Enhancing 3-D LiDAR Point Clouds With Event-Based Camera

Boyang Li, Hao Meng, Yuzhang Zhu, Rihui Song, Mingyue Cui, Gang Chen, and Kai Huang, *Member, IEEE*

***Abstract*— Point clouds from 3-D light detection and ranging (LiDAR) are useful for roadside units (RSU) applications in intelligent transportation system (ITS). High-density LiDAR products are restricted by high cost while the low-density and cheap ones are usually insufficient to perceive. Event-based cameras react to the changes in light intensity and output dense event streams consisting of triggered pixels. Unfortunately, there currently lacks depth information for event cameras. In order to address these problems, this article presents an approach to enhance sparse 3-D LiDAR point clouds with event pixels from an event-based camera. In our approach, the depth values of event pixels are estimated based on the distribution models which they belong to. The distribution models of event pixels are determined by the spatial information of the neighboring LiDAR points in a structural manner which we called the physical structure. To verify our approach, we conduct several real-world experiments about RSU applications in ITS. Results demonstrate that our approach can effectively improve 3-D point clouds density. The average accuracy of 3-D and 2-D vehicle detection increase by a factor of 14.6 and 8.8, respectively.**

***Index Terms*— Enhancement, event-based camera, intelligent transportation system (ITS), light detection and ranging (LiDAR), point clouds.**

I. Introduction

ROADSIDE units (RSU) are infrastructures equipped with smart sensors in intelligent transportation system (ITS) [1]. Sensor data from RSU not only enrich the source of perception but also provide information for traffic applications, such as monitoring and detection [2]. Because of the low price and the rich texture features [3], conventional cameras are chosen as the main sensors for RSU. However, the lack of direct accurate depth values is one of the urgent problems to be solved for cameras in ITS [4].

Manuscript received January 30, 2021; revised July 4, 2021; accepted July 6, 2021. Date of publication July 26, 2021; date of current version July 30, 2021. This work was supported in part by Shenzhen Basic Research Grants under Grant JCYJ20180507182508857 and Grant JCYJ20180508152434975, in part by the National Key Research and Development Program of China under Grant 2018YFB1802405, and in part by Guangdong Science and Technology Plan Project under Grant 2021A0505030024. The Associate Editor coordinating the review process was Dr. Yuya Koyama. *(Corresponding author: Kai Huang.)*

Boyang Li, Yuzhang Zhu, Rihui Song, Mingyue Cui, Gang Chen, and Kai Huang are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangdong 510275, China, and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou, Guangdong 510275, China (e-mail: liby3@mail2.sysu.edu.cn; zhuyzh9@mail2.sysu.edu.cn; songrh@mail2.sysu.edu.cn; cuimy@mail2.sysu.edu.cn; cheng83@mail.sysu.edu.cn; huangk36@mail.sysu.edu.cn).

Hao Meng is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangdong 510275, China, also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou, Guangdong 510275, China, and also with the Shenzhen Institute, Sun Yat-sen University, Guangzhou, Guangdong 510275, China (e-mail: mengh3@mail2.sysu.edu.cn).

Digital Object Identifier 10.1109/TIM.2021.3097862

In order to perceive the spatial environments, 3-D light detection and ranging (LiDAR) products are used to obtain depth information. Point clouds produced by 3-D LiDAR products have the potential for an effective complement to perceive on RSU because of the high-precision ranging results [5] and the ability to adapt to extreme lighting conditions. Considering the perception performance of LiDAR products, point clouds with higher density can significantly obtain more details of the environment [6]. Unfortunately, LiDAR products that can generate dense point clouds are generally expensive, which makes them not appropriate to be widely deployed on RSU. For instance, 32-channel and 64-channel Velodyne LiDAR products are about \$40 000 and \$80 000, respectively. In order to generate high-density point clouds, fusing multi-source sensor data to enhance LiDAR is a solid solution. To obtain the relationship between LiDAR and camera, Xie *et al.* [7] proposed a calibration method directly based on the alignment between the two sensor data. For the fusion method, Maddern and Newman [8] fused the 3-D point clouds with dense images from a stereo camera with a probability method. Depth information of each pixel from stereo images has been estimated before fusing. Actually, in the applications of RSU such as monitoring and vehicle detection, moving objects are more important than static background. Therefore, it is not necessary to process all pixels of the image in these scenarios.

The event-based camera is inspired by the working mechanism of the human retina [9]. Event pixels from this camera individually perceive the relative changes in light intensity and report an event if a threshold is reached [10]. The output of event-based cameras is not a series of frames, but a continuous stream of asynchronous digital events [11]. As a result, there are three characteristics of event-based cameras. First, event-based cameras perceive the environment at a high frequency. Second, objects in motion could be captured and the redundant background is discarded when the event-based cameras are deployed on the roadside. Thirdly, the high dynamic range makes event-based cameras adaptable in some extreme lighting conditions [10]. The characteristics of event-based cameras lead to potential in some scenes that are challenging for conventional cameras [12]. We believe the characteristics of event-based cameras can be used to increase the density of point clouds for RSU applications.

Fusing data from LiDAR and event-based cameras to generate denser point clouds is not straightforward. First, compared with point clouds from LiDAR, each event pixel from event-based cameras only has a 2-D pixel-wise index (x, y, t, p) [13], where (x, y) represents the pixel position, t represents time stamp and p indicates the polarity of light

intensity variety. Event-based cameras lack depth information. Second, depth values of LiDAR points cannot be directly used for event pixels. There is no direct relationship between LiDAR points and event pixels on the same object because of the difference in the number and distribution. It is necessary to find the correspondence between event pixels and LiDAR points.

To solve the aforementioned problems, this article presents an approach to enhance the sparse LiDAR point clouds with event-based cameras. The main idea of our approach is to estimate the depth of each event pixel based on the spatial information of the neighboring LiDAR points in a structural manner which we called the physical structure. Given a set of 3-D LiDAR points and event pixels, the neighborhood of each pixel is determined based on the area division by the Voronoi Diagram [14]. Then the physical structure where the event pixels are located is extracted by several LiDAR points in the neighborhood. Finally, depth values of the event pixels are estimated based on the depth change in physical structure. We evaluate our approach with real-world experiments. Results show that the density of point clouds increases by 9.6 times. With two open-sourced neural networks, the identification accuracy of 3-D and 2-D bounding boxes with the enhanced point clouds increase on average by 14.6 and 8.8 times, respectively. To sum up, the contributions of this article are as follows.

1) We propose an approach to enhance sparse LiDAR point clouds with event-based cameras for RSU applications in ITS. The approach includes the preprocessing process, neighborhood construction, and depth estimation.
2) In order to determine the physical structure and the distribution model of event pixels, the Voronoi Diagram is used to perform the area division for neighborhood construction for event pixels.
3) We conduct several real-world experiments and verify the availability of enhanced point clouds.

The remainder of this article is organized as follows. Section II reviews the related work. In Section III we present the motivation of our approach. Our approach is introduced in Section IV. Section V presents and analyses the experiment results. Finally, Section VI concludes.

II. Related Work

There have been several research studies on enhancing LiDAR point clouds with conventional sensors. However, there are few works on fusing event-based cameras with LiDAR product. In this section, we first discuss the researches on existing enhancing methods, then the characteristics and the applications of event-based cameras are reviewed.

Fusing LiDAR with conventional sensors has been widely studied. In [15], 2-D images are used to directly produce pseudo-LiDAR point clouds. In [16], a fusion method based on point clouds and monocular images is proposed. To employ the coordinate information of the existing sparse point clouds, the connection between pixels of images and 3-D points from point clouds is established. Then the density of point clouds is improved with the stereo images. In addition to these traditional methods, deep learning, whose development is very rapid in recent years, has also been applied in enhancing algorithms. Other kinds of deep neural networks in [17] and [18] are built to enhance the input sparse point clouds. However, both of the networks interpolate point clouds in the whole 3-D space, which significantly increases the data size.

TABLE I
Related Applications Based on Event-Based Cameras

Work	Sensors	Input	Output	Application
[23]	event-based & frame-based camera	2D events & images	2D boxes	fusion for detection
[25]	event-based camera with 2 channels	2D events & images	2D boxes	fusion for detection
[26]	binocular event-based cameras	2D events	3D events	stereo matching
[27]	binocular event-based cameras	2D events	3D events	stereo matching for reconstruction
[28]	monocular event-based camera	2D events	3D events	multi-view reconstruction
[29]	event-based & RGB-D camera	2D events & images	3D events	fusion for SLAM

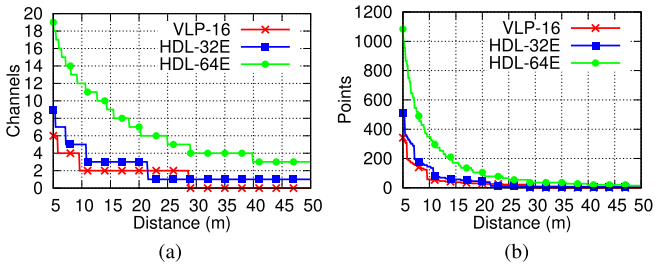


Fig. 1. Points and channels on 1 m^2 surface of object with different distances. (a) Number of channels. (b) Number of points.

Based on the characteristics of event-based cameras, there are several traffic applications with event-based cameras. In [19], a frame-based representation of event flow is combined with frames from conventional cameras to perform optical-flow estimation. Based on the work, a dataset produced by a synchronized stereo event-based camera system and LiDAR product is provided in [20]. The dataset records the data in the perspective of a moving vehicle, therefore, the events consist of both moving targets and the background case by ego-motion. DDD17 [21] and UZH-FPV Drone Racing Dataset [22] are also datasets containing event-based cameras captured on a moving vehicle or UAVs. To the best of our knowledge, there is no dataset containing event-based cameras available for RSU applications in ITS. A joint framework combining event-based and frame-based vision for vehicle detection is proposed in [23]. Spike neural network (SNN) is performed to transform the events into images so that the two kinds of data could be synchronized. Dempster–Shafer theory [24] is used to merge the two kinds of outputs from CNN into a joint decision model. Their fusion approach generates the merged 2-D results without any depth information. A fast pedestrian detection approach mixing the frame and event channels of Dynamic and Active Pixel Sensor (DAVIS) is proposed in [25]. The work performs well on 2-D pedestrian

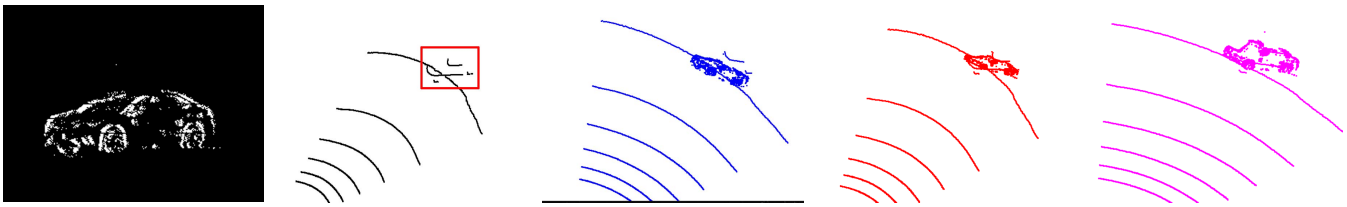


Fig. 2. Point cloud enhancement with different depth estimation methods. (a) Event. (b) LiDAR point clouds. (c) NN method. (d) IDW method. (e) Gaussian method.

detection, but cannot be used for 3-D detection because of the lack of depth.

Considering the 2-D events provided by event-based cameras, it is necessary to estimate the depth of events when fusing the events with 3-D point clouds. Table I compares the related fusion applications and methods based on event-based cameras. Two categories of algorithms on depth estimation with event-based cameras can be distinguished [30]. The first category is based on the binocular model or camera motion information. In [26], the method estimates the depth of events with the binocular model. However, the depth estimation results from event-based cameras are not as accurate as RGB images because fewer features are extracted from events. Based on the previous work, geometric constraints are added into consideration in [27]. Gabor Filter is used to extract edge features of target objects to calculate depth. In [28], a depth estimation algorithm based on the traditional Space-Sweeping algorithm [31] is developed. The algorithm estimates semi-dense 3-D structure from an event-based camera with a known trajectory. Although the algorithm is simple, it requires a known trajectory as an assumption, which is not suitable for real-world scenarios. The second category is to choose other sensors as a depth provider for depth estimation. In [29], a depth image from an RGB-D camera is projected into the event-based camera plane. Then the depth value of the nearest pixel is assigned to the event pixel. When choosing the nearest neighbor (NN) point as a depth provider in Euclidean Space, the offset error leads to the incorrect depth assignment.

In this article, only moving targets in point clouds are enhanced for applications on RSU in ITS. When fusing event-based cameras with LiDAR, 3-D LiDAR points are regarded as depth providers for 2-D events depth estimation. In order to select the suitable LiDAR points for events, we propose an approach to construct the neighborhood of events based on the Voronoi Diagram. The depth value will be calculated according to the spatial information estimated by the neighborhood.

III. Motivation

Sufficient points on moving objects are required in ITS. 3-D LiDAR can achieve high resolution in the horizontal plane. However, the vertical resolution of LiDAR depends on the number of beams [32]. In order to quantize the influence of laser beams on valid points, as well as to compare the point clouds enhancing methods, this section presents a roadside traffic scenario as a motivation example. First, the number of valid points is computed on the 1 m^2 surface of a target object with different distances. Second, enhanced point clouds from RSU are compared.

Fig. 1 shows the number of channels and points on 1 m^2 surface of object with different distances. Caused by the decline of the beams over the target object, it is clear that the number of points and channels obviously decrease when the distance increases. Even for the 64-channel LiDAR, channels and points drop to the same as low-channel LiDAR products starting from 30 m.

The illustrations of the enhanced point clouds are shown in Fig. 2. There is a moving car in the field of view. The first element is the event taken from the event-based camera, as shown in Fig. 2(a). The second element is the origin point clouds from LiDAR, moving object is marked in the red box, as shown in Fig. 2(b). The third to the fifth elements are the enhanced point clouds based on NN [29], inverse distance weighted (IDW) [33] and Gaussian Distribution Model [34] from the same perspective. The enhanced results are shown in Fig. 2(c)–(e).

The details of the car is enhanced by events comparing to origin point clouds. For the result based on NN, as shown in Fig. 2(c), most event pixels are assigned to a similar depth value because of choosing the nearest LiDAR point for depth estimation. The result is unreasonable. Results based on IDW and Gaussian are listed in Fig. 2(d) and (e). The results are better than NN. There are still some event pixels with unsuitable depth values on the car. The two methods collect neighboring LiDAR points and estimate the depth based on fixed linear weighted or Gaussian distribution. Yamaguchi *et al.* [34] is an improved interpolation method for images. It collects the eight-connected neighbors and calculates the weighting parameters according to the Gaussian function. The assumptions of the method are that the positions of pixels are spatially continuous in an image. Unfortunately, however, it is not applicable in our scenario to estimate the depth of events from LiDAR point clouds due to the discontinuity of the data.

In conclusion, low-beams LiDAR cannot be used for RSU applications because of lacking points. For point clouds enhancement based on event-based cameras and LiDAR, a fixed distribution model is insufficient for depth estimation. Therefore, an approach that automatically determines the distribution model based on the spatial information of the neighboring LiDAR points is preferable.

IV. Our Approach

Points on the same surface of a 3-D object have continuity and similarity in spatial coordinates. Therefore, when

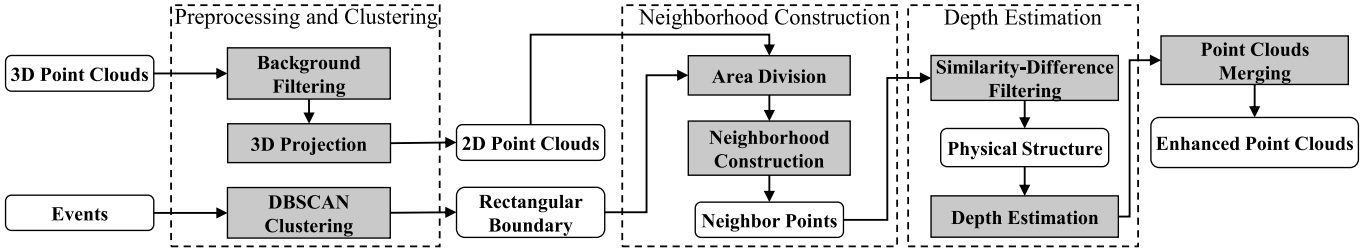


Fig. 3. Overview of our approach.

estimating the depth of an event pixel, we use the depth values of several LiDAR points distributed around the event pixel on 2-D plane as reference. Fig. 3 reveals the framework of the approach. The framework includes three parts: preprocessing and clustering, neighborhood construction based on area division, and depth estimation.

First, LiDAR points exceeding a certain distance threshold are filtered as noise points. 3-D LiDAR points are projected on the image plane and event pixels are clustered with a density-based clustering method DBSCAN [35]. Rectangular areas are constructed based on the clustering result. Second, area division is used for each rectangular area. The neighborhood of each event pixel is constructed through the area division. Third, based on the physical structure extracted from the neighborhood, the algorithm establishes the corresponding distribution model and estimates the depth. Finally, event pixels with estimated depth values will be merged into point clouds. The pseudo code of the preprocessing and clustering, and neighborhood construction are shown in Algorithm. 1.

A. Preprocessing and Clustering

3-D point clouds are defined as: $P_{3\mathrm{D}} = \{p3d_1, p3d_2, \ldots, p3d_n\}$, where n is the number of points. The target of our approach is to enhance objects in the sparse point clouds. In autonomous driving tasks, we focus on the objects in a certain range ahead. Just as the operation in [36], we select the LiDAR points in the range of 0–50 m. Therefore, a depth threshold $d_{\mathrm{threshold}}$ is set to crop the data.

After that, the filtered point clouds P'_{3D} are projected on $P_{2\mathrm{D}} = \{p2d_1, p2d_2, \ldots, p2d_{\acute{n}}\}$ by the intrinsic matrix of event-based cameras and the extrinsic matrix. The intrinsic matrix transforms the event pixels from the camera coordinate into the imaging plane coordinate based on the pin-hole model. The extrinsic matrix contains rotation and translation transformation from LiDAR to event-based cameras under the world coordinate. After the projection, LiDAR points and DVS event pixels are located on the same coordinate.

To establish the connection between event pixels of moving objects and corresponding LiDAR points, we adopt the density-based method to cluster the pixels. DBSCAN defines the cluster as the maximum set of points connected by density which is suitable for clustering event pixels (Line 1). Events E containing pixels together with the hyperparameters MinPts and clustering radius ϵ are the inputs of DBSCAN. As the events are captured by event-based camera, pixels of the moving objects are easily distinguished from the background based on the polarity of each pixel. DBSCAN selects one event pixel as the core point first, then finds the density-connected pixels under the control of MinPts and ϵ. Finally, all the event pixels on the plane are clustered into several sets through constant iterations.

For each cluster of E, the topmost (e_t), bottommost (e_b), leftmost (e_l), and rightmost (e_r) event pixels are selected to construct a rectangle (Lines 3–6) area based on their coordinates. The rectangle area contains all the event pixels from the corresponding cluster. The settings of MinPts and ϵ are chosen through the several empirical results which will be provided in Section V.

Event pixels and projected LiDAR points located in the same rectangular area can establish connection in this way. Since event-based cameras capture the movement of objects, the effective event pixels will be concentrated on moving objects. The rectangular boundary can be regarded as the contour range of the moving object determine by event pixels. The LiDAR points inside the same area belong to the same object and their depth can be helpful when estimating the depth of these event pixels.

B. Neighborhood Construction

After preprocessing for LiDAR points and clustering, the event pixels whose depth values to be estimated and corresponding LiDAR points are unified into the same rectangular area. The next step is to construct the neighborhood based on LiDAR points for each pixel inside the rectangular area.

The horizontal resolution of point clouds is much higher than the vertical resolution. If the neighbor of the pixel is only selected based on the category of Euclidean Distance, when the event pixel is closed to a laser line, the neighborhood is mainly composed of LiDAR points in the same direction as the laser line. It is not conducive to estimate the depth of the pixel between two laser lines. So we propose to construct the neighborhood of the event pixel by preforming area division based on the Voronoi Diagram inside the rectangular area.

The set of LiDAR points and event pixels inside the rectangular area are defined as: $S = \{p2d_1, p2d_2, \ldots, p2d_M\}$, $E = \{e_1, e_2, \ldots, e_N\}$,where M and N are the number of LiDAR points and event pixels inside the area, respectively, (Lines 8–12). For each rectangular area r, the algorithm constructs the Voronoi Diagram based LiDAR points set S for area division. A Voronoi Diagram uses discrete points as the center of the polygonal area to divide the rectangular area, ensuring that the distance between the point in each

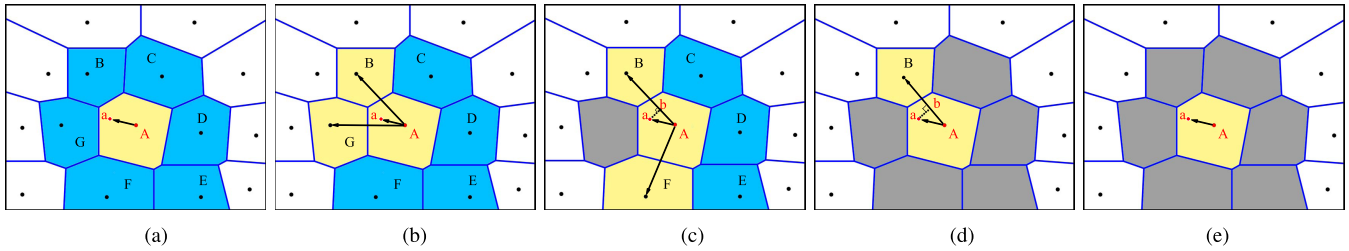


Fig. 4. Depth estimation based on area division by the Voronoi Diagram. Event pixels locate inside yellow areas. Blue areas are unselected neighborhoods. Gray areas are filtered areas because of the extremely different depth values. (a) Area division. (b) Smooth plane. (c) Sharp plane. (d) One point left. (e) Isolated point.

Algorithm 1 Neighborhood Construction of Events Based on Area Division

Input: P_{2D}: 2D point clouds
E: Events containing pixels
ϵ: Clustering radius
$MinPts$: Density threshold
Output: N: Neighborhood of each pixel in E

1: $C\{c_1, c_2, \ldots, c_{num}\} = \text{DBSCAN}(E, \epsilon, MinPts)$
2: Initialize neighbor set $N \leftarrow \{\}$
3: **for** cluster $c \in C$ **do**
4: Get e_t, e_b, e_l, e_r from c
5: $r \leftarrow$Rectangle(e_t, e_b, e_l, e_r)
6: Initialize $S \leftarrow \{\}$
7: **for** point $p2d \in P_{2D}$ **do**
8: **if** $p2d \in r$ **then**
9: Add $p2d$ into S
10: Remove $p2d$ from P_{2D}
11: **end if**
12: **end for**
13: $Area(p2d_i) \leftarrow \{p | d(p, p2d_i) \leq d(p, p2d_j), p2d_i, p2d_j \in S, i \neq j, p \in r\}$
14: **for** event pixel $e \in c$ **do**
15: $N_e \leftarrow \{p2d_j | Area(p2d_k) \cap Area(p2d_j) \neq \emptyset, e \in E, e \in Area(p2d_k), p2d_k, p2d_j \in S\}$;
16: Add N_e into N
17: **end for**
18: **end for**
19: return N

polygonal area and the center of the polygonal area is closer than the distance from the center of other polygonal areas. Each polygonal area corresponds to a LiDAR point and there are no overlaps between polygonal areas. In this way, each pixel will only be located in a polygonal area and the LiDAR point of this area is just the LiDAR point closest to the pixel. We define $p2d_i$ as the polygonal area of LiDAR point, where p is any point within the rectangular area (Line 13). Therefore, a set of 2-D LiDAR points inside the rectangle divide the area into several smaller polygonal areas.

The construction method of the Voronoi Diagram is based on the algorithm proposed in [37]. The construction of the Delaunay Triangle net is one of the steps to construct the Voronoi Diagram. Vertexes of the triangle in the triangle net are the center points of the polygonal area in the Voronoi Diagram. The construction of the Delaunay Triangulation will minimize the angle difference between the smallest angle in the triangle and other inner angles. Therefore, points on adjacent laser lines will be selected instead of all the points on the same laser line to form a triangle net.

The adjacent areas of event pixel e_i include two components. One is the polygonal area Area$(p2d_k)$ where e_i is located. We choose LiDAR point $p2d_k$ as the Seed Point which is the nearest LiDAR point of e_i in the plane according to the geometric properties of the Voronoi Diagram. The other is the polygonal area Area$(p2d_j)$ adjacent to Area$(p2d_k)$. $p2d_k$ and $p2d_j$ are the two vertexes of the same Delaunay triangle and there is a high probability that they are on adjacent laser lines. Meanwhile, there is a public edge between Area$(p2d_j)$ and Area$(p2d_k)$. Therefore, the neighborhood of e_i is composed of the LiDAR points corresponding to the adjacent areas (Lines 14-17). $Area(p2d_k) \cap Area(p2d_j) \neq \emptyset$ means that there is a public edge between the two areas.

The approach ensures that neighbor LiDAR points of events are distributed in several directions, which is helpful for estimating the depth of the event between the two laser lines.

C. Depth Estimation

Fig. 4 describes the depth estimation method. Before estimating the depth, the distribution model of each event pixel is determined based on the neighborhood of event pixel. Just as shown in Fig. 4(a), point A, B, C, D, E, F, and G make up the neighborhood of event pixel a. A is the Seed Point which is the LiDAR point closest to event pixel a and it corresponds to the yellow area in Fig. 4(a). The blue areas in Fig. 4(a) correspond to other points in the neighborhood. In order to filter out the LiDAR points with great difference from the Seed Point in 3-D space, the formula in (1) is defined to describe the similarity of LiDAR points and the Seed Point

$$\text{Diff}_{P_j} = \tanh\big(c1|A.\text{intensity} - P_j.\text{intensity}| + c2|A.\text{depth} - P_j.\text{depth}|\big), \quad P_j \in N_{e_i}. \tag{1}$$

A is the Seed Point, and P_j are the LiDAR points inside the neighborhood. The similarity contains the intensity and depth information of the two points. The weighted similarity is sent into a tanh function to make sure the output ranges from 0 to 1. In the experiments, c_1, c_2 are set to 0.5 and the threshold is set as 0.6 to judge whether the LiDAR point should be filtered.

Then, neighboring points, where the similarity with A is greater than the threshold are filtered out. The filtered points

are usually not located in the same surface with Seed Point A and they correspond to the gray areas in Fig. 4. Besides, Seed Point is considered to be on the same surface with a due to its NN characteristics in the 2-D plane. Therefore, the filtered points are not located in the same surface with a and they should be deleted when estimating the depth of a.

In our approach, target and direction vectors are defined to help determine the distribution model for the surface where the event pixel is located. The target vector is made up of the event pixel and the corresponding Seed Point, just as the $\vec{Aa}$ in Fig. 4(a). And the direction vector is made up of the Seed Point and the remaining neighbor points after filtering, just as $\vec{AB}$ and $\vec{AG}$ in Fig. 4(b). The algorithm seeks to find such two direction vectors before and after filtering, respectively.

1) Include the target vector in the direction.
2) The angle between the two direction vectors is the smallest.

According to whether the filtering causes the difference in two direction vectors, the distribution model is classified into two categories, smooth plane and sharp non-plane area. In Fig. 4(b), the two direction vectors are always $\vec{AB}$ and $\vec{AG}$ regardless of filtering. The surface determined by $\vec{AB}$ and $\vec{AG}$ is gentle in depth and the corresponding distribution model is called the smooth plane. In Fig. 4(c), point G is filtered and the two direction vectors change from $\vec{AB}$ and $\vec{AG}$ to $\vec{AB}$ and $\vec{AF}$. The variance shows that the depth of area, where event pixel a is located, changes dramatically. This distribution model is called the sharp non-plane.

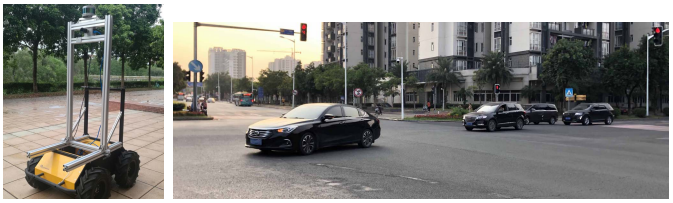
For the two distribution models, the depth of pixel is estimated in different ways. On a smooth plane, the trend of depth change between two points should be consistent with that of the entire plane. In Fig. 4(b), the target vector $\vec{Aa}$ can be represented by $\vec{AB}$ and $\vec{AG}$ in the form of weights. Then, the weight value can be used to estimate the depth change between A and a. The process is presented in (2), where d_a is the depth of a and $d_{\vec{AB}}$ is depth change between LiDAR point A and B

$$\begin{aligned} \vec{Aa} &= \alpha\vec{AB} + \beta\vec{AG} \\ d_a &= d_A + \alpha d_{\vec{AB}} + \beta d_{\vec{AG}}. \end{aligned} \tag{2}$$

On a sharp non-plane area, the depth difference between points is great. It is insufficient to use the two direction vectors to represent the depth change in such area. The direction vector closest to target vector is chosen for depth estimation instead. In Fig. 4(c), the target vector $\vec{Aa}$ is projected to the direction vector $\vec{AB}$ and the point of intersection is b. As shown in (3), the distance between point b and the two endpoints of the direction vector A and B is used for depth estimation

$$d_a = \alpha d_A + \beta d_B, \quad \alpha = \frac{|\vec{Bb}|}{|\vec{Ab}|+|\vec{Bb}|}, \ \beta = \frac{|\vec{Ab}|}{|\vec{Ab}|+|\vec{Bb}|}. \tag{3}$$

There are two special cases in our approach. If they are only one point left except Seed Point after filtering, as shown in Fig. 4(d), $\vec{Aa}$ is projected to the vector $\vec{AB}$, and (3) applies to this case. In Fig. 4(e), all the points except Seed Point are filtered out in the neighborhood. It means Seed Point A is an isolated point and the depth of A can be assigned to a directly.



(a) (b)

Fig. 5. Platform and real city scenarios. (a) Platform. (b) Scenario.

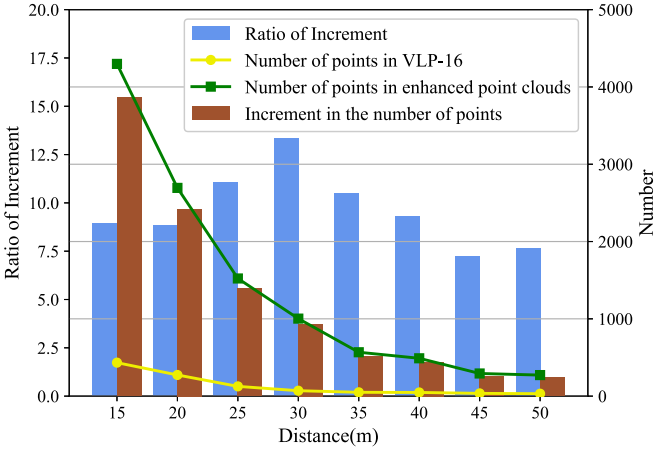


Fig. 6. Improvement of enhanced point clouds projected on the vehicles with different distance.

V. Experiment

We evaluate our approach with real city scenarios data captured by a Velodyne VLP-16 LiDAR[1] and a Celex4 Dynamic Vision Sensor(DVS)[2], which is a kind of event-based camera. As shown in Fig. 5, all the sensors are fixed on the Husky robot platform. VLP-16 can produce sparse point clouds at a frequency of 10 Hz, while Celex4 with a resolution of 640×768 to generate events. The experiments are conducted on a laptop with an Intel Core i7-6600U CPU at 2.60 GHz and a Raspberry Pi III with a Broadcom BCM2837 CPU at 1.40 GHz.

To associate the data from the two sensors, a leaky surface whose mechanism is similar to [38] is used to integrate the LiDAR frames and events. Then a coarse-to-fine registration pipeline is used to get the 6Dof extrinsic matrix. MATLAB Camera Calibration Toolbox [39] is used to get the intrinsic matrix of Celex4.

In order to evaluate our approach, we produce traffic dataset with the platform and compare the improvement of point clouds through different enhancing methods. Besides, to evaluate the availability of our enhancing approach, we use two kinds of open-sourced neural networks to perform the vehicle detection on our dataset. A video with more details about the enhancing result is shown in link (https://www.youtube.com/watch?v=Dag-S34WSLc).

A. Point Cloud Improvement Comparison

We first select several traffic scenarios on the dataset. Just as shown in Fig. 5(b), the scenarios are chosen at a traffic corner

[1]https://www.velodynelidar.com/products/
[2]https://www.celepixel.com/

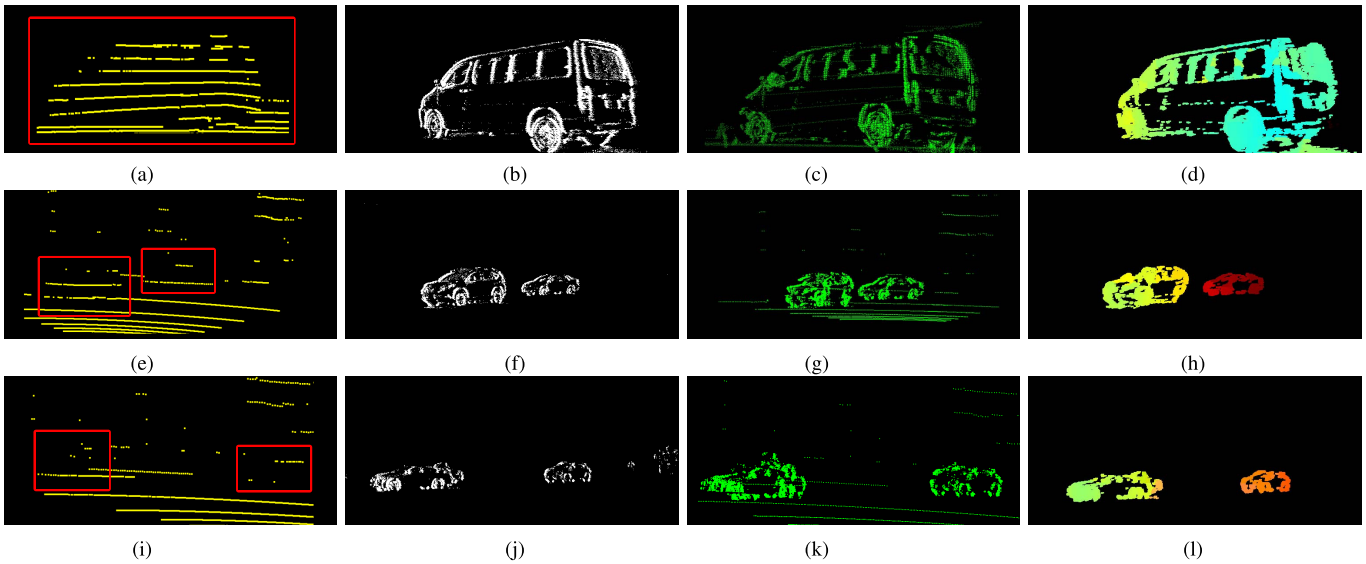


Fig. 7. Visualization of sparse LiDAR point clouds, events, enhanced point clouds, and estimated depth of events with distance 10, 20, and 30 m, respectively. Note that light colors represent close distances, dark colors represent far distances. (a) Original point clouds within 10 m. (b) Events within 10 m. (c) Enhanced point clouds within 10 m. (d) Depth of events within 10 m. (e) Original point clouds within 20 m. (f) Events within 20 m. (g) Enhanced point clouds within 20 m. (h) Depth of events within 20 m. (i) Original point clouds within 30 m. (j) Events within 30 m. (k) Enhanced point clouds within 30 m. (l) Depth of events within 30 m.

where vehicles come from four directions. Different vehicles and bicycles are included in the scenarios. The number of points reflected from objects can be considered as a function of distance. Therefore, we compare the number of incremental points and ratio of increment, respectively, with the distance ranging from 15 to 50 m. Fig. 6 reveals the number of points projected on the vehicles with the size of a cube 4.5 m $\times$ 2.0 m $\times$ 1.9 m on average. Obviously, we can observe the trend of four metrics in Fig. 6. The average number of points by VLP-16 varies from 433 to 32 dramatically. When the measuring distance is over 50 m, 32 points in point clouds are insufficient to perform any identification. As the ROI of event-based cameras reduces with distance, the number of event pixels on vehicles also decreases. And it causes that the number of enhanced point clouds changes from 4298 to 272. However, the increasing ratio of the enhanced point clouds with our method is impressive, with 13.3 times at most, 7.3 times at least, and 9.6 times on average. Even in the distance of 50 m, the increasing ratio is almost 7.6 times. The result demonstrates our approach improves the density of point clouds effectively.

To visualize the enhancement results, we display the origin sparse point clouds, events, enhanced point clouds, and corresponding depth maps. We select three scenarios with different numbers of vehicles and distances ranging from 10 to 30 m for visualization. The enhancements with different distances have been listed in the corresponding rows in Fig. 7. It is clear that the density of point clouds has been improved significantly. With measuring distance increasing, fewer laser beams are reflected from the vehicles, which causes fewer points in the point clouds. The second element is events from event-based cameras. The third element is the enhanced point clouds. The last element of each row is the depth map, and the color changes from cold color to warm color with depth values increasing.

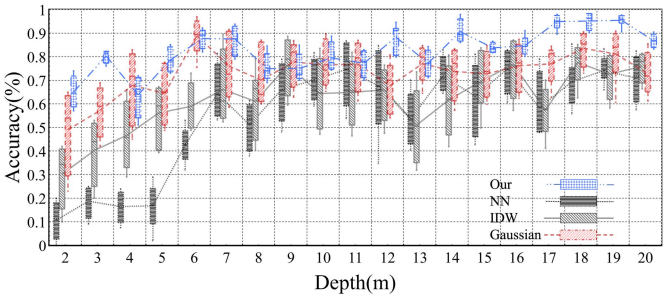


Fig. 8. Mean value and dispersion of estimation accuracy with different depth.

Fig. 7(a), (e), and (i) are origin point clouds, where the vehicles have been marked by red boxes. When the measuring distance is within 10 m, it is hard to identify the shape of vehicle. It becomes more difficult for identification in 20 or 30 m. The second and third column are the events and enhanced point clouds. It is obvious that the enhanced point clouds shown in Fig. 7(c), (g), and (k) reflect the shape of vehicles. These enhanced point clouds make it possible for vehicle detection. Fig. 7(d), (h), and (l) are depth maps, which can also reflect the shapes of vehicles.

To compare with other methods, we randomly select scenarios from our dataset. In each scenario, point clouds are enhanced through different methods including NN, IDW, Gaussian Distribution Model, and our method. The mean estimation depth accuracy and the dispersion of the results from multi scenarios are chosen as the metrics in Fig. 8.

From the perspective of mean value, all the methods do not perform well when the depth is shorter than 5 m. This is because the objects are located in the blind spot of LiDAR. In addition, the performance of depth estimation methods is affected by specific situations. NN performs worse than other

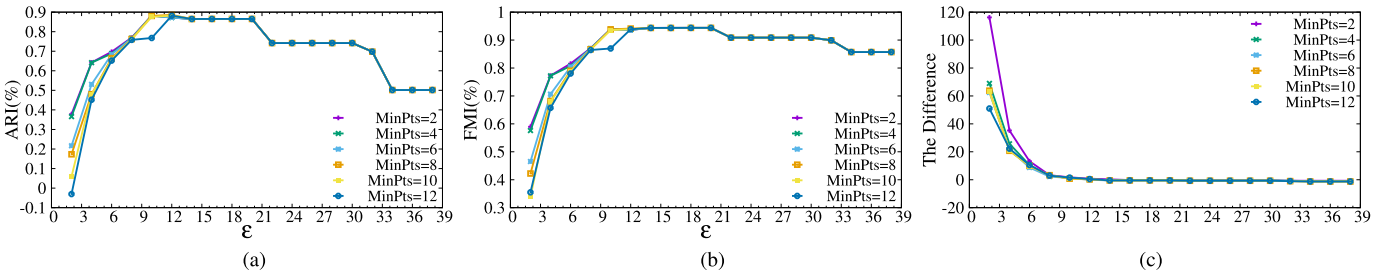


Fig. 9. Clustering results based on DBSCAN with different hyperparameters. (a) ARI results. (b) FMI results. (c) Number difference between clusters and ground truth.

methods in most scenarios. The reason is that this method amplifies the calibration error. The nearest LiDAR point will be used for depth assignment without further processing. The performance of the IDW and the Gaussian Distribution model are also affected by the scenarios. By contrast, our method performs more accurately and robustly than other methods, the average accuracy of our method is over 80%.

From the perspective of the dispersion of the data, the results through our method are better than any other method. For all the depth values, the differences between the maximum and the minimum are the smallest in our results. Furthermore, our data are centrally distributed around the mean value, which indicates that our method achieves the best stability and robustness compared to other methods under all depth conditions.

As a result, considering the combination of mean value and dispersion of the results, our method performs better than any other method, which is suitable for traffic scenarios.

B. Hyperparameter Experiments

The key hyperparameters of the DBSCAN method are MinPts and ϵ. When the MinPts is too small, the actual cluster may be split into multiple clusters. On the other hand, some neighbor clusters will be regarded as the same cluster. Therefore, we need to find the balance when setting the two parameters. The situations are the same for ϵ. In order to determine the settings of the hyperparameters, a series of DBSCAN clustering experiments are performed. In the experiments, 30 frames according to different scenarios are chosen and the labels of pixels from the events are marked as ground truth by manual. Some metrics are used to judge the results. The results are shown in Fig. 9.

Adjusted rand index (ARI) reflects the distribution similarity between the clustered data and the ground truth values. The range of ARI is from -1 to 1. When the value of ARI is closed to 1, it means the clustering result is accurate. As shown in Fig. 9(a), the average ARI changes with different hyperparameters, where the values of ϵ change from 2 to 38, and the values of MinPts change from 2 to 12. The trends of the curves rise rapidly with ϵ from 2 to 6. It means the clustering radius cannot be set too small. Then the peak of the curves remains when ϵ is set from 10 to 20. After that, the curves begin to fall. The most suitable MinPts is 10 according to the experiment. Besides, DBSCAN with different MinPts in our experiments generates the same results when ϵ grows.

Fowlkes–Mallows index (FMI) describes the geometric means of the recall and precision of the clustering results. It ranges from 0 to 1. A higher FMI metric indicates better clustering results. As shown in Fig. 9(b), when ϵ is set from 10 to 20 and MinPts is set 10, the curve achieves a peak value.

Fig. 9(c) describes the difference between the number of clusters and the ground truth values. From the figure, the difference reduces when the ϵ grows. When ϵ is larger than 8, the difference is closed to 0. Based on the results, MinPts and ϵ are set to 10 and 12.

C. Point Cloud Enhancement With Different Methods

In order to compare the enhancing results, three typical traffic scenarios described in Fig. 10 are used to give qualitative analysis of different methods.

Fig. 10(a) demonstrates the first example. There is a car and a truck in the view. As shown in Fig. 10(b), the number of laser beams on the truck is more than that on the car. Enhanced result based on NN is shown in Fig. 10(c). A part of the pixel values of the truck is not estimated correctly because of selecting the unsuitable LiDAR points for depth reference. Results based on IDW and Gaussian are shown in Fig. 10(d) and (e). Most of the event pixels are assigned to the suitable depth values in the enhanced point clouds. IDW is better than Gaussian in this scenario. There is a nonexistent line in the Gaussian result which means some pixels are estimated with the wrong depth. Our method can avoid the single depth reference or fixed distribution model. Point clouds of the car and the truck are enhanced correctly through our method, which can be observed in Fig. 10(f).

Fig. 10(g) shows the second example, there is a bicycle and a car in the field of view. Compared to the first scenario, the car locates behind the bicycle, and part of the car is blocked by the bicycle. From the RSU perspective, the pixels of the two objects are connected. Fig. 10(h) demonstrates the sparse point clouds from the VLP-16 LiDAR product. There are nine laser beams on the bicycle and only two laser beams on the car. Based on the NN method, the estimated depth values of most pixels are the same as those of the bicycle, which is shown in Fig. 10(i). NN directly searches the nearest LiDAR point for depth reference. It is difficult to choose the suitable LiDAR point when there are large unperceived areas on the surface of the object because of the sparse point clouds. For the results

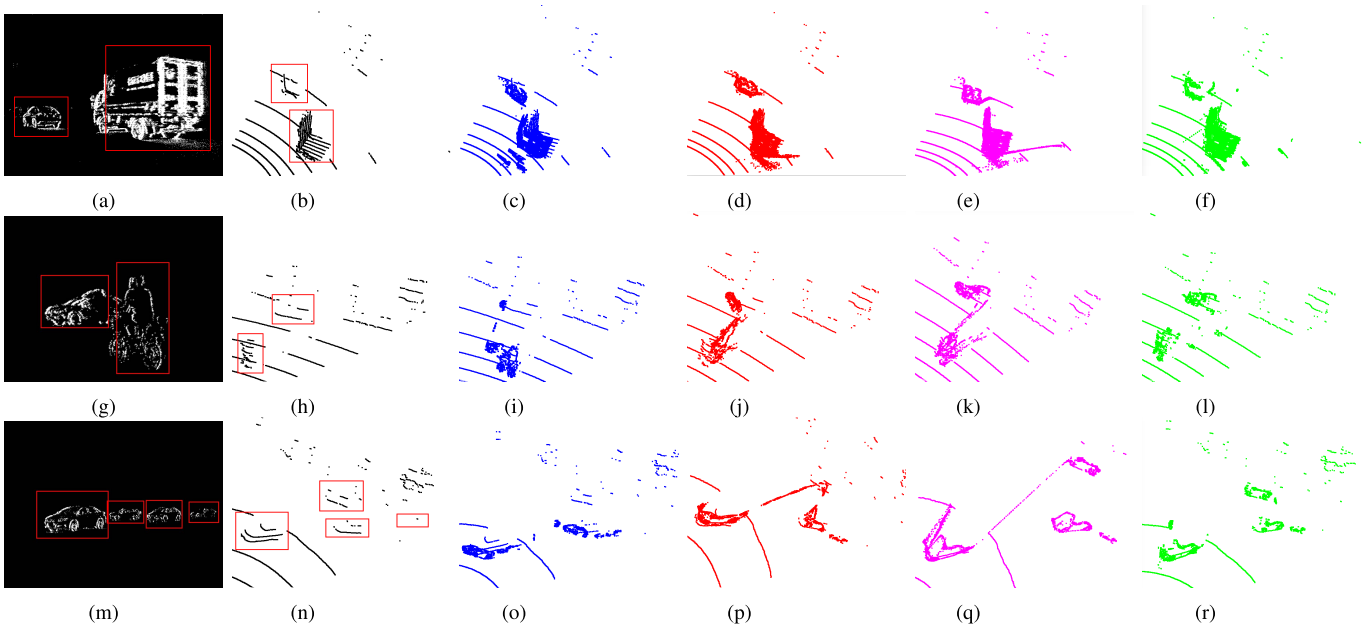


Fig. 10. Point clouds enhancement result with different depth estimation methods in selected scenarios. (a) Event from DVS. (b) LiDAR point clouds. (c) NN method. (d) IDW method. (e) Gaussian method. (f) Our method. (g) Event from DVS. (h) LiDAR point clouds. (i) NN method. (j) IDW method. (k) Gaussian method. (l) Our method. (m) Event from DVS. (n) LiDAR point clouds. (o) NN method. (p) IDW method. (q) Gaussian method. (r) Our method.

based on IDW and Gaussian, as shown in Fig. 10(j) and (k), there are nonexistent lines consisting of the pixels with wrong depth values between the two objects. IDW and Gaussian collect neighboring LiDAR points and estimate the depth based on fixed linear weighted or Gaussian distribution. To meet the requirements of their models, nonexistent lines are produced. Fig. 10(l) demonstrates the results through our method. Pixels belonging to different objects are distinguished. Point clouds of the bicycle and the car are enhanced correctly.

Fig. 10(m) shows the scenario of the third example. There are four cars driving on the road. Laser beams on the cars are more sparse than the second example, as shown in Fig. 10(n). Fig. 10(o) demonstrates that pixels are assigned to a similar depth through the NN method. Due to the difference in resolution between LiDAR and event-based cameras, a single LiDAR point is chosen as the depth provider for a group of pixels. As a result, event pixels on the object with sparse laser beams reflected are assigned similar depth values. Enhanced point clouds through IDW and Gaussian are shown in Fig. 10(p) and (q). Nonexistent lines between disjointed cars still remain in the scenario. Depth values of pixels are estimated according to a fixed distribution which is inconsistent to the real situation. Results through our method is shown in Fig. 10(r). Point clouds of the cars are enhanced correctly.

To make a conclusion, there are three advantages for our approach compared to existing methods. To make a clear discussion, a selected example is listed in Fig. 11. First, the neighbors are chosen based on both the spatial distance and the directions. Neighbors can be chosen from the constructed neighborhood by area division. As shown in the top left in Fig. 11, the nearest LiDAR point A and other LiDAR points in the horizontal and vertical directions are found. Second, a similarity-difference filtering operation is used to

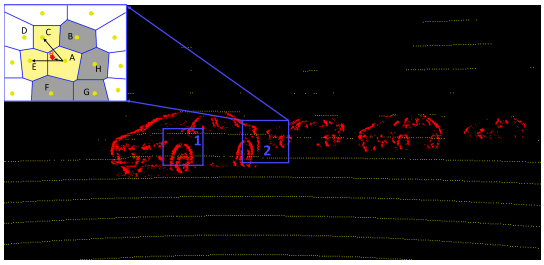


Fig. 11. Neighbor points selection and depth estimation. Yellow points are LiDAR points, red point is event.

filter out the noise points, background points or points belong to other object such as B, H, G, F are filtered out. Third, the physical structure based on the filtered out neighbors is computed to determine the spatial characteristics. For different structure such as sharp edge, smooth plane, line or isolated point, our approach estimate the depth value by the trends in depth changes on the objects, respectively. In the figure, the situation is regarded as the sharp edge. The depth of e is computed based on the depth changes the direction of A, C, E. As a result, the estimated results through our method is more reflective of the depth value.

Table II lists the average computation time of enhancing point cloud frame to compare the computation efficiency of the methods. The experimental results in the table are performed on the aforementioned laptop and Raspberry Pi III.

From Table II, it can be concluded that the average output frequency of our approach is close to 3 Hz on the laptop, 1 Hz on the Raspberry Pi III, respectively. For the clustering step in our approach, the average computation time is 34 ms on a laptop, 382 ms on Raspberry Pi, correspondingly. Compared to dense images, clustering on events requires less computation

TABLE II
COMPUTATION TIME WITH DIFFERENT METHODS

Method	Laptop (Core i7-6600U)	Raspberry Pi III (Broadcom BCM2837)
NN	239.27ms	465.40ms
IDW	276.98ms	757.06ms
Gaussian	287.02ms	857.48ms
Our	331.43ms	953.06ms

for two reasons. First, the events are sparse than images, only non-zero pixels are added for clustering. Second, the events distribution is more concentrated, which makes it easier for clustering algorithm. Although our algorithm requires a little more computation time than related methods on the two platforms, our estimation result performs the best of all the methods. Besides, we intend to run the algorithm on GPU in the future. Based on our knowledge, it is expected that our approach could achieve 10 Hz, which is the same as the output frequency of LiDAR product after acceleration on the GPU device.

D. Vehicle Detection

In order to evaluate our approach more effectively, we perform 3-D object detection tasks on the enhanced point clouds through two open-sourced neural networks SECOND-V1.5 [40] and PointPillars [36].

KITTI is one of the most important open-sourced dataset for autonomous driving tasks such as 3-D vehicle detection [41]. Due to the fact that KITTI does not contain events data, we collect the data and use our dataset for 3-D object detection. Before training, we first generate the enhanced point clouds through different methods and label the vehicles in enhanced point clouds with 3-D bounding boxes. Then our dataset is transformed into KITTI format and the KITTI official metrics are used to compare the performances of different methods. The metrics include 2-D Bounding Box (2D BBox), bird's eye view (BEV), 3D BBox, and average orientation similarity (AOS). IoU is short for intersection over the union. It represents the overlap ratio between the prediction box and the ground truth. According to KITTI, the IoU threshold of vehicles requires a bounding box overlap of 70%. Therefore, considering the fairness of the comparison, IoU is set as 0.7 in our article.

During training, default hyper-parameters of the network that perform best on the KITTI dataset are used to compare the difference between origin point clouds and enhanced point clouds by different methods.

The results of SECOND-V1.5 and PointPillars have been listed in Table III. Detection results through our method are improved dramatically. For SECOND-V1.5, the accuracies of 2D BBox, 3D BBox, BEV, and AOS increase by 6.5 times, 4.8 times, 0.9 times, and 4.1 times, respectively, compared with origin point clouds. For PointPillars, the accuracies are 9.2 times, 22 times, 4.9 times, and 6.1 times. Meanwhile, compared with the other three depth estimation methods, our method performs best of all the metrics.

For the metric of 2D BBox, the accuracy of origin point clouds is 8.03% in SECOND-V1.5 and 5.73% in PointPillars. Accuracies of detecting on all enhanced point clouds increase. This is because projections of the vehicles on the plane are enhanced with the event pixels. More details on the object surface can be captured by neural networks, which benefits the results of 2-D detection. For the NN method, the accuracy is 20.15% in SECOND-V1.5 and 14.26% in PointPillars. There is a certain chance that event pixels in the edge areas are estimated with the correct depth. For IDW and Gaussian method, the accuracy is 12.74%, 18.07% in SECOND-V1.5 and 10.60%, 11.81% in PointPillars. Event pixels on the edge areas are almost miscalculated due to the continuity of the model. This will harm the feature extraction of the neural network. To avoid the problems mentioned above, we only preserve the LiDAR points with similar depth values in the neighborhood of event pixels. The accuracy of our method is 60.21% and 58.17% in SECOND-V1.5 and PointPillars, respectively. To sum up, the 2-D accuracy of our method increases by a factor of 7.5 in SECOND-V1.5 and 10.2 in PointPillars. On one hand, our method avoids the single depth reference in the NN method. On the other hand, unlike continuous models, LiDAR points with large depth differences will not be used to estimate the depth of event pixels. As a result, our method performs best.

For the metric of 3D BBox, results of NN are worse than origin point clouds, which is 0.21% in SECOND-V1.5 and 0.15% in PointPillars. Results of IDW are similar to the original which is 4.55% in SECOND-V1.5 and 1.01% in PointPillars. But both of them perform worse than the Gaussian method. The accuracy of our method is 26.43% in SECOND-V1.5 and 23.60% in PointPillars. To sum up, the 3-D accuracy of our method increases by a factor of 5.8 in SECOND-V1.5 and 23.3 in PointPillars. Our method is better than all of the other methods and original point clouds. It is unavoidable that some event pixels are assigned the wrong depth values. These event pixels will affect the original features of the objects in point clouds when performing vehicle detecting.

For the NN method, it directly searches the nearest LiDAR points on the plane for the depth values of event pixels, which results in many event pixels being located on the same depth-value plane. IDW and Gaussian method collect neighboring points and estimate the depth values based on the linear weight or Gaussian model. Unfortunately, a fixed distribution calculating model is insufficient in complicated scenarios for depth estimation. For the center parts of the objects, the continuous models perform well. However, for the edge parts, depth values of event pixels are miscalculated. This results that the corresponding event pixels are located in the gaps of different objects, which can be regarded as the noise points. Our method estimates the depth value of each pixel based on the distribution model that it belongs to. This avoids many event pixels being assigned the same depth values and processes the event pixels on the edge areas more reasonably. For the metrics of BEV and AOS, the situations and reasons are the same as the 3D BBox.

TABLE III
DETECTION RESULT OF SECOND-V1.5 AND POINTPILLAR

Method / Metric	LiDAR Point Clouds		NN Method		IDW Method		Gaussian Distribution		Our Method	
	SECOND	PointPillar	SECOND	PointPillar	SECOND	PointPillar	SECOND	PointPillar	SECOND	PointPillar
car AP	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7	IoU=0.7
2D BBox AP	8.03	5.73	20.15	14.26	12.74	10.60	18.07	11.81	**60.21**	**58.17**
3D BBox AP	4.55	1.01	0.21	0.15	4.55	1.14	10.72	3.12	**26.43**	**23.60**
BEV AP	17.95	7.27	2.02	1.48	5.88	12.23	25.05	17.75	**34.14**	**43.14**
AOS AP	5.57	3.44	9.08	7.46	5.24	4.78	5.70	5.31	**28.36**	**24.59**

Compared to KITTI dataset, the same neural network based on our enhanced point clouds performs slightly worse on some of the metrics. We attribute it to the fact that our dataset is based on VLP-16 while point clouds of KITTI are produced by HDL-64. However, the 3-D detection results through our approach are greatly improved than origin VLP-16 and other methods. Our approach enhances sparse point clouds impressively and makes them more available in applications.

VI. CONCLUSION

In this article, we develop an approach to enhance sparse LiDAR point clouds with event-based cameras for RSU applications in ITS. To estimate the depth values of event pixels from the cameras, there are three steps in our approach. After the preprocessing for LiDAR points and clustering for event pixels, the neighborhood construction is performed based on area division by the Voronoi Diagram. Then, the distribution model is determined by spatial information of the neighboring LiDAR points in a structural manner. The depth values of event pixels are estimated based on the model. We evaluate our approach in real city traffic scenarios. Results demonstrate that our approach could enhance point clouds effectively. In the future, we intend to accelerate the speed of our approach with GPUs and add the noise reduction module for point clouds and events to make it more suitable for ITS.

REFERENCES

[1] J. A. Sanguesa *et al.*, "Sensing traffic density combining V2V and V2I wireless communications," *Sensors*, vol. 15, no. 12, pp. 31794–31810, 2015.

[2] A. Sarker *et al.*, "A review of sensing and communication, human factors, and controller aspects for information-aware connected and automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 7–29, Jan. 2020.

[3] T. Wen, Z. Xiao, K. Jiang, M. Yang, K. Li, and D. Yang, "High precision target positioning method for RSU in cooperative perception," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–6.

[4] S. Chiodini, R. Giubilato, M. Pertile, and S. Debei, "Retrieving scale on monocular visual odometry using low-resolution range sensors," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 8, pp. 5875–5889, Aug. 2020.

[5] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[6] S. Gargoum and K. El-Basyouny, "Effects of LiDAR point density on extraction of traffic signs: A sensitivity study," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2673, no. 1, pp. 41–51, Jan. 2019, doi: 10.1177/0361198118822295.

[7] S. Xie, D. Yang, K. Jiang, and Y. Zhong, "Pixels and 3-D points alignment method for the fusion of camera and LiDAR data," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 10, pp. 3661–3676, Dec. 2019.

[8] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2181–2188.

[9] C. Posch, "Bio-inspired vision," *J. Instrum.*, vol. 7, no. 1, Jan. 2012, Art. no. C01054.

[10] F. Baghaei Naeini *et al.*, "A novel dynamic-vision-based approach for tactile sensing applications," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 5, pp. 1881–1893, May 2020.

[11] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Attention mechanisms for object recognition with event-based cameras," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1127–1136.

[12] A. Nguyen, T.-T. Do, D. G. Caldwell, and N. G. Tsagarakis, "Real-time 6DOF pose relocalization for event cameras with stacked spatial LSTM networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–8.

[13] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," 2016, *arXiv:1607.03468*. [Online]. Available: http://arxiv.org/abs/1607.03468

[14] M. Erwig, "The graph Voronoi diagram with applications," *Networks*, vol. 36, no. 3, pp. 156–163, 2000.

[15] X. Weng and K. Kitani, "Monocular 3D object detection with pseudo-LiDAR point cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–10.

[16] S. W. Lee, "Enhancing point cloud density with stereo images," M.S. thesis, 2020. [Online]. Available: https://scholarworks.sjsu.edu/etd theses/5102, doi: 10.31979/etd.qwtw-xa9u.

[17] H. Liu, K. Liao, C. Lin, Y. Zhao, and M. Liu, "PLIN: A network for pseudo-LiDAR point cloud interpolation," *Sensors*, vol. 20, no. 6, p. 1573, Mar. 2020.

[18] F. Lu, G. Chen, S. Qu, Z. Li, Y. Liu, and A. Knoll, "PointINet: Point cloud frame interpolation network," 2020, *arXiv:2012.10066*. [Online]. Available: http://arxiv.org/abs/2012.10066

[19] A. Zihao Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," 2018, *arXiv:1802.06898*. [Online]. Available: http://arxiv.org/abs/1802.06898

[20] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2032–2039, Jul. 2018.

[21] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "DDD17: End-to-end DAVIS driving dataset," 2017, *arXiv:1711.01458*. [Online]. Available: http://arxiv.org/abs/1711.01458

[22] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? The UZH-FPV drone racing dataset," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6713–6719.

[23] J. Li, S. Dong, Z. Yu, Y. Tian, and T. Huang, "Event-based vision enhanced: A joint detection framework in autonomous driving," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 1396–1401.

[24] H. Lee, H. Kwon, R. M. Robinson, W. D. Nothwang, and A. M. Marathe, "Dynamic belief fusion for object detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–9.

[25] Z. Jiang *et al.*, "Mixed frame-/event-driven fast pedestrian detection," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8332–8338.

[26] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbrück, "Asynchronous event-based binocular stereo matching," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 347–353, Feb. 2012.

[27] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. B. Benosman, and B. Linares-Barranco, "On the use of orientation filters for 3D reconstruction in event-driven stereo vision," *Frontiers Neurosci.*, vol. 8, p. 48, Mar. 2014.

[28] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "EMVS: Event-based multi-view stereo–3D reconstruction with an event camera in real-time," *Int. J. Comput. Vis.*, vol. 126, pp. 1394–1414, Nov. 2017.

[29] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 359–364.
[30] G. Gallego *et al.*, "Event-based vision: A survey," 2019, *arXiv:1904.08405*. [Online]. Available: http://arxiv.org/abs/1904.08405
[31] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1996, pp. 358–363.
[32] M. Cao, P. Su, H. Chen, S. Tang, and Y. Liu, "3-D dense rangefinder sensor with a low-cost scanning mechanism," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, 2021.
[33] I. Ashraf, S. Hur, and Y. Park, "An investigation of interpolation techniques to generate 2D intensity image from LiDAR data," *IEEE Access*, vol. 5, pp. 8250–8260, 2017.
[34] T. Yamaguchi, M. Ikehara, and Y. Nakajima, "Image interpolation based on weighting function of Gaussian," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 1193–1197.
[35] T. N. Tran, K. Drab, and M. Daszykowski, "Revised DBSCAN algorithm to cluster data with dense adjacent clusters," *Chemometric Intell. Lab. Syst.*, vol. 120, pp. 92–96, Jan. 2013.
[36] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.
[37] L. J. Guibas, D. E. Knuth, and M. Sharir, "Randomized incremental construction of delaunay and Voronoi diagrams," *Algorithmica*, vol. 7, nos. 1–6, pp. 381–413, Jun. 1992.
[38] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3867–3876.
[39] J.-Y. Bouguet. (2008). *Camera Calibration Toolbox for MATLAB*. [Online]. Available: http://www.vision.caltech. edu/bouguetj/calib_doc
[40] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
[41] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

Boyang Li received the B.Sc. and M.Sc. degrees in software engineering from Sun Yat-sen University, Guangzhou, China, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree, majoring in computer science, under the instruction of Kai Huang.

His research interests include cooperative simultaneous localization and mapping (SLAM) and event-based cameras.

Hao Meng received the B.Sc. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2019, where he is currently pursuing the M.Sc. degree, majoring in computer science, under the instruction of Kai Huang.

His research interest includes autonomous driving.

Yuzhang Zhu received the B.S. degree in software engineering from Sun Yat-sen University, Guangzhou, Guangdong, China, in 2019, where he is currently pursuing the master's degree with the School of Data and Computer Science.

His research interest includes autonomous driving technology.

Rihui Song received the B.Sc. and M.Sc. degrees in software engineering from Sun Yat-sen University, Guangzhou, China, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree with the School of Biomedical Engineering.

His research interests include autonomous driving, medical robots, computer vision, and artificial intelligence.

Mingyue Cui received the B.Sc. degree in software engineering from Chongqing Normal University, Chongqing, China, in 2014, and the M.Sc. degree in software engineering from Sun Yat-sen University, Guangzhou, Guangdong, China, in 2017, where he is currently pursuing the Ph.D. degree, majoring in computer science, under the instruction of Kai Huang.

His research interests include autonomous driving and the Internet of Things (IoT). He is recently focused on simultaneous localization and mapping (SLAM) and edge computing in autonomous driving.

Gang Chen received the B.E. degree in biomedical engineering, the B.S. degree in mathematics and applied mathematics, and the M.S. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2008, and 2011, respectively, and the Ph.D. degree in computer science from the Technical University of Munich, Munich, Germany, in 2016.

He is currently an Associate Professor with Sun Yat-sen University, Guangzhou, China. His current research interests include embedded systems, high-performance computing, real-time systems, and robotics.

Kai Huang (Member, IEEE) received the B.Sc. degree from Fudan University, Shanghai, China, in 1999, the M.Sc. degree from Leiden University, Leiden, The Netherlands, in 2005, and the Ph.D. degree from ETH Zurich, Zurich, Switzerland, in 2010.

He joined Sun Yat-sen University as a Professor in 2015. He was appointed as the Director of the School of Data and Computer Science, Institute of Unmanned Systems, in 2016. He was a Senior Researcher with the Computer Science Department, Technical University of Munich, Munich, Germany, from 2012 to 2015, and the Research Group Leader of Fortiss GmbH, Munich, in 2011. His research interests include techniques for the analysis, design, and optimization of embedded/systems/cyber-physical systems (CPS), particularly in the automotive, medical, and robot domains.

Dr. Huang has served as a member of the Technical Committee on cybernetics for Cyber-Physical Systems of the IEEE Systems, Man, and Cybernetics (SMC) Society, Embedded Systems for China Computer Federation, Vehicle Control, and Intelligence for Chinese Association Automation, and Robotic Intelligence for Chinese Association of Automation. He received the Youth Overseas High-Level Talent Introduction Plan 2014 and was granted the Chinese Government Award for Outstanding Self-Financed Students Abroad 2010. He was a recipient of Best (Student) Paper/Candidates Awards IEEE Intelligent Vehicles Symposium (IV) 2018, Human-Friendly Robotics (HFR) 2018, the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2017, the National Conference on Embedded System Technology (ESTC) 2017, Embedded Systems for Real-Time Multimedia (ESTIMedia) 2013, the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS) 2009, ESTIMdedia 2009, and the General Chairs' Recognition Award for Interactive Papers in the IEEE Conference on Decision and Control (CDC) 2009. He also serves as the Deputy Secretary for the Technical Committee on Intelligent Robotics of China Computer Federation.