delta t^2 - \mathbf{v}_i\Delta t) \\ \mathbf{R}_i^\top(\mathbf{v}_j - \mathbf{g}\Delta t - \mathbf{v}_i) \\ \mathbf{q}_i^{-1} \otimes \mathbf{q}_j \\ \mathbf{b}_{aj} - \mathbf{b}_{ai} \\ \mathbf{b}_{gj} - \mathbf{b}_{gi} \end{bmatrix} \tag{16}$$

is also the preintegration, but computed from the states $\mathbf{x}_i$ and $\mathbf{x}_j$ to be optimized. The IMU preintegration factor wants to minimize the difference between the preintegrations computed from the two paths. Please refer to [24] for the derivation details.

Eq. 5 is a nonlinear least square problem and can be optimized by the Gauss-Newton method. We employ Ceres Solver[25] for the implementation.

D. State Predictor

The optimized poses output by the sliding window optimizer are not real-time poses. They are delayed by the elapsed time of semantic detection and optimization. To output real-time poses, we maintain a state predictor in a separate thread. The state predictor maintains a *base state* and the IMU stream that arrives after the base state. The real-time poses are integrated from the IMU stream using the base state as the initial condition. The base state is regularly updated when an optimized pose is produced from the sliding window optimizer. In our system, the sliding window optimizer runs at 10Hz, while the state predictor is triggered by a 100Hz timer.

IV. Experiments

Our test vehicle platform consists of a front camera, an IMU, and the wheel encoder, which run at 10Hz, 100Hz, and 50Hz, respectively. A NovAtel PwrPak7D-E1 INS module is installed side to obtain a pseudo ground truth trajectory, which can achieve $< 10cm$ accuracy with RTK enabled. Fig. 4 shows our test platform. The LiDAR is not used in our experiments. The program runs on an Nvidia Xavier computer with an 8-core ARM CPU, and a 512-core Volta GPU, and 32GB memory. We test our methods on both the highway and the tunnel environments. The HD map vector maps are provided by a third-party map provider, geo-referenced in WGS84 coordinates, as shown in Fig. 5. The vector map contains abundant classes of elements, while currently, only lane lines and poles are used in our experiments. The map is converted to local ENU (East-North-Up) coordinates for distance transform alignment.

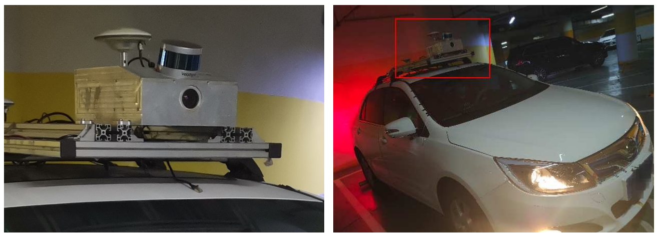


Fig. 4: The test platform used in our experiments.

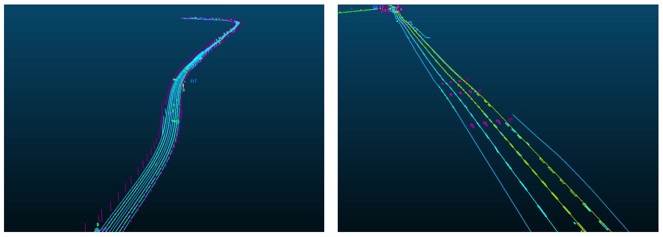
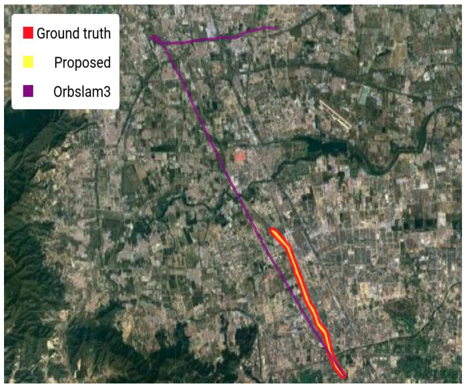

(a) G7 Highway (b) Tunnel

Fig. 5: The two HD vector maps used in our experiments.

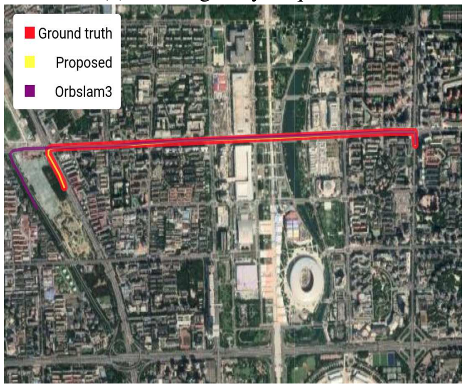
A. Quantitative Evaluation

We test our method on two sequences. The first sequence makes a round trip on the 11km highway, totaling a distance of 22km. The second sequence navigates through four consecutive long tunnels, totaling 1.2km. The trajectory output from the NovAtel INS module is used as ground truth. Fig. 6 shows the quantitative evaluation results. Not that the localizer does not use the GPS during the whole ride except for initialization.

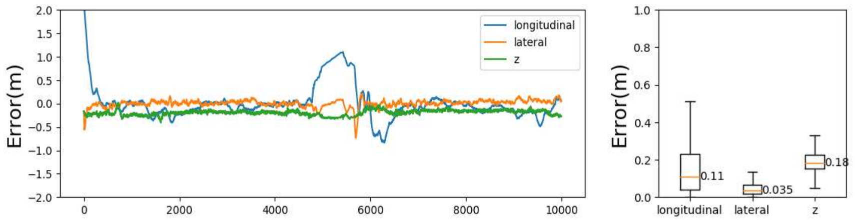
For the highway sequence, the lateral and longitudinal errors are within 10cm and 20cm, respectively, for the majority of cases. Some interesting cases worth elaboration. The first few frames in Fig. 6c shows large errors on the xy-plane. This is caused by the rough initial position given. In frame 5000 to 5700, there exists a roughly one-meter longitudinal error. This is caused by a temporal traffic jam, as shown in Fig. 7a. The car has only moved for a short distance during this period, and the only longitudinal clues are three very distant poles. Unlike close poles, distant poles are less effective in constraining longitudinal position since, for faraway poles (e.g., 100m away), a longitudinal difference (e.g., 1m) in 3D only causes a limited difference (e.g., 1px) in image coordinates.



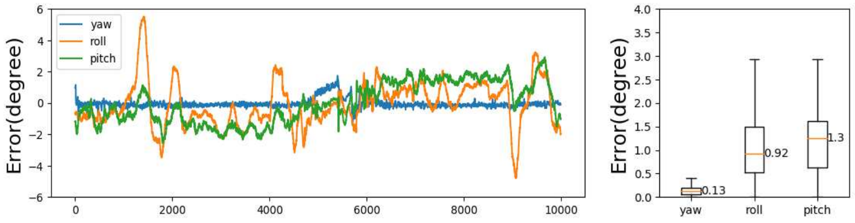
(a) The highway sequence.



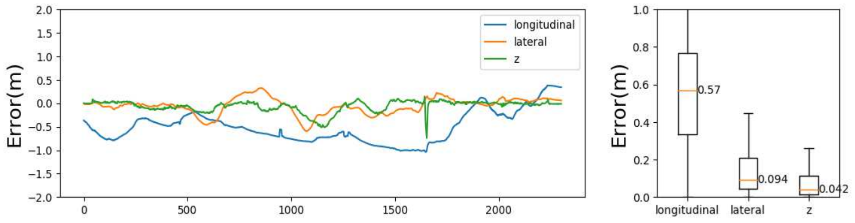
(b) The tunnel sequence.



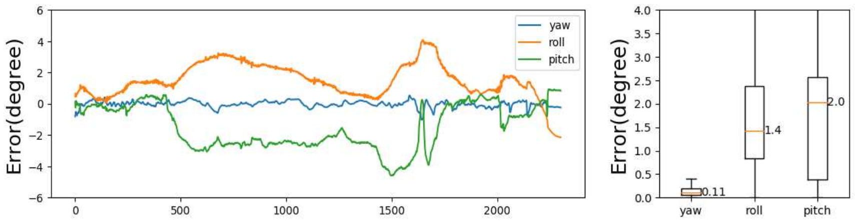
(c) Translation errors on the highway sequence.



(d) Rotation errors on the highway sequence.



(e) Translation errors on the tunnel sequence.



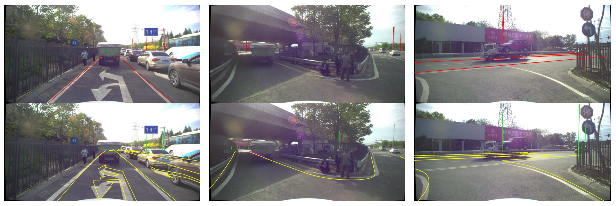
(f) Rotation errors on the tunnel sequence.

Fig. 6: A quantitative evaluation of the proposed localizer on both highway and tunnel environments. The RTK trajectory from the NovAtel INS module is used as ground truth.

For the tunnel sequence, the lateral errors are within 10cm for most cases. However, the longitudinal errors are much larger, which are within 60cm for most cases. This is caused by the lack of longitudinal clues (i.e. poles) inside the tunnels. Errors in roll and pitch also tend to be larger compared to the highway sequence. We suspect the reason lies in the ground truth trajectory. Since the GPS is blocked under the consecutive tunnels, and the tunnels contain many up-hill and down-hill segments during the course, the ground trajectory might also accumulate attitude drift. The hypothesis is supported by vector map projection induced by the GNSS pose in Fig. 8, which shows visible pitch error.

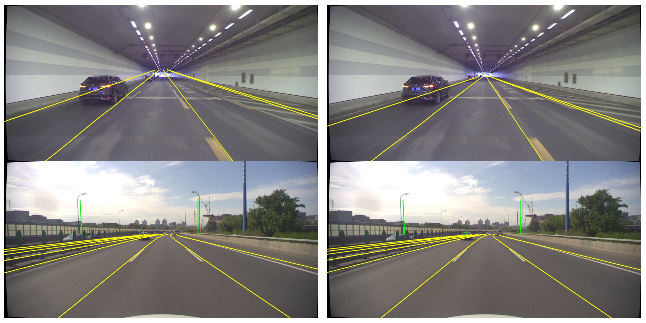
B. Qualitative Evaluation

To visualize the solution quality more intuitively, we project the HD vector map onto the image using the localizer's output pose. Fig. 7 illustrates a few examples across various scenarios. In the test sequences, the width of the lane lines and the dash segments is 15cm. If a lane line's projection lies inside the 15cm road mark, it means the projection is offset by at most 7.5cm. Our projections lie within the 15cm wide road marks in most cases. An interesting phenomenon is that poles at the right image boundary seem to have larger misalignments than others. The



(a) Traffic jam (b) First left turn (c) Second left turn

Fig. 7: Frames 5300, 5700, and 6000 from the highway sequence. First row is the semantic detections. Second row is the vector map projections.



(a) NovAtel INS module (b) Ours

Fig. 8: Comparison of vector map projections between NovAtel INS module and ours. Although NovAtel's trajectories are used as ground truth, they are not always accurate when GPS is blocked. (a) Both projctions look similar in highway. (b) NovAtel's projection has larger pitch error in tunnel.

following reason could cause this phenomenon. A 1° yaw error could result in a 30 to 70-pixel shift in our 1920px-wide image. The closer the pole, the larger the shift. Since the vehicle is driving on the right side of the road, we get closer poles from the right border than from the left. Hence yaw errors are more visually magnified on the right image border.

We also compare our approach to the start-of-the-art visual-inertial method ORB-SLAM3[3]. However, it is difficult to create an ORB feature map that aligns well with an existing vector map since there is no obvious method to associate between sparse features and vectors directly, and there will exist non-rigid differences between the maps. Therefore, we opt for a qualitative comparison. Specifically, we run ORB-SLAM3 on the two test sequences and compare the trajectories to ours. Since the output trajectories are in their own local coordinates, we manually align them to the map. The aligned trajectories are shown as purple in Fig. 6a and Fig. 6b. The trajectories show large orientation drift and scale drift. Note that due to experiment restriction, this is by no means a fair comparison, but it shows intuitively how the vector map could be exploited to turn a drifted odometry into a map-aligned trajectory.

C. *Ablation Study*

We perform an ablation study to access the contributions from poles and lane lines respectively.

Fig. 9a shows the errors of the pole-only trajectory when lane lines are removed. The result error curve demonstrates similar statistics to the full method in Fig. 6c. The lateral median error becomes larger, while the longitudinal error stays relatively the same. Albeit a slightly worse lateral error, the experiment implies that poles alone provide enough constraints for a good enough localization in both lateral and longitudinal directions. During frame 5000 to 5700 traffic jam, the pole-only setting demonstrates the same longitudinal drift behavior to the full method. This is expected since few lane lines can be observed during this same, making full method effectively degenerate to the pole-only setting during these frames.

Fig. 9b shows the trajectory errors of the lane line-only trajectory when poles are disabled. The trajectory maintains excellent lateral accuracy but demonstrates considerable longitudinal drift. The reason is that a longitudinal drift on a straight road does not affect the lane lines' appearance in the front image. As time goes by, the small drifts accumulate into a large drift in the longitudinal direction. At frame 5700, there exists a sudden peak in the lateral direction. This is caused by the vehicle making its first left turn from the main road into the u-turn ramp, which effectively converts the longitudinal error into the lateral error. The drift continues to increase by suddenly drops to zero at frame 6000. This is when the vehicle is making the second left turn to merge back into the main road. At this moment, the main road's lane lines all appear horizontally in the image, providing strong longitudinal constraints, which brings the system's longitudinal error back to zero.

In summary, the lane line-only setting has better lateral accuracy than the pole-only setting but has poor longitudinal accuracy. The pole-only solution has slightly worse lateral accuracy but good longitudinal accuracy. Fig. 9a and Fig. 9b also compares the vector map's projections on the u-turn ramp. Albeit having an unknown drift, the lane line-only setting shows a well-aligned vector map, while the pole-only solution shows laterally drifted lane lines.

D. *Runtime*

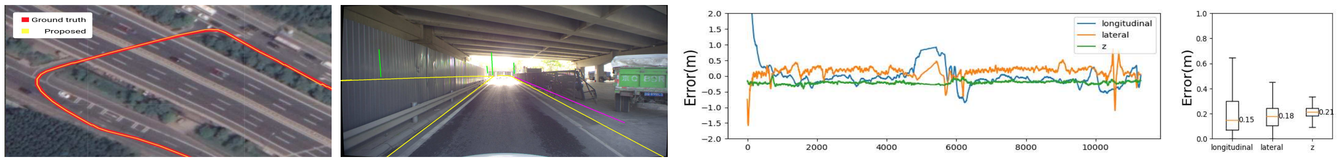
The semantic detection module can run at slightly more than 10Hz on GPU, while the distance transform computation and sliding window optimization combined can run greater than 10Hz on CPU. Sicne the optimization is triggered when a new DT image is received, and the front image stream is configured to 10Hz, the optimization thread runs at 10Hz. Since the CPU and the GPU tasks are pipelined, the optimized poses are at least one frame late. These late poses are used to update the base state in the state predictor introduced in Section III-D, which then outputs real-time fresh poses by integrating the IMU measurements starting from the base state.

V. Conclusion

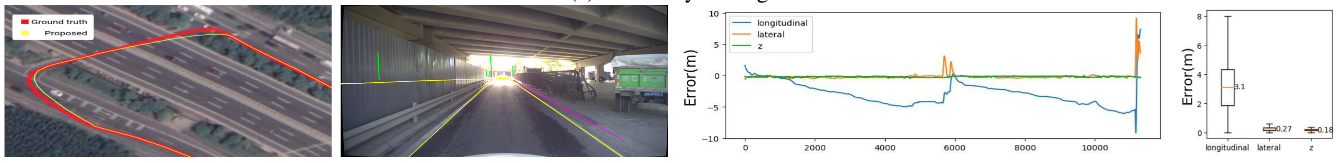
We have a presented monocular visual localization system that exploits the HD vector map directly as the localization target. The system does not require explicit data association but instead makes use of distance transforms of semantic detections to enable a non-explicit differentiable data matching process. The system is robust due to the semantic features used and is able to achieve centimeter-level and submeter-level accuray in lateral and logitudinal directions respectively. A future work is to extend the method to multi-view camera systems and to more complex urban scenes.

References

[1] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, pp. 231–242, 2010.



(a) Pole-only setting.



(b) Lane line-only setting.

Fig. 9: An ablation study by removing lane lines and poles respectively from the vector map. The satellite images show the data segment between frame 5000 to 7000, where the vehicle encountered a traffic jam and then made a large U-turn. (a) The pole-only setting sees degradation in both lateral and longitudinal accuracies but still manage to achieve $< 30cm$ errors for most cases. (b) The lane line-only setting maintains an excellent lateral accuracy but degrades significantly in the longitudinal direction. Please see the text for detailed elaboration.

[2] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1672–1679, IEEE, 2018.

[3] 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.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 15–22, IEEE, 2014.

[5] 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.

[6] 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.

[7] 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.

[8] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term visual localization using semantically segmented images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6484–6490, IEEE, 2018.

[9] 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.

[10] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–183, IEEE, 2014.

[11] Y. Kim, J. Jeong, and A. Kim, "Stereo camera localization in 3d lidar maps," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, IEEE, 2018.

[12] 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.

[13] 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.

[14] 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.

[15] Z. Xiao, D. Yang, T. Wen, K. Jiang, and R. Yan, "Monocular localization with vector hd map (mlvhm): A low-cost method for commercial ivs," *Sensors*, vol. 20, no. 7, p. 1870, 2020.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] J. Kim and C. Park, "End-to-end ego lane estimation based on sequential transfer learning for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 30–38, 2017.

[18] J. Philion, "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11582–11591, 2019.

[19] X. Li, J. Li, X. Hu, and J. Yang, "Line-cnn: End-to-end traffic line detection with line proposal unit," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 248–258, 2019.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[22] Z. Li, "Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending," 2020.

[23] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE intelligent vehicles symposium (IV)*, pp. 286–291, IEEE, 2018.

[24] 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.

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