An Event-based Stereo 3D Mapping and Tracking Pipeline for Autonomous Vehicles

Anass El Moudni, Fabio Morbidi, Sebastien Kramm, Rémi Boutteau

An Event-based Stereo 3D Mapping and Tracking Pipeline for Autonomous Vehicles

Anass El Moudni[1], Fabio Morbidi[2], Sebastien Kramm[1], Rémi Boutteau[1]

***Abstract*— Event cameras are bio-inspired, motion-activated sensors which generate asynchronous events instead of intensity images at a fixed rate. These sensors have been shown to outperform traditional frame-based cameras by large margins, in case of high-speed motions and scenes with high dynamic range. Next-generation intelligent vehicles are expected to greatly benefit from these novel cameras, especially in adverse lighting conditions, and their potential is still largely untapped.**

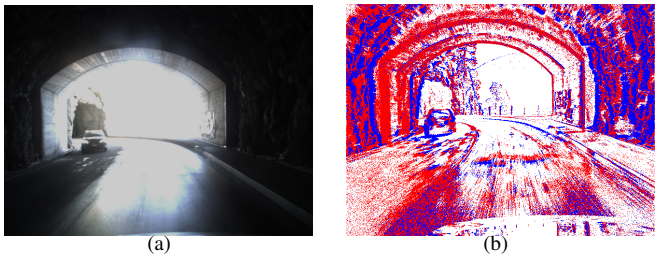
In the last decade, the continuous stream of events produced by an event camera has been exploited in numerous 3D perception tasks (depth estimation, 6-DoF tracking, visual-inertial odometry, etc.). In this paper, we propose an event-based stereo pipeline for simultaneous 3D mapping and tracking. The mapping module relies on DSI (Disparity Space Image) fusion, and the tracking module makes use of time surfaces as anisotropic distance fields, to estimate the pose of the stereo camera. Numerical experiments with a publicly-available event dataset recorded by a car in different urban environments, show the effectiveness of the proposed architecture.

***Index Terms*— Event camera, Stereo depth estimation, Visual odometry, Disparity space image, Intelligent vehicle**

I. Introduction

Intelligent vehicles, or more generally, Advanced Driver-Assistance Systems (ADAS), offer solutions to traffic-related problems and they are meeting with a growing success today. Typical objectives include reducing the number of accidents (94% of them are caused by human errors), decreasing the level of stress induced by driving in congested areas, and facilitating the transport of freight and people with mobility issues [1]. For all these tasks, *perception* plays a vital role since it ensures a vehicle's understanding of the surrounding environment. Modern cars are equipped with a suite of advanced sensors, such as multiple color cameras (8 cameras in Tesla Model Y), radars [2], lidars [3], RGB-D cameras [4], and, more recently, *event cameras* [5].

Event cameras are asynchronous, neuro-inspired sensors which have lately brought about a paradigm shift in visual perception. In fact, unlike conventional cameras that output images at a fixed rate, event-based sensors only react to moving objects in a scene, by detecting pixel-wise brightness changes. Their high dynamic range (up to 140 dB vs. 60 dB of traditional cameras), allows them to correctly operate in challenging illumination conditions (e.g. twilight, entrance/exit of a tunnel, see Fig. 1). Other attractive properties, such as low latency, reduced energy consumption and high temporal resolution (in the order of microseconds), make event cameras an ideal choice for automotive applications.



(a) (b)

Fig. 1. Passing through a dark tunnel, a notoriously challenging task for a driver: (a) RGB image from the left camera of a stereo system in the DSEC dataset [6]; (b) Reconstructed frame using event data.

The main challenge, today, is to design algorithms which are able to process the continuous stream of events and extract the relevant information, either by fusing it with measurements from other sensors [7], [8], or by devising event representations which mimic the output of frame-based cameras [9].

The majority of the existing works address the problems of camera tracking and 3D mapping, as two separate tasks. For the mapping module, a simple solution consists in reconstructing a grayscale image from the events by simply accumulating them over time (or by using a recurrent neural network [10]), and then applying classic computer vision algorithms. Other methods rely on data association [11], [12] (matching to calculate the disparity), or are correspondence-free, but the knowledge of camera's trajectory is necessary to estimate the depth by performing a re-projection into a reference view [13]. As far as the camera tracking problem is concerned, a variant of the Kalman filter has been used in [14] to predict the 6-DoF camera motion. Spiking neural networks have been also proposed to perform numerical regression and estimate the angular velocity of an event camera [15].

From the previous literature review, it emerges that algorithms for monocular event cameras are prevalent. ESVO (Event-based Stereo Visual Odometry) [16] is one of the few existing algorithms for *stereo event cameras*, and in this paper we upgrade it by introducing an accurate and self-contained (i.e. no need to assume a perfect knowledge of camera's pose) 3D mapping and tracking pipeline.

*This work was supported by the French National Research Agency through the CERBERE project (ANR-21-CE22-0006).

[1]Université Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Université, LITIS UR 4108, Rouen, France, {anass.el-moudni, sebastien.kramm, remi.boutteau}@univ-rouen.fr

[2]Université de Picardie Jules Verne, MIS laboratory, UR 4290, Amiens, France, fabio.morbidi@u-picardie.fr

More specifically, the original contributions of this paper can be summarized as follows:

- Replacement of the mapping module in ESVO with a more efficient one, based on the fusion of disparity space images (DSIs) [17],
- Validation of the full 3D mapping and tracking pipeline with an event-oriented dataset (DSEC [6]) in different driving scenarios.

The remainder of this paper is organized as follows. Sect. II reviews the related work on event cameras for autonomous vehicles. The problem studied in the paper is formulated in Sect. III. Sect. IV presents the results of our experiments with real event data in different urban environments. Finally, the main contributions of the paper and some possible avenues of future research are discussed in Sect. V.

II. Related work

In this section, we briefly recall some relevant event representations, with special emphasis on time surfaces (Sect. II-A). We also review related work on event-based depth estimation (Sect. II-B), and camera-pose estimation and tracking (Sect. II-C).

A. Event representations

As mentioned in Sect. I, event cameras generate an asynchronous stream of events induced by the motion of the camera and/or of the observed scene. In order to provide more context and extract more discriminant information about the environment, the events are often aggregated in space and/or time before being processed. The following event representations are the most common.

Individual events or **spikes**: Individual events are represented by quadruples, $\boldsymbol{e}_k = (x_k,\, y_k,\, t_k,\, p_k)$, where $(x_k,\, y_k)$ are the pixel coordinates of k-th event, t_k is the timestamp, and $p_k \in \{-1,\, +1\}$ is the polarity of the event ($p_k = +1$ if the log brightness change at pixel $(x_k,\, y_k)$ is positive, and $p_k = -1$, otherwise).

Group of events: The spatio-temporal neighborhood information carried by a single event is generally poor. For this reason, the events can be grouped into packets (or clusters) of N_e elements: $\mathcal{E} = \{\boldsymbol{e}_k\}_{k=1}^{N_e}$. The choice of N_e depends on the complexity of data processing algorithm and on the available hardware (processor, memory capacity).

Event frames: Conventional computer vision algorithms can be used with event data, by defining *Event Frames* (or *Event Images*). The events are accumulated over a time window of fixed length T and represented as an image, with three possible values per pixel ($+1$ if the polarity of the event is positive, -1 if the polarity is negative, and 0 if no event has been detected in the interval T).

Time surfaces (TS): Time surfaces are 2D maps that store the timestamp of the most recent event at a given pixel location [18]. The intensity, in these pseudo-images, encodes the time and motion information. Hence, old events have low intensities compared to the most recent ones (in classical computer vision, these 2D maps are generally referred to as "motion history images"). In theory, the map should be updated at each incoming event using a kernel (to retain sparsity and asynchronicity). However, in practice, since a single event is informative enough, multiple events are accumulated over a time window of length T (parameter T should be selected with care: it depends on the nature of the observed scene and on the technical specifications of the event camera).

Other event representations (such as, voxel grids, voxel cubes, adaptive TS [19], graph-based, etc.) exist in the literature, and they have different pros and cons, depending on the application at hand. For more details, the reader is referred to [5, Sect. 3.1].

B. Depth estimation

The problem of depth estimation with event cameras has been widely studied in recent years, and it has been attacked from different angles.

Most of the existing works use a synchronized *stereo event camera* for instantaneous depth estimation. A two-step approach is usually adopted: first, the matching problem is solved (by leveraging, for example, the epipolar constraint), and then triangulation is performed by knowing the intrinsic and extrinsic parameters of the two event cameras. The main idea is to define a matching cost function and minimize it, while processing the events via a sliding spatio-temporal window. Belief propagation [11] and event-driven semi-global matching (ESGM) [12] are two methods that have been used to minimize such a cost function. The main drawback of these methods is the computational cost, the data-association step being the major bottleneck. To get around this issue (and minimize the number of outliers), one can exploit extra sources of information, such as edge orientation or event polarity. In [20], the authors rely on camera's velocity information for depth estimation: they use it to generate a time-synchronized event disparity volume, where regions with the correct disparity are in focus.

The methods based on *monocular event cameras* for depth estimation, take a radically different approach, since the temporal correlation between the events in the left and right camera of a stereo rig, is no longer available. Some of the existing algorithms assume a prior knowledge of camera's motion. This is the case with the space-sweep method, which leverages the sparsity of events for depth estimation without the need for a data-association step [13].

By and large, the existing depth estimation methods either process events individually or in packets, depending on the event representation adopted. The "instantaneous" methods work in real time and have decent accuracy, whilst the methods which process the events in packets are very accurate but computationally expensive. This accuracy-efficiency trade-off has yet to be quantified in the literature, and in this paper, we propose a new stereo 3D mapping and tracking pipeline (see Sect. III), in order to get the best of both worlds.

C. Camera-pose estimation and tracking

Event cameras are often combined with other incumbent sensors for more accurate and robust pose estimation: for example, an inertial measurement unit (IMU) for visual-inertial

odometry [21], or a frame-based camera [22]. The great majority of the existing methods solve the event-based visual odometry problem using feature tracking, by treating events and conventional grayscale images, conjointly. More specifically, there exist two main families of methods in the literature. *Indirect methods* process events through an intermediate representation, typically 2D maps (e.g. time surfaces [18]), and therefore extract keypoints to track. Although these methods work relatively well with standard grayscale images [23], event-based features are much harder to track because of their erratic behavior and the lack of local information (spatial neighborhood). In contrast, *direct methods* process single events, and satisfactory results, in terms of accuracy, are reported in [24]: however, the existing event-feature extractors and trackers [25], [26] cannot be easily adapted to a visual odometry problem, since they suffer from poor accuracy and stability.

The previous methods have been generally conceived to guarantee low latency (i.e. to estimate, in principle, the camera pose at each incoming event). However, the information encoded into a single event is typically too poor for reliable pose estimation. Probabilistic filters [27], [28] have been shown to be more suitable for this specific task (the estimated pose is updated every time a new event is triggered), while still preserving the asynchronous nature of the incoming stream of events.

Closer to our goal, in [16], the mapping and tracking problems are treated simultaneously with a stereo event camera: the initial depth estimate is computed from a pair of TS via semi-global matching, while the tracking module uses the TS as anisotropic distance fields. The main idea is to align the "dark" regions in two successive TS, and to find the rigid transformation that gives the best correlation score. This optimization problem is solved using the Lucas-Kanade method [29]. The mapping module, on the other hand, leverages the notions of epipolar constraint and event co-occurrence: in fact, each point on the edges of an observed object, generates two simultaneous events on corresponding epipolar lines, in the left and right camera. In practice, the matching problem is solved by minimizing a similarity cost function based on the warped time surfaces in the two cameras.

III. Problem formulation

Our objective, in this paper, is to design an accurate event-based stereo 3D mapping and tracking pipeline. We build upon ESVO [16] and replace its mapping module with a more efficient one, based on the fusion of DSIs [17]. For the reader's convenience, the complete flowchart of our method is reported in Fig. 3. The input events are treated in two different ways. The stream of individual events $\boldsymbol{e}_k = (x_k, y_k, t_k, p_k)$ is first processed to extract the TS using the exponential kernel,

$$\mathcal{T}(\mathbf{x}, t) = \exp\left(-\frac{t - t_{\text{last}}(\mathbf{x})}{\eta}\right), \tag{1}$$

where t_{last} is the time of the most recent event at pixel $\mathbf{x} = (x_k, y_k)$ and $\eta > 0$ is the decay-rate parameter.

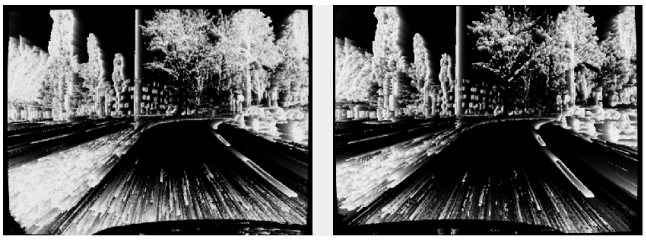


Fig. 2. A pair of time surfaces generated from the event data of "Zurich04" sequence in the DSEC dataset [6].

TS are memory efficient and store the motion history of the edges in the scene (in fact, light pixels are more recent than dark pixels in the pseudo-image). In ESVO [16] and in the variant T-ESVO [19], the TS are used in both the mapping and tracking modules, whereas in this work we only exploit them in the tracking module (see the cyan blocks in Fig. 3).

After an initialization step in which a modified semi-global matching (SGM) method is used, a preliminary depth map is built and the first pose is estimated by correlating the depth map with the re-projected TS, using the estimated rigid transformation. The poses are stored in a database and they are accessible at any moment in time, thanks to an interpolation on SE(3). The camera pose is then used, alongside with the raw events, to perform a back-projection (DSI creation), via the EMVS (Event-based Multi-View Stereo) algorithm [13]. Finally, the mapping module fuses the DSIs across cameras and over time [17]: the back-projected rays have maximum density at the correct depth Z, which allows to recover the 3D locations (these locations can be stored in the point-cloud database as well). The local depth map can ultimately be used to refine the camera-pose estimates.

A. Tracking module

The tracking module (cyan blocks in Fig. 3) uses the TS as anisotropic distance fields (see the example in Fig. 2). Large values in the TS correspond to recent edges, while small values correspond to older edge locations. This is an interesting property of TS, which can be used to track the motion of the camera: in fact, it is sufficient to monitor the growth rate of the values on the TS. The goal is to align the dark regions in the negative TS, defined by,

$$\overline{\mathcal{T}}(\mathbf{x}, t) = 1 - \mathcal{T}(\mathbf{x}, t), \tag{2}$$

with the inverse local depth maps, when projected into a given camera frame (usually the left one). The pose that ensures the best alignment between the dark regions of the negative TS and the points in the depth map, is then the sought-after solution. More formally, this optimization problem can be stated as follows. Let us assume that a packet of pixels $\mathcal{S}_{\mathcal{F}_{\text{ref}}} = \{\mathbf{x}_i\}_{i=1}^{n}$ with known depth Z_i in the reference frame $\mathcal{F}_{\text{ref}}$, and that a negative TS, referred to as $\overline{\mathcal{T}}_{\text{left}}(\cdot, k)$, are available at time k. Then, one seeks to find the pose $\mathbf{T} \in \text{SE(3)}$ which guarantees the best alignment between the minima of $\overline{\mathcal{T}}_{\text{left}}(\cdot, k)$ and the warped semi-dense map $\mathbf{T}(\mathcal{S}_{\mathcal{F}_{\text{ref}}})$. In other words, we search for the vector $\boldsymbol{\theta}^*$

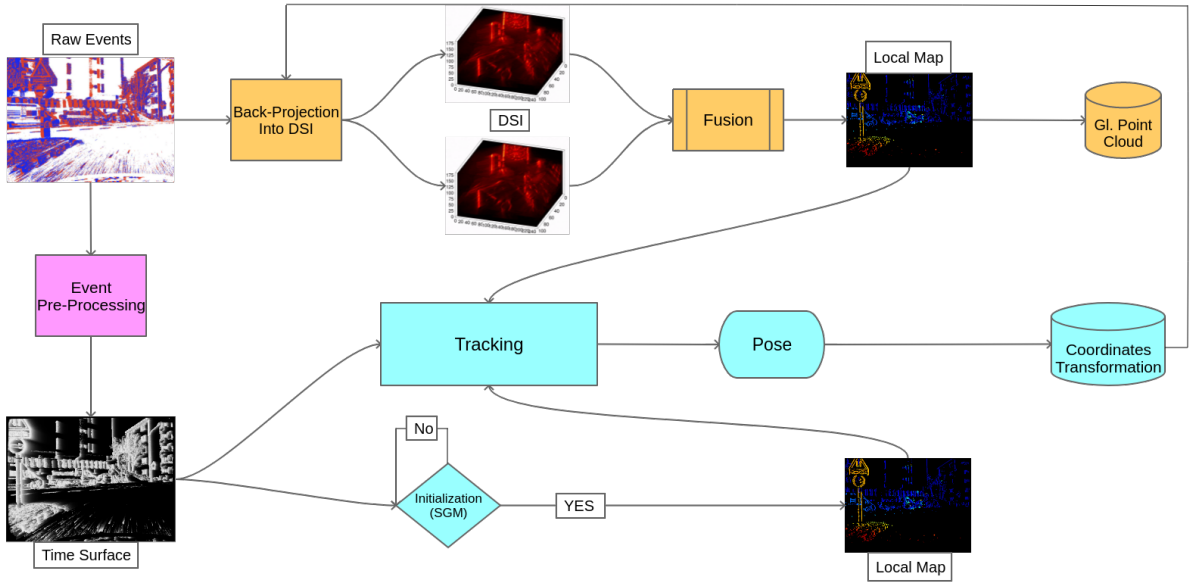


Fig. 3. Overview of the proposed 3D mapping and tracking pipeline. The mapping module is marked in orange, the tracking module in cyan, and the blocks for output storage are represented as cylinders (flowchart adapted from [16]).

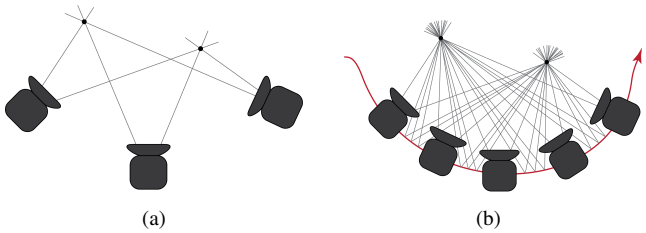
that satisfies:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\mathbf{x} \in \mathcal{S}_{\mathcal{F}_{\text{ref}}}} \overline{\mathcal{T}}^2_{\text{left}}\big(W(\mathbf{x}, Z, \boldsymbol{\theta}), k\big), \qquad (3)$$

where $W(\mathbf{x}, Z, \boldsymbol{\theta})$ is the warping function that maps points from $\mathcal{F}_{\text{ref}}$ to a given reference frame. Three consecutive transformations are performed to find the matrix $\mathbf{T}$ which guarantees the best alignment. First, an event at location $\mathbf{x}$ is projected into the 3D space, using the estimated depth. This point is multiplied by the candidate transformation matrix $\mathbf{T}$, and the new 3D position is finally re-projected into the left camera frame by using the inverse projection function. Matrix $\mathbf{T}$ is recovered via the function $\boldsymbol{G}(\boldsymbol{\theta}) : \mathbb{R}^6 \mapsto \text{SE}(3)$, which maps the vector $\boldsymbol{\theta}$, stacking the 3 CGR (Cayley-Gibbs-Rodrigues) parameters [30] for orientation and the 3 parameters for translation, to the three-dimensional special Euclidean group. For more details on the mathematical formulation and resolution methods used in the tracking module, the reader is referred to [16].

B. Mapping module

The back-projection method is instrumental in generating a disparity space image (DSI), which plays a central role in our mapping module (orange blocks in Fig. 3). In classical computer vision, the *space-sweep* approach [31] allows to solve the multi-view stereo (MVS) problem, and a 3D reconstruction can be obtained without the need for matching or data association across cameras. Unlike the majority of existing methods, that leverage the pixel intensities to solve the MVS problem in two steps (matching and triangulation), the space-sweep approach only relies on binary edges in the images, and this sparse information closely resembles that generated by event cameras. More precisely, the *classical* space-sweep approach proceeds in three steps as follows (see Fig. 1(a)):



(a) (b)

Fig. 4. Space-sweep method with conventional and event cameras. (a) Two points in space are observed by a moving frame-based camera; (b) The same two points are observed by a moving event camera: the number of rays is significantly larger because of the continuous acquisition process.

1) Back-project or warp image features (edges in our case) as rays into a DSI. In other words, every time an edge point triggers an event, counts it as a ray through the DSI after back-projection,
2) Create a ray density function which is incremented every time a ray crosses a voxel of the DSI,
3) Choose which voxel corresponds to the actual 3D position of each pixel on the edges. For this, use a threshold on the ray density function.

When comes to *event cameras*, no edge-detection algorithm (e.g., Canny or Sobel edge detectors) is needed, since the events are triggered by brightness changes naturally occurring along the edges. In Fig. 4(b), we can see that the edges tend to trigger events from too many "consecutive" viewpoints, because of the continuous and asynchronous acquisition process. The event-based space-sweep method then takes packets of events $\{\boldsymbol{e}_k\}_{k=1}^{N_e}$ as input, to replace

the point features that will be subsequently back-projected into the DSI, since the position of the camera (viewpoints) is supposed to be known at each time instant, according to the MVS assumption. Obviously, the higher the number of viewing rays generated by the multiple viewpoints, the easier the detection of the regions of interest in the DSI (in fact, it comes to a simple ray-density analysis). To create the DSI, we start by choosing a virtual reference view at time $t = T/2$ where T is the time window of processed subset of events. Then, a volume V which is consistent with the technical specifications of the event camera, is defined. The size of the volume depends on the image resolution of the camera (width w and height h), and depth resolution (number of depth planes N_z). Hence, the size of V is $w \times h \times N_z$. The score is then stored by using a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}_{>0}$, which counts the back-projected viewing rays passing through each voxel of V. A threshold is introduced to select the voxels with maximum-density rays: by this means, the exact depth location of the observed object in the scene is obtained. Visually, on the DSI, an object will be blurry when faraway from the correct depth plane, and in focus, otherwise.

Different fusion functions (which are special cases of Hölder mean) have been considered in [17] to maximize the accuracy of depth estimates and minimize the number of outliers. The DSIs are fused across the two cameras and over time, by subdividing the temporal window T into N_s sub-intervals. The harmonic mean for the fusion across cameras, and the arithmetic mean for the fusion over time, have been empirically shown to be the best combination in [17]. This stands to reason: in fact, the harmonic mean tends to give more importance to DSI regions with high, similar values, which can be interpreted as maximizing process of the correlation score between the refocused events across the cameras.

IV. Experimental results

In this section, we briefly describe the dataset used in our numerical experiments (Sect. IV-A), and discuss our qualitative and quantitative results (Sect. IV-B and Sect. IV-C).

A. Dataset

The proposed pipeline has been evaluated on the publicly-available DSEC dataset [6]. To create this dataset, the authors equipped a car with two (hardware) synchronized Prophesee Gen 3.1 event cameras, two RGB cameras, one lidar, and one RTK-GPS. The ground-truth depth maps are obtained from the lidar and the stereo RGB camera mounted on the car roof, using a two-step procedure. The dataset comprises 53 sequences recorded in three Swiss cities, under different conditions (broad daylight/darkness, urban/peri-urban environments, multiple moving objects including pedestrians, cars, trucks, etc.).

We selected five representative sequences, two from "Interlaken" and three from "Zurich", for our qualitative and quantitative analyses (on average, the sequences have a duration of 25 seconds and count 17 million events/s). The dataset consists of a collection of files containing the events, their rectified spatial coordinates, the RGB images from the stereo rig, and the disparity maps projected on the left event camera, to assess their accuracy.

B. Qualitative evaluation

Different metrics exist in the literature to evaluate the performance of a depth-estimation algorithm. A general and commonly-used metric is pixel-level accuracy with respect to the ground truth. While this metric is well suited for static scenes, in complex driving scenarios, other criteria, such as the density of estimated depth map or the ability to detect different moving objects in front of the vehicle, may be more appropriate. Therefore, for a fair evaluation of a depth-estimation algorithm, one should go beyond a pure quantitative analysis.

The mapping module described in Sect. III-B, produces accurate depth maps, which are, however, less dense than those yielded by ESVO. This is evident in Fig. 5, where the first (second) row reports the depth maps obtained with our method (ESVO) in two sequences of the DSEC dataset. The edges in the depth maps generated by our method are sharper, which makes the detection (e.g., by an off-the-shelf deep learning algorithm such as YOLO [32]) of different objects in the scene (traffic light, tree, etc.), easier.

Driving scenarios are challenging for event-based vision, since more events are generated on the periphery of the image rather than at the center, where the apparent motion is weaker. The presence of *moving objects* in the scene (cars, trucks, pedestrians, etc.) is another issue, and no results exist in the literature with non-static cameras (cf. [16], [17]). The back-projection of moving edges is indeed problematic, since the number of viewpoints of the same object is generally small, which results in an increase in the number of outliers or in a decrease in the size of the 3D point cloud (the ray density function always remains below the threshold

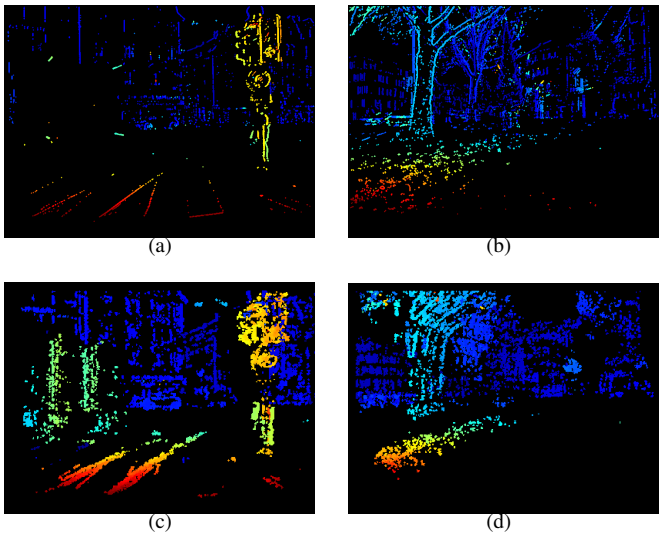
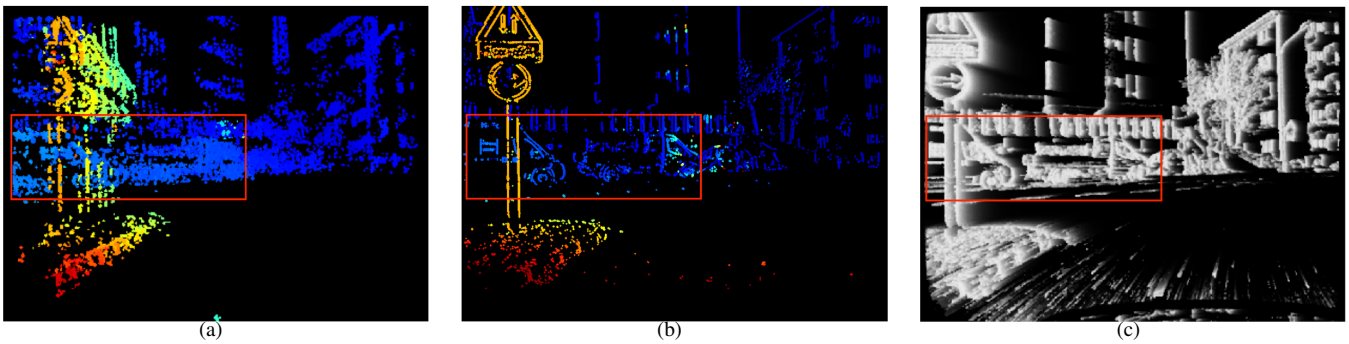


Fig. 5. Comparison of the depth maps generated with: (a), (b) Our method, and (c), (d) ESVO (best viewed in color). With our method, the objects in the scene have sharper contours: the traffic light and the tree, respectively, can then be easily identified.



(a) (b) (c)

Fig. 6. Comparison of two mapping modules in the "Zurich04_a" sequence: (a) ESVO's mapping module, (b) DSI fusion and (c) corresponding time surface. Note that the two moving vehicles in front of the car inside the red box, are clearly distinguishable in (b) and (c), but not in (a).

value). In spite of these challenges, our method provides satisfactory results with objects moving in a direction which is roughly orthogonal to that of the car (see Fig. 6). In particular, by comparing Fig. 6(a) with Fig. 6(b), we can see that the depth maps generated by ESVO's mapping module are denser than those produced by DSI fusion, which results in larger 3D points clouds. However, as indicated in [17], and confirmed by our experiments, DSI fusion is the clear winner in terms of accuracy.

It is worth pointing out here that the performances of the algorithm may significantly vary, depending on the apparent motion of the objects in the scene: weak motions, corresponding, for example, to other vehicles heading in the same direction as the car, are more problematic. In these cases, enough events (edges) are generated to detect a moving object like a truck, which is clearly distinguishable on the time surfaces. However, these events are too sparse to back-project them all and guarantee continuous edges on the depth map. The small number of events generated by dynamic objects with weak apparent motion, accounts for the "empty" (black) zones in the estimated depth maps. The reflecting and texture-less surfaces of certain vehicles further exacerbate this problem, a priority area for future research.

C. Quantitative evaluation

For a rigorous evaluation of the proposed pipeline, we also conducted a quantitative analysis by using the three urban sequences, "Interlaken00_c", "Interlaken00_d" and "Zurich02_a" in the DSEC dataset. Table I reports the mean absolute error (MAE), median error (MED), and root-mean-square error (RMSE) of the depth estimates in meters, and the relative error (RE) in percentage. The results have been obtained by comparing the estimated depths projected on the left event camera, with the disparity maps projected on the same reference frame. The results in Table I are on par with those reported in [17]. Although our MAE is the second best in one sequence, our MED is always smaller (as is known, MED is less sensitive to outliers than MAE). Table I also reveals some variability in the MAE over the three sequences: this is imputable to the different data-recording conditions, event-camera settings, and number and speed of moving objects in the scene. The same variability is also apparent within the single sequences (see Fig. 7), and it depends on the type of maneuver executed by the car driver.

In conclusion, our analysis has shed some light on the potential of the proposed architecture, and it has helped to identify possible areas for improvement as well. Albeit preliminary, our results are promising and indicate that our event-based stereo pipeline is successful in capturing the intricate spatial details present in complex man-made environments, thereby contributing towards better scene understanding and, in the long term, improved road safety.

V. Conclusion and Future Work

In this paper, we have presented a complete pipeline for 3D mapping and tracking with a stereo event camera mounted on a moving vehicle. In particular, the mapping module used in ESVO [16], has been replaced with a more efficient one based on DSI fusion, which does not require the time-consuming data association step. Numerical experiments with real event data from the DSEC dataset,

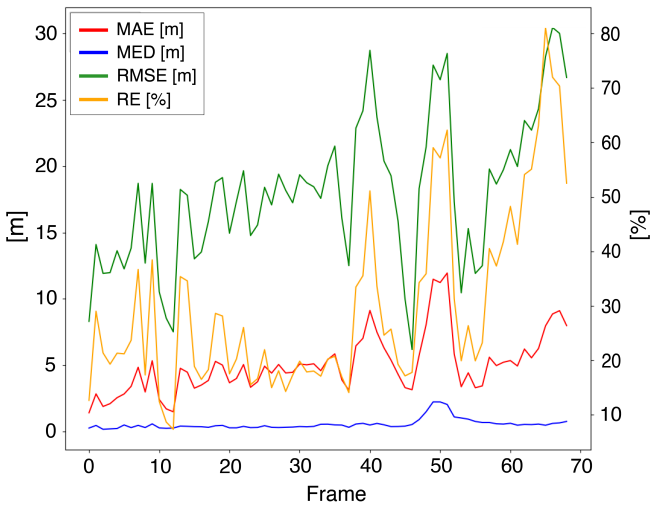


Fig. 7. Time evolution of MAE (red), MED (blue), RMSE (green) in meters, and of RE (yellow) in percentage, in the "zurich04_b" sequence. The peaks correspond to sharp turns or the existence of several moving objects.

TABLE I

QUANTITATIVE COMPARISON OF OUR METHOD WITH TWO STATE-OF-THE-ART APPROACHES, IN THREE SEQUENCES OF DSEC DATASET. THE BEST VALUES ARE IN BOLDFACE.

Method	Interlaken00_c				Interlaken00_d				Zurich02_a			
	MAE	MED	RMSE	RE [%]	MAE	MED	RMSE	RE [%]	MAE	MED	RMSE	RE [%]
EMVS [13]	4.6847	2.5641	3.6981	28	4.5647	2.0254	3.8247	25	5.2647	2.5471	3.6941	28
ESVO [16]	3.5817	2.1276	**3.2946**	11	4.2791	1.2035	**3.2350**	13	**3.9210**	1.5820	**3.6691**	**10**
Ours	**2.6765**	**0.6682**	10.0547	**10**	**3.3667**	**0.6109**	8.5041	**12**	4.7438	**0.8251**	10.7409	12

have shown the accuracy and robustness of the proposed architecture.

This work opens up several interesting perspectives for future research. First of all, we plan to reduce the number of degrees of freedom of our tracking module in order to tailor it to the simplified dynamics of a car. In the near future, we are going to build our own multi-modal dataset to test the proposed pipeline in different traffic and weather conditions, and we would like to fuse the event data with measurements coming from other on-board sensors (e.g., lidar [8], [33], RGB camera, IMU). The adoption of a unified event representation in the mapping and tracking modules, is also expected to improve the time efficiency and robustness of our pipeline. Finally, the reconstruction of depth maps could benefit from an additional processing step in which connected areas are detected.

REFERENCES

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[2] F. Roos, J. Bechter, C. Knill, B. Schweizer, and C. Waldschmidt, "Radar Sensors for Autonomous Driving: Modulation Schemes and Interference Mitigation," *IEEE Microw. Mag.*, vol. 20, no. 9, pp. 58–72, 2019.

[3] S. Royo and M. Ballesta-Garcia, "An Overview of Lidar Imaging Systems for Autonomous Vehicles," *Applied Sciences*, vol. 19, no. 9, p. 4093, 2019.

[4] J. Yang, C. Wang, H. Wang, and Q. Li, "A RGB-D Based Real-Time Multiple Object Detection and Ranging System for Autonomous Driving," *IEEE Sensors J.*, vol. 20, no. 20, pp. 11 959–11 966, 2015.

[5] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-Based Vision: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, 2022.

[6] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "DSEC: A Stereo Event Camera Dataset for Driving Scenarios," *IEEE Rob. Autom. Lett.*, vol. 6, no. 3, pp. 4947–4954, 2021.

[7] A. Lee, Y. Cho, Y. s. Shin, A. Kim, and H. Myung, "ViViD++: Vision for Visibility Dataset," *IEEE Rob. Autom. Lett.*, vol. 7, no. 3, pp. 6282–6289, 2022.

[8] V. Brebion, J. Moreau, and F. Davoine, "Learning to Estimate Two Dense Depths from LiDAR and Event Data," in *Proc. Scandinavian Conf. Image Anal.*, 2023, pp. 517–533.

[9] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1731–1740.

[10] 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, 2019.

[11] Z. Xie, S. Chen, and G. Orchard, "Event-Based Stereo Depth Estimation Using Belief Propagation," *Front. Neurosci.*, vol. 11, 2017, article no. 535.

[12] Z. Xie, J. Zhang, and P. Wang, "Event-based stereo matching using semiglobal matching," *Int. J. Adv. Rob. Syst.*, vol. 15, no. 1, 2018.

[13] H. Rebecq, G. Gallego, D. Scaramuzza, and E. Müggler, "EMVS: Event-Based Multi-View Stereo - 3D Reconstruction with an Event Camera in Real-Time," *Int. J. Comput. Vision*, vol. 126, no. 12, pp. 1394–1414, 2018.

[14] H. Kim, S. Leutenegger, and A. Davison, "Real-time 3D reconstruction and 6-DOF tracking with an event camera," in *Proc. Europ. Conf. Comput. Vis.*, 2016, pp. 349–364.

[15] M. Gehrig, S. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-Based Angular Velocity Regression with Spiking Networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4195–4202.

[16] Y. Zhou, G. Gallego, and S. Shen, "Event-based Stereo Visual Odometry," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1433–1450, 2021.

[17] S. Ghosh and G. Gallego, "Multi-Event-Camera Depth Estimation and Outlier Rejection by Refocused Events Fusion," *Adv. Intell. Syst.*, vol. 4, no. 12, p. 2200221, 2022.

[18] X. Lagorce, G. Orchard, F. Galluppi, B. Shi, and R. Benosman, "HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1346–1359, 2017.

[19] Z. Liu, D. Shi, R. Li, Y. Zhang, and S. Yang, "T-ESVO: Improved Event-Based Stereo Visual Odometry via Adaptive Time-Surface and Truncated Signed Distance Function," *Adv. Intell. Syst.*, p. 2300027, 2023.

[20] A. Zhu, Y. Chen, and K. Daniilidis, "Realtime Time Synchronized Event-based Stereo," in *Proc. Europ. Conf. Comput. Vis.*, 2018, pp. 433–447.

[21] W. Guan, P. Chen, Y. Xie, and P. Lu, "PL-EVIO: Robust Monocular Event-based Visual Inertial Odometry with Point and Line Features," *arXiv:2209.12160v*, September 2022.

[22] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided Direct Sparse Odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5771–5780.

[23] R. Mur-Artal and J. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[24] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[25] E. Müggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *Proc. British Machine Vis. Conf.*, 2017, pp. 1–8.

[26] I. Alzugaray and M. Chli, "Asynchronous corner detection and tracking for event cameras in real time," *IEEE Rob. Autom. Lett.*, vol. 3, no. 4, pp. 3177–3184, 2018.

[27] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 703–710.

[28] G. Gallego, J. Lund, E. Müggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-Based, 6-DOF Camera Tracking from Photometric Depth Maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2402–2412, 2018.

[29] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[30] M. Shuster, "A Survey of Attitude Representations," *J. Astronaut. Sci.*, vol. 41, no. 4, pp. 439–517, 1993.

[31] R. Collins, "A space-sweep approach to true multi-image matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1996, pp. 358–363.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[33] K. Ta, D. Bruggemann, T. Brödermann, C. Sakaridis, and L. Van Gool, "L2E: Lasers to Events for 6-DoF Extrinsic Calibration of Lidars and Event Cameras," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11 425–11 431.