To appear in *Advanced Robotics*
Vol. 35, No. 5, March 2021, 1–16

FULL PAPER

Stereo Dense Depth Tracking Based on Optical Flow using Frames and Events

Antea Hadviger*, Ivan Marković, Ivan Petrović

University of Zagreb Faculty of Electrical Engineering and Computing
Laboratory for Autonomous Systems and Mobile Robotics (LAMOR)

(*v1.0 released April 2020*)

Event cameras are biologically inspired sensors that asynchronously detect brightness changes in the scene independently for each pixel. Their output is a stream of events which is reported with a low latency and high temporal resolution of a microsecond, making them superior to standard cameras in highly dynamic scenarios when they are sensitive to motion blur. Event cameras can be used in a wide range of applications, one of them being depth estimation, in both stereo and monocular settings. However, most known event-based depth estimation methods yield sparse depth maps due to the nature of the sparse event stream. We present a novel method that fuses information from both events and standard frames, as well as odometry, to exploit the advantages of both sensors. We propose to estimate dense disparity from standard frames at the point of their availability, predict the disparity using odometry information, and track the disparity asynchronously using optical flow of events between the standard frames. We present the performance of the method through several experiments in various setups, including synthetic data, KITTI dataset enhanced with events, MVSEC dataset, as well as our own stereo event camera recordings.

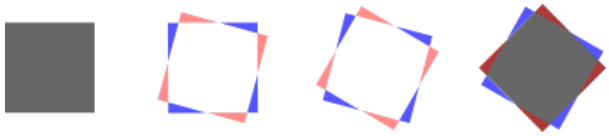
Keywords: event cameras, stereo vision, depth estimation

1. Introduction

Event cameras, such as the dynamic and active-pixel vision sensor (DAVIS) [1], are biologically inspired sensors that operate fundamentally different than standard cameras. They detect only changes in brightness intensity for each pixel of the scene independently and report them asynchronously in the form of *events*, immediately upon detection. An event $e_k = (\mathbf{x}_k, t_k, p_k)$ carries the information about the pixel location $\mathbf{x}_k = (x_k, y_k)^\top$, timestamp t_k, and polarity $p_k \in \{+1, -1\}$ of the brightness change, i.e., increase or decrease in brightness. Event cameras offer important advantages over standard cameras in a number of aspects. Since their output is a sparse event stream without redundant information, their power consumption and bandwidth requirements are lower. One of their most appealing features is their low latency and high temporal resolution of a microsecond, making them immune to motion blur, which standard cameras are prone to in highly dynamic scenarios. Furthermore, very high dynamic range of 140 dB, opposed to only 60 dB of standard cameras, makes event cameras a very useful sensor in situations of challenging illumination and environmental conditions, often encountered by autonomous mobile robots and vehicles. Most event cameras also provide standard image frames at a fixed frame rate.

A recent survey on event cameras offers an exhausting overview of all state-of-the-art event-based applications and methods [2]. They are widely used for all standard computer vision tasks,

*Corresponding author. Email: antea.hadviger@fer.hr



(a) frames and events (color indicates polarity)



(b) disparity with frames only (not tracked between frames)



(c) disparity with frames and events (proposed method)

Figure 1.: Images show data in four points in time (left to right) obtained by capturing a rotating dark grey square. First row shows raw data, i.e., frames (seen as dark grey square) and events (red and blue; color indicates polarity). Note that standard frames are available only at first and fourth timestamp. Second row shows disparity obtained with frames only, which changed only upon the arrival of the standard frame. The last row presents the benefits of our method, as disparity map is continuously changed, no matter if standard frames are available.

including depth (disparity) estimation. Depth estimation is usually tackled by using two synchronized stereo cameras and applying the standard stereo paradigm. The first step is to determine stereo correspondences between events, which is very challenging to do on a per-event basis due to lack of context and information that individually observed events contain. Most approaches match events by forming local context descriptors based on their spatial neighborhoods and temporal simultaneity, thus observing only events with most recent timestamps [3, 4]. Some works proposed to estimate event lifetimes prior to event aggregation in order to yield more accurate results [5, 6]. In order to get smoother depth maps, global approaches incorporate smoothness constraints across neighboring points, typically demonstrated in scenes with static cameras and few moving objects [7, 8]. Monocular depth estimation is appropriate for 3D reconstruction in visual odometry and SLAM [9–12], since this approach requires recovering information about the relative pose between the camera and the scene as it is not possible to adhere to the standard stereo paradigm. Similarly, a stereo method proposed in [13] uses the velocity of the camera to temporally synchronize events.

Most of the aforementioned methods produce sparse disparities, only at the pixels where events are present. The work in [14] attempts to recover semi-dense disparity by fusing information from multiple camera viewpoints, while [5] proposes to estimate dense disparity by performing interpolation in the areas uncovered by events. A first deep learning-based stereo approach that produces dense results was proposed in [15]. It showed significantly better performance than the other methods tested on the Multi Vehicle Stereo Event Camera Dataset [16], which is the most relevant benchmark for stereo event-based methods.

Naturally, event and standard cameras offer mutually complementary advantages which can be exploited simultaneously to build a robust and more accurate system. In one such attempt, the method in [17] detects features in standard frames and track them asynchronously using events for the purpose of visual odometry. To the best of our knowledge, our proposed method is the first to combine events and frames for the task of depth estimation. We suggest to use standard frames to perform dense disparity estimation with any standard stereo matching method, e.g., SGM [18], and use events to track the disparity asynchronously between the frames, an illus-

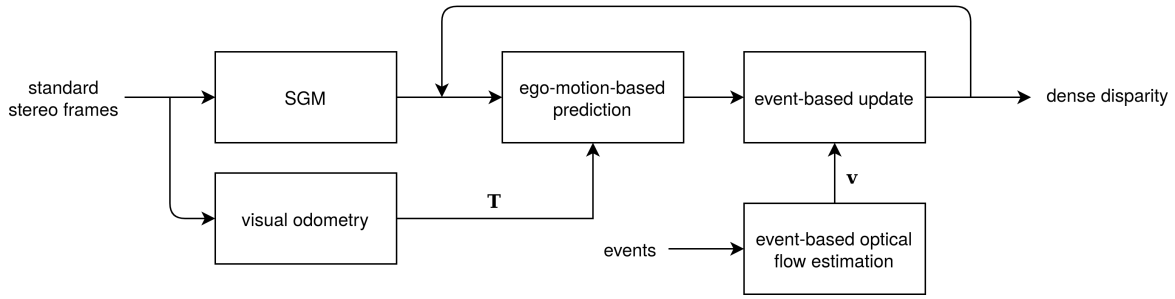


Figure 2.: Flow chart depicting the proposed method. Dense disparity is estimated using SGM from the standard frames, and then continuously updated using ego-motion and events.

tration of which is shown in Figure 1. Additionally, we propose to also use camera ego-motion to predict dense disparity between the frames in the areas of the image where no events are present. Such approach was used earlier in [19] where they exploit the previous frame to predict the disparity of the current frame. They suggest to perform standard stereo matching only for the pixels where prediction was unsucessful, therefore reducing the complexity of stereo matching. A similar method that rejects unreliable predictions in the areas where moving objects are detected was proposed in [20]. The work in [21] aims to reduce the disparity search space based on the prediction and use pixel-wise Kalman filtering to combine the predicted disparity and the newly matched one, whereas [22] extends the method by exploiting the difference between the predicted and matched disparity to detect moving objects. Our method relies on the most recent information on camera ego-motion and assumes constant motion in the subsequent interval between the frames to predict dense disparity at any point in time, while also performing asynchronous event-based updates. The proposed approach is validated through multiple experiments, including real-world and simulated data. The method performs in real-time in a single thread, with the ability to run our implementation of SGM multithreaded, making it suitable for implementation in autonomous systems that operate in highly dynamic scenarios or have significant computational constraints.

The rest of the paper is organized as follows. Our novel method for dense depth map estimation by using events and frames is presented in Section 2. In Section 3 we provide a brief overview the event-based optical flow method proposed in [23], since it constitutes an important part of our method. Finally, we present the experimental results in Section 4 and conclude with final remarks in Section 5.

2. Dense Depth Estimation and Tracking

Disparity maps obtained from standard frames are available only at fixed points in time, depending on the frame rate of the camera, thus essentially resulting in blind intervals between the frames. In this section we propose a method for estimating dense disparity maps that relies on both standard frames and events, as well as the camera ego-motion. We suggest to estimate dense disparity maps from standard frames at the points of their availability, and predict the disparity at any point in time between the frames by using the ego-motion information. To enable prediction of the moving objects' disparity, where the prediction based on ego-motion would be invalid, we suggest to exploit the events to track the disparity map asynchronously and instantaneously at the time of arrival of each event. This means that dense disparity maps are readily available at any point in time between the standard frames. Figure 2 shows the diagram of the proposed method.

2.1 *Frame-based Disparity Estimation*

Our method is invariant to the choice of a dense frame-based disparity estimation algorithm. It can work in both stereo and monocular setup. However, since events are mostly generated on boundaries of image segments, it is desirable that the chosen frame-based disparity estimation method yields sharp and precise results in these areas, i.e., at disparity discontinuities. Otherwise, updating the disparity using events would not be effective. We used our implementation of semi-global matching (SGM) for a stereo setup presented in [22], which supports AVX2 instructions and multithreading to support real-time performance. This algorithm aggregates stereo matching costs along several linear paths as follows:

$$L_{\mathbf{r}}(\mathbf{p},d) = C(\mathbf{p},d) + \min \begin{cases} L(\mathbf{p}-\mathbf{r},d) \\ L(\mathbf{p}-\mathbf{r},d\pm 1) + P_1 \\ \min_i L(\mathbf{p}-\mathbf{r},d+i) + P_2 \end{cases}, \tag{1}$$

where $\mathbf{p}$ denotes the pixel location and d is the potential disparity. $L(\mathbf{p},d)$ is the accumulated loss along direction $\mathbf{r}$, and $C(\mathbf{p},d)$ is the matching cost between the referent image patch and the considered potential match. The final accumulated cost is the sum of all costs $L_{\mathbf{r}}(\mathbf{p},d)$. In our implementation, we use the matching cost based on the census transform. SGM introduces parameters P_1 and P_2 as discontinuity penalties in order to obtain smooth disparity maps. P_1 is the penalty for a disparity change of 1 pixel, while P_2 penalizes any disparity change i larger than 1. This, however, consequently leads to loss of precision on disparity boundaries as the algorithm resists to change the disparity, which is unfavorable for our method. For this reason we chose smaller values of P_1 and P_2 than it is usually recommended, resulting in better discontinuity sharpness, but at the price of less smoothness.

2.2 *Odometry-based Disparity Prediction*

In order to predict the dense disparity for any timestamp between the frames, it is required to determine the ego-motion of the camera. In all our experiments, we used the ground truth odometry provided in the datasets, but any ego-motion estimation method can be used. We assume constant motion of the camera in the whole interval until the next camera frame arrives, i.e., until the arrival of new ego-motion information. Thus, we use the most recent ego-motion information as a prediction for the future camera motion. Disparity prediction can be done at an arbitrary frequency, depending on the available computational resources and required temporal resolution of dense disparity availability, It can depend on the event rate or the velocity itself, but it can be fixed as well. In any case, prediction is based on the dense disparity that has been continuously updated using events.

Let $\mathbf{T}_k^{k+1}$ represent the predicted homogenous transformation (rotation and translation) from the next coordinate frame of the left camera at the time t_{k+1} to the coordinate frame of the previous dense disparity prediction (or estimation based on frames) at the time t_k: We obtain this transformation by multiplying the last known camera velocity with the interval since the last known dense disparity $\Delta t = t_{k+1} - t_k$. Furthermore, let $\omega = (x, y, d)^\top$ represent the homogenous coordinates in the disparity space centered in the image, and let $\mathbf{M} = (X, Y, Z)^\top$ represent the coordinates in the Euclidean space. Disparity prediction is given by the following equation:

$$\begin{bmatrix} \omega^{k+1|k} \\ 1 \end{bmatrix} = \mathbf{H}_k^{k+1} \begin{bmatrix} \omega^k \\ 1 \end{bmatrix}, \tag{2}$$

Herein, $\mathbf{H}_k^{k+1}$ is the 4×4 transformation matrix defined as:

$$\mathbf{H}_k^{k+1} = \mathbf{\Gamma}\mathbf{T}_k^{k+1}\mathbf{\Gamma}^{-1}, \tag{3}$$

where $\mathbf{\Gamma}$ is the projection matrix that transforms the coordinates ω between the disparity space and the coordinates $\mathbf{M}$ in the Euclidean space as follows:

$$\begin{bmatrix}\omega\\1\end{bmatrix} \simeq \mathbf{\Gamma}\begin{bmatrix}\mathbf{M}\\1\end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix}f & 0 & 0 & 0\\0 & f & 0 & 0\\0 & 0 & 0 & fb\\0 & 0 & 1 & 0\end{bmatrix} \tag{4}$$

with f and b being the focal length of the cameras and the baseline of the stereo rig, respectively [24].

When the camera is moving forward, the predicted disparity map D is incomplete due to the *zooming effect*. We attempt to fill in the disparity D for the pixels $\mathbf{p} = (x, y)^\top$ whose disparity could not be predicted by defining their value as the average of two of their closest horizontal or vertical neighbors:

$$D(x,y) = \begin{cases} \frac{1}{2}(D(x-1,y) + D(x+1,y)), & \text{if } |D(x-1,y) - D(x+1,y)| < \gamma \\ \frac{1}{2}(D(x,y-1) + D(x,y+1)), & \text{if } |D(x,y-1) - D(x,y+1)| < \gamma \\ \text{invalid}, & \text{otherwise} \end{cases}. \tag{5}$$

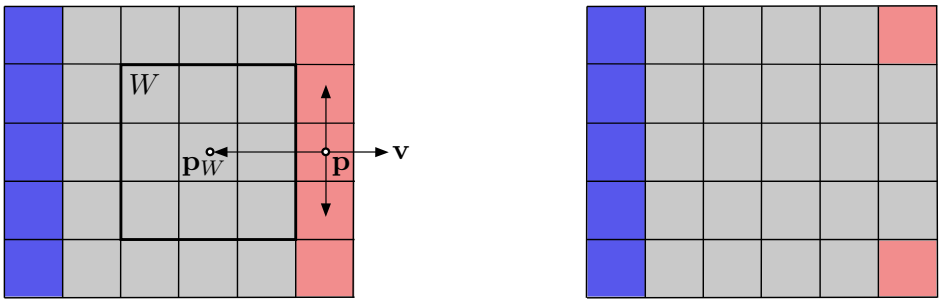
However, we only use the average value to fill in the missing disparities if the neighboring disparities are similar enough, up to a certain threshold value γ. This way we ensure that we do not use an incorrect value to fill in the missing disparities appearing at the disparity discontinuities.

2.3 *Event-based Disparity Tracking*

If an event occurs at a certain pixel, this is an indication that the disparity at the given pixel might have changed. Since events are triggered by motion (translational and/or rotational), they are present only at the moving object's edges, and not the whole surface of the objects, if they are not textured. Thus, our method works best if the object moves parallel to the image plane, i.e., does not change depth, due to not being able to estimate depth of the moving objects in the areas where no events are present. Although this assumption is a certain limitation of our method, it is a good starting point for developing more robust methods for real-world situations, by considering image segmentation and moving object detection to enable disparity prediction for the entire areas of the moving objects.

For illustration of the method, let us observe a homogenous, non-textured object moving through a plane parallel to the image plane on a homogenous background without camera motion, as depicted in Figure 3. In this case, events are caused by changes in brightness subsequent to two different changes in object position: object leaving the space it was covering before (depicted in blue) and object arriving into the space it has not covered before (depicted in red). It is important to notice that, in this case, red events should have different disparities than the blue events, since the former are located on the area covered by the object in the foreground, which is closer to the camera, and the latter are located in the area containing the background of the image. Therefore, we need to associate an event to the segment of the image that triggered it, in order to track its disparity correctly. Since events do not carry enough information to infer this, we need to use past standard frames and their disparity map as well.

If a brightness change is moving through the scene, it triggers events in different pixels at different points in time. Therefore, these events correspond to the same image segment, and should thus share the same disparity. If we can estimate the location of the previous event triggered by the same brightness change as the current event, i.e., by the same movement in the scene, then we can associate the disparity of that previous event to the current one. We propose to find the location of the previous event by leveraging optical flow, which is estimated



(a) disparity and events (b) tracked disparity

Figure 3.: Dense disparity tracking using events. (a) shows the disparity and events (red and blue) for a block moving towards right. For the event at the pixel $\mathbf{p}$, we estimate its optical flow $\mathbf{v}$ and update the disparity at its location by leveraging disparities in the window W located in the direction opposite to the optical flow vector. As seen in (b), we update the two disparities located at the pixels perpendicular to the optical flow as well.

as explained later on in Section 3. More specifically, if we take the position of the current event and move opposite from the direction of the optical flow vector for a displacement of one pixel, we will land at the location of the event previously triggered by the same brightness change. Formally, for the event position $\mathbf{p}$, this can be expressed as:

$$\mathbf{p}_{\text{prev}} = \mathbf{p} - \hat{\mathbf{v}}, \tag{6}$$

where $\mathbf{v} = (v_x, v_y)^\top$ is the optical flow vector, and $\hat{\mathbf{v}}$ is the corresponding unit vector.

To determine the disparity $D(\mathbf{p})$ at the current event's location and asynchronously update the dense disparity map, we propose to form a spatial window W around the pixel $\mathbf{p}_W$ and set the value of the disparity as the median of the previously estimated disparities in the window W (Figure 3):

$$D(\mathbf{p}) = \text{median } \{D(\mathbf{q}), \mathbf{q} \in W\}. \tag{7}$$

The disparity map D has continuously been tracked since its original values were obtained using frames, meaning that some values might have already changed in the described manner using the events. Position $\mathbf{p}_W$ for the window size $(2w-1) \times (2w-1)$ is determined as:

$$\mathbf{p}_W = \mathbf{p} - w\hat{\mathbf{v}}, \tag{8}$$

and rounded to the nearest integer coordinates. Using the median is a more robust approach as opposed to just considering the single disparity of the previous event. It also ensures that the determined disparity value will be the same as some already present in the disparity map, which would not be the case if we used, e.g., averaging. We propose to use a square 3×3 window W, as shown in Figure 3, to ensure robustness to noise, but still keep the computational load small. Size and shape of the window can be adapted if needed, and even contain only one pixel at $\mathbf{p}_{\text{prev}}$ if optical flow is very accurate.

Since events are most commonly caused by moving edges, it is safe to assume that events should appear along a line perpendicular to the optical flow of the moving edge. Therefore, apart from the position of the incoming event, we can also change the disparity at the neighboring pixels located along the direction perpendicular to the event's optical flow, for a displacement of one pixel in each direction (Figure 3). This is achieved in the following manner:

$$D(\mathbf{p} + \hat{\mathbf{v}}_p) = D(\mathbf{p}), \tag{9}$$

$$D(\mathbf{p} - \hat{\mathbf{v}}_p) = D(\mathbf{p}), \tag{10}$$

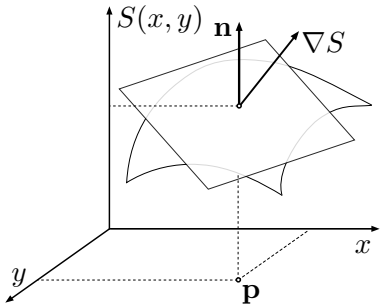


Figure 4.: Event-based optical flow estimation using the planar approximation of the surface of active events S.

where $\mathbf{v}_p$ is equal to $(-v_y, v_x)^\top$, which is the vector perpendicular to the optical flow vector v, and $\hat{\mathbf{v}}_p$ is the corresponding unit vector. $D(\mathbf{p})$ is already computed according to (7). However, we suggest to perform these additional disparity updates only if the event stream is too sparse to update depth or estimate optical flow correctly. If the event stream is dense enough, then there is no need to do the additional updates as it can have a negative effect on accuracy. We do not employ any form of postprocessing.

3. Event-based Optical Flow

By leveraging optical flow of each event, we aim to estimate the location of the previous event triggered by the same brightness change as the current event, which in turn makes us able to track the dense disparity map, as explained in Section 2. In this section we provide a brief overview of the event-based optical flow estimation method originally proposed in [23], as it plays an important role in our method (Figure 4).

3.1 *Method Overview*

A data structure in the spatio-temporal domain that maps the coordinates of each pixel of the image to the timestamp of the last event that occurred at the given position is called a surface of active events (SAE). We can formally define SAE as a function $S : \mathbb{R}^2 \to \mathbb{R}, S(p) = t$, where $\mathbf{p} = (x, y)^\top$ are the coordinates of the pixel and t is the timestamp of the last corresponding event.

The planar approximation of the SAE at the position $\mathbf{p}$ corresponds to its first order Taylor expansion:

$$S(\mathbf{p} + \Delta\mathbf{p}) \approx S(\mathbf{p}) + (S_x(\mathbf{p}), S_y(\mathbf{p}))\, \Delta\mathbf{p}, \tag{11}$$

where S_x and S_y denote the first partial derivatives, $\frac{\partial S}{\partial x}$ and $\frac{\partial S}{\partial y}$, respectively. Given that time t is a naturally increasing quantity, S is also a monotonically increasing function, which allows us to use the inverse function theorem around the position $\mathbf{p}$:

$$\frac{\partial S}{\partial x}(x, y_0) = \frac{1}{v_x(x, y_0)}, \frac{\partial S}{\partial y}(x_0, y) = \frac{1}{v_y(x_0, y)}, \tag{12}$$

showing the relation of the gradient to the optical flow velocities v_x and v_y. The gradient of S can now be expressed as:

$$\nabla S(\mathbf{p}) = \left(\frac{1}{v_x(\mathbf{p})}, \frac{1}{v_y(\mathbf{p})}\right)^\top. \tag{13}$$

The normal $\mathbf{n}(\mathbf{p}) = (n_1, n_2, n_3)^\top$ of the tangent plane to the SAE at the point $\mathbf{p}$ is given by:

$$\mathbf{n}(\mathbf{p}) \propto (1, 0, S_x(\mathbf{p}))^\top \times (0, 1, S_y(\mathbf{p}))^\top = (-S_x(\mathbf{p}), -S_y(\mathbf{p}), 1)^\top, \tag{14}$$

which yields:

$$\mathbf{n}(\mathbf{p}) \ \propto \ (-v_x^{-1}, -v_y^{-1}, 1)^\top. \tag{15}$$

Finally, we are able to express the optical flow velocities as:

$$v_x = -\frac{n_3}{n_1}, v_y = -\frac{n_3}{n_2}. \tag{16}$$

In the sequel, we explain the plane-fitting algorithm used to obtain the tangent plane parameters.

3.2 *Local Plane-fitting Algorithm*

Since the sensor is imperfect and prone to noise, parameters of the local planes tangent to the SAE are estimated iteratively using RANSAC to assure robustness to outliers and filter out noise. We consider only the past events in the $N \times N$ spatial window surrounding the current event, as proposed in [25], to avoid latency that would be introduced by considering future events as well.

The plane is first fitted to the current event and the two random past events in the aforementioned spatio-temporal window, which we consider hypothetical inliers. Then, we iterate through all other events in the window and proclaim them as inliers if their distance to the plane is below a certain threshold. If there were not enough inliers found, a new random set of events is chosen in the first step and the process is repeated. However, in case that the minimum number of inliers is not found in a certain number of attempts, the current event is proclaimed as noise. In the end, least squares minimization is used to fit a plane to all the inliers:

$$\mathbf{n}_{\text{LS}} = \arg\min_{\mathbf{n}} \|\mathbf{A}\mathbf{n} - \mathbf{b}\|^2, \tag{17}$$

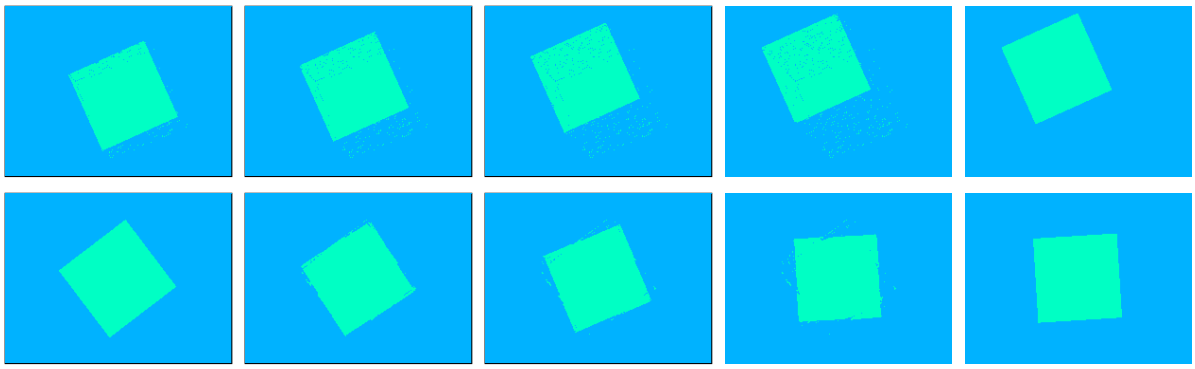
where $\mathbf{A}$ is the $n \times 3$ matrix of the n points of the SAE (x_i, y_i, t_i) identified as inliers, and $\mathbf{b} = (1 \cdots 1)^\top$.

4. Experimental Results

In this section we will demonstrate the results yielded by our method through several experiments. Experiments were run on a personal computer with Intel Core i7-7700HQ CPU at 2.80GHz on a single core. The method is implemented in C++ and adapted for the ROS environment [26]. The execution time of the method depends on the event rate. It takes approximately 5 μs to process a single event. Since our method depends only on the result of frame-based depth estimation, and not the specific method, in experiments in Sections 4.1 and 4.2, we demonstrate the results obtained by using ground truth depth maps instead of actually performing frame-based depth estimation.

4.1 *ESIM Experiments*

Experiments were conducted on the data obtained with ESIM: an Open Event Camera Simulator [27]. Since the simulator provides accurate ground truth depth maps, we used it to illustrate the benefits of our method that are most evident when sharp and precise dense disparity frames



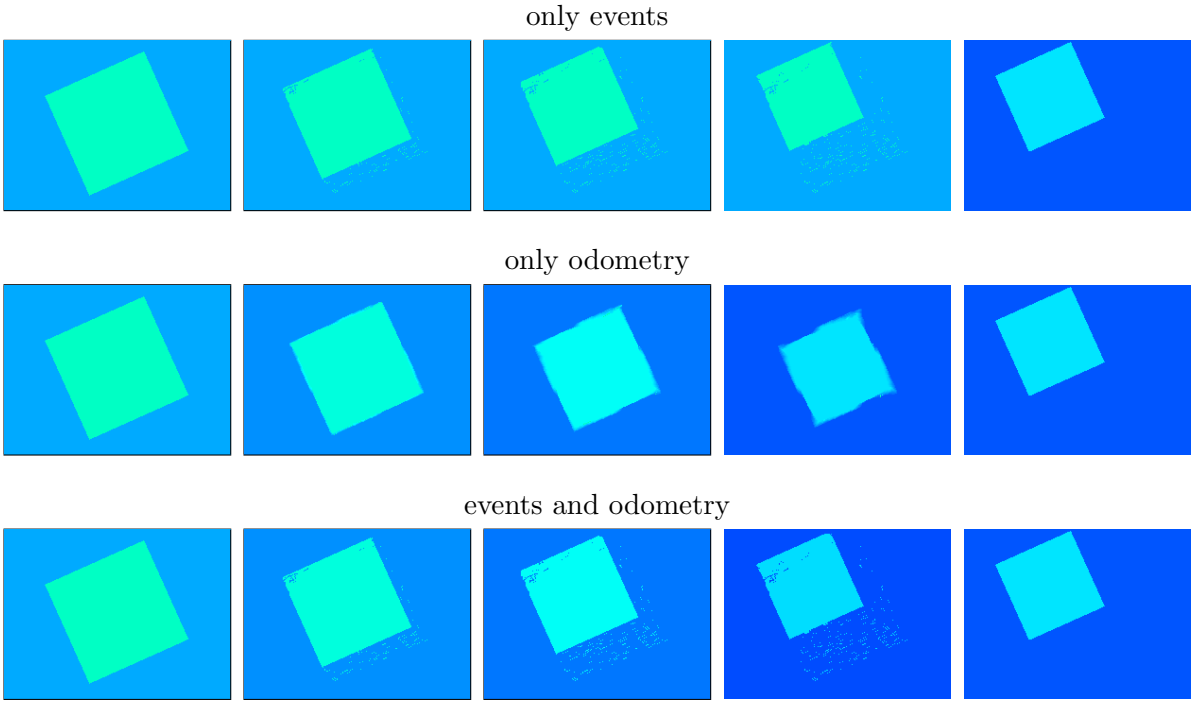
(a) GT at $t = t_1$ (b) $t = t_1 + \tau$ (c) $t = t_1 + 2\tau$ (d) $t = t_2$ (e) GT at $t = t_2$

Figure 5.: Experiments 1 and 2 with simulated data. The object is moving parallel to the ground plane, while the camera is still. The first and second row depict translational and rotational motion, respectively. First, (a) are the dense depth maps obtained from the simulator. Next, (b) and (c) show the depth maps tracked using events inbetween the camera frames. Finally, (d) shows the final tracked depth map at the point of arrival of the new ground truth map (seen in (e)), with the starting square position outlined to better illustrate the movement. Greener shade indicates smaller depth.

are available. As an input, the simulator takes a description of the scene 3D model, as well as the desired camera trajectory. The scene used in our experiments is a simple dark-colored block positioned on a light-colored plane. All surfaces are non-textured in order to eliminate redundant events. Experiments are set up in a way that the camera image plane is parallel to the ground plane. The simulator does not support moving objects in the scene. Since we need at least one object in the scene to move in order to make use of the event data, we chose to discard the x and y components of the camera velocity (set them to 0) and keep only the z component. Effectively, this results in a scenario where the object is moving parallel to the ground plane, and the camera is moving perpendicular to the ground plane. Dense depth maps obtained in four experiments are shown in Figs. 5, 6, and 7. Therein, first and last column show the ground truth depth maps provided by the simulator, available at the timestamps t_1 and t_2 (camera frame rate), while the other columns show the depth maps inbetween standard frames.

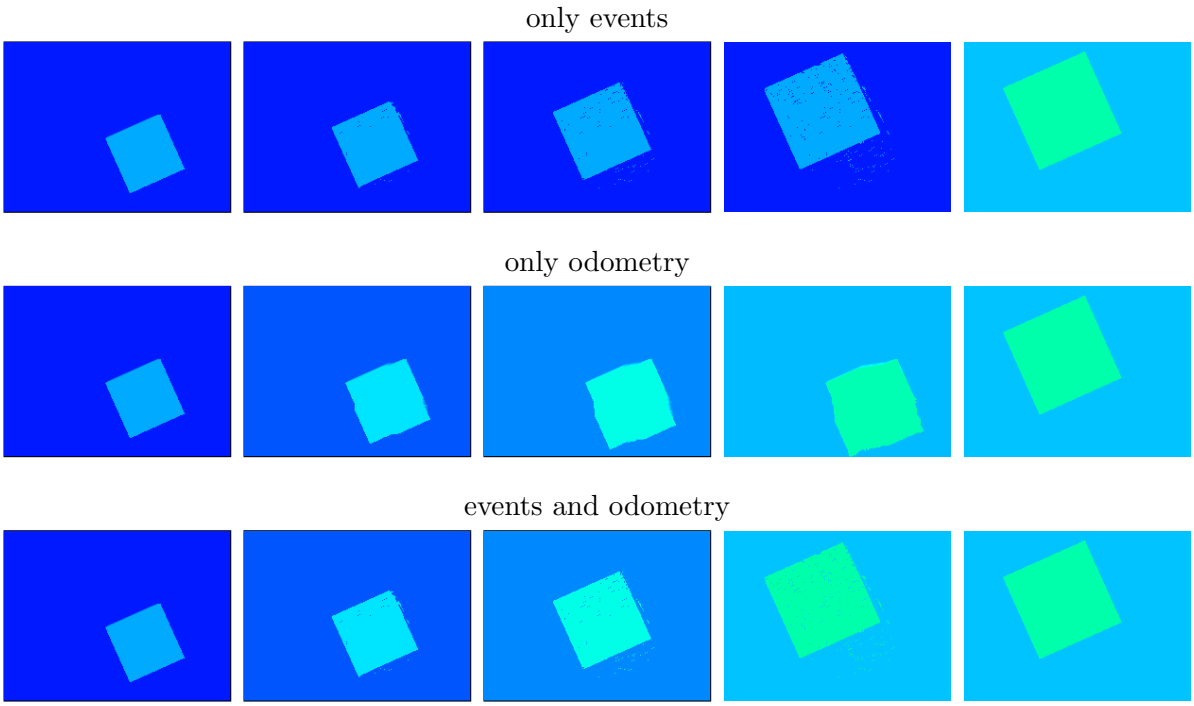
Experiments 1 and 2 (Figure 5) shows very good performance of the method in case when the object is moving parallel to the ground plane, either translating or rotating, while the camera is static; hence, depth of the objects is unchanged. The same results are expected if the object is static and the camera is moving parallel to the image plane.

In Figure 6 and Figure 7, we illustrate the benefit of using the ego-motion information. These experiments present the scenario where the camera is moving perpendicular to the ground plane and the object is moving parallel to the ground plane. The first row of the images shows the results obtained by using only events, while the second row shows the output with the ego-motion information taken into account. Since in these experiments the camera is moving upwards/downwards, the depth of the whole scene changes, which is not possible to track using only events that are present only at the edges of the objects, as shown in the first row of the experiment images. The second row shows what happens when events are not used. As it is not possible to track object movement using only odometry, the disparities in (d) and (e) are shifted in x and y directions. In the images in the third row, both odometry and events are used. Depth is continuously changing in the whole scene, as opposed to only in those areas where the events are present, and object movement is tracked correctly. However, using ego-motion would not enable us to predict the depth of the moving objects in case that they were moving perpendicular to the ground plane, which we are not able to show due to the limitations of the simulator. Our result would still be more informative than it would be by using only the standard frames as the depth at the objects' edges where the events are present would be closer to reality.



(a) GT at $t = t_1$ (b) $t = t_1 + \tau$ (c) $t = t_1 + 2\tau$ (d) $t = t_2$ (e) GT at $t = t_2$

Figure 6.: Experiment 3 with simulated data. The object is moving parallel to the ground plane, while the camera is moving upwards, perpendicular to the ground plane. Lighter and greener color indicates smaller depth.



(a) GT at $t = t_1$ (b) $t = t_1 + \tau$ (c) $t = t_1 + 2\tau$ (d) $t = t_2$ (e) GT at $t = t_2$

Figure 7.: Experiment 4 with simulated data. The object is moving parallel to the ground plane, while the camera is moving downwards, perpendicular to the ground plane.

4.1.1 Quantitative Evaluation

For the experiments with simulated data, we are able to provide quantitative evaluation due to availablity of the ground truth depth. Results are shown in Table 1. Outliers are mostly present in the form of noise, caused by the inability to estimate optical flow correctly. All experiments were performed without any postprocessing, which would certainly help reduce the problem of noise and improve accuracy. As expected, the number of outliers accumulates over time, but accuracy is still very high.

		Events + Odometry (%)			Odometry (%)			
Experiment	Figure	$t_1+\tau$	$t_1+2\tau$	t_2	$t_1+\tau$	$t_1+2\tau$	t_2	Events
1 (translation)	Figure 5	1.15	1.37	1.70	6.88	13.19	18.95	152328
2 (rotation)	Figure 5	0.67	0.72	0.80	11.36	17.24	21.72	54385
3 (upwards)	Figure 6	1.02	1.20	1.33	26.78	40.21	49.92	133743
4 (downwards)	Figure 7	0.55	0.82	1.27	9.07	19.24	32.75	164216

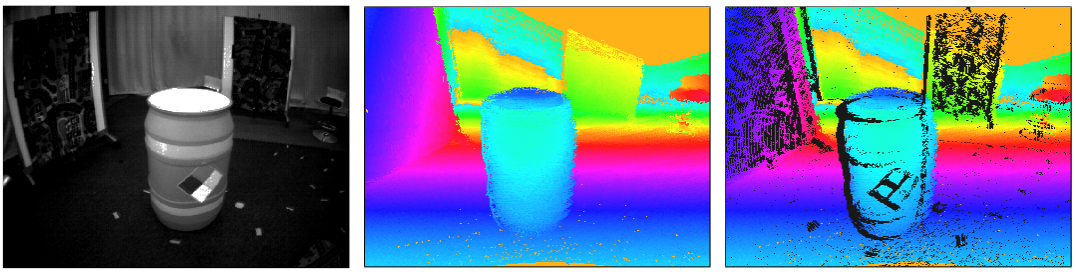
Table 1.: Quantitative evaluation of four simulated experiments. Outliers are defined as pixels whose estimated depth differs from ground truth by more than 5 %. Accuracy of the proposed method (using events and odometry) is significantly better than the accuracy obtained by using only odometry. Total number of events between t_1 and t_2 is shown in the last column.

4.2 *MVSEC Experiments*

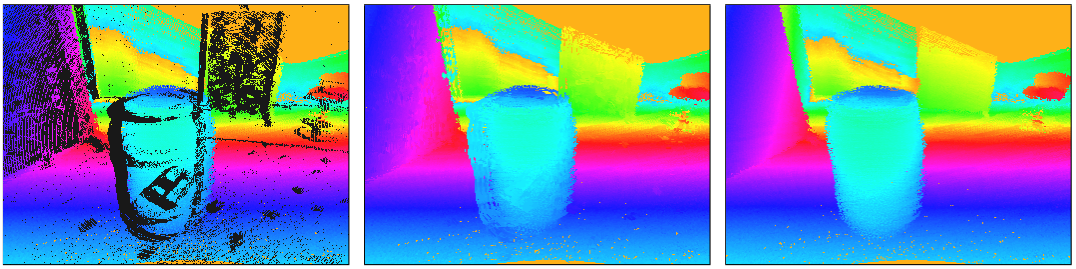
In another set of experiments, we used the publicly available Multi Vehicle Stereo Event Camera Dataset (MVSEC) [16]. This dataset provides multiple stereo event sequences collected with two DAVIS346 cameras, including ground truth information about pose and depth obtained by lidar SLAM. Extrinsic calibration parameters that transform lidar to camera frame are also available. However, due to imperfect calibration and lack of hardware timestamp synchronization between the lidar and the camera, which authors of the dataset tried to overcome with manual fine tuning, events are still not properly overlapping with the corresponding disparity at times. This dataset does not include moving objects, so we discard the odometry information and use only events for disparity updates, effectively treating every object in the scene as a moving object. Since precise dense disparity maps are the important prerequisite for our method, we chose those dataset intervals in which the overlapping is satisfying at least in some parts of the scene, as per visual evaluation. As in Section 4.1, we did not perform depth estimation on the standard frames, but we used the ground truth depth instead. The results can be seen in Figure 8, where we also show raw event data in (b) and (c). It is clear that the misalignment of events and edges in the depth map leads to inaccuracy in disparity tracking, which is mostly visible when observing the right edge of the object in the foreground. Since the events on the left edge are correctly aligned with the depth map, the tracked disparity is also accurate in this area.

4.3 *KITTI Experiments*

To demonstrate the performance of our method in a real-world scenario, we used the well-known KITTI dataset [28]. Since the dataset is recorded using only standard cameras, it does not include events. However, we generated synthetic event data using the open-source *Video to Events* tool provided by [29]. Prior to synthesizing events, the standard frames of the KITTI dataset that were captured with the frequency of 10 Hz were upsampled with a frame interpolation package (also available in the said tool) based on [30] to achieve better results. In the experiment in Figure 9, the camera is still and the bus is moving towards left. Dense disparity was estimated using SGM, shown in (a) and (f). Disparity is tracked between the frames using only events, since no camera movement is present. Final tracked disparity in (e) almost completely aligns

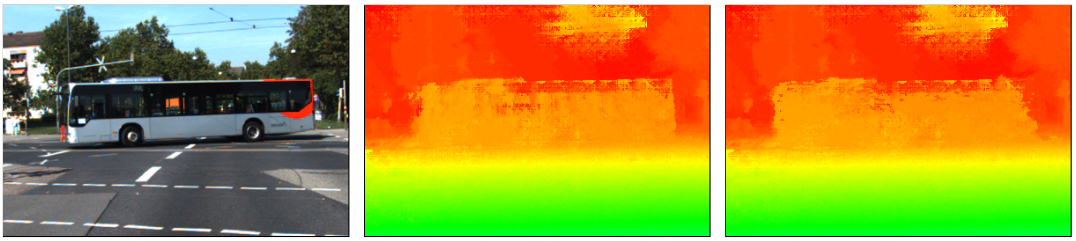


(a) grayscale (b) GT at $t = t_1$ (c) $t = t_1 + \tau$

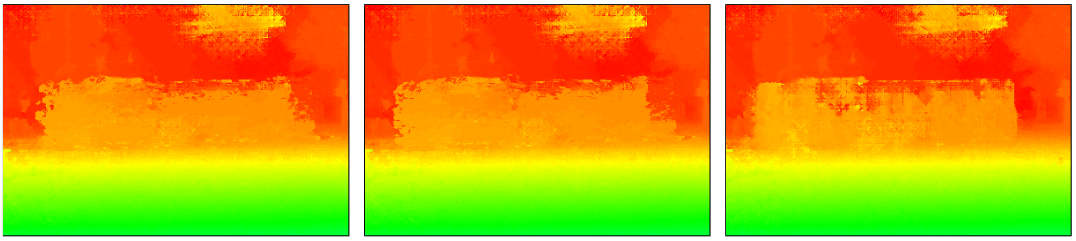


(d) $t = t_1 + 2\tau$ (e) $t = t_2$ (f) GT at $t = t_2$

Figure 8.: Experiment with a sequence from the MVSEC dataset. The five images represent the same data as seen in Figure 5, except that (b) and (c) do not show the updated disparity, but aggregated raw events (in black) to illustrate the misalignment of events and depth map. This, in turn, results with a tracked depth map different than the ground truth in the area where events were misaligned (right edge of the object).



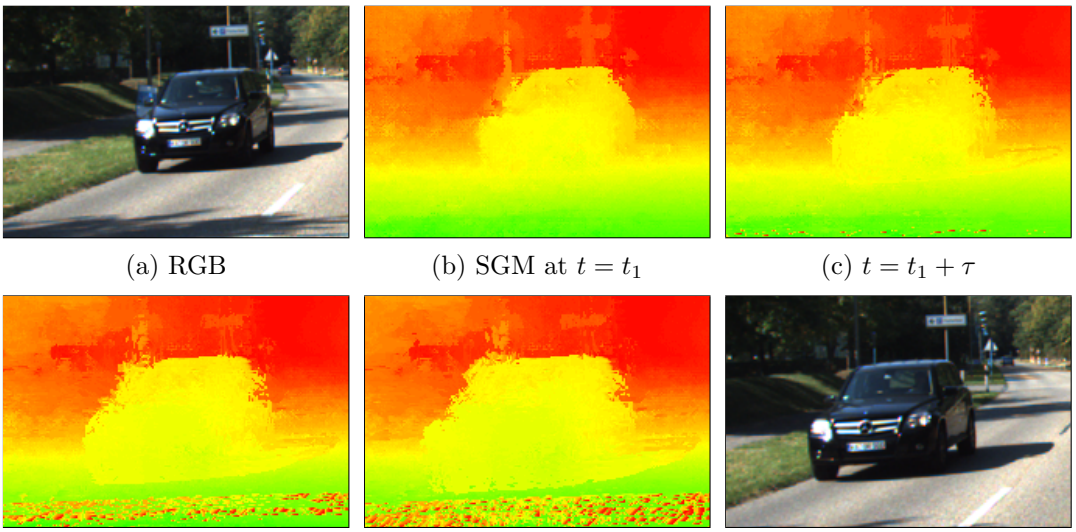
(a) RGB (b) SGM at $t = t_1$ (c) $t = t_1 + \tau$



(d) $t = t_1 + 2\tau$ (e) $t = t_2$ (f) SGM at $t = t_2$

Figure 9.: Experiment 1 with a sequence from the KITTI dataset. The standard RGB frame is seen in (a), while (b) and (f) are the dense disparity maps obtained with SGM. Disparity tracked by events is seen in (c) through (e).

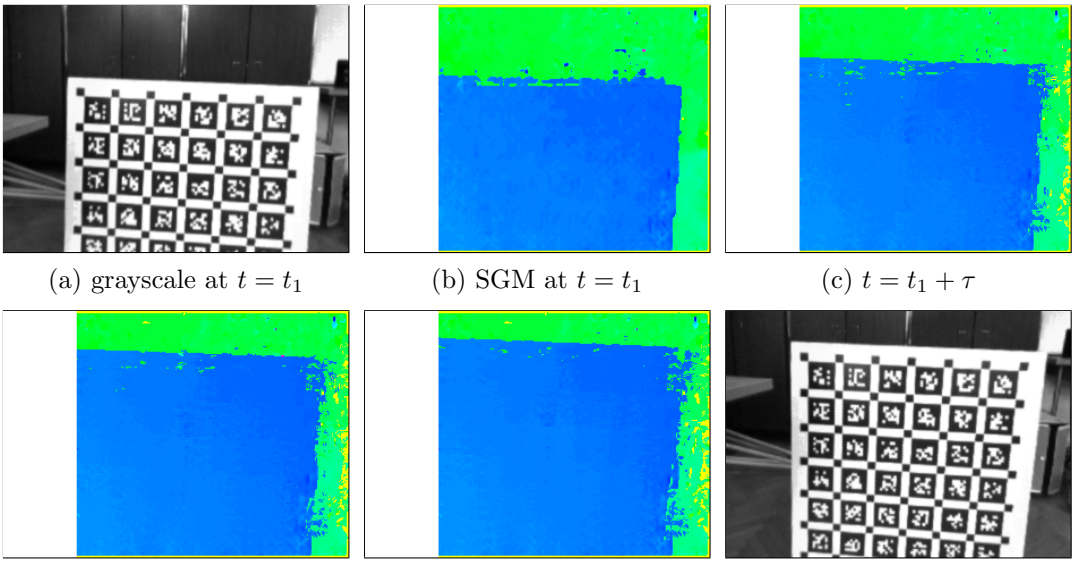
with the dense disparity at the same timestamp, apart from the bottom right corner of the bus where no events are generated due to the intensity of the pixels of the bus being almost the same as the intensity of the background. The second experiment in Figure 10 shows a scene where the car is approaching while the camera is moving forwards. Disparity is updated using both events and odometry. Final tracked disparity reflects the movement of the approaching car, as well as the camera. Invalid disparities in the road area (seen as red) are the result of the zooming effect.



(a) RGB (b) SGM at $t = t_1$ (c) $t = t_1 + \tau$

(d) $t = t_1 + 2\tau$ (e) $t = t_2$ (f) RGB at $t = t_2$

Figure 10.: Experiment 2 with a sequence from the KITTI dataset. The standard RGB frames are seen in (a) and (f), while (b) is the dense disparity map obtained with SGM. Disparity updated using events and odometry, as seen in (c) through (e).



(a) grayscale at $t = t_1$ (b) SGM at $t = t_1$ (c) $t = t_1 + \tau$

(d) $t = t_1 + 2\tau$ (e) $t = t_2$ (f) grayscale at $t = t_2$

Figure 11.: Experiment with a sequence obtained with two stereo DAVIS346 sensors. (a) and (f) are showing the standard frames, while (b) is the dense depth obtained with SGM. Disparity tracked by events is seen in (c) through (e). White area in the depth maps is the part of the scene which is not visible with both cameras.

4.4 *Stereo DAVIS346 Experiments*

Experimental data was collected with two DAVIS346 sensors mounted on a stereo rig with a baseline of 10 cm. The sensor provides both standard frames and event data with a spatial resolution of 346×280 pixels. Stereo rectification parameters were determined using the Kalibr toolbox [31] by considering only standard frames. Frame-based disparity was estimated with semi-global matching (see Section 2.1). The sensors are observing a scene with an april tag board set in the foreground, as seen in Figure 11, and the camera is moving downwards. We have lowered the frame rate of the cameras to better illustrate the results of our method. Even though the

interval between the standard frames is fairly long (almost 1 s) and the scene differs significantly, depth was successfully tracked between the frames using events, without any postprocessing.

5. Conclusion

In this paper, we have presented a novel method for dense disparity estimation fuses information from standard frames, events and odometry. Since standard frames are only available at discrete timestamps, we have proposed to predict the dense disparity maps using odometry information, and use events to asynchronously track the disparity based on optical flow of events, making dense disparity available at any point in time. We have demonstrated the performance of the method through several experiments that cover different scenarios, including synthetic data from the event simulator, KITTI dataset enhanced with events, MVSEC dataset, and our own stereo DAVIS recordings. Future work will aim at enhancing performance of the method in real-world scenarios, such as autonomous driving, by detecting moving objects in the scene and ensuring robustness to noise in the event data.

Acknowledgment

This work has been supported by the FLAG-ERA JTC 2016 and the Ministry of Science and Education of the Republic of Croatia under the project Rethinking Robotics for the Robot Companion of the Future (RoboCom++) and the Croatian Science Foundation under contract No. I-3137-2019. The research has been carried out within the activities of the Centre of Research Excellence for Data Science and Cooperative Systems supported by the Ministry of Science and Education of the Republic of Croatia.

References

[1] Brandli C, Berner R, Yang M, Liu SC, Delbruck T. A 240× 180 130 db 3 μs Latency Global Shutter Spatiotemporal Vision Sensor. IEEE Journal of Solid-State Circuits. 2014;49(10):2333–2341.

[2] Gallego G, Delbruck T, Orchard G, Bartolozzi C, Taba B, Censi A, Leutenegger S, Davison A, Conradt J, Daniilidis K, et al.. Event-based Vision: A Survey. arXiv preprint arXiv:190408405. 2019;.

[3] Kogler J, Humenberger M, Sulzbachner C. Event-based Stereo Matching Approaches for Frameless Address Event Stereo Data. In: International Symposium on Visual Computing. Springer. 2011. p. 674–685.

[4] Zou D, Guo P, Wang Q, Wang X, Shao G, Shi F, Li J, Park PK. Context-aware Event-driven Stereo Matching. In: 2016 IEEE International Conference on Image Processing (ICIP). IEEE. 2016. p. 1076–1080.

[5] Zou D, Shi F, Liu W, Li J, Wang Q, Park PK, Shi CW, Roh YJ, Ryu HE. Robust dense depth map estimation from sparse DVS stereos. In: British Machine Vis. Conf.(BMVC). Vol. 3. 2017.

[6] Hadviger A, Marković I, Petrović I. Stereo Event Lifetime and Disparity Estimation for Dynamic Vision Sensors. In: 2019 European Conference on Mobile Robots (ECMR). IEEE. 2019. p. 1–6.

[7] Piatkowska E, Belbachir A, Gelautz M. Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. 2013. p. 45–50.

[8] Xie Z, Zhang J, Wang P. Event-based stereo matching using semiglobal matching. International Journal of Advanced Robotic Systems. 2018;15(1):1729881417752759.

[9] Kim H, Leutenegger S, Davison AJ. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In: European Conference on Computer Vision. Springer. 2016. p. 349–364.

[10] Rebecq H, Gallego G, Mueggler E, Scaramuzza D. EMVS: Event-based multi-view stereo3D reconstruction with an event camera in real-time. International Journal of Computer Vision. 2018; 126(12):1394–1414.

[11] Rebecq H, Horstschäfer T, Gallego G, Scaramuzza D. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. IEEE Robotics and Automation Letters. 2016; 2(2):593–600.
[12] Gallego G, Rebecq H, Scaramuzza D. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 3867–3876.
[13] Zhu AZ, Chen Y, Daniilidis K. Realtime time synchronized event-based stereo. In: European Conference on Computer Vision. Springer. 2018. p. 438–452.
[14] Zhou Y, Gallego G, Rebecq H, Kneip L, Li H, Scaramuzza D. Semi-dense 3D reconstruction with a stereo event camera. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018. p. 235–251.
[15] Tulyakov S, Fleuret F, Kiefel M, Gehler P, Hirsch M. Learning an event sequence embedding for dense event-based deep stereo. In: Proceedings of the IEEE International Conference on Computer Vision. 2019. p. 1527–1537.
[16] Zhu AZ, Thakur D, Özaslan T, Pfrommer B, Kumar V, Daniilidis K. The Multi Vehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. IEEE Robotics and Automation Letters. 2018;3(3):2032–2039.
[17] Kueng B, Mueggler E, Gallego G, Scaramuzza D. Low-latency visual odometry using event-based feature tracks. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2016. p. 16–23.
[18] Hirschmuller H. Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence. 2007;30(2):328–341.
[19] Dobias M, Sara R. Real-time global prediction for temporally stable stereo. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). IEEE. 2011. p. 704–707.
[20] Jiang J, Cheng J, Chen B, Wu X. Disparity prediction between adjacent frames for dynamic scenes. Neurocomputing. 2014;142:335–342.
[21] Fućek L, Marković I, Cvišić I, Petrović I. Dense disparity estimation in ego-motion reduced search space. IFAC-PapersOnLine. 2017;50(1):10122–10127.
[22] Popović G, Hadviger A, Marković I, Petrović I. Computationally efficient dense moving object detection based on reduced space disparity estimation. IFAC-PapersOnLine. 2018;51(22):360–365.
[23] Benosman R, Clercq C, Lagorce X, Ieng SH, Bartolozzi C. Event-based Visual Flow. IEEE transactions on neural networks and learning systems. 2014;25(2):407–417.
[24] Demirdjian D, Darrell T. Motion estimation from disparity images. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. Vol. 1. IEEE. 2001. p. 213–218.
[25] Mueggler E, Forster C, Baumli N, Gallego G, Scaramuzza D. Lifetime Estimation of Events from Dynamic Vision Sensors. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2015. p. 4874–4881.
[26] Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY. ROS: an open-source Robot Operating System. In: ICRA workshop on open source software. Vol. 3. Kobe, Japan. 2009. p. 5.
[27] Rebecq H, Gehrig D, Scaramuzza D. ESIM: an open event camera simulator. Conf on Robotics Learning (CoRL). 2018 oct;.
[28] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 2012. p. 3354–3361.
[29] Gehrig D, Gehrig M, Hidalgo-Carrió J, Scaramuzza D. Video to events: Recycling video datasets for event cameras. In: IEEE conf. comput. vis. pattern recog. (cvpr). 2020 June.
[30] Jiang H, Sun D, Jampani V, Yang MH, Learned-Miller E, Kautz J. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 9000–9008.
[31] Maye J, Furgale P, Siegwart R. Self-supervised Calibration for Robotic Systems. In: 2013 IEEE Intelligent Vehicles Symposium (IV). IEEE. 2013. p. 473–480.