


ORIGINAL RESEARCH

Towards high-definition vector map construction based on multi-sensor integration for intelligent vehicles: Systems and error quantification

Runzhi Hu¹ | Shiyu Bai¹  | Weisong Wen¹ | Xin Xia² | Li-Ta Hsu¹
¹Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, Hong Kong

²Department of Civil and Environmental Engineering, University of California, Los Angeles, California, USA

Correspondence

Shiyu Bai, Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, Hong Kong.
Email: shiyu.bai@polyu.edu.hk

Funding information

Guangdong Basic and Applied Basic Research Foundation, Grant/Award Number: 2021A1515110771; University Grants Committee of Hong Kong under the scheme Research Impact Fund, Grant/Award Number: R5009-21

Abstract

A lightweight, high-definition vector map (HDVM) enables fully autonomous vehicles. However, the generation of HDVM remains a challenging problem, especially in complex urban scenarios. Moreover, numerous factors in the urban environment can degrade the accuracy of HDVM, necessitating a reliable error quantification. To address these challenges, this paper presents an open-source and generic HDVM generation pipeline that integrates the global navigation satellite system (GNSS), inertial navigation system (INS), light detection and ranging (LiDAR), and camera. The pipeline begins by extracting semantic information from raw images using the Swin Transformer. The absolute 3D information of semantic objects is then retrieved using depth from the 3D LiDAR, and pose estimation from GNSS/INS integrated navigation system. Vector information (VI), such as lane lines, is extracted from the semantic information to construct the HDVM. To assess the potential error of the HDVM, this paper systematically quantifies the impacts of two key error sources, segmentation and LiDAR-camera extrinsic parameter error. An error propagation scheme is first formed to illustrate how these errors fundamentally influence the accuracy of the HDVM. The effectiveness of the proposed pipeline is demonstrated through our code available at <https://github.com/ebhrz/HDMap>. The performance is verified using typical datasets, including indoor garages and complex urban scenarios.

1 | INTRODUCTION

Accurate, reliable, and absolute positioning is the base of the intelligent transportation system [1]. It is significant for autonomous systems with navigation requirements, such as unmanned aerial vehicles (UAV) [2] and autonomous driving vehicles (ADV) [3]. The multi-sensor integrated localization solutions [4–6] are extensively investigated by combining the global navigation satellite system (GNSS) [7], inertial navigation system (INS), light detection and ranging (LiDAR), and camera sensors. Satisfactory positioning accuracy can be achieved in open areas with satisfactory satellite visibility. However, the absolute positioning accuracy is significantly degraded in urban canyons with excessively tall buildings reflecting the GNSS signals, leading to poor GNSS positioning accuracy [8]. In this case, high-definition (HD) maps currently are regarded as the

most accurate localization method [9], and researchers sought to include the pre-built HD map into the multi-sensor integrated localization, leading to the GNSS/INS/LiDAR/HD map-based localization solution [10–22]. Specifically, absolute positioning can be obtained by matching the real-time 3D point clouds with the pre-built 3D point cloud map. In other words, the globally referenced positioning is guaranteed even when the GNSS is not available. Due to the high reliability of the GNSS/INS/LiDAR/HD map-based localization solution, it is extensively used by the research community. In short, the pre-built map is significant for reliable autonomous driving localization in complex urban canyons.

Generating a high-accuracy HD map composed of dense 3D point clouds depends heavily on the carrier's pose. Hence, many methods are investigated based on multi-sensor fusion to improve the pose estimation [4, 23, 24]. Specifically, the

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2024 The Author(s). *IET Intelligent Transport Systems* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

high-accuracy pose estimation is obtained by complementarily integrating the measurements from the GNSS, INS, and LiDAR sensors. Then, the high-accuracy 3D point cloud map can be obtained by accumulating the continuous 3D point clouds from 3D LiDAR based on the high-accuracy pose estimation from multi-sensor integration. Although the 3D point cloud map is a frequently used pipeline, it is cumbersome and hard to maintain. It is a significant bottleneck problem that limits the large-scale deployment of the 3D point cloud map, mainly caused by the structureless point cloud form. To fill this gap, the high-definition vector map (HDVM) [25–30] is investigated as a potential replacement for the conventional cumbersome 3D point cloud map. Unlike the conventional 3D point cloud map, the HDVM is lightweight and mainly includes vectorized information (VI), such as the lane lines and poles, which are massively available in typical urban scenarios. Moreover, the HDVM is easy to maintain and can even be updated using the crowd-sourced images collected by large-scale users [31, 32]. In short, as pointed out by a recent survey paper [33], the HDVM is a promising solution for the reliable navigation of autonomous driving vehicles. However, several issues still affect the usage of HDVM in complex urban environments.

To tackle the challenges listed above, this paper presents a generic pipeline for HDVM automatic construction based on GNSS, INS, camera, and LiDAR. The proposed pipeline extracts semantic information from camera images using the recently developed Swin transformer [34, 35]. Second, the depth of the semantic information is recovered using the 3D point clouds captured by the 3D LiDAR. Then, the reprojection is performed, and a local semantic point cloud is gathered to be an HDVM with the help of pose estimation from a high-accuracy RTK-GNSS/INS integrated navigation system. Meanwhile, the VI includes the lane lines and poles, the two most essential components for the map-based localization of ADV. Finally, the error quantification model is introduced to quantify the potential error of HDVM based on the two listed vital factors.

Therefore, our contributions are mainly three points below:

- (1) This paper proposes a pipeline that enables reliable and incremental HDVM generation based on onboard sensors. Unlike existing methods, the proposed pipeline encompasses both horizontal (ground) and vertical components. Moreover, the proposed pipeline is encapsulated and open-sourced to the community via <https://github.com/ebhrz/HDMap>.
- (2) This paper first presents a theoretical and experimental quantification analysis between the listed two key factors and the potential error of the generated HDVM. This error quantification model can benefit the academic and industry community in developing the HDVM, especially for dense urban environments.
- (3) This paper verifies the effectiveness of the proposed pipelines using datasets collected in indoor garages and challenging outdoor urban canyon scenarios.

The rest of the paper is structured as follows: Section 2 presents an overview of the framework with pre-definitions of important mathematical symbols. Concretely, the specific procedure of each step is described in Section 3. The error analysis will be discussed in detail in Section 4. Moreover, the experimental results and discussions are displayed in Section 5. Conclusions and prospects are given in Section 6. Section 7 is the acknowledgments.

2 | RELATED WORK

2.1 | HDVM generation

Generally, the construction of an HDVM involves two essential components: (1) Semantic segmentation, leveraging sensors like LiDAR and cameras and (2) VI extraction from the segmentation results, rendering the map more lightweight. Current methodologies in semantic segmentation can predominantly be categorized into two classes, focusing either on point cloud analysis or image processing.

A conventional approach within the field consists of performing the semantic segmentation directly, utilizing the intensity of the point cloud within the 3D domain. In 2008, Kammel [36] utilized the Radon transform to extract lane lines, but recently, the works in [37–39] all directly used reflective intensity as a filter to find the lane lines. Based on LiDAR scans, the extracted lane lines can easily retrieve the depth information in 3D. Unfortunately, these LiDAR intensity-based methods could only extract lane lines or road signs on the ground, as the LiDAR intensity is mainly reliable for the lane lines with high reflectivity. The poles, trees, traffic lights, and obstacles cannot be fully recovered based on LiDAR intensity. Moreover, owing to the substantial advancements achieved by convolutional neural networks (CNNs) and transformer in the realm of object detection and segmentation within 2D images, commonly referred to as pixel-based methods, several researchers have endeavoured to employ CNNs and transformer to process LiDAR point cloud-based segmentation. PointNet [40] and PointNet++ [41] tried to directly do semantic segmentation on the point cloud and used symmetry function to solve the problem caused by unstructured point cloud. Nevertheless, compared to pixel-based object detection and segmentation, voxel CNN-based [40, 42] and transformer-based [43] object detection on point cloud require lots of computation due to the sparse and unstructured points. Although some improvements [44, 45] are obtained, the performance of direct object detection in 3D point clouds is still limited by the sparsity of the point clouds compared with the image-based segmentation. Thus, RangeNet++ [46] projected the point cloud to a panorama grey scale image and did the segmentation on 2D domain. But this method also is limited to the poor texture information.

Unlike the sparse 3D LiDAR point clouds, the pixels in an image are structured in the 2D domain, providing rich texture information. Since AlexNet [47], CNN-based methods have developed dramatically, skeletons like InceptionNet [48],

VGGNet [49], and ResNet [50], with some recent architecture such as RCNN and Transformer [51, 52], have vastly increased the accuracy in object detection and instance semantic segmentation. Inspired by the strong capability of the image-based learning networks, the works in [30, 53, 54] performed semantic segmentation in the image domain for HDVM map construction. However, these methods rely on a strong assumption that the vehicles are driving on a plane road with necessary lane lines. Therefore, the VI can be reprojected to the 3D domain based on inverse perspective mapping (IPM) [55]. In short, the image has rich texture information, easily segmented by leveraging the recent dramatic progress in deep learning. However, the image cannot provide the depth information in the 3D domain, which the HDVM requires.

As an extension, the LiDAR-image-fused solution was investigated. The work from [56, 57] used GNSS to get the positioning solution and then projected the LiDAR points to OpenStreetMap to get the semantic information, where OpenStreetMap is regarded as an image. However, the accuracy of the OpenStreetMap is not guaranteed. Interestingly, the effort from [58] utilized a CNN to segment the image. The output is regarded as the prior probability to estimate the posterior probability of the sparse parts of the 3D point clouds. The work in [59] used CNN to detect road surfaces and lane lines in the image. Then, the lane lines were projected to the 3D domain using the depth information from LiDAR. As an aggressive method, the work in [60] introduced an end-to-end method, which utilizes a vast network to simultaneously process image and point cloud, which directly outputs a topological map. However, the generalization capability of the end-to-end method is limited by the training dataset applied. In conclusion, contemporary methods that fuse camera and LiDAR data also tend to emphasize the ground components, often predicated on the assumption of a planar road surface. This method fails to utilize the depth information to recover vertical components fully. Regarding 2D localization, ground components like lane lines are limited to determining only one degree of freedom. The localization can only attain accuracy when vertical components are considered and integrated into the model.

2.2 | HDVM error quantification

The existing research on the HDVM generation [56, 59] mainly focused on the open area with limited traffic participants and stable illumination conditions. As a result, the semantic information can be easily detected and segmented. Thus, the accuracy of the HDVM can be reliably guaranteed. Unfortunately, this is not true for complex urban scenarios with dense traffic. Given the additional challenges from the complex urban scenarios for HDVM construction, it is of great significance to quantify the potential errors of the HDVM before its massive deployment in safety-critical ADV applications. Unfortunately, there are few works on the systematic quantification of the errors for the HDVM. It is recognized that [61] the error from the (1) inaccuracy of semantic segmentation of the image to extract the VI and (2) the depth recovery of the VI arising

from the inaccuracy of the LiDAR-camera extrinsic parameters are the two significant sources dominating the accuracy of the HDVM based on LiDAR-camera-fusion. Quantifying the impacts of the listed two key error sources on the HDVM is highly expected.

Instead of quantifying the error of the HDVM construction, some work is devoted to analysing the error of the HDVM update, given the potential crowdsourcing of image data. Zhang [62] used the metric of intersection over union (IoU). In [31], Kim proposed an approach based on the shell structure, which can calculate the distance between the map points and the new income points on a shell structure. However, these metrics are all evaluated in the experiment step, where the quantification of the errors from extrinsic parameters and semantic segmentation errors are not presented, which can reduce the accuracy of the HD map as the potential safety risk is not revealed in advance.

3 | OVERVIEW OF THE PROPOSED METHOD

The overview of the proposed pipeline is shown in Figure 1. Our construction consists of three kinds of raw data: image from Zed 2 camera at 30 Hz, point cloud from Velodyne LiDAR at 10 Hz, and the 6 degrees of freedom (DoF) pose derived from a high-accuracy GNSS/INS integrated positioning system, namely the SPAN-CPT, operating at a high-frequency rate of 100 Hz [63].

Before the map construction, preprocessing is required to calibrate intrinsic and extrinsic parameters between LiDAR and the camera. The most common camera calibration method is from Zhang [64], and it will provide intrinsic parameters and distortion parameters. This calibration process yields exact intrinsic parameters, thereby minimizing the potential introduction of significant mapping errors. LiDAR-camera extrinsic parameters are fundamental and will significantly affect the accuracy of the HD map. Most methods are based on perspective-*n*-points (PnP) [65] and iterative closest point (ICP) [66], which are the 3D-to-2D and 3D-to-3D methods [49], respectively. Besides, it is postulated that no extrinsic parameters exist between the LiDAR and SPAN-CPT. This assumption is grounded in the positioning of the LiDAR directly above the SPAN-CPT, resulting in minimal translational differences. Additionally, the yaw angle remains consistent, and typically, there is an absence of pitch and roll variations when the vehicle operates on level terrain.

In addition, the effects of image and point cloud distortion need to be considered to obtain refined raw data. Image distortion is due to the convex spherical lenses—the image magnification increases with the distance from the optical axis. Point cloud distortion is caused by the simultaneous rotation of the LiDAR and the vehicle's motion. Fortunately, LiDAR provides the accurate time delay of each point, and high-frequency pose estimation from SPAN-CPT can help fix the issue.

After rectifying the distortion, it is essential to capture the image and point cloud simultaneously. Typically, hardware trig-

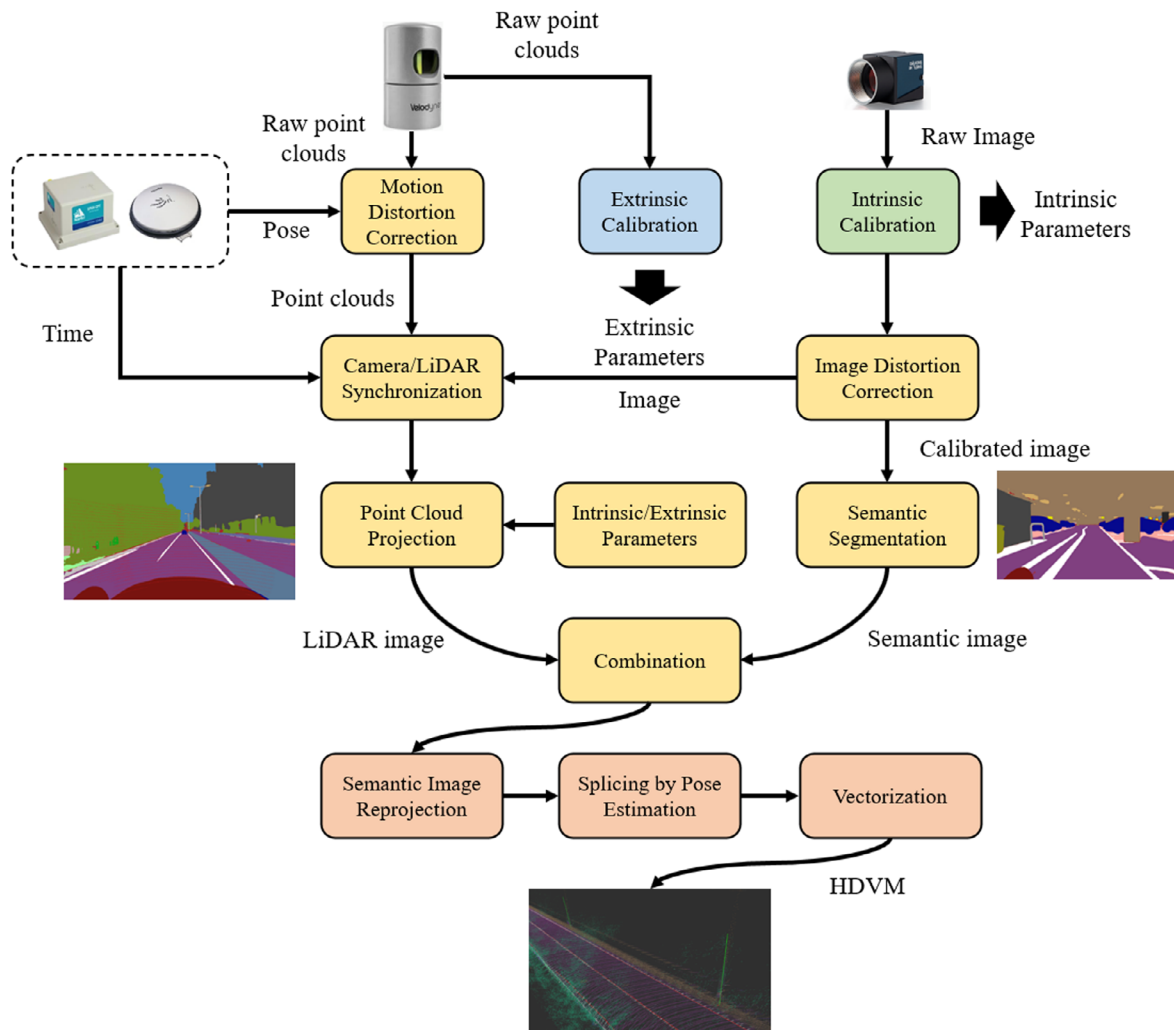


FIGURE 1 The framework of the proposed pipeline includes four parts. (1) System intrinsic calibration in green part, (2) system extrinsic parameters calibration in blue part, (3) integration of the information from light detection and ranging (LiDAR) and camera in yellow part, and (4) High-definition vector map (HDVM) construction. The two calibrations are the preprocess before the map construction. Integration of LiDAR and camera is the key to the map construction. Lastly, it is about how to build the vector map.

gers are employed to synchronize LiDAR and the camera. However, leveraging high-frequency pose estimation facilitates frame alignment by displacing the positions of the closest point cloud frame to match the image frame. The temporal difference between the nearest point cloud frame, captured at a frequency of 10 Hz, and the image frame, captured at 30 Hz, is less than 0.07 s. It ensures that the two frames exhibit minimal parallax.

After the synchronization, the image goes through the semantic segmentation neural network to be a semantic image. It is also a key point of map construction because the accuracy of the HDVM relies heavily on the accuracy of the semantic segmentation.

There are lots of segmentation networks, such as Mask R-CNN [52], FCN [67], and UNet [68]. In our implementation, we choose the current state-of-the-art Swin transformer as the solution, and the model is pre-trained with the Mapillary Vistas dataset [69]. Meanwhile, the point cloud synchronized with the image is projected to the camera plane by intrinsic and extrin-

sic parameters, and in this paper, it is called a LiDAR image. The semantic image with depth information can be generated by merging LiDAR and the semantic image. Reprojecting the semantic image can get a semantic point cloud. Then, instance segmentation is performed by the DBSCAN [70] algorithm, as the number of clusters within the point clouds is unknown. For the last step, each instance with extracted semantics can be fitted with different models to get vectorized. Specifically, the lane lines are continuous, and a growing strategy is used to get it vectorized, while poles are separated and extracted by the top centre point. Based on the above process, the generation of HDVM has been finished.

Before diving into the expanded introduction of the proposed method, the coordinates involved are clarified as follows:

The LiDAR frame $\{ \cdot \}$: Originated at the geometric center of the LiDAR with the x -axis, y -axis, and z -axis pointing to the right, front, and up, respectively.

The camera frame $\{c\}$: Originated at the geometric centre of the camera with the x -axis, y -axis, and z -axis pointing to the right, down, and front, respectively.

The image frame $\{i\}$: Originated at the top left corner of an image with the x -axis and y -axis pointing to the right and down, respectively.

The east-north-up (ENU) frame $\{e\}$: Originated at the initial vehicle position with the x -axis, y -axis, and z -axis pointing to the east, north, and up, respectively. It is also the world frame in this paper.

The symbol ${}^b_a\{\cdot\}$ denotes a transformation from frame a to frame b .

By default, all matrix operations employed in this paper, such as point transformations, are conducted in homogeneous form. Any operation not originally in this form is converted to homogeneous form before actual computation.

4 | HDVM GENERATION VIA MULTI-SENSOR INTEGRATION

This section presents the construction of the HDVM based on the integration of GNSS/INS/camera/LiDAR.

4.1 | Pose estimation and interpolation by GNSS-RTK/INS integration

Due to the high frequency of pose estimation from SPAN-CPT at 100 Hz, the motion can be regarded as uniform between two adjacent frames. Thus, the interpolation can be used to obtain the nearest point to achieve distortion correction and synchronization. The interpolation strategy is as follows:

${}^n\mathbf{T}_n$ denotes the n th pose estimation from SPAN-CPT and is the ground truth at a discrete time t_n , providing position \mathbf{t}_n including east, north, up, and orientation \mathbf{q}_n , which is a quaternion. An interpolated pose estimation at a given time t is represented as ${}^n\mathbf{T}_t$, which also contains position \mathbf{t}_t and orientation \mathbf{q}_t . The two kinds of pose estimation can be expressed below:

$${}^n\mathbf{T}_n = \{\mathbf{t}_n, \mathbf{q}_n, t_n\} \quad (1)$$

$${}^n\mathbf{T}_t = \{\mathbf{t}_t, \mathbf{q}_t, t\} \quad (2)$$

The two adjacent pose estimation frames can be found as:

$$\{{}^n\mathbf{T}_{n-1}, {}^n\mathbf{T}_n | t_{n-1} < t < t_n\} \quad (3)$$

Then, the interpolated pose can be calculated by following equations:

$$k = \frac{t - t_{n-1}}{t_n - t_{n-1}} \quad (4)$$

$$\mathbf{t}_t = \mathbf{t}_{n-1} + k * (\mathbf{t}_n - \mathbf{t}_{n-1}) \quad (5)$$

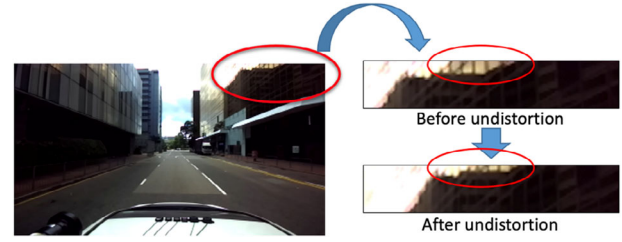


FIGURE 2 The image distortion and the correction result.

$$\mathbf{q}_t = \mathbf{q}_{n-1} (\mathbf{q}_{n-1}^{-1} \mathbf{q}_n)^k \quad (6)$$

After getting the interpolated pose estimation, the following parts describe the details of the framework.

4.2 | Image distortion correction

Image distortion includes radial distortion and tangential distortion. Tangential distortion is caused by the lens not paralleling the imaging plane, but its influence is unnoticeable under modern manufacturing technology. On the other hand, radial distortion results from the different refractive indices of the camera lens surface. In short, the better the lens's materials, the less the radial distortion is. Hence, the radial distortion should be corrected, and its model [71] can be described by Taylor expansion as:

$$x' = x (1 + k_1 r^2 + k_2 r^4 + \dots + k_n r^{2n}) \quad (7)$$

$$y' = y (1 + k_1 r^2 + k_2 r^4 + \dots + k_n r^{2n}) \quad (8)$$

where x' and y' denote a corrected pixel position, while x and y are the original pixel positions. k_n denotes the coefficients. r is the distance between the pixel and the image centre. Usually, the first and the second orders are accurate enough for modelling. These parameters can be obtained by camera calibration. An image distortion correction sample is shown in Figure 2.

4.3 | Point cloud motion distortion correction

In a typical situation, points from a point cloud frame are expected to be captured at the same time. However, LiDAR works by rotating its sensor and getting the reflected signal to measure the distance; reflected signals in one point cloud frame are not received simultaneously. Once the LiDAR moves, motion distortion occurs. Figure 3 depicts the theoretical principle of distortion and an example. In Figure 3a, The blue solid line is the vehicle trajectory. The red dots are the real point clouds that the LiDAR captures. The blue dots are the point clouds that the LiDAR captures statically. In Figure 3b, the area in the red circle is a wall captured by LiDAR, and the wall is split due to motion distortion. This distortion can be ignored

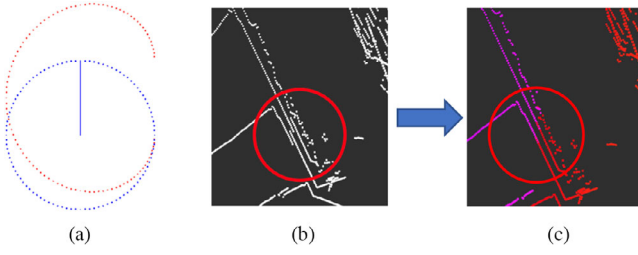


FIGURE 3 The theoretical principle of the distortion and an example with correction.

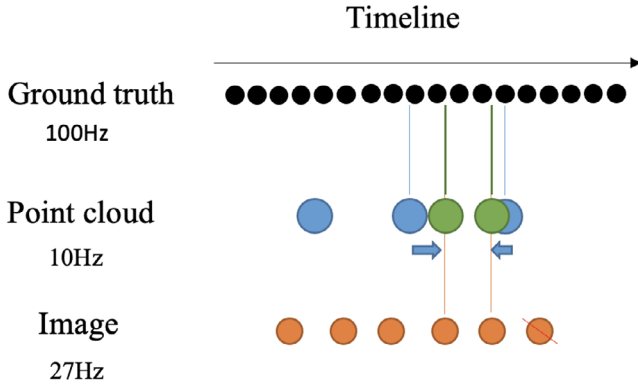


FIGURE 4 The framework of time synchronization.

in a low-speed situation, but in a high-speed situation, it will be fatal.

Fortunately, most LiDAR sensors provide the scan's start time and each point's return time. The motion distortion can be fixed by the equation below:

$${}^l\mathbf{p}_f = {}^l\mathbf{T}_p {}^n\mathbf{T}_s {}^l\mathbf{p}_d \quad (9)$$

where ${}^l\mathbf{p}_f$ is the fixed coordinate of a LiDAR point, ${}^l\mathbf{T}_p$ is the transformation from ENU frame to LiDAR frame at the each point's return time gotten from ground truth, ${}^n\mathbf{T}_s$ is the transformation from LiDAR frame to ENU frame at the scan's start time gotten from ground truth, and ${}^l\mathbf{p}_d$ is the coordinate of the distortion point in LiDAR frame. In Figure 3c, the distortion in (b) is fixed by Equation (9).

4.4 | Camera LiDAR time synchronization

LiDAR and camera should capture the scene simultaneously. However, the two devices can only be configured with distinct sampling rates in most situations. A solution is to design hardware to trigger both, which is expensive. Besides, a software solution relies on high-frequency pose estimation, moving the point cloud to align the nearest camera frame available, as shown in Figure 4.

The proximate image frame can be identified for a specified point cloud frame, and their pose estimations can be acquired. Points in the point cloud are directly transformed through the pose estimations. It is like LiDAR motion distortion, and the

detail is as follows:

$${}^l\mathbf{p}_i = {}^l\mathbf{T}_i {}^n\mathbf{T}_l {}^l\mathbf{p}_l \quad (10)$$

where ${}^l\mathbf{p}_l$ represents the original coordinate of the points in the moving point cloud. ${}^n\mathbf{T}_l$ is the transformation from the LiDAR frame to the ENU frame at the LiDAR data receiving time. ${}^l\mathbf{T}_i$ is the transformation from ENU to LiDAR frame at the image receiving time, and ${}^l\mathbf{p}_i$ is the aligned coordinate.

4.5 | Image semantic segmentation

Semantic elements constitute a necessary component of the HDVM. With the development of deep learning and pattern recognition, automatic annotation by computer is available. Convolution is the most powerful method to extract the features. However, in contrast with 2D convolution, 3D convolution is less mature because the raw point cloud is too large to be convoluted. Although down-sampling is performed, voxel-based convolution [42] still needs many calculations. Degenerating the voxel to point pillars [44] will create a pseudo image to be processed with the 2D convolution. Thus, in this paper, we do the semantic segmentation in the image domain and reproject it to the 3D domain via LiDAR and extrinsic parameters.

To achieve semantic segmentation, in this paper, a Swin transformer [34, 35] model is employed to do this step. Swin transformer is an architecture proposed by Meta, and the model is pre-trained on the Mapillary Vistas dataset [69]. The Mapillary Vistas consists of 25,000 high-resolution images and 66 semantic object categories. The images were taken in various weather, seasons, and times of day globally, covering six continents. It guarantees the robustness of the model. The model training on the dataset was iterated 3,000,000 times, making the mIOU reach 60.8%. Moreover, the model received images of any size and directly outputted a tensor of the same size in the first two dimensions. The size of the third dimension is 66, containing the probability of each category. Accordingly, the third dimension will be merged into one by picking up the serial number of the categories which has the maximum probability. It can be formulated as follows:

$$S_1(u, v) = \{P_1, P_2, P_3 \dots P_{66}\} \quad (11)$$

$$S_2(u, v) = \operatorname{argmax}_n P_n, P_n \in S_1(u, v) \quad (12)$$

where u and v are the locations of a pixel, and S_1 represents the semantic segmentation model, which outputs the probability P_n corresponding to the semantic categories of a given pixel at location (u, v) . Each P_n represents the probability that the pixel belongs to the n th category. S_2 refers to the function that identifies the category with the maximum probability in S_1 , thereby determining the most probable classification for the pixel. In short, in this step, the input is an RGB image of size $W \times H$, and the output is $W \times H \times 1$.

In our implementation, dynamic objects can be easily classified through our semantic segmentation. In Mapillary dataset, lane line is defined as 24 and pole is defined as 45, while car is defined as 55. In the map construction, we only concern these two elements, lane line and pole, and filter other objects to build our map.

4.6 | Point cloud projection

Map elements are segmented in the image domain; thus, reprojecting them to the 3D domain needs depth information. The depth information comes from LiDAR, which means that the LiDAR should emulate the camera, not only be at the same place but also use the same intrinsic parameters. Hence, the point cloud in the LiDAR frame should first be transformed into the camera frame via extrinsic parameters between LiDAR and the camera. Then, the camera's intrinsic parameters are used to project these points to the image plane. The following equations can formulate this:

$${}^c\mathbf{p} = {}^c\mathbf{T} {}^l\mathbf{p} \quad (13)$$

$$d \begin{Bmatrix} u \\ v \\ 1 \end{Bmatrix} = \mathbf{K} {}^c\mathbf{p} \quad (14)$$

where ${}^c\mathbf{p}$ and ${}^l\mathbf{p}$ denote the point coordinate in the camera frame and LiDAR frame, and ${}^c\mathbf{T}$ denotes the transformation from the LiDAR frame to the camera frame, which is the extrinsic matrix. \mathbf{K} represents the intrinsic matrix, while u and v denote the coordinates within the image. The depth information is encapsulated in the variable d . The image under consideration exhibits sparsity, featuring several distinct lines. Notably, each pixel with a non-zero value is embedded with the depth information.

4.7 | Combination and semantic points reprojection

In this step, the semantic result and the depth information are merged in the image domain. Suppose the image size is $m \times n$, then the semantic output size is $m \times n \times 1$, and the “image” size from LiDAR is also $m \times n \times 1$. Combine the two sources pixel by pixel. The shape of the output turns to be $m \times n \times 2$, which contains both the semantic and depth information. Figure 6c shows the visualization of the combination.

After the combination, some pixels possess depth information, and reprojection is subsequently conducted. Each pixel is first projected from the image frame to the camera frame and then transformed from the camera frame to the LiDAR frame. The detail is as follows:

$${}^c\mathbf{p} = \mathbf{K}^{-1} d \begin{Bmatrix} u \\ v \\ 1 \end{Bmatrix} \quad (15)$$

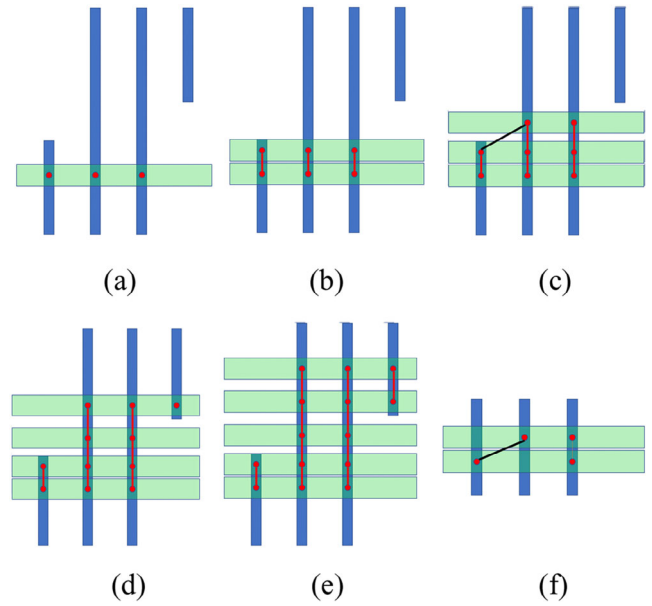


FIGURE 5 The process of vectorization for lane lines.

$${}^l\mathbf{p} = ({}^c\mathbf{T})^{-1} {}^c\mathbf{p} \quad (16)$$

Based on Equations (15) and (16), the semantic points with depth are reprojected to the LiDAR frame.

4.8 | Splicing and vectorization

To this point, the semantic image has been reprojected to the 3D domain, but it should be noticed that only a sparse point cloud is obtained. With the help of the pose estimation from SPAN-CPT or LiDAR odometry and mapping (LOAM), the sparse point cloud can be stacked to be a dense point cloud. It is stored as the HD semantic map (HDSM).

This paper mainly vectorises lane lines and poles as the two elements can represent the horizontal and vertical features, significantly aiding map-aided localization [72]. Notably, while lane lines are continuous, poles are discrete, necessitating distinct vectorization methods for each.

Vectorization for lane lines uses a growth strategy. Due to the sparse point cloud, a sliding window is utilized to maintain ten frames of the lane line point cloud, chosen from the whole semantic point cloud. DBSCAN [70] then segments the instance by clustering the lane line points. Once each lane line is determined, the centre point is calculated to represent the current lane line. Given lane lines' continuity, clustering focuses on points within a set distance (5 m, e.g.). With the moving of the vehicle, new centre points and previous centre points are connected to be a polyline, and lane lines are vectorised. The detail of the process is shown in Figure 5.

Illustrations from (a) to (e) in Figure 5 show the vectorization process for lane lines. Blue lines are the lane lines, the green box is the sliding window, and the red points denote the cluster centre point. First, in (a), centre points are generated, and in (b), new centre points are generated with the window

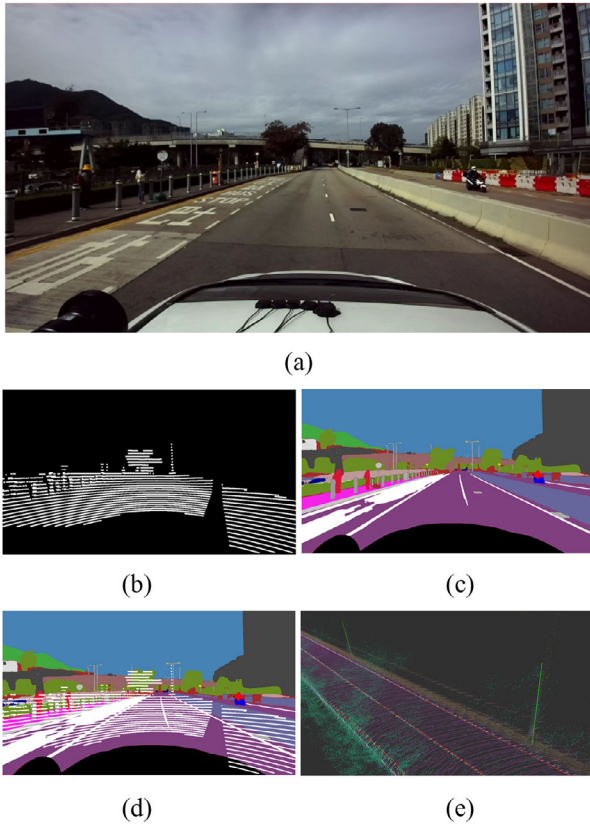


FIGURE 6 (a) The original image; (b) the LiDAR image; (c) the semantic image; and (d) the integration of LiDAR image and semantic image. The depth information is in the white lines. (e) The generated HDVM and HDVM. After vectorization, the red and green solid lines are the lane lines and poles.

moving. Calculate all the distance between each previous and current centre point and connect the nearest point in the previous centre points. However, when lanes are merging in case (c), this will make an issue that one current centre point connects two previous centre points. Another case (f) is that the cluster fails to connect the current centre point with the previous one in the same lane, causing different lane lines to be connected. The black lines are the mismatched lines. To prevent these two cases, a limitation should be added: the connection distance should not be longer than the width of the lanes. In the context of lane bifurcation, as depicted in Figure 5d, it is noteworthy that no complications arise. It can be attributed to connections emanating from the previously established centre points. As a result, two points on the same lane line in two different time frames must have the shortest distance. Figure 5e acts as a sequential extension of Figure 5d, and the progression remains uninterrupted.

Vectorization for poles is simpler than lane lines because different poles are discrete, and the serial number of the pole category can also filter the pole point cloud. Therefore, DBSCAN can directly segment all the poles, and for each pole cluster, the centre points are calculated, and its z value is the maximum z -axis value of its cluster.

Figure 6 visualizes the result of each step of the HDVM construction: (a) is the original image, (b) is the semantic segmentation result, (c) is the LiDAR and semantic image

combination, and (d) includes the generated HDVM and HDVM.

5 | HDVM ERROR QUANTIFICATION

In this framework, many factors can cause errors and affect the HDVM accuracy, such as camera intrinsic parameters, camera LiDAR time synchronization, etc. However, there are two primary error sources. The first is the semantic segmentation accuracy, and the second is the LiDAR-camera extrinsic parameters accuracy. These two factors enormously influence semantic map accuracy. Quantifying the errors in these two specific factors is crucial for ensuring the safety of using the map and for making necessary updates or improvements. This section will try quantifying the error propagation scheme.

5.1 | Process review

First, the primary process of the proposed pipeline in Section 4 needs to be reviewed to help perform the error quantification, which can be formulated as follows:

- 1) Transform the points in the LiDAR frame to the camera frame via extrinsic parameters and project them to the image plane via intrinsic parameters using Equations (13) and (14).

The data can be expressed as a 2D array D :

$$D(u, v) = d \quad (17)$$

$D(*)$ can be regarded as a function in which the input u and v are the pixel coordinates, and d is the depth information.

- 2) Perform the semantic segmentation in the image domain, and the semantic result can also be expressed as a 2D array S :

$$S(u, v) = c \quad (18)$$

Also, the segmentation can be regarded as a function $S(*)$, and c is the pixel class.

- 3) Combine the projection in Equation (17) and semantic information in Equation (18), and a new semantic array $C(*)$ with depth information can be obtained:

$$C(u, v) = \{d, c\} \quad (19)$$

- 4) Reproject these points to the 3D domain via the depth information, intrinsic and extrinsic parameters using Equations (15) and (16).

To clearly describe the issue, $\|_{m:n}$ and $\|_n$ denote the m th to the n th elements and the n th element in the vector. Moreover, to formulate this problem, assume that the z -axis coordinate is a constant g in the error quantification due to the lane line on the

ground. Furthermore, the extrinsic parameters matrix between LiDAR and the camera is obtained based on the Euler angle α, β, γ , which follows the $\tilde{x}-x-\tilde{z}$ rotation order and translation i, j, k :

$${}^c_l \mathbf{T} = \begin{bmatrix} \mathbf{R}(\alpha, \beta, \gamma) & \mathbf{t} \\ 0 & 1 \end{bmatrix} \quad (20)$$

where:

$$\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_{\tilde{x}}(\gamma) \mathbf{R}_x(\beta) \mathbf{R}_{\tilde{z}}(\alpha) \quad (21)$$

$$\mathbf{R}_{\tilde{x}}(\theta_{\tilde{x}}) = \begin{bmatrix} \cos\theta_{\tilde{x}} & -\sin\theta_{\tilde{x}} & 0 \\ \sin\theta_{\tilde{x}} & \cos\theta_{\tilde{x}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (22)$$

$$\mathbf{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad (23)$$

$$\mathbf{t} = [i, j, k]^T \quad (24)$$

The intrinsic parameters are denoted as:

$$\mathbf{K} = \begin{bmatrix} k_1 & 0 & p_1 \\ 0 & k_2 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (25)$$

where p_1 and p_2 represent the coordinate of the optical center in the image, k_1 and k_2 represent the focal length in terms of pixel units along the x-axis and y-axis respectively.

The coordinate of the point is defined as:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ g \end{bmatrix} \quad (26)$$

The homogeneous form is:

$$\hat{\mathbf{p}} = \begin{bmatrix} x \\ y \\ g \\ 1 \end{bmatrix} \quad (27)$$

The two kinds of errors are introduced in the combination step, as shown in Equation (19). Extrinsic parameters error makes the pixel project to a wrong location in the camera frame; thus, its influence will reflect in both the image domain and vectorised lane lines. Moreover, semantic segmentation error exists due to the accuracy of the segmentation neural network, and its effect will act on the vectorised lane lines. Therefore, the error quantifications are represented in three aspects:

- 1) LiDAR-camera extrinsic parameter errors in the image domain.

- 2) Semantic segmentation errors for vectorised lane line in the 3D domain.
- 3) LiDAR-camera extrinsic parameter errors for vectorised lane line in 3D domain.

5.2 | LiDAR-camera extrinsic parameter errors in the image domain

Errors in the extrinsic parameters will result in an identical spatial point projection to a disparate location within the image domain. Extrinsic parameter errors include the following errors: $\Delta ff, \Delta fi, \Delta fl, \Delta i, \Delta j, \Delta k$, which are the errors of Euler angle and translation. The affected transformation matrix changes from ${}^c_l \mathbf{T}$ to ${}^c_l \tilde{\mathbf{T}}$, which is the extrinsic parameters transformation with error:

$${}^c_l \tilde{\mathbf{T}} = \begin{bmatrix} \mathbf{R}(\alpha + \Delta ff, \beta + \Delta fi, \gamma + \Delta fl) & \mathbf{t}' \\ 0 & 1 \end{bmatrix} \quad (28)$$

where:

$$\mathbf{t}' = [i + \Delta i, j + \Delta j, k + \Delta k]^T$$

The location of the projected point in the image domain is:

$$\begin{bmatrix} u_e \\ v_e \\ 1 \end{bmatrix} = \frac{1}{d} \mathbf{K} [{}^c_l \tilde{\mathbf{T}} \hat{\mathbf{p}}]_{1:3} \quad (29)$$

where u_e and v_e are the location of the pixel projected by \mathbf{T}_e . Meanwhile, the ground truth of the pixel is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{d} \mathbf{K} [{}^c_l \mathbf{T} \hat{\mathbf{p}}]_{1:3} \quad (30)$$

Therefore, the distance between the pixel (u_e, v_e) and (u, v) can represent the error in the image domain.

5.3 | Semantic segmentation errors for vectorised lane line in the 3D domain

The lane line constitutes a unique element within a map, wherein its \tilde{x} -axis value can be regarded as a constant, denoted as g . This is a basic assumption for the error evaluation in this part. Semantic segmentation error is on the image domain. It can be described as the offset $\Delta u, \Delta v$ for the pixel location u, v . Then,

$$u + \Delta u$$

the pixel coordinate changes to $[v + \Delta v]$. Define the two depth

$$1$$

information d_1 and d_2 , and reproject the pixel point using the following equation:

$$\mathbf{p} = {}^c_l \mathbf{T}^{-1} \mathbf{K}^{-1} d_1 \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (31)$$

$$\mathbf{p}' = {}^c\mathbf{T}^{-1} \mathbf{K}^{-1} d_2 \begin{bmatrix} u + \Delta u \\ v + \Delta v \\ 1 \end{bmatrix} \quad (32)$$

Meanwhile, there is the assumption that:

$$[\mathbf{p}]_3 = \begin{bmatrix} {}^c\mathbf{T}^{-1} \mathbf{K}^{-1} d_1 \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \end{bmatrix}_3 = g \quad (33)$$

$$[\mathbf{p}']_3 = \begin{bmatrix} {}^c\mathbf{T}^{-1} \mathbf{K}^{-1} d_2 \begin{bmatrix} u + \Delta u \\ v + \Delta v \\ 1 \end{bmatrix} \end{bmatrix}_3 = g \quad (34)$$

Due to the constant assumption in equations in (33) and (34), d_1 and d_2 can be calculated. Then, substitute it into Equation (30), and the coordinate of the reprojected point can be calculated. The distance between \mathbf{p} and \mathbf{p}' can denote the error in the vectorized lane line.

5.4 | LiDAR-camera extrinsic parameter errors for vectorised lane line in the 3D domain

As Equation (28) shows, the transformation matrix turns to ${}^c\mathbf{T}$ from ${}^l\mathbf{T}$. For lane lines, extrinsic parameter errors make the original point and error point project to the same location. In an image frame, it can be formulated as:

$$\frac{1}{d'} \mathbf{K} [\tilde{\mathbf{T}}\hat{\mathbf{p}}]_{1:3} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{d} \mathbf{K} [{}^c\mathbf{T}\hat{\mathbf{p}}]_{1:3} \quad (35)$$

Then, there are equations below:

$$d = \left[\mathbf{K} [{}^c\mathbf{T}\hat{\mathbf{p}}]_{1:3} \right]_3 \quad (36)$$

$$d' = \left[\mathbf{K} [\tilde{\mathbf{T}}\hat{\mathbf{p}}']_{1:3} \right]_3 \quad (37)$$

$$\mathbf{p}' = \frac{d'}{d} {}^c\tilde{\mathbf{T}}^{-1} {}^c\mathbf{T}\hat{\mathbf{p}} \quad (38)$$

Since it is assumed that the coordinate z value is a constant and then:

$$[\mathbf{p}']_3 = \frac{d'}{d} \left[{}^c\tilde{\mathbf{T}}^{-1} {}^c\mathbf{T}\hat{\mathbf{p}} \right]_3 = g \quad (39)$$

d' is the only unknown in Equation (39), so it can be solved. Taking the d' to Equation (38), and \mathbf{p}' can be solved. The distance between \mathbf{p} and \mathbf{p}' can denote the error in the vectorized lane line.



FIGURE 7 Sensor setup deployment during the experimental data collection. (a) The real sensor kit and the sensors we used are labelled. (b) The sensor kit on our Honda Fit around the science park. LiDAR stands for the light detection and ranging, while SPAN stands for the synchronous position, attitude navigation.

6 | EXPERIMENTAL VALIDATION

6.1 | Experimental setup

6.1.1 | Sensor setups

To validate the effectiveness of the proposed method, the experiments were conducted in two typical scenarios in Hong Kong. The first is an indoor scenario, the garage on the Hong Kong Polytechnic University (PolyU) campus, and the second is the driving road under the open sky around the science park in Tai Po, Hong Kong. The sensor kit in Figure 7 is equipped with a Velodyne HDL-32 LiDAR and a Zed 2 camera. Besides, the ground truth of the positioning with centimetre-level accuracy is postprocessed by commercial software from NovAtel using an integrated GNSS real-time kinematic and fibre optics gyroscope inertial system, under the open sky. All the data were collected and synchronized using the robot operating system (ROS) [73]. The LiDAR-Camera-RTK/INS extrinsic and camera intrinsic parameters were calibrated beforehand. Figure 8a,b indicates the actual garage and driving road scenarios. The garage and whole driving road lengths are 110 and 10000 m. In the experiment, the annotation needs to be manually set for evaluation. Considering that the results are not affected by the distance length, only a 500-m section of driving road was selected to validate the proposal efficiently.

Figure 8a is the scenario in the garage of PolyU. We only used Velodyne HDL-32 LiDAR and Zed 2 camera in this scenario. (b) is the scenario around the science park, Hong Kong. We used Velodyne HDL-32 LiDAR, Zed 2 camera, and SPAN-CPT in this open sky scenario. (c) and (d) are the HDSM for the garage and science park, respectively. (e) and (f) are the HDVM of the garage and science park, respectively.

Section 6.2 will quantify the theoretical error and explore the error propagation scheme. Sections 6.3 and 6.4 will show the experiments in garage and science park scenarios, respectively. The garage HDVM contains lane line vectors, and the science park HDVM contains lane line and pole vectors. To evaluate the accuracy of our HDVM, cloud-to-cloud distance is used as the metric. Specifically, the ground truth and HDVM vector points are interpolated at 0.01-m intervals to be a dense point cloud. Subsequently, the ground truth point cloud is used as the

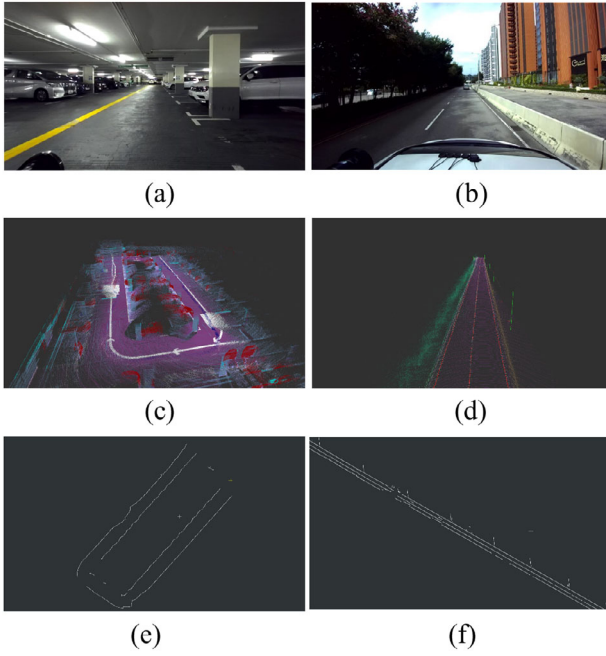


FIGURE 8 (a) The garage scenario and (b) the science park scenario. (c,d) are their HDSM in the colour point cloud, while (e,f) are the HDVM for them, respectively.

reference to find the nearest point for each point in the HDVM point cloud. The distance between the point in HDVM and its nearest point in ground truth is defined as the error of the point.

6.2 | Error quantifying

This section will quantify the error from semantic segmentation and LiDAR-camera extrinsic parameters. First, the error of the extrinsic parameters will be quantified in the 2D domain via Equations (29) and (30). Then, the lane line elements will be quantified in the 3D domain using Equations (32) and (38). In this part, our calibrated transformation matrix from LiDAR to the camera is as follows:

$${}^c\mathbf{T}_{ours} = \begin{bmatrix} 0.9999 & 0.0048 & -0.0010 & 0.1135 \\ -0.0009 & -0.0245 & -0.9997 & -0.1617 \\ -0.0048 & 0.9997 & -0.0245 & 0.0516 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (40)$$

Once the sensors are fixed on the sensor kit, extrinsic parameters are invariant in normal conditions. Thus, extrinsic parameters are off-line calibrated to guarantee that all sensors coincide.

The intrinsic parameters of our camera are:

$$\mathbf{K} = \begin{bmatrix} 543.5046 & 0 & 630.7183 \\ 0 & 540.5383 & 350.9063 \\ 0 & 0 & 1 \end{bmatrix} \quad (41)$$

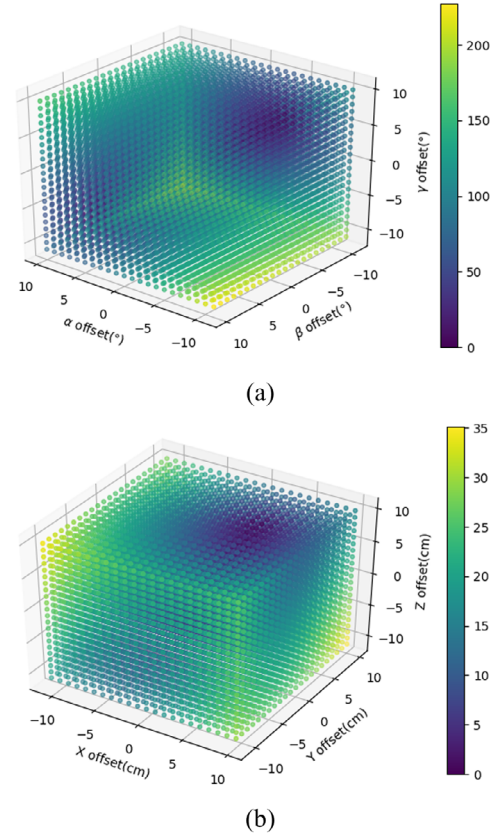


FIGURE 9 (a,b) The distance errors by rotation and translation errors using our extrinsic parameters.

The ground truth point which is chosen for evaluation is:

$$\mathbf{p} = [1, 3, -1.1]^T \quad (42)$$

The proposed method can provide more trustworthy initial guesses than the conventional method in the scenario constructed for this experiment. Moreover, the proposed method can complete the entire matching and pose estimation process in the challenging scenario, which is an essential improvement over the conventional method.

Since extrinsic parameters includes rotation error and translation error, in our quantification, we fix one when evaluating the other one. And there still are three error sources both in rotation error and translation error, thus, the axis x, y, z represents $\Delta f_f, \Delta f_i, \Delta f_l$ in rotation error in Figures 9a and 11a, and $\Delta i, \Delta j, \Delta k$ in translation error in Figures 9b and 11b. And the colour indicates the value of the error. The colour closer to yellow means a more significant error. For Figure 10, the x and y -axis denote the horizontal and vertical pixel errors on image, and the z -axis denotes the error value.

6.2.1 | LiDAR-camera extrinsic parameter errors quantifying in the image domain

The error quantifying in the image domain is to measure the distance between the ground truth and the projected point in

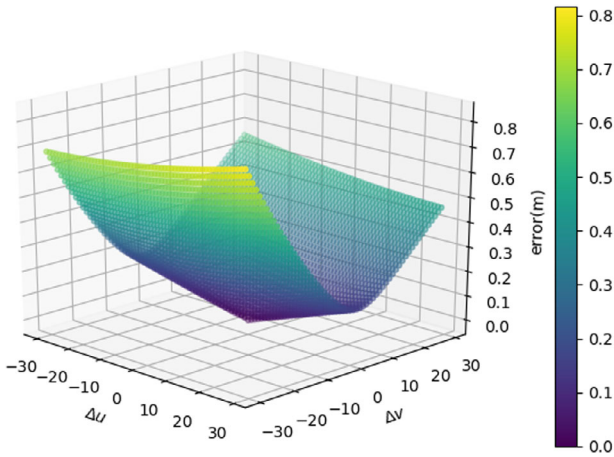


FIGURE 10 The impact of semantic segmentation error.

the image. According to Equations (29) and (30), the error in pixel position, defined as the discrepancy between the point location projected via our calibrated extrinsic parameters, can be formulated as:

$$e = \sqrt{(u - u_e)^2 + (v - v_e)^2} \quad (43)$$

The variables in the equation are the errors of the extrinsic parameters, including Δf , Δf_i , Δf_l , Δi , Δj , Δk . The pixel position errors derived from the angle error Δf , Δf_i , Δf_l and the translation error Δi , Δj , Δk are evaluated, in which the intervals of angle error and translation error are from -10° to 10° and -10 to 10 cm, respectively. The unit of the error is one pixel.

All the resulting computations utilize the intrinsic matrix \mathbf{K} specified in Equation (41). Both (a) and (b) employ our transformation delineated in Equation (40). As shown in Figure 9, angle error dramatically affects the result, as the maximum error is about 300 pixels, especially the Δf , which is the first rotation angle offset around z -axis. As for translation error, it has less influence, as the maximum error is about 37 pixels.

6.2.2 | Error quantifying for vectorised lane lines in the 3D domain

In the theoretical analysis, the error reprojected point \mathbf{p}' caused by segmentation and extrinsic parameters errors are obtained from Equations (32) and (38), respectively. Therefore, the metric can be the Euclidean distance between \mathbf{p}' and the ground truth point \mathbf{p} :

$$e = \|\mathbf{p} - \mathbf{p}'\|_2 \quad (44)$$

In this evaluation, the ground truth point is on the right front, and its coordinate is $\mathbf{p} = [1, 3, -1.1]^T$, which means the ground in the LiDAR frame is 1.1 m below the LiDAR, and the constant g is 1.1 m.

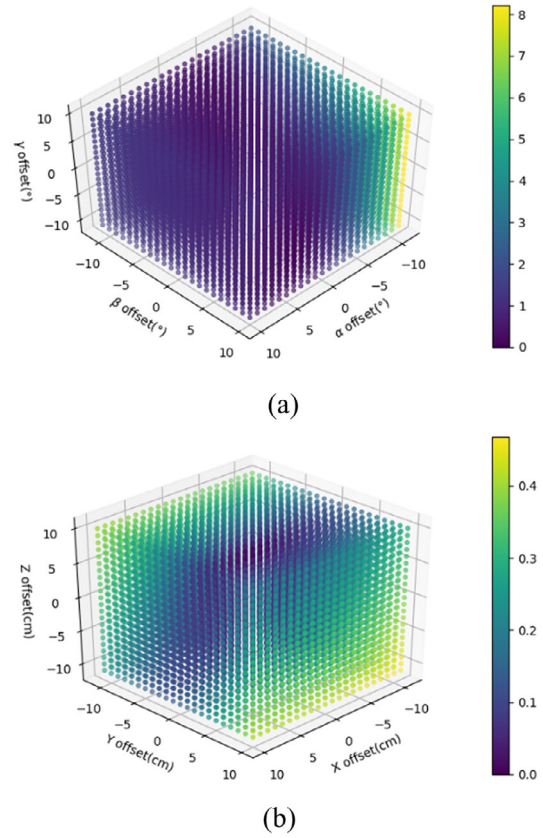


FIGURE 11 The qualified rotation and translation error of the evaluated point.

Semantic segmentation errors

Semantic segmentation error is affected by Δu , Δv . The impact of semantic segmentation error is shown in Figure 10, which indicates errors using our extrinsic parameters when pixel location offsets fall between -30 pixels and 30 pixels.

In this situation, it can be inferred that the error in the horizontal direction affects the accuracy less. However, a little error in the vertical direction can cause a huge error.

LiDAR-camera extrinsic parameter errors

As for the error of the extrinsic parameters, it consists of Δf , Δf_i , Δf_l , Δi , Δj , Δk . To show the result, the rotation in Figure 11a is changed from -10° to 10° for each Euler angle, and the translation in Figure 11b is changed from -10 to 10 cm. Still, the error level is represented by different colours, and colours close to yellow mean more error.

Figure 11a shows that changing the Δf angle to -10° and the Δf_i angle to 10° can cause the most significant error. Δf influences the projection in the vertical direction. In the context of the function $\tan(\alpha)$, even a tiny change in the angle α can cause a substantial change in the value of the function. Because the point $[1, 3, -1.1]^T$ is on the right, changing the Δf_i to 10° means swerving to the left. These can explain why changing Δf to the increasing direction and Δf_i to the decreasing direction in the ENU frame can cause the most significant error.

TABLE 1 The MIOU of the point cloud semantic segmentation algorithms on semantic KITTI.

Methods	PointNet	PointNet++	SPLATNet (sparse lattice network)	RangeNet++	Ours
mIoU	14.6	20.1	22.8	52.2	23.3

Figure 11b shows how the translation error affects the result. The error in the z -axis affects the result most. However, in contrast to the rotation error, translation error has less influence, and the worst situation is only 0.4 m. It shows that the rotation parameters are more significant to be finetuned and optimized.

6.3 | Point cloud semantic segmentation performance on semantic KITTI

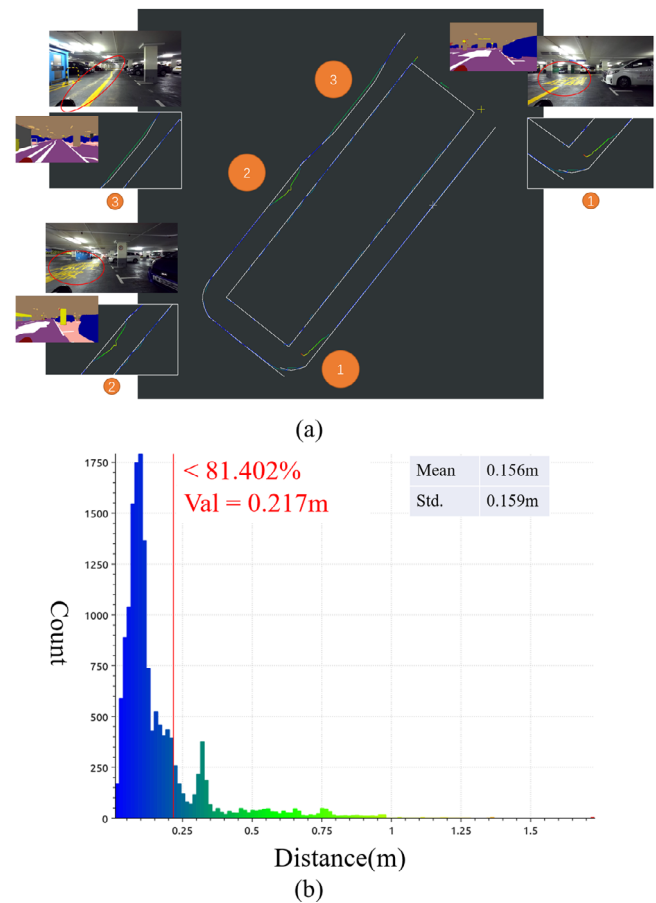
To evaluate the robustness and accuracy of our pipeline, we conduct an indirect validation on Semantic KITTI [74]. However, the official evaluation of Semantic KITTI requires classifying all the points on their testing dataset, and our method is limited to the field-of-view (FOV) of the front camera, which means our method can only process the front part of the point cloud in one frame. Thus, we can only test our method on the training dataset because it provides the ground truth. So, the evaluation is based on two assumptions: (1) Our method performs similarly inside and outside the FOV. (2) Our method performs similarly on the training dataset and testing dataset of Semantic KITTI. We compared our methods to PointNet [40], PointNet++ [41], SPLATNet [75] and RangeNet53++ [46]. The results are shown in Table 1.

Our source code can be found in our repository. The deep learning part of our pipeline is not trained on this dataset, but the performance still exceeds PointNet, PointNet++ and SPLATNet. RangeNet++ performs the best on the dataset, but there is a possibility that their model might fail on different datasets. The generalization of our method proves to be effective, and it will lead to a decrease in the HDVM construction error.

6.4 | Experimental results in the garage of PolyU campus

To evaluate the performance of the proposed method, the framework is first run to construct the HDSM and HDVM of the garage in PolyU. The road is elliptical, about 40 m long, and 15 m wide under the ground. It is a typical indoor scenario, and the GNSS system is unavailable. Only LiDAR and the camera of the sensor kit are used. It will lead to a lack of LiDAR distortion correction. Therefore, the sensor kit is put on a trolley and is pushed to move slowly to avoid distortion. Due to a lack of GNSS, LIO-SAM [76] was utilized to generate the localizations and orientations, which ensures an average accuracy of 0.12 m. The generated HDVM can be seen in Figure 8c.

In the garage scenario, the lane line is the main element to be vectorised. Figure 12 visualizes our HDVM and the ground

**FIGURE 12** The HDVM for the garage in PolyU. Std stands for the standard deviation, while Val stands for the value.

truth. The white line is the manually annotated ground truth, and the coloured line is the HDVM. Different colours represent the error magnitudes of each point, which is changed from blue to yellow with the error increment.

As a result, the mean error of this scenario is 0.156 m, and the standard deviation (Std) is 0.159 m. The histogram in Figure 12b shows that the error of 81.402% points is less than 0.217 m. In statistics, the percentile 75% (Quartile 3, Q3) is usually chosen to be concerned, but if we choose the less percentage, the error value will be smaller. So, we choose the bound as 80%. Given that 80% cannot be precisely labelled on the figure. Therefore, we choose the nearest points, which is 81.402%. It can be observed from Figure 12a that inaccurate parts mainly appear in parts 1, 2, and 3. In parts 1 and 2, some features are mistaken for lane lines, leading to relatively significant errors. As for part 3, the real lane line and the deprecated lane line beside it are all segmented as lane lines, which makes the wrong cluster centre and leads to the error.

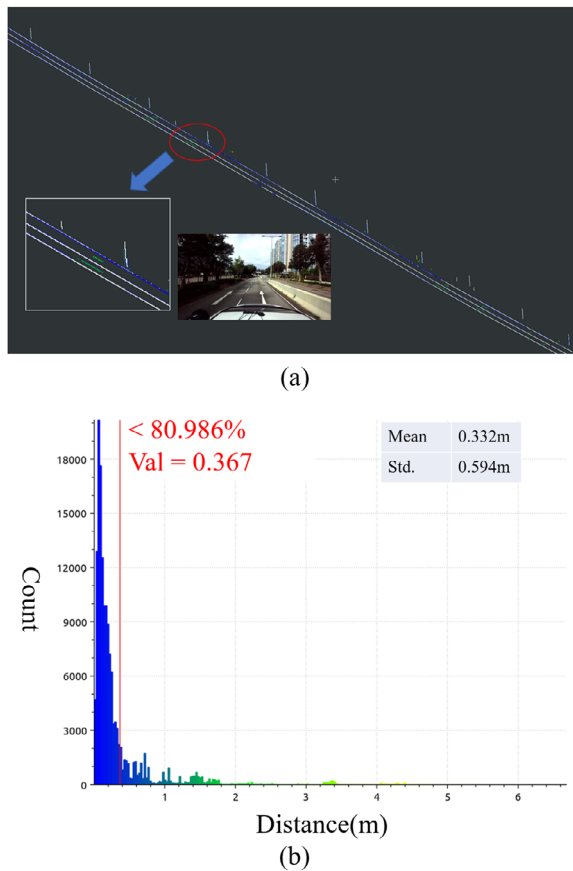


FIGURE 13 The HDVM around the science park. Std stands for the standard deviation, while Val stands for the value.

6.5 | Experimental results in science park

A real driving scenario was tested around the science park. In this scenario, the lane line and pole are vectorised. SPAN-CPT was used to get high-frequency and accurate localizations and orientations under the open sky. The road for evaluation is about 0.52 km. The construction wholly followed our proposed pipeline. The generated HDVM can be seen in Figure 8f.

In Figure 13a, the white solid lines are the manually annotated ground truth, and the coloured lines are the generated HDVM. Also, each colour represents its error magnitude, which changes from blue to red as the error increases.

In this scenario, the mean error is 0.332 m, and the Std is 0.594 m. The outliers shown in Figure 13a are the traffic signs on the ground, which are classified as lane lines in Mapillary Vistas but not annotated in our ground truth. In general, the error of 80.986% of HDVM points is less than 0.367 m, as shown in Figure 13b. The selection of 80.986% shares the same explanation as the previous section.

6.6 | Discussion

In the theoretical analysis in the last section, semantic segmentation and rotation errors play essential roles in promising

accuracy. However, these two experiments show that the error is mainly from the significant missing and incorrect parts caused by semantic segmentation. The tiny error caused by small shifting pixels from semantic segmentation and slight rotation or translation from extrinsic parameters will not exert a significant influence due to the clustering step. Only the central point in the cluster will be used during this step. As a result, those inaccurate points located on the cluster's edge will be ignored. Therefore, offline calibration of extrinsic parameters with minor errors is acceptable. Meanwhile, valid semantic segmentation is fundamental and directly influences the accuracy of the result.

7 | CONCLUSIONS AND PROSPECTS

This paper proposes an automatic HDVM construction framework using the camera, LiDAR, and GNSS/INS integrated navigation system. The proposed method sets up a refined pipeline for HDVM construction, considering internal and external sensor errors at the front end of the process. Unlike the previous work, our pipeline considers the ground and vertical elements. On this basis, the recently developed Swin transformer is used in the pipeline to extract semantic information, which can achieve better effects for semantic segmentation. The pipeline combines the semantic information and depth information to reconstruct the HDVM. The theoretical error is quantified by formulating the error propagation scheme. From the results of indoor and outdoor experiments, our pipeline can achieve sub-meter accuracy. In short, the proposed method for HDVM construction enables reliable and incremental HDVM generation.

For future studies, we would think about more elements that can be vectorised to aid localization and navigation. Besides, we would further improve the accuracy of the HDVM from semantic segmentation and extrinsic parameters calibration. Though minor errors from extrinsic parameters would be mitigated by clustering, it is still vital to HDSM. HDSM has enormous potential on the server side, especially in using crowdsourced data to update the HD map because HDVM drops much information.

AUTHOR CONTRIBUTIONS

Runzhi Hu: Conceptualization; methodology; software; validation; writing—original draft. **Shiyu Bai:** Methodology; investigation; writing—review and editing. **Weisong Wen:** Conceptualization; formal analysis; funding acquisition; validation. **Xin Xia:** Writing—review and editing. **Li-Ta Hsu:** Project administration; supervision.

ACKNOWLEDGEMENTS

This work was supported by the Guangdong Basic and Applied Basic Research Foundation (2021A1515110771) and University Grants Committee of Hong Kong under the scheme Research Impact Fund (R5009-21). This research was also supported by the Faculty of Engineering, The Hong Kong Polytechnic University under the project “Perception-based GNSS PPP-RTK/LVINS integrated navigation system for unmanned autonomous systems operating in urban canyons”. The authors

also would like to thank their colleague Feng Huang, Yihan Zhong and Dr. Guohao Zhang for their kind help with the data collection.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

Data available on request from the authors

ORCID

Shiyu Bai  <https://orcid.org/0009-0001-6082-6520>

REFERENCES

- Zhao, C., Lv, Y., Jin, J., Tian, Y., Wang, J., Wang, F.-Y.: Decast in transverse for parallel intelligent transportation systems and smart cities: Three decades and beyond. *IEEE Intell. Transp. Syst. Mag.* 14(6), 6–17 (2022)
- Yao, H., Qin, R., Chen, X.: Unmanned aerial vehicle for remote sensing applications—A review. *Remote. Sens.* 11(12), 1443 (2019)
- Wen, W., Zhou, Y., Zhang, G., Fahandezh-Saadi, S., Bai, X., Zhan, W., Tomizuka, M., Hsu, L.-T.: Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes. In: *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2310–2316. IEEE, Piscataway, NJ (2020)
- Chang, L., Niu, X., Liu, T., Tang, J., Qian, C.: Gns/Ins/Lidar-Slam integrated navigation system based on graph optimization. *Remote. Sens.* 11(9), 1009 (2019)
- Ando, T., Kugimiya, W., Hashimoto, T., Momiyama, F., Aoki, K., Nakano, K.: Lateral control in precision docking using Rtk-Gns/Ins and Lidar for localization. *IEEE Trans. Intell. Veh.* 6(1), 78–87 (2020)
- Fernandez, A., Wis, M., Silva, P.F., Colomina, I., Pares, E., Dovic, F., Ali, K., Friess, P., Lindenberger, J.: Gns/Ins/Lidar integration in urban environment: Algorithm description and results from Atenea Test Campaign. In: *Proceedings of the 2012 6th ESA Workshop on Satellite Navigation Technologies (Navitec 2012) & European Workshop on GNSS Signals and Signal Processing*, pp. 1–8. IEEE, Piscataway, NJ (2012)
- Groves, P.D.: Principles of Gns, inertial, and multisensor integrated navigation systems. *IEEE Aerosp. Electron. Syst. Mag.* 30(2), 26–27 (2015)
- Wen, W., Zhang, G., Hsu, L.T.: Correcting Nlos by 3d Lidar and building height to improve Gns single point positioning. *Navigation* 66(4), 705–718 (2019)
- Chalvatzaras, A., Pratikakis, I., Amanatiadis, A.A.: A Survey on map-based localization techniques for autonomous vehicles. *IEEE Trans. Intell. Veh.* 8(2), 1574–1596 (2022)
- Levinson, J., Montemerlo, M., Thrun, S.: Map-based precision vehicle localization in urban environments. In: *Robotics: Science and systems*. Citeseer (2007)
- Wan, G., Yang, X., Cai, R., Li, H., Zhou, Y., Wang, H., Song, S.: Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In: *Proceedings of the 2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4670–4677. IEEE, Piscataway, NJ (2018)
- Ding, W., Hou, S., Gao, H., Wan, G., Song, S.: Lidar inertial odometry aided robust Lidar localization system in changing city scenes. In: *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4322–4328. IEEE, Piscataway, NJ (2020)
- Kim, S., Kim, S., Seok, J., Ryu, C., Hwang, D., Jo, K.: Road shape classification-based matching between lane detection and Hd map for robust localization of autonomous cars. *IEEE Trans. Intell. Veh.* 8(5), 3431–3443 (2023)
- Zhang, Z., Zhao, J., Huang, C., Li, L.: Learning visual semantic map-matching for loosely multi-sensor fusion localization of autonomous vehicles. *IEEE Trans. Intell. Veh.* 8(1), 358–367 (2022)
- Li, L., Yang, M., Li, H., Wang, C., Wang, B.: Robust localization for intelligent vehicles based on compressed road scene map in urban environments. *IEEE Trans. Intell. Veh.* 8(1), 250–262 (2022)
- Tao, Z., Bonnifait, P.: Sequential data fusion of Gns Pseudoranges and Dopplers with map-based vision systems. *IEEE Trans. Intell. Veh.* 1(3), 254–265 (2016)
- Hansson, A., Korsberg, E., Maghsood, R., Nordén, E.: Lane-level map matching based on Hmm. *IEEE Trans. Intell. Veh.* 6(3), 430–439 (2020)
- Zhang, K., Liu, S., Dong, Y., Wang, D., Zhang, Y., Miao, L.: Vehicle positioning system with multi-hypothesis map matching and robust feedback. *IET Intel. Transport Syst.* 11(10), 649–658 (2017)
- Schweizer, J., Bernardi, S., Rupi, F.: Map-matching algorithm applied to bicycle global positioning system traces in Bologna. *IET Intel. Transport Syst.* 10(4), 244–250 (2016)
- Song, C., Yan, X., Stephen, N., Khan, A.A.: Hidden Markov model and driver path preference for floating car trajectory map matching. *IET Intel. Transport Syst.* 12(10), 1433–1441 (2018)
- Lee, S., Choi, J., Seo, S.W.: Ego-Lane index-aware vehicular localisation using the Deeproad network for urban environments. *IET Intel. Transport Syst.* 15(3), 371–386 (2021)
- Zhong, Y., Huang, F., Zhang, J., Wen, W., Hsu, L.T.: Low-cost solid-state Lidar/Inertial-based localization with prior map for autonomous systems in urban scenarios. *IET Intel. Transport Syst.* 17(3), 474–486 (2023)
- Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monroy, A., Ando, T., Fujii, Y., Azumi, T.: Autoware on board: Enabling autonomous vehicles with embedded systems. In: *Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPs)*, pp. 287–296. IEEE, Piscataway, NJ (2018)
- Qian, C., Liu, H., Tang, J., Chen, Y., Kaartinen, H., Kukko, A., Zhu, L., Liang, X., Chen, L., Hyypä, J.: An integrated Gns/Ins/Lidar-Slam positioning method for highly accurate forest stem mapping. *Remote. Sens.* 9(1), 3 (2016)
- Xiao, Z., Yang, D., Wen, T., Jiang, K., Yan, R.: Monocular localization with vector Hd map (Mlvhm): A low-cost method for commercial IVS. *Sensors* 20(7), 1870 (2020)
- Xiao, Z., Jiang, K., Xie, S., Wen, T., Yu, C., Yang, D.: Monocular vehicle self-localization method based on compact semantic map. In: *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3083–3090. IEEE, Piscataway, NJ (2018)
- Zhang, C., Liu, H., Li, H., Guo, K., Yang, K., Cai, R., Li, Z.: Dt-Loc: Monocular visual localization on Hd vector map using distance transforms of 2d semantic detections. In: *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1531–1538. IEEE, Piscataway, NJ (2021)
- Zhang, C., Liu, H., Xie, Z., Yang, K., Guo, K., Cai, R., Li, Z.: Avp-Loc: Surround view localization and relocalization based on Hd vector map for automated valet parking. In: *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5552–5559. IEEE, Piscataway, NJ (2021)
- Qin, T., Zheng, Y., Chen, T., Chen, Y., Su, Q.: A light-weight semantic map for visual localization towards autonomous driving. In: *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11248–11254. IEEE, Piscataway, NJ (2021)
- Qin, T., Chen, T., Chen, Y., Su, Q.: Avp-Slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot. In: *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5939–5945. IEEE, Piscataway, NJ (2020)
- Kim, K., Cho, S., Chung, W.: Hd map update for autonomous driving with crowdsourced data. *IEEE Rob. Autom. Lett.* 6(2), 1895–1901 (2021)
- Li, B., Song, D., Kingery, A., Zheng, D., Xu, Y., Guo, H.: Lane marking verification for high definition map maintenance using crowdsourced images. In: *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2324–2329. IEEE, Piscataway, NJ (2020)
- Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: common practices and emerging technologies. *IEEE Access* 8, 58443–58469 (2020)

34. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L.: Swin transformer V2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11999–12009. IEEE, Piscataway, NJ (2022)
35. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9992–10002. IEEE, Piscataway, NJ (2021)
36. Kammel, S., Pitzer, B.: Lidar-based lane marker detection and mapping. In: Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, pp. 1137–1142. IEEE, Piscataway, NJ (2008)
37. Ghallabi, F., Nashashibi, F., El-Haj-Shade, G., Mittet, M.: Lidar-based lane marking detection for vehicle positioning in an Hd map. In: Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2209–2214. IEEE, Piscataway, NJ (2018)
38. Hata, A., Wolf, D.: Road marking detection using Lidar reflective intensity data and its application to vehicle localization. In: Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 584–589. IEEE, Piscataway, NJ (2014)
39. Jung, J., Che, E., Olsen, M.J., Parrish, C.: Efficient and robust lane marking extraction from mobile Lidar point clouds. ISPRS J. Photogramm. Remote Sens. 147, 1–18 (2019)
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 77–85. IEEE, Piscataway, NJ (2017)
41. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems 30, (2017)
42. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499. IEEE, Piscataway, NJ (2018)
43. Liu, Y., Tian, B., Lv, Y., Li, L., Wang, F.-Y.: Point cloud classification using content-based transformer via clustering in feature space. IEEE/CAA J. Autom. Sin. 11(1), 231–239 (2024)
44. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12689–12697. IEEE, Piscataway, NJ (2019)
45. Zhou, Y., Sun, P., Zhang, Y., Angelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in Lidar point clouds. In Proceedings of the Conference on Robot Learning, pp. 1–10. PMLR, Microtome Publishing, Brookline, MA (2020)
46. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate Lidar semantic segmentation. In: Proceedings of the 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 4213–4220. IEEE, Piscataway, NJ (2019)
47. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM 60(6), 84–90 (2017)
48. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778. IEEE, Piscataway, NJ (2016)
49. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, (2014)
50. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Angelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9. IEEE, Piscataway, NJ (2015)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. 30, (2017)
52. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-Cnn. In Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988. IEEE, Piscataway, NJ (2017)
53. Xiang, Z., Bao, A., Su, J.: Hybrid bird's-eye edge based semantic visual slam for automated valet parking. In: Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 11546–11552. IEEE, Piscataway, NJ (2021)
54. Yu, Y., Li, J., Guan, H., Jia, F., Wang, C.: Learning hierarchical features for automated extraction of road markings from 3-D mobile Lidar point clouds. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8(2), 709–726 (2015)
55. Huang, Y., Li, Y., Hu, X., Ci, W.: Lane detection based on inverse perspective transformation and Kalman filter. KSII Trans. Internet Inf. Syst. 12(2), 643–661 (2018)
56. Guo, C., Kidono, K., Meguro, J., Kojima, Y., Ogawa, M., Naito, T.: A low-cost solution for automatic lane-level map generation using conventional in-car sensors. IEEE Trans. Intell. Transp. Syst. 17(8), 2355–2366 (2016)
57. Mattys, G., Wang, S., Fidler, S., Urtasun, R.: Hd maps: Fine-grained road segmentation by parsing ground and aerial images. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3611–3619. IEEE, Piscataway, NJ (2016)
58. Paz, D., Zhang, H., Li, Q., Xiang, H., Christensen, H.I.: Probabilistic semantic mapping for urban autonomous driving applications. In: Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2059–2064. IEEE, Piscataway, NJ (2020)
59. Elhousni, M., Lyu, Y., Zhang, Z., Huang, X.: Automatic building and labeling of Hd maps with deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 13255–13260. MIT Press, Cambridge, MA (2020)
60. Li, Q., Wang, Y., Wang, Y., Zhao, H.: Hdmapnet: A local semantic map learning and evaluation framework. arXiv:2107.06307, (2021)
61. Liu, R., Wang, J., Zhang, B.: High definition map for automated driving: Overview and analysis. J. Navig. 73(2), 324–341 (2020)
62. Zhang, P., Zhang, M., Liu, J.: Real-time Hd map change detection for crowdsourcing update based on mid-to-high-end sensors. Sensors (Basel) 21(7), (2021)
63. Hsu, L.-T., Kubo, N., Wen, W., Chen, W., Liu, Z., Suzuki, T., Meguro, J.: Urbannav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas. In: Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), pp. 226–256. Institute of Navigation, Manassas, VA (2021)
64. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. 22(11), 1330–1334 (2000)
65. Li, S., Xu, C., Xie, M.: A robust O(N) solution to the perspective-N-point problem. IEEE Trans. Pattern Anal. Mach. Intell. 34(7), 1444–1450 (2012)
66. Besl, P.J., McKay, N.D.: A method for registration of 3D shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14(2), 239–256 (1992)
67. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440. IEEE, Piscataway, NJ (2015)
68. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, pp 234–241. Springer, Cham (2015)
69. Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P.: The Mapillary vistas dataset for semantic understanding of street scenes. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5000–5009. IEEE, Piscataway, NJ (2017)
70. Ester, M., Krieger, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press, Washington D.C (1996)
71. Mallon, J., Whelan, P.F.: Precise radial un-distortion of images. In: Proceedings of the 17th International Conference on Pattern Recognition, pp. 18–21. IEEE, Piscataway, NJ (2004)
72. Wen, T., Xiao, Z., Wijaya, B., Jiang, K., Yang, M., Yang, D.: High precision vehicle localization based on tightly-coupled visual odometry and vector Hd map. In: Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 672–679. IEEE, Piscataway, NJ (2020)

73. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: An open-source robot operating system. *ICRA Workshop Open Source Software* 3(3.2), 1–5 (2009)
74. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of Lidar sequences. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9296–9306. IEEE, Piscataway, NJ (2019)
75. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2530–2539. IEEE, Piscataway, NJ (2018)
76. Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., Rus, D.: Lio-Sam: Tightly-coupled Lidar inertial odometry via smoothing and mapping. In: *Proceedings of the 2020 IEEE/RSJ International Conference on*

Intelligent Robots and Systems (IROS), pp. 5135–5142. IEEE, Piscataway, NJ (2020)

How to cite this article: Hu, R., Bai, S., Wen, W., Xia, X., Hsu, L.-T.: Towards high-definition vector map construction based on multi-sensor integration for intelligent vehicles: Systems and error quantification. *IET Intell. Transp. Syst.* 18, 1477–1493 (2024). <https://doi.org/10.1049/itr2.12524>