Dense Depth-Map Estimation Based on Fusion of Event Camera and Sparse LiDAR

Mingyue Cui, Yuzhang Zhu, Yechang Liu, Yunchao Liu, Gang Chen, *Member, IEEE*, and Kai Huang, *Member, IEEE*

***Abstract*— Depth-map estimation reflects the geometry of the visible surface in the environment directly and plays an important role in perception and decision for intelligent robots. However, sparse LiDAR only provides low-resolution depth information, which is a huge challenge for accurate sensing algorithms. To address this problem, this article proposes a novel fusion framework to generate dense depth-map based on event camera and sparse LiDAR. The approach uses the geometric information provided by the point cloud as prior knowledge and clusters point cloud data by an improved density clustering algorithm. Combined with the 3-D surface model of each cluster, the approach can provide 3-D reconstructions of the coordinate points of events and further obtain the dense-depth map by depth expansion and hole filling. Finally, we deploy our approach in MVSEC datasets and real-world applications. Experimental results show that, compared with other approaches, our approach can obtain more accurate depth information.**

***Index Terms*— Data fusion, depth estimation, event camera, intelligent robot, sparse LiDAR.**

I. Introduction

LIDAR has a wider view field (e.g., 360 in horizontal directions) and provides more direct 3-D environment information, which is widely used in intelligent robots [1]. Many applications [2], [3] rely on the perception of surroundings and use depth information to reason and react accordingly. In terms of point density, depth maps can be categorized as sparse depth maps [4] or dense depth maps [5]. Compared with sparse depth maps, dense depth maps have higher resolution and more details of the environment. However, dense depth-map needs abundant point cloud data scanned by high-beam LiDARs that are at high prices. For example, a general 64-line LiDAR of Velodyne costs $80 000 [6].

Manuscript received August 10, 2021; revised December 17, 2021; accepted December 24, 2021. Date of publication January 19, 2022; date of current version February 24, 2022. This work was supported in part by Shenzhen Basic Research Grants under Grant JCYJ20180507182508857 and Grant JCYJ20180508152434975, in part by the Science and Technology Planning Project of Guangzhou, China, under Grant 202007050004, and in part by the National Key Research and Development Program of China under Grant 2018YFB1802405. The Associate Editor coordinating the review process was Donghoon Kang. *(Corresponding author: Kai Huang.)*

Mingyue Cui, Yuzhang Zhu, Yechang Liu, Yunchao Liu, and Gang Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China, and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou 510275, China (e-mail: cuimy@mail2.sysu.edu.cn; zhuyzh9@mail2.sysu.edu.cn; liuych66@mail2.sysu.edu.cn; liuych65@mail2.sysu.edu.cn; cheng83@mail.sysu.edu.cn).

Kai Huang is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China, and also with the Shenzhen Institute, Sun Yat-sen University, Guangzhou 510275, China (e-mail: huangk36@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TIM.2022.3144229

The sparse LiDAR limits the performance and operation range of many depth-dependent sensing algorithms, especially those with depth as input. There are many studies on the fusion of LiDARs and vision cameras [7]–[9]. Caltagirone *et al.* [10] propose a context-based multisensor system, which is applied to pedestrian detection in urban environments. Their approach uses fully convolutional neural networks to detect roads by fusing LiDAR point clouds and camera images. Shin *et al.* [11] describe a framework for direct visual simultaneous localization and mapping (SLAM) combining a monocular camera with sparse depth information from LiDARs. These studies above are all based on LiDARs and traditional cameras. However, traditional cameras have some limitations, such as motion blur and low dynamic range, which are not suitable for high-speed dynamic scenes.

The emerging event camera, which outputs the light intensity changes of the surrounding environment, can be much more promising. Compared with traditional vision camera, event camera has a higher dynamic measurement range (140 dB) and time resolution (in the order of *us*) [12]. Event pixels from this camera perceive the relative changes in light intensity and report an event if a threshold is reached individually. The characteristics of event cameras make it easier to cope with challenging scenarios especially in high-speed and high dynamic range scenes [13], [14], such as feature extraction and tracking, and optical flow estimation. Fig. 1 shows the progress of the fusion of events and sparse point clouds.

Fusion of event camera data and sparse LiDAR data is, however, a challenging task. First, the dimension and structure of output data from LiDAR and event camera are different. Event camera outputs a continuous stream of events but LiDAR scans output 3-D point cloud data at a fixed frequency. It is difficult to unify two kinds of data under the same coordinate system. Second, there is no direct relationship between LiDAR points and event pixels on the same object. The distribution of the point cloud is random and scattered. The event stream only responds to the gradient of the ambient light change, which makes it difficult to completely cover the entire object. Last but not least, compared with the static view [15], the event stream of dynamic view in the intelligent robots has more noisy information and extra environmental information. Therefore, a sophisticated approach is needed to make the fusion of low-beam LiDAR and event camera to obtain dense depth maps effectively.

In this article, we introduce a novel framework of data fusion based on event camera and LiDAR to obtain dense depth

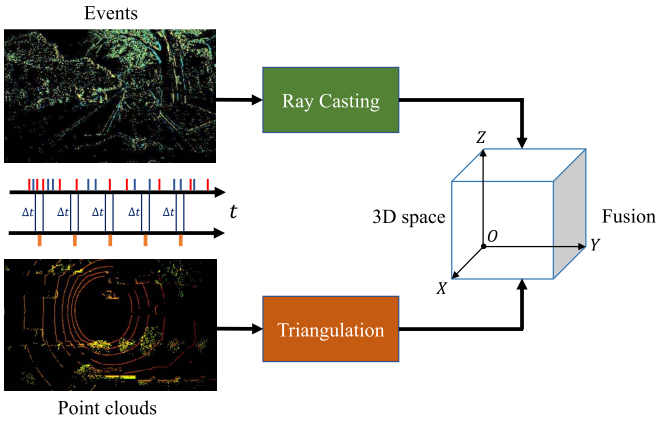


Fig. 1. Fusion of events and sparse point clouds. First, the event stream is turned into frames according to the frequency of LiDAR data. For each frame, the events are backprojected into 3-D space by ray casting, and triangulation is used for surface models' construction of point clouds. Then, the two types of data can be fused in the 3-D space.

information. The key point of data fusion is to estimate the depth of 2-D events in 3-D space combined with 3-D point cloud data. Our approach mainly includes the following steps: preprocessing module, ray casting of events, surface models' construction, depth estimation, and dense depth completion. First, in the preprocessing module, we extract the point cloud information in a specific range and convert the event stream into image frame sequences by time truncating. Besides, clustering with a designed weighted Euclidean distance and triangulation is introduced to achieve surface models' construction of point cloud. Then, based on ray casting of events, we use intersection detection with constraints to reconstruct points and estimate depth. Finally, depth expansion and holes' filling operations in morphology are applied to fill the area without references and further obtain dense depth information. We evaluate our approach on the MVSEC dataset [16] and two real-world applications. Compared with other approaches, our approach can achieve more accurate results. In summary, our contributions are given as follows.

1) We propose a dense depth-map estimation approach based on event camera and sparse LiDAR, which can achieve depth estimation efficiently under dynamic outdoor scenes.
2) We evaluate our approach on different metrics and real-world applications. In terms of depth estimation error, our approach is better than other works at different cutoff depth distances in most of the scenarios. In vehicle detection, compared with the raw point cloud, the accuracy of ours is improved by 10.59% on average. At the same time, the root mean square error (RMSE) of the absolute pose error (APE) is reduced by 22.19% in LiDAR odometry.

II. Related Work

A. LiDAR and Event Camera

With the development of autonomous vehicles, LiDAR has been attracting great attention from industry and academia, especially for the motorized optomechanical scanning LiDAR. LiDAR is a remote sensing method used to measure the distance of objects. It is usually composed of multiple channels of transmitters and receivers stacked vertically, a rotating motor, information processing systems, and so on. LiDAR can fire hundreds of thousands of pulses per second in the field of view (FoV). These light waves bounce off objects and return to the LiDAR sensor. By using the returning time for each pulse, the distance of objects can be calculated. Each of these pulsed laser measurements is then processed into a point cloud according to the time-of-flight (ToF). LiDAR works at a fixed frequency and generates a set of point clouds $P = \{(x_i, y_i, z_i)|i = 1, 2, \ldots, n\}$ at every frame, where x_i, y_i, and z_i indicate the x, y, and z coordinates of the point cloud with index i.

Event cameras are bioinspired vision sensors that capture the brightness changes in the environment. Unlike traditional cameras acquiring full images at a specific rate, event cameras are asynchronous sensors, which means that they sample light based on scene dynamics. Event cameras asynchronously and independently respond to the brightness changes for every pixel all the time. When a pixel sends an event, it memorizes the log intensity and continuously monitors for a change of sufficient magnitude from this memorized value. Once the change exceeds a threshold, the event camera sends an event, which is transmitted from the chip with the u, v location, the time t, and the polarity p of the change. Therefore, different from image frames generated by traditional cameras, the output of event cameras is a continuous event stream $E = \{(u, v, t, p)\}$, where u, v, t, and p indicate the pixel position, timestamp, and polarity, respectively.

B. Previous Research

In recent years, there are some studies [17]–[19] on depth estimation based on LiDAR, event cameras, or fusion with other sensors. We briefly introduce studies on these topics in this section.

1) Sparse LiDAR : The potential of deep learning for depth estimation on sparse point clouds is widely explored. Uhrig *et al.* [20] propose a convolutional neural network with simple yet effective sparse convolution layers operating on sparse point clouds. The sparse convolution layers can consider the location of missing data during the convolution operation, which makes results better than the baseline. Huang *et al.* [21] introduce multiscale features to make depth estimation and propose three novel sparsity-invariant operations. Based on these, a sparsity-invariant multiscale encoder–decoder network (HMS-Net) is designed for handling sparse inputs and sparse feature maps. Chodosh *et al.* [22] propose a novel deep recurrent autoencoder based on compressed sensing and alternating directional neural networks (ADNNs) for depth completion. Although the network is designed to have fewer layers and parameters, the performance is not degraded. The approach of deep learning requires huge computing power. To solve this problem, Ku *et al.* [23] rely only on basic image processing operations to perform depth completion of sparse LiDAR depth data. Their ip_basic algorithm ranks first on the KITTI test server among all published approaches. However, those approaches do not fundamentally solve the problem of information loss caused by low-density LiDAR.

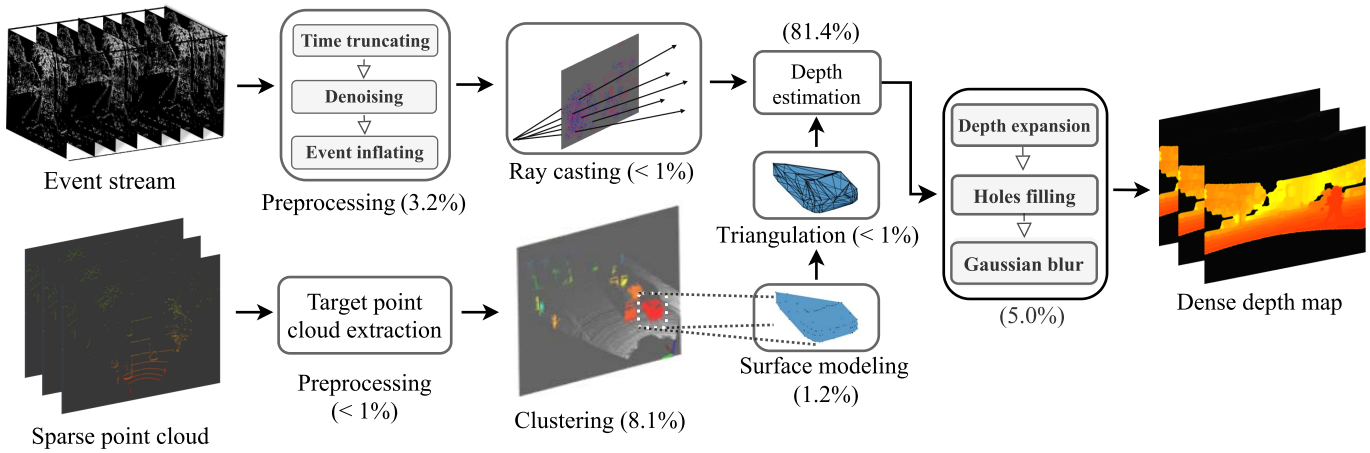


Fig. 2. Overview of our approach, which includes the average runtime ratio of each module.

2) Event Camera: Event camera is a new type of sensor, which is inspired by the working mechanism of the human retina. Due to the limited information provided, there are few studies directly applied to depth estimation. Kim *et al.* [24] point out that depth estimation from moving event cameras should be possible. They perform real-time 3-D reconstruction from a single hand-held event camera with no additional sensing, which enables it to work in unstructured scenes and without prior knowledge. Schraml *et al.* [25] convert the stream of event data into image frames at constant time intervals and then use stereo matching methods of traditional cameras for depth estimation. Rebecq *et al.* [26] propose an event-based multiview stereo (EMVS) method, which is able to produce accurate, semi-dense depth maps, without requiring any explicit data association or intensity estimation. Gallego *et al.* [27], [28] further propose multiple focus loss functions for event alignment and introduce a simple way to process events by maximizing event alignment along point trajectories on the image plane. Zhu *et al.* [29] propose a novel framework for unsupervised learning for event cameras that learns motion information. Their method shows superior performance in depth estimation, optical flow, and so on. Hidalgo-Carrió *et al.* [30] use a recurrent architecture to learn monocular dense depth from events. Quantitative experiments show up to 50% improvement in average depth error with respect to previous event-based methods. Recently, Gehrig *et al.* [31] open source a stereo camera dataset named DSEC, which contains data from event camera, color camera, LiDAR, and so on.

3) Multisensor Fusion: Maddern and Newman [32] propose a probabilistic method for fusing sparse 3-D LiDAR data with stereo images. The method takes advantage of the complementary error characteristics of LiDAR range sensing and dense stereo to obtain accurate dense depth maps. Weikersdorfer *et al.* [33] present a novel 3-D simultaneous localization and mapping algorithm based on the fusion of RGB-D camera and event camera. They use the nearest method to find the depth value for each event in the dense depth map of the RGB-D camera. Brandli *et al.* [34] propose the combination of a bioinspired, redundancy-suppressing event camera with a pulsed line laser to allow fast terrain reconstruction. With the proposed algorithm, temporal correlations between the pulsed stimulus and the recorded signal can be extracted, which is used as a filtering criterion for the stripe extraction. Van Gansbeke *et al.* [8] introduce a novel algorithm to accurately complete sparse LiDAR maps guided by RGB images. Confidence masks are exploited in order to take into account the uncertainty in the depth predictions from each modality. Recently, Gehrig *et al.* [35] propose a novel network architecture, which generalizes traditional RNNs to handle asynchronous and irregular data from multiple sensors.

TABLE I
DIFFERENCES WITH THE WORK OF [15]

Approach	Sensors	Platform	Ways	Evaluation	Object
[15]	LiDAR & Event camera	Stationary	2D	Self-collected	Enhancing
Ours	LiDAR & Event camera	Moving	3D	MVSEC	Dense depth map

The combination of event camera and LiDAR is a promising opportunity for a new type of visual processing. However, up until today, there have not been many thorough prior studies on dense depth-map estimation based on the fusion of event camera and sparse LiDAR, especially evaluated with real-life applications, which prompted this work. A recent study [15] proposes a fusion approach to enhancing LiDAR point clouds with the event-based camera. However, their approach is oriented to the stationary sensor platform rather than the moving platform. Besides, the goal of their study is to enhance point clouds of moving objects, not to obtain dense depth maps. Finally, they need to convert point cloud data to 2-D space for neighborhood construction, but we do surface modeling directly in 3-D space. The specific differences can be seen in Table I.

III. APPROACH

As shown in Fig. 2, we introduce the framework of our approach. First, during preprocessing, the event stream data are converted into image frame sequences and synchronized according to the frequency of LiDAR. Events are further denoised and inflated on the 2-D pixel plane to reduce noise and expand the overall pixel values. Besides, point cloud in

the overlap of the event camera and LiDAR field of view is extracted. Second, we reconstruct the point cloud data and events, respectively. Clustering and point cloud triangulation are used to construct the surface models that contain geometric information of objects in the environment. In order to achieve better point cloud clustering effects, we design a weighted Euclidean distance metric for clustering. For event stream information, we use backprojection to make a spatial sweep. Then, by intersection detection with rays and surface models, 3-D points corresponding to events are reconstructed. Finally, we use depth expansion and holes' filling algorithms to obtain dense depth maps.

A. Preprocessing

Event camera outputs a continuous stream of events, but LiDAR scans outputs point cloud data at a fixed frequency, which requires preprocessing module to make these two different types of data synchronization. The event stream is defined as a tuple (u, v, t, p) to represent pixel position, timestamp, and polarity, respectively. In the preprocessing, events are synchronized with LiDAR through timestamps, and a sliding time window is applied to turn the event stream into frames. Specifically, it is assumed that the frequency of LiDAR is f, the time stamp of the current frame is T, and the sliding time window is set as Δt. The event E_T at the corresponding time is projected onto a 2-D plane, which can be described as

$$E_T = \left\{(u, v, t, p) | T - \Delta t < t < T, 0 < \Delta t < \frac{1}{f}\right\}. \quad (1)$$

Considering that photosensitive components of the event camera at Δt may output events of different polarity, we choose the event whose response time is closest to T as the output.

To further reduce noise caused by light intensity sensitivity, we use the arithmetic mean filter algorithm [36] to preserve the polarity of events. Threshold e_{th} is set for the arithmetic mean filter

$$f(u, v) = \begin{cases} 1, & \frac{1}{M \times N} \sum_{(s,t) \in S_{uv}} |g(s, t)| >= e_{\text{th}} \\ 0, & \frac{1}{M \times N} \sum_{(s,t) \in S_{uv}} |g(s, t)| < e_{\text{th}} \end{cases} \quad (2)$$

where $S_{u,v}$ is an $M \times N$ filling window whose coordinate of the center point is (u, v). $g(s, t)$ is the event polarity in $S_{u,v}$. In addition, we use the dilation operation [37] in computer vision to inflate events to enhance the event stream information, which can be described as

$$E_d = E_T \oplus S = \{(u, v, t, 1) | (S)_{uv} \cap E_T \neq \varnothing\} \quad (3)$$

where E_d is the inflating event and S is the convolution kernel. The denoising and inflating event results can be shown in Fig. 3.

After synchronization, we extract the point clouds in the overlap of the event camera and LiDAR field of view. For most autonomous driving tasks, the objects in a certain range ahead are concerned. Just as the operation in [38], in this article, we extract the point cloud with the distance in 0–50 m and height in 0–3 m except for the ground. In addition, for the point cloud, noise and outliers data caused by occlusion are filtered in the preprocessing module.

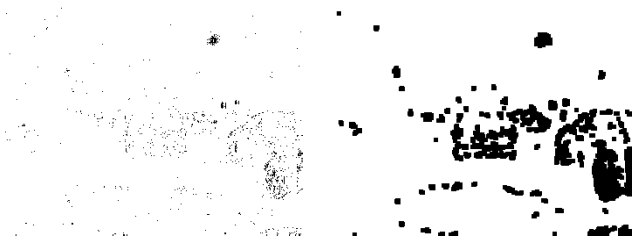


Fig. 3. Comparison before and after event preprocessing. (a) Raw event. (b) Denoising and inflating results.

B. Ray Casting of Events

The event stream obtained by denoising and inflating belongs to 2-D space, which cannot be directly fused with point cloud data. Considering the projection of point clouds into 2-D space, the geometric characteristics of space will be lost. It is necessary to backproject the events into 3-D space and calculate the orientation vector of each event. The event camera can be represented in the LiDAR coordinate system using both the projection matrix and the intrinsic matrix. Then, we backproject the events stream according to the principle of ray casting and pin-hole model of cameras. The backprojected events are defined as

$$\mathbf{r} = C + \mathbf{D}l \quad (4)$$

in which C is the optical center of the camera in the LiDAR coordinate system, $\mathbf{D}$ represents the normalization vector starting from C, and l is the length of the ray to be determined. By ray casting, the event stream is described as the quadruples and their orientation vector.

C. Surface Models' Construction of Point Clouds

The point cloud is a random discrete sparse point set, in which points do not match events one by one. Thus, in this part, we need to construct the surface models of objects with the point cloud. Based on the 3-D plane representation of target objects, we can unify point clouds and event streams under the same coordinate system by orientation vector intersection detection.

To obtain point sets belonging to different objects basically, clustering is applied to divide the point cloud. We do not require precise division of the point cloud, but it is necessary to ensure that points belonging to the same object are in the same set as much as possible. However, points are usually dense in the horizontal direction but sparse in the vertical direction. The traditional Euclidean distance metric in clustering often divides the points close to the object or the points on the upper and lower parts of the object wrongly. Thus, we introduce different weights information on axes in the coordinate system and customize the weighted Euclidean distance

$$\begin{aligned} D(a, b) &= \sqrt{w_1(a_x - b_x)^2 + w_2(a_y - b_y)^2 + w_3(a_z - b_z)^2} \\ &= \sqrt{\sum_{i=1}^{3} w_i(a_i - b_i)^2}, w_1 = w_2 > w_3 \end{aligned} \quad (5)$$

where a and b are two different coordinate points, and w_i is the weight of different coordinate axes in the 3-D Cartesian

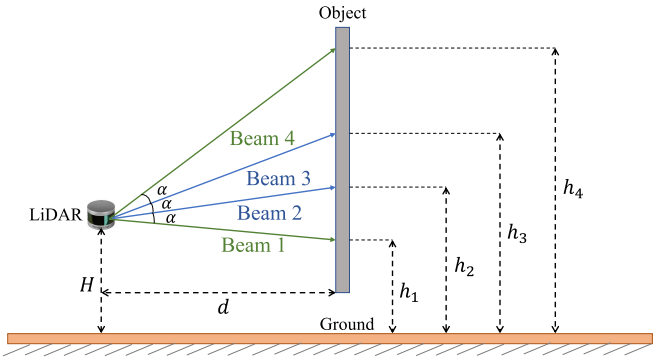


Fig. 4. Schematic of laser beam projection, in which "Beam 2" and "Beam 3" are the laser real beams and "Beam 1" and "Beam 4" are virtual beams.

coordinate system. Empirically, we set w_1, w_2 to 2, and w_3 to 1.

After clustering, the point sets are considered as surfaces discrete sampling of different objects. The convex hulls of each cluster are rough representations of the surface of each object. However, in actual situations, most objects reflect only a few or even one laser beam. Such a cluster is hard to represent the surface of an object. To make the constructed surface models close to actual surfaces, some virtual points that assist in surface models' construction are added to clusters. The values of virtual points on x- and y-axes are the same as that of points in clusters, while the values on the z-axis need to be recalculated.

As shown in Fig. 4, we show the schematic of laser beam projection, where the height of LiDAR is H and the angular resolution is α. The depth of points in the cluster is denoted as d. "Beam 2" and "Beam 3" are the lowest and highest beams in the same cluster, which are all the laser real beams. Their heights are denoted as h_2 and h_3, respectively. For each cluster, we add two corresponding virtual beams "Beam 1" and "Beam 4." Then, the height h_1 of the laser "Beam 1" can be described by $g(\alpha, d)$

$$g(\alpha,d)=\begin{cases} H-d*\tan\left(\alpha-\arctan\left(\dfrac{h_2-H}{d}\right)\right), & h_2 \geqslant H \\ H-d*\tan\left(\alpha+\arctan\left(\dfrac{H-h_2}{d}\right)\right), & h_2 < H. \end{cases} \tag{6}$$

Similarly, the height h_4 of the laser "Beam 4" is described by $f(\alpha, d)$

$$f(\alpha,d)=\begin{cases} d*\tan\left(\alpha+\arctan\left(\dfrac{h_3-H}{d}\right)\right)+H, & h_3 \geqslant H \\ d*\tan\left(\alpha-\arctan\left(\dfrac{H-h_3}{d}\right)\right)+H, & h_3 < H. \end{cases} \tag{7}$$

Here, to make equations look simple, we suppose that h_1 is less than H when h_2 is larger than H and h_4 is larger than H when h_3 is less than H.

However, considering that the angular resolution is α, the height of virtual points added in the cluster should be larger than h_1 and less than h_4. Otherwise, the height of virtual points will overlap on the z-axis, which causes distant events to be incorrectly estimated to nearby objects. At the same time, the distant virtual points will be very high or very low. Thus, we add more constraints as follows:

$$h_1 = \max\left(g\left(\frac{\alpha}{2}, d\right), \quad z_i - 0.5, 0\right) \tag{8}$$

$$h_4 = \min\left(f\left(\frac{\alpha}{2}, d\right), \quad z_i + 0.5\right). \tag{9}$$

We use half of the LiDAR angular resolution to calculate the height and consider the height of objects in the real road environment.

The convex hulls consisting of parts of points and virtual points can help to determine the approximate location of events in space. However, it is still unable to accurately obtain the depth information of the event. In order to make full use of the known depth information from the point cloud, we need to divide the surface of the target object into as small patches as possible. In this article, we use the Delaunay triangulation algorithm [39] to divide the convex hulls into triangles that do not overlap each other and maximize the minimum internal angle.

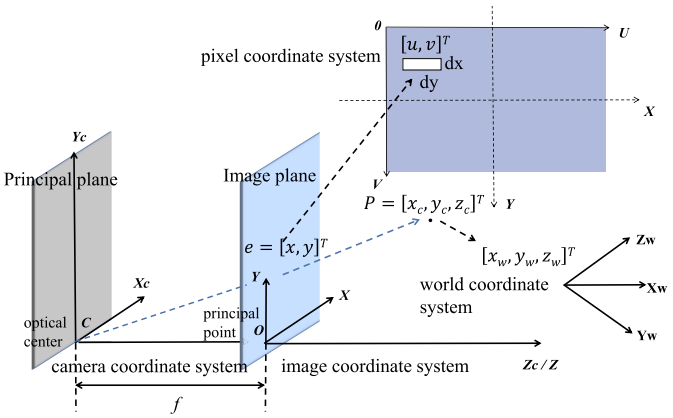


Fig. 5. Transformation relationship of different coordinate systems.

D. Depth estimation

Before depth estimation, we use the pinhole camera model [40] to put the event and the point cloud data in the same coordinate system. The transformation relationship between different coordinate systems is shown in Fig. 5. During the above processing, we obtain rays of events and surface models of point clouds in the form of triangular patches. Then, we determine whether a ray in space intersects a triangle and calculate the intersection point. The point P in the triangle patch V with vertices (V_0, V_1, V_2) can be represented by the barycentric coordinate system as follows:

$$\begin{aligned} P &= a(V_1 - V_0) + b(V_2 - V_0) + V_0 \\ &= (1-a-b)V_0 + aV_1 + bV_2, a > 0, b > 0, a + b < 1. \end{aligned} \tag{10}$$

Combine (4) and solve the simultaneous equation as follows:

$$\begin{bmatrix} l \\ a \\ b \end{bmatrix} = \frac{1}{(\mathbf{D} \times \mathbf{E_2}) \cdot \mathbf{E_1}} \begin{bmatrix} (\mathbf{E_3} \times \mathbf{E_1}) \cdot \mathbf{E_2} \\ (\mathbf{D} \times \mathbf{E_2}) \cdot \mathbf{E_3} \\ (\mathbf{E_3} \times \mathbf{E_1}) \cdot \mathbf{D} \end{bmatrix} \tag{11}$$

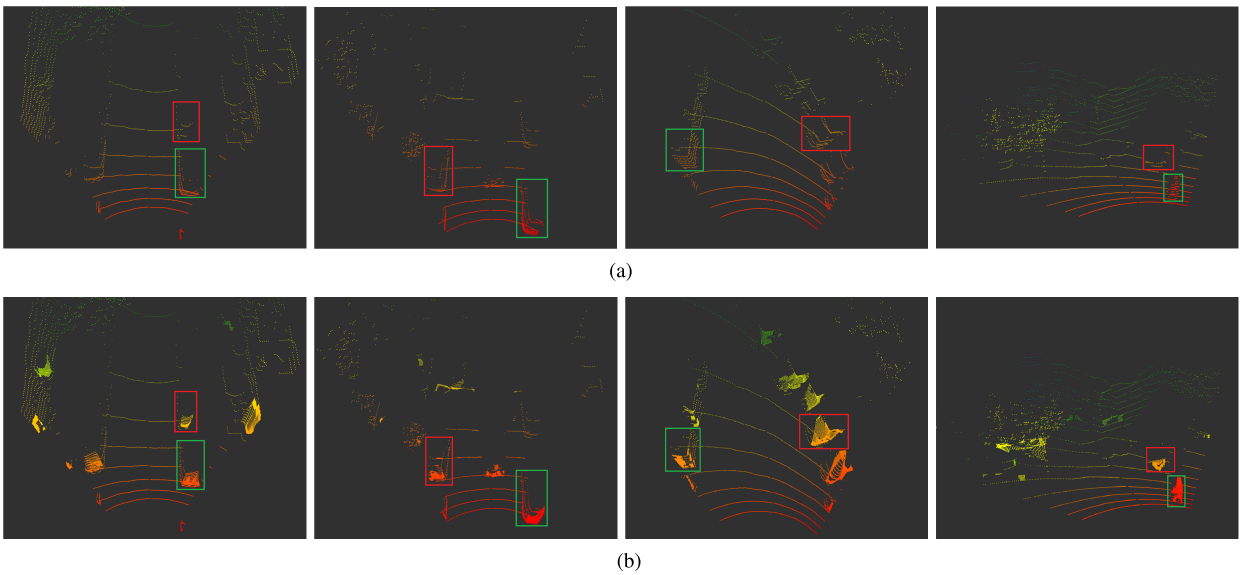


Fig. 6. Comparison of quality performance before and after fusion. (a) Raw sparse point cloud. (b) Fusion results.

in which $\mathbf{E_1} = V_1 - V_0, \mathbf{E_2} = V_2 - V_0, \mathbf{E_3} = C - V_0$. By substituting the solved values for l, a, and b in (4), we can calculate the intersection point and depth information.

In 3-D space, there may be multiple triangle patches intersecting a ray. Considering the problem of occlusion of light, we constrain the orientation vector of the event and select the surface models closest to the sensor that intersects the ray. Finally, we use (4) and (11) to reconstruct the point $P_r(x_r, y_r, z_r)$. The specific constraints are given as follows:

$$P_r(x_r, y_r, z_r) = C + \mathbf{D}l \begin{cases} \arg\min_l C + \mathbf{D}l \\ \arg\min_y A_q(y) < y_r < \arg\max_y A_q(y) \\ \arg\min_z A_q(z) < z_r < \arg\max_z A_q(z) \end{cases} \quad (12)$$

in which A_q is the obtained cluster intersecting the ray.

E. Dense Depth Map

After points' reconstruction, the effective geometric information of the event is integrated into the sparse point cloud. The local area of the point cloud is dense, but there are many gaps in the space between the points. Thus, we use depth expansion and hole filling algorithms for depth completion to obtain the dense depth map.

1) Depth Expansion: There are still many pixels without depth information between different laser beams. These blank pixels are most likely to have the same depth as the near point cloud, so the existing depth information is used to expand the depth of these blank pixels. We design a custom kernel to estimate depth. The size of the kernel is set to be 5×5, which can prevent the kernel from being too large to estimate nonobject pixels or too small to miss valid pixels. The kernel shape is designed such that the most likely pixels with the same values are dilated to the same value.

2) Hole Filling: Depth expansion does not perform depth filling on a large scale, so there are still a few blank pixels between the point clouds. For small holes in the depth map, depth estimation can be performed by connecting the edge depth of objects near pixels without depth. Then, we use the closing operation in morphology to close the small holes. For large holes in the depth map, since they are far away from the point cloud with accurate depth, we use a 7×7 full kernel to fill in any remaining empty pixels while leaving valid pixels unchanged.

The depth expansion and holes' filling operations only perform on pixels with depth information, which avoids depth errors increasing. After that, we use the Gaussian blur to smooth the local plane, especially the sharp object edge. Nevertheless, there still exists a large blank space in the depth map. These areas are often beyond the range of LiDAR, such as unobstructed foreground and sky. In this article, we ignore these areas to avoid erroneous depth estimation. Fig. 6 shows the comparison of quality performance for point cloud before and after fusion. We can see that, compared with the raw point cloud data, the fused results have higher density and more details. In the next section, we further evaluate the performance of depth information.

IV. Experiment

In this section, we verify our approach with the offline evaluation. Compared with other approaches, the experimental results show the effectiveness of our approach.

A. Experiment Settings

The proposed framework is extensively tested on the public MVSEC dataset [16] to prove its effectiveness. The MVSEC is composed of various data collected by different sensors, such as LiDAR, event camera, IMU, and GPS. The model of LiDAR is Velodyne VLP 16 PUCK LITE and the model of event camera is DAVIS 346B. The MVSEC dataset contains multiple scenarios of the vehicle-mounted platform, which have 36 560 frames. The specific dataset information can be shown in Table II. Fig. 7 [16] shows the setup of sensors used in MVSEC dataset. In this article, we take the event camera images from the DAVIS left camera and the point cloud data from Velodyne LiDAR.

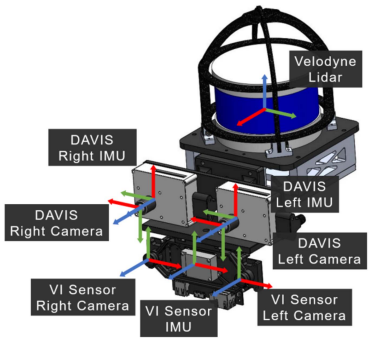


Fig. 7. Setup of sensors.

TABLE II
SPECIFIC INFORMATION OF THE MVSEC DATASET

Sequence	Time (s)	Frame	Size (Gbytes)
outdoor_day1	262	5238	8.6
outdoor_day2	653	13068	26.5
outdoor_night1	262	5237	8.1
outdoor_night2	375	7485	12.4
outdoor_night3	277	5532	9.0

1) Evaluation Metrics: In terms of the clustering algorithm, Davies-Bouldin Index (DBI) [41], Calinski-Harabasz Index (CHI) [42], and Silhouette Coefficient (SC) [43] are used to evaluate the quality of clustering. DBI shows the similarity between cluster categories, and CHI is the ratio of the covariance between clusters to the covariance within clusters. SC is the combination of its degree of cohesion and separation, which is widely accepted as a standard cluster validation measure in the large data clustering application software, such as MAEXPLORER (maexplorer.soureforge.net) for analyzing DNA microarray. The higher the CHI and SC index are, the better the quality effect of clustering, while DBI is the opposite.

For depth error, the mean absolute error (MAE) is chosen, as shown in the following:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \overline{y_i}| \tag{13}$$

where y_i and $\overline{y_i}$ represent the estimated depth value and the ground truth.

2) Hyperparameter Set: The clustering algorithm [44] has two key hyperparameters: ϵ and MinPts. ϵ represents the domain distance threshold between data samples. MinPts is the threshold of the number of samples within the distance ϵ. As shown in Fig. 8, we use different (ϵ, MinPts) for clustering to explore the optimal parameters. With the growth of ϵ or MinPts, CHI, DBI, and SC tend to their limits, but the point cloud is not reasonably divided. Since the clusters are also increasing, a large number of spatial coordinate points are classified into one category. From Fig. 8, we can see that the clustering quality is the best when the parameter is near ($\epsilon = 1$, MinPts $= 1$).

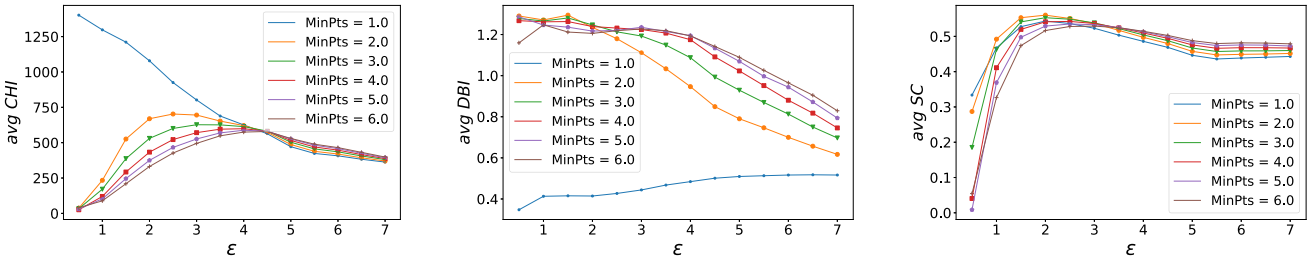
The kernel size used for depth expansion also has an impact on the accuracy of depth estimation. A small kernel cannot effectively cover most of the edge gaps in the depth map after the reconstruction of spatial coordinate points. Nevertheless, a large kernel often exceeds the range of the target object. Therefore, we empirically set the kernel size to 5×5 [4].

3) Baselines: We compare our approach with others, including image-based, event-based, LiDAR-based, and event-image fused algorithms. The event-based approaches are the work of Zhu *et al.* [29] and Hidalgo-Carrió *et al.* [30]. The ip_basic algorithm [23] ranks first on the KITTI depth completion benchmark among all published approaches, which is based on the LiDAR. The image-based approaches are MonoDepth [45] and MegaDepth [46]. What is more, we apply MegaDepth to the image frames reconstructed from the event using E2VID [47] to predict the depth, which is named MegaDepth$^+$, for a fuller comparison. The event-image fused approach [35] introduces recurrent asynchronous multimodal (RAM) networks to handle irregular data from multiple sensors.

B. Performance

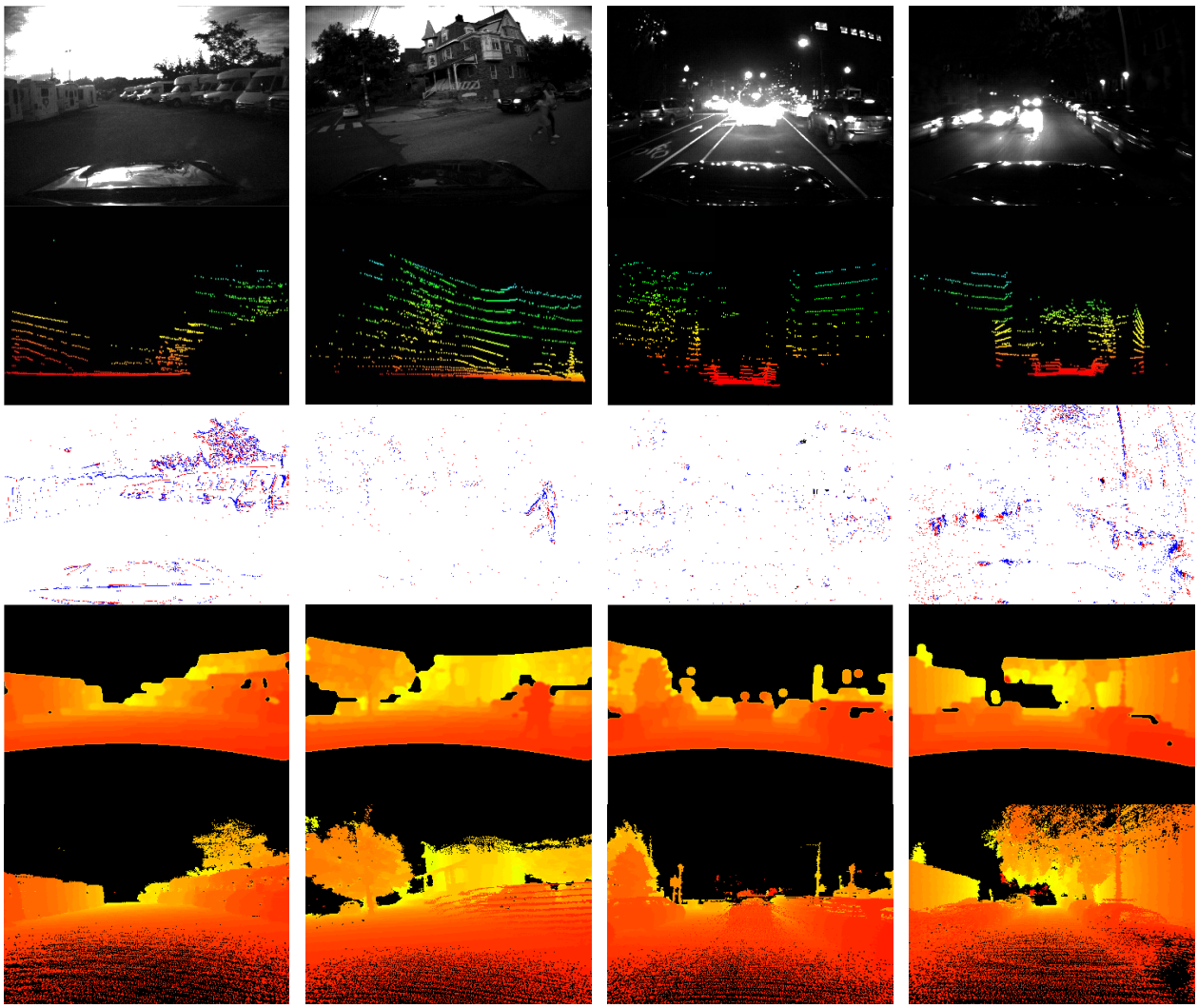
As shown in Fig. 9, we visualize the depth map after data fusion on the MVSEC dataset. We can observe that the raw point cloud data is quite sparse, and objects such as vehicles parked on the roadside are very blurred. For the event information, we can only see the general outline information. After data fusion, the shape of the vehicle is well restored, which has better recognition than the raw point cloud, as shown in Fig. 6. In addition, compared with the ground truth, the depth map of our approach still has a little gap. On the one hand, more sensors are used to generate ground truth, such as IMU. On the other hand, our approach is only based on the current frame, while the ground truth uses the historical and future point cloud data, which can supplement the blind area of the current frame. With limited information, our approach still achieves satisfactory results. Note that, in this article, we filter the depth information of ground truth whose distance is greater than 0–50 m, to ensure a more accurate comparison. For the test sequences of the MVSEC dataset, our proposed algorithm runs at 56 Hz on the Intel Core i7-7700K Processor, which means that 8596 LiDAR points and 1839 events are processed per second on average.

In Table III, we further compare depth errors of different approaches with thresholds up to 10, 20, and 30 m. First, our algorithm fully outperforms the image-based algorithm and the event camera-based algorithm in most scenarios, especially in daytime scenes. Taking full advantage of the prior knowledge from LiDAR, our algorithm has a maximum improvement of 33.13% compared to the work of Hidalgo-Carrió *et al.* [30] and a maximum improvement of 54.41% compared to the work of Zhu *et al.* [29] at 10 m. In the range of 20 m, compared with the best results of the two event-based algorithms, the depth estimation error of our work improves by 51.51%, 42.67%, 18.63%, and 17.17% in each scene, respectively. Second, the ip_basic algorithm based on LiDAR is head of image- and event-based algorithms in most scenarios. Compared with other sensors, the LiDAR can provide more accurate prior depth for depth estimation. Nevertheless, our approach outperforms the ip_basic algorithm in all scenarios, and the overall depth error has been reduced by 7.98% on average. The lack of depth estimation can be



(a) (b) (c)

Fig. 8. Distribution of *CHI*, *DBI*, and *SC* in the outdoor night1 sequence under different (ϵ, MinPts). (a) *CHI* index distribution. (b) *DBI* index distribution. (c) *SC* index distribution.



(a) (b) (c) (d)

Fig. 9. Qualitative performance of the four test sequences of the MVSEC dataset. The first row shows DAVIS grayscale frames, and the second row and the third row correspond to point cloud data and events' information, respectively. The fourth row is the result of our approach, and the ground-truth depth maps can be shown in the fifth row. Each column represents a test scenario of the MVSEC dataset. (a) Outdoor day1. (b) Outdoor day2. (c) Outdoor night1. (d) Outdoor night2.

compensated by the event camera, which can also provide extra depth information. Finally, with the increase in the cutoff depth distances, the depth error increases gradually. Compared with the other algorithms, the depth error of the LiDAR-based method increases more obviously with the increase of cut-off depth distance. It means that LiDAR is more sensitive to distance, especially for the low-beam sparse LiDAR.

V. Applications

In order to further verify the effectiveness of the proposed approach, we evaluate it with two real-life applications of intelligent robotics.

A. Vehicle Detection

Vehicle detection [48], [49] is an important part of the perception system for intelligent robots, which has become a

TABLE III
COMPARISON OF THE AVERAGE ABSOLUTE DEPTH ERROR (IN METERS) AT DIFFERENT CUTOFF DEPTH DISTANCES

Sequence	Distance (m)	Image-based			Event-based		Point cloud-based	Event and image	
		MonoDepth	MegaDepth	MegaDepth$^+$	Zhu *et al.*	Hidalgo-Carrió *et al.*	ip_basic	RAM	**Ours**
outdoor day1	10	3.44	2.37	3.37	2.72	1.85	1.39	1.39	**1.24**
	20	7.02	4.06	5.65	3.84	2.64	1.50	2.17	**1.28**
	30	7.02	4.06	5.65	3.84	3.13	5.07	**2.76**	4.87
outdoor day2	10	-	-	-	-	-	1.95	-	**1.72**
	20	-	-	-	-	-	2.07	-	**1.77**
	30	-	-	-	-	-	4.56	-	**4.35**
outdoor night1	10	3.49	2.54	2.40	3.13	3.38	2.31	2.50	**2.26**
	20	6.33	4.15	4.20	4.02	3.82	2.37	3.19	**2.19**
	30	9.31	5.60	5.80	4.89	4.46	5.81	**3.82**	4.50
outdoor night2	10	5.15	3.92	3.39	2.19	1.67	1.93	**1.21**	1.88
	20	7.80	5.78	4.99	3.15	2.63	2.20	2.31	**2.14**
	30	10.03	7.05	6.22	3.92	3.58	4.73	**3.28**	4.67
outdoor night3	10	4.67	4.15	4.56	2.86	1.42	1.82	**1.01**	1.78
	20	8.96	6.00	5.63	4.46	2.33	2.00	2.34	**1.93**
	30	13.36	7.24	6.51	5.05	**3.18**	4.65	3.43	4.55

significant technique for many key tasks, including obstacle avoidance, path planning, and so on. In this experiment, we compare our approach with the ip_basic algorithm [23] and the work of Li *et al.* [15]. Two classic open-source object recognition neural networks (Second-v1.5 [50] and PointPillars [38]) are used to evaluate the performance of point cloud data before and after fusion. Since Second-v1.5 and PointPillars both use the KITTI dataset, we convert the MVSEC dataset to the official KITTI format. Before training, the vehicles in the point cloud before and after fusion are manually marked with 3-D bounding boxes. Intersection over Union (IoU) is the overlap degree between candidate bound and ground truth bound, which is set to the default of 0.7. For training, we manually marked 1200 frames with 3-D bounding boxes, which contains 600 before-fusion frames and 600 after-fusion frames. Those frames are randomly divided into 1000 training and 200 testing frames. Default hyperparameters of networks that perform best on the KITTI dataset are used.

For evaluation metrics, we use the KITTI official indicators and verification sets to compare the performance of the network trained. Official indicators mainly include four metrics: 2-D Bbox (the accuracy of the 2-D bounding box), BEV (the accuracy of the bounding box in the birds-eye view), 3-D Bbox (the accuracy of the 3-D bounding box), and AOS (the accuracy of average orientation similarity). According to the official evaluation rules, when the overlap of the 3-D bounding box recognized by the neural network and the truth value labeled exceeds 70%, the detection is considered to be correct.

The detection results of different approaches can be shown in Table IV. From these data, we have the following observations. First, compared with the raw point cloud data, the fusion results of our approach achieve significant improvement, especially for 2-D BBox and 3-D BBox. Second-V1.5 and Pointpillars are boosted by 12.62% and 9.45% under 2-D BBox. For the 3-D BBox, Second-v1.5 and PointPillars increase by 21.18% and 8.61%. After data fusion, more 3-D characteristic information can be learned by neural networks, which makes the detection result more accurate. Second, our approach is superior to the ip_basic algorithm in almost all metrics. It is obvious that the event camera provides more 2-D information, and the projection of the object on the 2-D plane is improved. Finally, compared to the work of Li *et al.*, our approach achieves a better detection effect. The main reason is that the density-based clustering of events in dynamic scenes cannot achieve ideal results.

TABLE IV
COMPARISON OF QUALITY PERFORMANCE FOR VEHICLE DETECTION

	Methods	2D BBox	BEV	3D BBox	AOS
Raw data	SECOND-V1.5	37.03	78.58	27.27	26.67
	PointPillars	42.49	76.96	41.48	37.18
Ip_basic	SECOND-V1.5	45.13	65.07	37.35	**36.57**
	PointPillars	48.28	83.84	42.91	38.64
Li *et al.*	SECOND-V1.5	48.24	72.82	37.21	33.89
	PointPillars	50.08	84.22	41.31	37.23
Ours	SECOND-V1.5	**49.65**	**79.05**	**48.42**	34.79
	PointPillars	**51.94**	**84.88**	**50.10**	**39.88**

B. LiDAR Odometry

Simultaneous localization and mapping (SLAM) [33], [51] is used for autonomously constructing a descriptive map of passed area and localization by 3-D LiDAR scanner. In this experiment, we use HDL_GRAPH_SLAM algorithm [52] to evaluate the odometry performance of our approach. The HDL_GRAPH_SLAM is based on normal distributions transform (NDT) [53] scan matching, which is not constrained by the specific number of LiDAR beams. We use the APE and relative pose error (RPE) to evaluate SLAM accuracy. Besides, the mean error (ME) and RMSE of the APE and the RPE are calculated for a better comparison.

As shown in Table V, we compare the pose accuracy of point cloud data before and after fusing event information. We can observe that the fusion data based on our approach achieves higher pose accuracy. The main reason is that the fusion data can provide more feature information, which makes the point cloud registration easier and more accurate. There is a significant improvement in all metrics. Specifically, for the

TABLE V
COMPARISON OF POSE ACCURACY

	APE		RPE	
	ME	RMSE	ME	RMSE
Raw point cloud	16.24	21.09	1.25	1.34
Ip_basic	14.81	18.32	1.20	1.27
w/o-event	15.22	17.98	1.23	1.28
Ours	**13.91**	**16.41**	**1.10**	**1.21**
Improvement	14.35 %	22.19 %	12.00 %	9.70 %

APE, the RMSE is reduced by 22.19%, and the ME is reduced by 14.35%. For the RPE, the RMSE and the ME decrease by 9.70%, 12.00% respectively. In addition, it can be seen that the fusion with event cameras effectively improves the performance.

VI. CONCLUSION

In this article, we propose a dense depth-map estimation approach based on event stream and sparse point cloud. The approach completes ray-casting of filtered events and builds 3-D surface models with point clouds using a clustering algorithm. By intersection detection with rays and surface models, we reconstruct 3-D points corresponding to events and estimate depth. The experimental results show that, compared with other approaches, our approach can obtain more accurate depth information. In this article, we investigate the possibility of dense depth estimation based on event camera and sparse LIDAR. We believe that the scheme of combining event camera and LIDAR can help to improve intelligent robots.

REFERENCES

[1] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "An integrated framework for 3-D modeling, object detection, and pose estimation from point-clouds," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 3, pp. 683–693, Mar. 2015.

[2] D. Eigen, C. Puhrsch, and R. Fergus, *Depth Map Prediction From a Single Image Using a Multi-Scale Deep Network*. Cambridge, MA, USA: MIT Press, 2014.

[3] G. He, X. Yuan, Y. Zhuang, and H. Hu, "An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.

[4] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4796–4803.

[5] L. Ding and G. Sharma, "Fusing structure from motion and lidar for dense accurate depth map estimation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2017, pp. 1283–1287.

[6] (2018). *Velodyne Lidar*. [Online]. Available: https://velodynelidar.com/

[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, Oct. 2019.

[8] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy LiDAR completion with RGB guidance and uncertainty," in *Proc. 16th Int. Conf. Mach. Vis. Appl. (MVA)*, May 2019, pp. 1–6.

[9] Y. Li, Y. Ruichek, and C. Cappelle, "Optimal extrinsic calibration between a stereoscopic system and a LIDAR," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 8, pp. 2258–2269, Aug. 2013.

[10] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar-camera fusion for road detection using fully convolutional neural networks," *Robot. Autom. Syst.*, vol. 111, pp. 125–131, Oct. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018300496

[11] Y.-S. Shin, Y. S. Park, and A. Kim, "Direct visual SLAM using sparse depth for camera-LiDAR system," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5144–5151.

[12] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 10, pp. 259–275, Jan. 2010.

[13] F. B. Naeini, A. M. Alali, R. Al-Husari, A. Rigi, and Y. H. Zweiri, "A novel dynamic-vision-based approach for tactile sensing applications," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 5, pp. 1881–1893, May 2020.

[14] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 142–149, Feb. 2017.

[15] B. Li, H. Meng, Y. Zhu, R. Song, and K. Huang, "Enhancing 3-D LiDAR point clouds with event-based camera," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 9511712.

[16] 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.

[17] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse LiDAR data with depth-normal constraints," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 2811–2820.

[18] M. Jaritz, R. D. Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with CNNs: Depth completion and semantic segmentation," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 52–60.

[19] G. Gallego *et al.*, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2022.

[20] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant CNNs," in *Proc. Int. Conf. 3D Vis. (3DV)*, 2017, pp. 11–20.

[21] Z. Huang, J. Fan, S. Cheng, S. Yi, X. Wang, and H. Li, "HMS-Net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion," *IEEE Trans. Image Process.*, vol. 29, pp. 3429–3441, 2020.

[22] N. Chodosh, C. Wang, and S. Lucey, "Deep convolutional compressed sensing for LiDAR depth completion," in *Proc. Asian Conf. Comput. Vis.* Springer, 2018, pp. 499–513.

[23] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," in *Proc. 15th Conf. Comput. Robot Vis. (CRV)*, 2018, pp. 16–22.

[24] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DOF tracking with an event camera," in *Computer Vision*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 349–364.

[25] S. Schraml, A. N. Belbachir, and H. Bischof, "An event-driven stereo system for real-time 3-D 360° panoramic vision," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 418–428, Oct. 2016.

[26] 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, no. 12, pp. 1394–1414, Dec. 2018.

[27] 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.

[28] G. Gallego, M. Gehrig, and D. Scaramuzza, "Focus is all you need: Loss functions for event-based vision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12280–12289.

[29] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 989–997.

[30] J. Hidalgo-Carrió, D. Gehrig, and D. Scaramuzza, "Learning monocular dense depth from events," in *Proc. Int. Conf. 3D Vis. (3DV)*, 2020, pp. 534–542.

[31] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "DSEC: A stereo event camera dataset for driving scenarios," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4947–4954, Jul. 2021.

[32] 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.

[33] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 359–364.

[34] C. Brandli *et al.*, "Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor," *Frontiers Neurosci.*, vol. 7, p. 275, Oct. 2014.
[35] D. Gehrig, M. Ruegg, M. Gehrig, J. Hidalgo-Carrio, and D. Scaramuzza, "Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2822–2829, Apr. 2021.
[36] X. Jiang, "Iterative truncated arithmetic mean filter and its properties," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1537–1547, 2011.
[37] N. Jawas and N. Suciati, "Image inpainting using erosion and dilation operation," *Int. J. Adv. Sci. Technol.*, vol. 51, pp. 127–134, Feb. 2013.
[38] 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.*, Oct. 2019, pp. 12697–12705.
[39] L. Chen and J.-C. Xu, "Optimal delaunay triangulations," *J. Comput. Math.*, vol. 4, pp. 299–308, Dec. 2004.
[40] P. Sturm, *Pinhole Camera Model*. Springer, 2014.
[41] J. Xiao, J. Lu, and X. Li, "Davies bouldin index based hierarchical initialization K-means," *Intell. Data Anal.*, vol. 21, no. 6, pp. 1327–1338, Nov. 2017.
[42] S. Łukasik, P. A. Kowalski, M. Charytanowicz, and P. Kulczycki, "Clustering using flower pollination algorithm and Calinski-Harabasz index," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Oct. 2016, pp. 2724–2728.
[43] D.-T. Dinh, T. Fujinami, and V.-N. Huynh, "Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient," in *Proc. Int. Symp. Knowl. Syst. Sci.* Springer, 2019, pp. 1–17.
[44] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, Jan. 1996, pp. 226–231.
[45] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 270–279.
[46] Z. Li and N. Snavely, "MegaDepth: Learning single-view depth prediction from internet photos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2041–2050.
[47] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 1964–1980, Jun. 2019.
[48] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications," *IEEE Sensors J.*, vol. 20, no. 9, pp. 4901–4913, May 2020.
[49] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.
[50] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
[51] R. Munguía and A. Grau, "Closing loops with a virtual sensor based on monocular SLAM," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 8, pp. 2377–2384, Aug. 2009.
[52] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, Mar. 2019, Art. no. 172988141984153.
[53] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Oct. 2003, pp. 2743–2748.

Mingyue Cui received the B.Sc. degree in software engineering from Chongqing Normal University, in 2014, the M.Sc. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2017, where he is currently pursuing the Ph.D. degree (major in computer science).

His research interests are in the areas of autonomous driving and Internet-of-Things (IoT). He is recently focused on computer vision and edge computing in autonomous driving.

Yuzhang Zhu received the B.S. degree in software engineering from Sun Yat-sen University, Guangzhou, 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.

Yechang Liu is currently pursuing the B.Sc. degree with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, under the supervision of K. Huang.

His research interests include the area of autonomous driving and edge computing.

Yunchao Liu is currently pursuing the B.Sc. degree with Sun Yat-sen University (major in computer science), Guangzhou, China.

He joined the Robotic and Intelligence Computing Laboratory, Sun Yat-sen University, in 2020, where he is currently a student under the Instruction of K. Huang. His research interests include the area of autonomous driving and robotics.

Gang Chen (Member, IEEE) 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, 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, and Peng Cheng National Laboratory, China. His research interests include deep learning and neural networks.

Dr. Chen received the best paper awards on DATE 2021, ICET 2021, and ESTImedia 2020 and best paper candidate awards on CODES+ISSS 2020. He is an Associate Editor of the *Journal of Circuits, Systems and Computers*. He also serves as a Reviewer for several Tier-top conferences/journals.

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

He joined Sun Yat-Sen University, Guangzhou, China as a Professor in 2015. He was a Senior Researcher with the Computer Science Department, Technical University of Munich, Munich, Germany from 2012 to 2015, and a Research Group Leader with fortiss GmbH, Munich, Germany, in 2011. His research interests include techniques for the analysis, design, and optimization of embedded/CPS systems, particularly in the automotive, medical, and robotic domains.

Prof. Huang was a recipient of best paper awards/candidates for a number of conferences.