IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

HDMI-Loc: Exploiting High Definition Map Image for Precise Localization via Bitwise Particle Filter

Jinyong Jeong[1], Younggun Cho[1] and Ayoung Kim[1*]

***Abstract*— In this paper, we propose a method for accurately estimating the 6-Degree Of Freedom (DOF) pose in an urban environment when a High Definition (HD) map is available. An HD map expresses 3D geometric data with semantic information in a compressed format and thus is more memory-efficient than point cloud maps. The small capacity of HD maps can be a significant advantage for autonomous vehicles in terms of map storage and updates within a large urban area. Unfortunately, existing approaches failed to sufficiently exploit HD maps by only estimating partial pose. In this study, we present a full 6-DOF localization against an HD map using an onboard stereo camera with semantic information from roads. We introduce an 8-bit representation for road information, which allow for effective bitwise operation when matching between query data and the HD map. For the pose estimation, we leverage a particle filter followed by a full 6-DOF pose optimization. Our experimental results show a median error of approximately 0.3 m in the lateral and longitudinal directions for a drive of approximately 11 km. These results can be used by autonomous vehicles to correct the global position without using Global Positioning System (GPS) data in highly complex urban environments. The median operation speed is approximately 60 msec supporting 10 Hz.**

I. Introduction

Despite the increased interest in autonomous vehicle research, perception-based localization in a complex urban environment still addresses challenges. The widely used GPS is at times limited due to the multi-path issue caused by high-rise buildings. To overcome this limitation, perceptual sensors such as the Light Detection and Ranging (LiDAR) sensor and cameras have been highlighted as a solution. Several methods involving LiDAR simultaneous localization and mapping (SLAM) [1] and visual SLAM [2] reported incredible performance without requiring prior knowledge of an environment.

However, *what if the prior knowledge is available?* With an available prior map, SLAM problems become localization problems for achieving localization accuracy that can overcome potentially different sensor modalities between the query and the prior map. Widely adopted prior maps can be categorized into three types: two-dimensional (2D) digital maps, pointclouds, and vectorized semantic labeled maps.

J. Jeong, Y. Cho and A. Kim are with the Department of Civil and Environmental Engineering, KAIST, Daejeon, S. Korea [jjy0923, yg.cho, ayoungk]@kaist.ac.kr

This work is supported through a grant by [Localization in changing city] project funded by Naver Labs Corporation. J. Jeong was financially supported by Korea MOLIT as Innovative Talent Education Program for Smart City.

A. 2D Digital Map-based Localization

Recent studies indicated that even publicly available 2D digital maps (e.g., low-resolution aerial images and vectorized 2D maps) may be sufficiently informative for approximate global positioning. Hu et al. [3] used road markings and poles extracted from aerial images as landmarks. Pole-shaped features were estimated from pointclouds obtained through LiDAR, and road markings such as arrows, stop-lines, and pedestrian crossings were detected using images. In this study, the vehicle pose was estimated by matching these features with landmarks. Pink et al. [4] used Support Vector Machine (SVM) algorithm to identify lane pixels in aerial images. The markings were detected using a Canny edge detector [5], and the centroids of each marking were estimated. Finally, the vehicle position was calculated by matching lane markings from an aerial image with the estimated centroids. Roh et al. [6] utilized the direction of walls extracted from an aerial image to correct only the vehicle heading angle. Unfortunately, public aerial images are limited in terms of resolution, which affects the accuracy of pose estimation. Moreover, a pose estimation error may occur depending on the method in which features are extracted from aerial images.

Roh et al. [7] and Vysotska and Stachniss [8] used 2D digital maps that store building exterior wall data extracted from aerial images in the form of vectors. By utilizing the digital map, stable performance could be guaranteed by minimizing the errors that may occur when extracting features from aerial images. However, as aerial images and digital map data only contain 2D information, it is insufficient to estimate the 6 DOF pose in global coordinate.

B. 3D Pointcloud Map-based Localization

Prior maps can also be constructed from LiDARs. Several studies approached the global positioning problem by localizing against prior three-dimensional (3D) pointcloud maps. Barsan et al. [9] used the intensity information of pointcloud map with deep learning techniques to estimate vehicle pose. Wolcott and Eustice [10] estimated a vehicle's pose by calculating the Normalized Mutual Information (NMI) between rendered images of the 3D pointcloud in the synthetic view and the actual camera image. In this method, the performance of localization depends on the resolution of the synthetic view: The higher the resolution, the greater the amount of computation. As each LiDAR sensor has its own intensity characteristics, performance generalization is challenging when using intensity values. Kim et al. [11] performed localization on a 3D map using stereo camera

images. By using the depth of each pixel calculated from a stereo image, the pose was estimated through registration between a pointcloud of a stereo camera and a 3D map.

To enhance visual localization against a pointcloud map, certain studies have reported methods of inserting associated visual features into the map [12, 13]. When building a prior pointcloud map using cameras and LiDARs, image features corresponding to 3D points were embedded into map points. Unfortunately, the feature-based maps are difficult to generalize over different types of features and are greatly affected by appearance discrepancies from the map acquisition time.

As reported in the literature, geometric information in pointclouds are less affected by illumination variance and are thus robust to visual changes in the environment. Nevertheless, using 3D pointcloud to express the same area requires more storage than other data formats. This is a potentially feasible solution providing accurate localization performance for small area; however, once expanded to a larger scale, storage and real-time data transmission limitations become a critical issue.

C. HD Map-based Localization

The major limitation of the aforementioned aerial image and pointcloud maps involves memory-ineffectiveness. To resolve this issue, the compact and semantic 3D representation of HD maps have become a focus. HD maps include semantic information in the form of compressed 3D vectors, constructed from pointclouds obtained from high-precision Mobile Mapping System (MMS) or from high-resolution aerial/satellite images. As HD maps have semantic information and global coordinates in a vector form, the map can represent large areas with small capacities. For example, HD map data provided by Naver Labs [13] encapsulate a 11×11 km area within 17 Mbyte. This is 30 times more efficient, considering that the size of the pointcloud map data of the same area in the Complex Urban Dataset [14] is approximately 500 Mbyte. Smaller amounts of data can extend the coverage of the map and allow for faster map updates online.

Ma et al. [15] used HD maps consisting of lane and traffic sign information. The authors compared query LiDAR data against HD maps, thereby reducing errors in the lateral (lanes) and longitudinal (traffic signs) directions. Contrary to the well maintained lateral localization, longitudinal localization performed less effectively due to data sparsity. Welte et al. [16] estimated the vehicle position using markings stored on a map via the Kalman smoothing process. Unfortunately, this paper only considers the accuracy of the lateral direction. Poggenhans et al. [17] used an HD map that included road markings and road border information. Each detector extracted features from the stereo image input and used the Unscented Kalman Filter (UKF) to estimate a location by matching the features with the map. Vivacqua et al. [18] estimated position by matching the points of lane markings that were estimated by a camera with the lane information of an HD map.

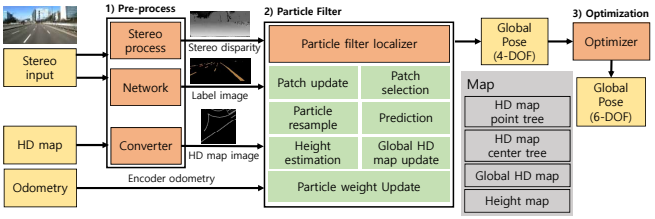


Fig. 1: Given stereo camera images, HD maps, and odometry, we estimate a full 6 DOF pose in global coordinates.

However, as HD map data is not a form of data directly obtained from LiDARs or cameras, pose estimation using HD maps requires matching between heterogeneous data. All of these aforementioned existing methods only provide a partial pose estimation, not a full 6-DOF. The most critical limitation of these existing methods is the lower accuracy in the longitudinal direction, especially when road information is sparse. This paper proposes a global localization method with sub-meter level accuracy by using label information included in HD maps and semantic information estimated from images through Convolutional Neural Network (CNN). For matching measurements and map data, a bitwise particle filter was used instead of an extended Kalman filter (EKF) or histogram filter, which were used in previous researches. Since the particle filter can adjust the number and distribution of particles according to the environment, it may be more robust to the various environments than other filters.

D. Overview and Contribution

We believe that using HD maps in the localization phase would improve both the localization accuracy and memory efficiency. Sharing the same motivation with existing HD map-based localization, we believe that HD maps are highly useful representation that requires less storage space. Unlike the previous methods, we estimate a full 6-DOF pose against an HD map together with a substantially improved localization performance. More importantly, the proposed method presents well maintained longitudinal localization even in environments with sparse road information. The following are the contributions of the proposed method.

- The proposed method infers a full 6-DOF pose against a prior HD map using only stereo camera and odometry data.
- Compared to odometry, we achieve substantial localization accuracy improvements for the longitudinal direction regardless of road marking sparsity.
- We introduce an 8-bit image representation for semantic road information and implement bitwise operations for fast computation speeds.

II. Proposed Method

The HD map used in the algorithm is vector-format data that are made publicly available by Naver Labs [13]. The map data in the shapefile format consists of labels that include lanes, stop lines, direction signs, and surface signs. On the query side, we exploit stereo camera images and

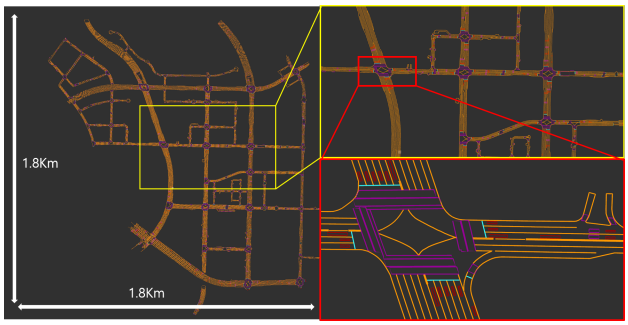


Fig. 2: HD map 3D vectorized data of the Pangyo area used in the experiment. The total area of the map is approximately 1.8×1.8 km and consists of lanes (orange), stop lines (sky blue), arrow signs (red), and other surface signs (purple).

odometry data from the same location [14]. Odometry was calculated based on differential wheel model using encoder counter data such as [6].

The proposed method is divided into three processes, with the overall process depicted in Fig. 1. First, disparity images and semantic labels are prepared from stereo images. In this preparation phase, we also convert HD map data from shapefiles to 8-bit images. Next, using the particle filter, we estimate the 4-DOF partial pose (i.e., 3-DOF translation and heading) of the vehicle through image matching of the patch and the global HD map image. Both query patches and the global HD map image are 8-bit images, thus the bitwise AND operation is used for fast computation. For the final step, optimization completes the full 6-DOF pose with respect to the HD map by additionally calculating roll and pitch.

A. Notation

Matrix $\mathbf{T}_B^A \in SE(3)$ represents the rigid body transformation that registers the data defined in coordinate system B to coordinate system A. This matrix is also represented by a tuple $\mathbf{t}_B^A \in \mathbb{R}^6$, where $\mathbf{t}_B^A = \{t_x, t_y, t_z, r_x, r_y, r_z\}$. t_x, t_y, t_z represent the translation along each axis in meters, and r_x, r_y, r_z represent the rotation along each axis in radian. ${}^C P_{D,t}$ represents the pointcloud expressed in coordinate C obtained from sensor D at time t. The subscripts S, V, G, and p_i refer to stereo, vehicle, global, and the i-th patch.

B. Pre-processing and 8-bit Representation

Prior to matching, we convert the HD map vector data into an 8-bit representation called *tile* and save it as image files. We also simultaneously construct a tile search tree to find the tile closest to the most recent vehicle pose. This pre-processing is run once before the main process.

1) HD map to 8 bit image tiles: On the HD map side, we convert the original shapefile (Fig. 2) to an 8-bit image. The first, second, and third bit of the 8-bit image represent lanes, stop lines, and signs, respectively (i.e., 128: lane, 64: stop line, 32: sign). The converted HD map image is divided according to a predefined map size ($l_{map} = 30$) and stored with the Universal Transverse Mercator (UTM) global coordinate of each image center point. The images in the red grid in Fig. 3 are the divided images, which are referred to as *tiles* in this manuscript. In the same manner, the 32-bit images containing the height information of the HD map are created and stored together.

Fig. 3: Candidate tileset creation for matching. The image in the red grid is a tile. Each bit of the 8-bit image tile represents a label. The neighboring tiles are found based on the estimated global coordinates of the vehicle and the tileset are incrementally expanded and contracted.

Fig. 3 shows a global HD map image composed of tiles that were stored in the pre-process. Each pixel of the image is represented in the 8-bit format, and each bit includes the label information from the HD map. The global HD map image is expanded and contracted based on the vehicle pose that is estimated by a particle filter and the global location of the patches. This expansion and contraction process is performed during the global HD map update process.

2) Search Tree for HD Map Tiles: During the pre-processing, an HD map center tree and an HD map point tree are generated. The HD map center tree is composed of the center coordinates of each tile. It is utilized to quickly retrieve the nearest tile based on the vehicle pose during the candidate tileset creation process (the expansion and contraction). The HD map point tree is composed of points constituting the shapefile. This tree is used to quickly obtain height values at the pose calculated by the particle filter.

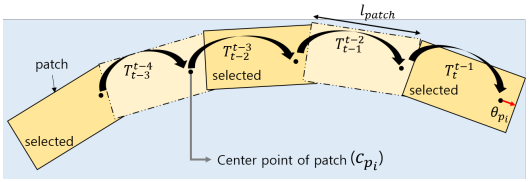
3) Stereo image to labeled pointcloud: On the query image side, disparity images are extracted from the stereo images using the stereo box matching algorithm (StereoBM [19]). CNN is utilized to estimate the semantic label for road markings in the stereo images. DeepLabv3+ network [20] is used as a base network, and trained using Apolloscape Dataset [21] and find-tuned using our own labeled data. Fig. 4 shows the resulting semantic image, where the color of each pixel represents the label. This semantic image is then combined with the disparity to construct a semantic labeled pointcloud that is fed to the main process in the following section.

4) Labeled pointcloud to Subpatch: From this labeled pointcloud, we create an 8-bit image by projecting the label pointclouds obtained from stereo cameras onto the z-plane (the bird's-eye view projection). This 8-bit image representation is referred to as a *subpatch* and will be appended to a *patch*. We mainly use the patch for matching against a candidate tileset from the HD map. Each bit of pixel of

(a) Lane only environment

(b) Sample direction sign, stop line and pedestrian crossing

Fig. 4: Result of semantic segmentation using CNN. Each color represents the label of each pixel (yellow: lane, red: direction sign, cyan: stop line, purple: pedestrian crossing).



(a)

(b)

Fig. 5: Illustration of patches. (a) Each patch includes an 8-bit image such as (b), and each bit of the image represents semantic information obtained from the stereo image. Patches also include the global coordinate of the vehicle pose, with the last frame as the center point ($\mathbf{c}_{p_i}$). Each patch also has a transformation ($\mathbf{T}_k^{k-1}$) relative to the previous patch.

the patch image represents a label. All patches include the global coordinate of the vehicle pose of the last frame as the center point (c_{p_i}) in addition to the associated transformation relative to the previous patch as based on this center point. A patch consists of l_{patch} sequential subpatches ($l_{patch} = 5$ used in this paper). Patches along the travel distance are as shown in Fig. 5.

Using this 8-bit patch-to-tile comparison yields substantial computational efficiency. High sparsity in the patch-based representation allows us to avoid every pixel-wise comparison within a candidate tileset; only a bit-wise pixel-level comparison is performed over the patches. The following section explains the patch update and selection in detail.

C. Patch Maintenance

1) Patch Update: For each incoming subpatch, we append and update the existing patch. To update the patch, the labeled pointcloud is converted into the global coordinates

$$^{G}P_{S,t} = \mathbf{T}_V^G \mathbf{T}_S^V {}^{S}P_{S,t}, \tag{1}$$

where $^{S}P_{S,t}$ is the labeled pointcloud in stereo camera coordinate, and $\mathbf{T}_V^G$ and $\mathbf{T}_S^V$ are the transformations of the vehicle pose and extrinsic parameter, respectively. The vehicle pose in the global coordinate comes from the accumulating odometry data. The converted pointcloud is projected onto the image coordinate of the patch image.

$$^{p_0}P = \pi_0(^{G}P_{S,t}). \tag{2}$$

The projection function (π_0) maps the pointcloud in global coordinate ($^{G}P_{S,t}$) onto an image coordinate (z-plane) of the 0-th patch (the most recent patch). The result of the projection contains image coordinate and label information (i.e. $^{p_0}P^{(k)} = \{u^{(k)}, v^{(k)}, l^{(k)}\}$), which are added to 0-th patch image. If the patch length is larger than l_{patch}, new patches are created, and the age of each patch is decreased by α.

$$a_{p_i} = \alpha a_{p_{i-1}} \tag{3}$$

Parameter α indicates decreasing age and is smaller than 1 ($\alpha < 1$). The age of each patch (a_{p_i}) prevents old patches from affecting cumulative odometry error.

2) Patch Selection: We further improve the computational efficiency by minimizing the number of patches (n_{patch}) used for matching. By only prioritizing a more informative patch, we are able to improve the longitudinal direction accuracy, overcoming the limitations of existing methods. For example, patches containing only straight lanes contribute less than patches including stop lines and road markings in terms of localization along the travel direction.

To determine the importance of patches, the number of points of a label (n_{stop}, n_{mark}) included in each patch image is used, and the number of points per unit length ($\bar{n}_{lane}$) is used in the case of lanes. A patch is considered as a best patch if $\bar{n}_{lane}$, n_{stop}, and n_{mark} all exceed the first thresholds (ξ_{lane_1}, ξ_{stop_1}, ξ_{mark_1}) or if $\bar{n}_{lane}$, and n_{stop} exceed the second thresholds (ξ_{lane_2}, ξ_{stop_2}). A patch is considered a bad patch if $\bar{n}_{lane}$ is smaller than the minimum threshold ($\xi_{lane_{min}}$). Finally, when selecting patches, at least three patches, including at least one best patch, are selected. All bad patches are ignored during this step. In our experiment, the first thresholds (ξ_{lane_1}, ξ_{stop_1}, ξ_{mark_1}) are set to 12, 40, and 50, and the second thresholds (ξ_{lane_2}, ξ_{stop_2}) are set to 10 and 70. The minimum threshold for lane pixel ($\xi_{lane_{min}}$) is set to 10.

D. Bitwise Particle Filter

For the localization backend, we adopt a particle filter and calculate the t_x, t_y, t_z, and r_z of the vehicle pose ($\bar{\mathbf{T}}_V^G$) according to the global coordinates. In the proposed method, we assume that we know the accurate initial pose to initialize the particles. $\mathbf{T}_V^G$ is the pose estimated using odometry data, and $\bar{\mathbf{T}}_V^G$ is the final pose estimated by the algorithm. The state of each particle consists of three components (t_x, t_y and r_z), and the 3-DOF poses of the particles become the center of the most recent patch. The global coordinate of the previous patch's center point can be estimated using a relative transformation that the patches contain. In the proposed method, the number of particles is adjusted according to the number of selected patches. Therefore, the re-sampling

process is first performed according to the number of patches before the particle weight update.

1) Particle Re-sampling: To avoid increasing computation time, the number of particles is adjusted according to the number of patches.

$$n_{particle} = min(n_{min}\frac{l_{max}}{n_{patch} \cdot l_{patch}}, n_{max}), \quad (4)$$

where n_{patch} is the number of patches selected in the previous step and $n_{particle}$ is the number of particles for re-sampling. n_{min} is the minimum number of particles. l_{max} is the maximum sum of patch lengths to be used for matching, and l_{patch} is the length of patches.

If the best patch is located nearby and fewer patches are selected, a large number of particles are utilized to increase the accuracy of the pose. Conversely, if the best patch is located far away and the number of selected patches is large, the number of particles is decreased. n_{min}, n_{max}, l_{patch}, and l_{max} are set to 500, 2000, 5, and 100 respectively.

2) Particle Prediction: In the particle prediction step, the re-sampled particles are used via a motion model. Gaussian noise is added during propagation.

3) Candidate Tileset Creation: Fig. 3 shows the candidate tileset creation process. In this step, the global HD map image is incrementally expanded by loading an 8-bit image tile close to the current estimated vehicle pose. To update the particle weights according to the patch-to-tile matching score, we first need to prepare the candidate tile image. This process is depicted in Fig. 3. To quickly search for nearby tiles, the HD map center tree created during the pre-processing is utilized.

4) Particle Weight Update: Using the candidate tile images and selected informative patches, particle weights are updated according to the bit-wise matching score. For the matching, the patch image (I_{p_i}) is rotated by the difference between the heading angle of the particle pose ($\hat{\theta}_{p_i}$) and the heading angle calculated by the odometry (θ_{p_i}) around the center point (c_{p_i}) of the patch image.

$$\hat{I}_{p_i} = t(\mathbf{c}_{p_i})r(\hat{\theta}_{p_i} - \theta_{p_i})t(-\mathbf{c}_{p_i})I_{p_i}, \quad (5)$$

where t and r are the translation by the input vector and the rotation by the input angle θ, respectively. The matching process is then performed using the global HD map tile image and patch images. Given the global HD map tile image I_G and the rotated patch image $\hat{I}_{p_i}$, we perform an AND operation as follows:

$$I_{M,p_i} = \text{AND}(I_G, \hat{I}_{p_i}). \quad (6)$$

Next, the weight of the particles is updated using the number of pixels of lanes, stops, and markings ($n_{l,p_i}, n_{s,p_i}, n_{m,p_i}$) in an 8-bit result image (I_{M,p_i}) computed by a bitwise AND operation. The final weight of each particle is calculated using the weight ($\omega_l, \omega_s, \omega_m$) of each label and the age parameter (a_{p_i}) that was calculated in the patch update step.

$$w_t^{(k)} = \sum_{i=0}^{n_{patch}} a_{p_i}(\omega_l n_{l,p_i} + \omega_s n_{s,p_i} + \omega_m n_{m,p_i}) \quad (7)$$

5) Pose and Height Estimation: After updating the weights of the particles, the final vehicle pose is estimated using the updated weights. The average pose of the top 3% of particles in terms of weight size is determined as the final pose estimation.

As the particle filter in the proposed method utilizes 2D image matching, only t_x, t_y, and r_z can be calculated using this method. In this height estimation process, the HD map point closest to the estimated vehicle pose is found from the pre-established HD map point tree. The global height data (i.e., t_z) of the current state is updated using the height information of the determined HD map point.

E. Optimization

In the previous process, t_x, t_y, t_z, and r_z of the vehicle pose are calculated using the particle filter. The final optimization process estimate roll and pitch to complete a full 6-DOF pose estimation. Pointclouds from a stereo image containing only road points are converted into vehicle coordinates as follows:

$$^{V}P_{S,t} = \mathbf{T}_S^{VS} P_{S,t}. \quad (8)$$

The planar equation can be obtained from the pointcloud that was converted into vehicle coordinate using the Random Sample Consensus (RANSAC) algorithm [22].

$$^{V}\mathbf{n}_t = \{^{V}n_{t,x}, ^{V}n_{t,y}, ^{V}n_{t,z}, ^{V}d_t\}, \quad (9)$$

where $^{V}n_{t,x}$, $^{V}n_{t,y}$, and $^{V}n_{t,z}$ are the components of the normal vector of the plane and $^{V}d_t$ is the constant value of the plane equation.

Lastly, optimization is performed using the sum of plane distance between the estimated plane and the global HD map points converted into vehicle coordinate by using the target transformation matrix ($\bar{\mathbf{T}}_G^V = (\bar{\mathbf{T}}_V^G)^{-1}$).

$$^{V}P_{map} = \bar{\mathbf{T}}_G^{VG} P_{map}. \quad (10)$$

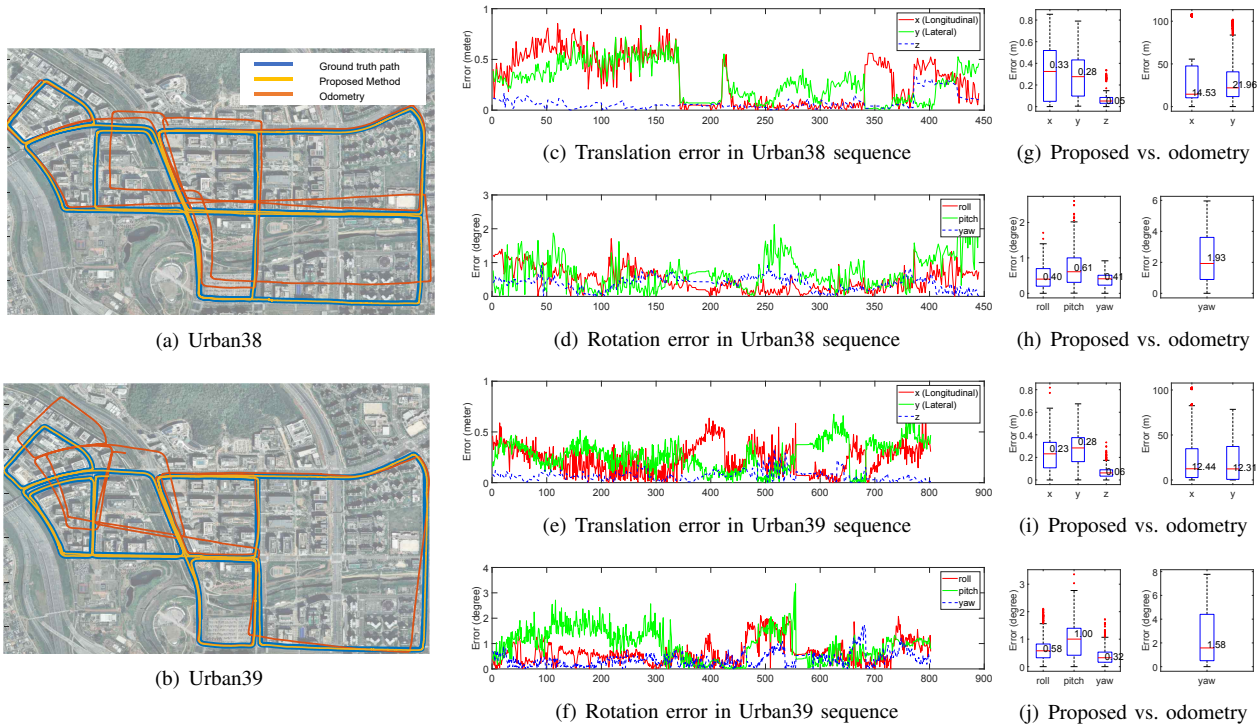
We optimize for $\bar{\mathbf{T}}_G^V$ while fixing (t_x, t_y, t_z, and r_z) and estimating only roll (r_x) and pitch (r_y) using the Levenberg-Marquardt (LM) method. The target parameters are initialized using optimized value of previous frame.

$$r_x^*, r_y^* = \underset{r_x, r_y}{\text{argmin}} \sum_{k=0}^{N} (^{V}\mathbf{n}_t \cdot ^{V}P_{map}^{(k)}) \quad (11)$$

Through this process, a full 6-DOF vehicle pose is estimated in global coordinate.

III. Experimental Results

For evaluation, we chose `Urban38` and `Urban39` in the Complex Urban Dataset [14] because the high-precision HD map data of Naver Labs [13] was available for the same location. The two test sequences are highly realistic and challenging, each containing dynamic objects and illumination variations over 11 km. No other available public dataset supports both HD maps and sensor data over the same region.



(a) Urban38

(b) Urban39

(c) Translation error in Urban38 sequence

(d) Rotation error in Urban38 sequence

(e) Translation error in Urban39 sequence

(f) Rotation error in Urban39 sequence

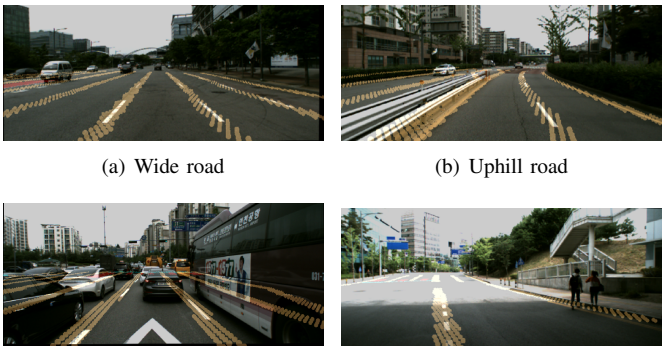
(g) Proposed vs. odometry

(h) Proposed vs. odometry

(i) Proposed vs. odometry

(j) Proposed vs. odometry

Fig. 6: Experimental results using Urban38 and Urban39 from the Complex Urban Dataset [14]. (a)–(b) The blue path is the ground truth path, the yellow path is the result of the proposed method, and the orange path is the odometry path. The total travel path of the two data is approximately 11 km. (c)–(f) Translational and rotational localization errors are depicted with respect to the image frame number. (g)–(j) Box plots of the estimated pose error (proposed method result is on left, odometry result is on the right).



(a) Wide road

(b) Uphill road

(c) Occlusion by traffic

(d) Shadow under the bridge

Fig. 7: Results of projecting the HD map data onto an image using estimated vehicle pose in various environments.

A. Qualitative Evaluation of Localization Accuracy

Fig. 6 shows the experimental results of the two sequences. The blue line is the ground truth trajectory provided by the dataset. This ground truth data was calculated using Virtual Reference Station (VRS)-GPS data, navigation sensor data, and LiDAR-based matching results in the graph SLAM framework. The orange route is the trajectory of the vehicle using only the encoder data. Although the odometry data provided in the dataset is well-calibrated, there is an inevitable cumulative error in the odometry over time. The yellow route is the trajectory estimated using the proposed method. In both sequences, the vehicle pose is stably estimated throughout the entire 11 km distance despite the dynamic objects and ambiguity in the environment (Fig. 7).

B. Quantitative Evaluation of Localization Accuracy

Fig. 6(c)–Fig. 6(j) show quantitative evaluations over the ground truth provided in the Complex Urban Dataset. However, we found that the optimized ground truth was less accurate when the VRS-GPS data was not fully accurate. Hence, for concrete and conservative quantitative evaluation, we only evaluated the pose when VRS-GPS received sufficient satellite signals for estimating a global position. Fig. 6(c)–Fig. 6(e) show the change of estimated pose error in the two sequences. In this figure, x and y refer to the translation error in the longitudinal and lateral directions of the vehicle, respectively, and z is the translation error of the global height. The rotation error is an error expressed in degrees based on the vehicle coordinates. In most cases, the translation error did not exceed 0.5 m in both the longitudinal and lateral directions. However, in Fig. 6(c), the translation error of the frames between 30 and 160 was relatively larger than other frames. This result came from a long, straight road in a wide, ten-lane-only environment with no road markings (Fig. 7(a)). In this environment, even though the road marking information for constraints in both directions

TABLE I: Performance comparison against existing methods. In the third column, we re-implemented Vivacqua et al. [18] using our dataset.

	Ours	Re-impl [18]
Method	Bitwise MCL	variable gain filter
Complexity	High	
Test env	Pangyo (S. Korea)	
Path length (km)	11.06	
Eval	lat/lon	lat/lon
Mean error	0.24 / 0.30	0.28/ 0.31
Max error	0.55 / 0.67	0.63 / 0.70
Map size	17 Mbyte	17 Mbyte
Computation time	10 Hz	10 Hz

were included in the selected patch, the distance of this patch was too far. Thus, this patch did not significantly benefit the pose estimation. The rotation error did not exceed two degrees. However, pitch error was relatively large as it changed very dynamically compared to other rotation parameters.

The box plots of the localization error further describe the performance, as shown in Fig. 6(g)–Fig. 6(j). Unlike the proposed method, since odometry is the result of encoder data, only 2D motion can be estimated. Comparing the two datasets, the error shows a similar trend. In the case of translation, the median error was maintained below 0.4 m in all directions, and the error in the z-direction had a median of approximately 0.05 m. In the case of roll and yaw rotation, the median error was approximately 0.5 degrees, and pitch rotation had a median error of approximately 1 degree. The most important aspect of autonomous driving is accuracy in the lateral and longitudinal directions. Generally, the average lane width is 3 m and the average vehicle width is 1.8 m, therefore a median error of 0.5 m is within the allowable error width. The median error is also sufficient for the purpose of correcting the global position in place of GPS, which exhibits very low accuracy in complex urban areas.

We further provide a comparison against existing HD map based methods [18] in Table I. We re-implemented [18] as there was no source code available for the algorithm and applied it to our dataset. Specifically, we have replaced patch selection to their history of 240 m and exploit lanes as in [18]. The proposed method yielded an improvement along longitudinal direction even applied to the environment with higher complexity. The performance drop in the re-implementation of [18] could be because of this higher complexity in the test environment.

C. HD Map Projection onto Query Image

To further validate the localization performance, we projected 3D HD map data onto image coordinate using the estimated vehicle pose, as shown in Fig. 7. The projection over the entire sequence can be seen in the attached video (`result.mp4`). As shown in Fig. 7(b), even in hills with large variations in pitch angle, the motion was accurately estimated so that the HD map data was projected at the correct location. Fig. 7(c) shows a situation where lane information

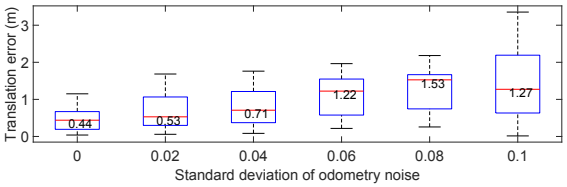


Fig. 8: The pose estimation error due to odometry noise. Gaussian noise was added to the odometry input (x, y, yaw) with different standard deviations.

is not visible because of the numerous surrounding vehicles. Fig. 7(d) shows a place where the dynamic range of the image is large due to the shadow under the bridge. Under these various circumstances, we could qualitatively confirm that the vehicle pose was accurately estimated.

D. Ablation Studies

The performance of the proposed method could be affected by noise level in odometry, semantic labels, and patch size. Via this ablation study, we examined the effect of each component.

Fig. 8 shows the pose estimation performance when noise was added to odometry. In the experiments, the standard deviation of the particle distribution was set in proportion to the standard deviation of the odometry noise. Experimental results show that the error of the pose estimation increased in proportion to the odometry noise. However, the algorithm maintained some performance when the noise was smaller than the standard deviation of 0.04 . When the standard deviation of odometry noise was greater than 0.1 , localization failed during the experiment.

Table II further shows the ablation study results depend on semantic label information included in the particle weight update process and the patch size. The result shows that lanes affect the localization performance of lateral direction, and stop lines and markings affect to longitudinal direction.

Many HD map-based localization researches such as [15, 17] often leverage sensor measurement in the current time frame. This strategy can be simulated by using the latest single patch to update the particle weight without the patch selection process (`Single Patch` in Table II). The result of a single patch test shows large localization errors in the

TABLE II: The pose estimation error depends on the semantic label and patch size (`L`: Lane, `S`: Stop line, `M`: Marking, `x`: longitudinal, `y`: lateral, `z`: height, and unit: meter). The `single Patch` used only a recent patch without a patch selection process.

		L+S+M	L+S	L+M	L	Single Patch
Mean	x	0.308	0.316	0.304	0.476	1.428
	y	0.241	0.259	0.255	0.256	0.779
	z	0.068	0.067	0.067	0.065	0.075
Max	x	0.771	0.687	0.710	3.98	14.981
	y	0.659	0.739	0.739	0.77	3.835
	z	0.177	0.167	0.177	0.198	0.198

TABLE III: Computation time of each process (msec). The particle weight update process took the most time, and most processes are, on average, completed within 2 ms except for the weight update and optimization processes.

	Patch update	Patch selection	Prediction	Resample	HD map update	Height estimation	Weight update	Optimization	Total
Min	0.035	0.0002	0.156	0.067	0.016	0.006	4.575	3.285	10.01
Median	0.96	0.0019	0.292	0.130	0.025	0.029	48.84	8.509	61.58
Max	29.42	0.098	8.58	6.0	37.72	8.045	89.82	39.68	104.05

lateral and longitudinal directions because it depends on only recent measurements of the current environment.

E. Computational Time

In total, two PCs with hardware specifications of an Intel i7-7700 CPU, 32Gbyte Ram, and 1080Ti GPU were utilized for the implementation. All processes, with the exception of the stereo process and semantic labeling prediction, ran on one PC, as the stereo and semantic prediction process required a significant CPU and GPU computation. Table III shows the computational time of the localization module. The proposed localization algorithm can be divided into eight parts, and the median operation time was less than 5 msec for most processes.

IV. Conclusion and Future Works

In this paper, we presented a full 6-DOF global pose estimation exploiting semantic road information obtained from HD maps and query stereo images. Using HD maps can significantly reduce the storage space and alleviates the cost of updating and transmitting the map. We achieved global accuracy at the sub-meter level through the use of only stereo camera and odometry supporting at 10 Hz.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Proc. Robot.: Science & Sys. Conf.*, vol. 2, no. 9, 2014.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] H. Hu, M. Sons, and C. Stiller, "Accurate global trajectory alignment using poles and road markings," in *Proc. IEEE Intell. Vehicle Symposium*, 2019, pp. 1186–1191.

[4] O. Pink, F. Moosmann, and A. Bachmann, "Visual features for vehicle localization and ego-motion estimation," in *Proc. IEEE Intell. Vehicle Symposium*, 2009, pp. 254–260.

[5] P. Bao, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 27, no. 9, pp. 1485–1490, 2005.

[6] H. Roh, J. Jeong, and A. Kim, "Aerial image based heading correction for large scale SLAM in an urban canyon," *IEEE Robot. and Automat. Lett.*, vol. 2, no. 4, pp. 2232–2239, 2017.

[7] H. Roh, J. Jeong, Y. Cho, and A. Kim, "Accurate mobile urban mapping via digital map-based SLAM," *IEEE Sensors J.*, vol. 16, no. 8, p. 1315, 2016.

[8] O. Vysotska and C. Stachniss, "Exploiting building information from publicly available maps in graph-based SLAM," in *IEEE Sensors J.*, 2016, pp. 4511–4516.

[9] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to localize using a LiDAR intensity map." in *CoRL*, 2018, pp. 605–616.

[10] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2014, pp. 176–183.

[11] Y. Kim, J. Jeong, and A. Kim, "Stereo camera localization in 3D LiDAR maps," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, Oct. 2018, pp. 1–9.

[12] H. Kim, T. Oh, D. Lee, and H. Myung, "Image-based localization using prior map database and monte carlo localization," in *Proc. Intl. Conf. on Ubiquitous Robots and Ambient Intell.*, 2014, pp. 308–310.

[13] N. Labs. (2019) NAVER LABS HDMap Dataset. [Online]. Available: https://hdmap.naverlabs.com/

[14] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *Intl. J. of Robot. Research*, vol. 38, no. 6, pp. 642–657, 2019.

[15] W.-C. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun, "Exploiting sparse semantic HD maps for self-driving vehicle localization," 2019.

[16] A. Welte, P. Xu, P. Bonnifait, and C. Zinoune, "Estimating the reliability of georeferenced lane markings for map-aided localization," in *Proc. IEEE Intell. Vehicle Symposium*, 2019, pp. 1225–1231.

[17] F. Poggenhans, N. O. Salscheider, and C. Stiller, "Precise localization in high-definition road maps for urban regions," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 2167–2174.

[18] R. P. D. Vivacqua, M. Bertozzi, P. Cerri, F. N. Martins, and R. F. Vassallo, "Self-localization based on visual lane marking maps: An accurate low-cost approach for autonomous driving," *IEEE Trans. Intell. Transport. Sys.*, vol. 19, no. 2, pp. 582–597, 2017.

[19] K. Konolige. Stereobm algorithm in of opencv. [Online]. Available: https://docs.opencv.org/3.4/d9/dba/classcv_1_1StereoBM.html

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. European Conf. on Comput. Vision*, 2018, pp. 801–818.

[21] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE Trans. Pattern Analysis and Machine Intell.*, 2019.

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