Int J Comput Vis
DOI 10.1007/s11263-017-1050-6

EMVS: Event-Based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time

Henri Rebecq[1,2] · **Guillermo Gallego**[1,2] · **Elias Mueggler**[1,2] · **Davide Scaramuzza**[1,2]

Received: 6 January 2017 / Accepted: 19 October 2017

Abstract Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, because the output is composed of a sequence of asynchronous events rather than actual intensity images, traditional vision algorithms cannot be applied, so that a paradigm shift is needed. We introduce the problem of event-based multi-view stereo (EMVS) for event cameras and propose a solution to it. Unlike traditional MVS methods, which address the problem of estimating *dense* 3D structure from a set of known viewpoints, EMVS estimates *semi-dense* 3D structure from an event camera with known trajectory. Our EMVS solution elegantly exploits two inherent properties of an event camera: (1) its ability to respond to scene edges—which naturally provide semi-dense geometric information without any pre-processing operation—and (2) the fact that it provides continuous measurements as the sensor moves. Despite its simplicity (it can be implemented in a few lines of code), our algorithm is able to produce accurate, semi-dense depth maps, without requiring any explicit data association or intensity estimation. We successfully validate our method on both synthetic and real data. Our method is computationally very efficient and runs in real-time on a CPU.

Keywords Multi-view stereo · Event cameras · Event-based vision · 3D reconstruction

Multimedia Material

A supplemental video for this work is available at https://youtu.be/EFpZcpd9XJ0

1 Introduction

An event camera, such as the dynamic vision sensor (DVS) (Lichtsteiner et al. 2008), works very differently from a traditional camera. It has *independent* pixels that only send information (called "events") in presence of brightness changes in the scene at the time they occur. Thus, the output is not an intensity image but a stream of asynchronous events at microsecond resolution, where each event consists of its space–time coordinates and the *sign* of the brightness change (i.e., no intensity). Since events are caused by brightness changes over time, an event camera naturally responds to edges in the scene in presence of relative motion.

Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds, low power consumption, and a very high dynamic range (130 dB compared to 60 dB of standard cameras). These properties make the sensors ideal in all those applications where fast response and high efficiency are crucial and also in scenes with wide variations of illumination. Additionally, since information

Communicated by Edwin Hancock, Richard Wilson, Will Smith, Adrian Bors, Nick Pears.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11263-017-1050-6) contains supplementary material, which is available to authorized users.

✉ Henri Rebecq
rebecq@ifi.uzh.ch

[1] Robotics and Perception Group, Department of Informatics, University of Zurich, Zurich, Switzerland

[2] Robotics and Perception Group, Department of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland

Published online: 07 November 2017

is only sent in presence of brightness changes, the sensor removes all the inherent redundancy of standard cameras, thus requiring a very low data rate (KB vs MB). However, since event cameras became commercially available only recently (Lichtsteiner et al. 2008), little related work exists, and, because their output is significantly different from that of standard cameras, traditional vision algorithms cannot be applied, which calls for new methods to process the data from these novel cameras, and therefore be able to unlock their potential.

1.1 Contribution

In this paper, we address the problem of structure estimation (i.e., 3D reconstruction) with a *single* event camera by introducing the concept of event-based multi-view stereo (EMVS) (Sect. 4), and we propose an algorithm to solve this problem.

Our approach (Sects. 5–7) follows a space-sweep (Collins 1996) voting and maximization strategy to estimate semi-dense depth maps at selected viewpoints, and then we merge the depth maps to build larger 3D models. We evaluate the method on both synthetic and real data (Sect. 8). The results are analyzed and compared with ground truth, showing the successful performance of our approach.

This paper is based on our previous work (Rebecq et al. 2016), which we extend in several ways:

- We provide a justification of the choice of perspective sampling of space by analyzing the operation of event back-projection (Sect. 6).
- We show how event back-projection can be efficiently implemented and parallelized using homographies to enable real-time performance, and we quantify the computational performance of our method (Sect. 7).
- We improve structure estimation by means of simple processing techniques, such as bilinear voting in the disparity space image (Sect. 7.1) and median filtering of the semi-dense depth map (Sect. 5.2.5).
- We include additional experiments (Sect. 8), showing the applicability of our method.

2 Event Cameras and Applications

Event cameras are biologically inspired sensors that present a new paradigm on the way that dynamic visual information is acquired and processed. Each pixel of an event camera operates independently from the rest, continuously monitoring its intensity level and transmitting only information about brightness changes of given size ("events") whenever they occur, asynchronously, with microsecond resolution. Specifically, if $L(\mathbf{u}, t) \doteq \log I(\mathbf{u}, t)$ is the logarithmic brightness or intensity at pixel $\mathbf{u} = (x, y)^\top$ in the image plane, an

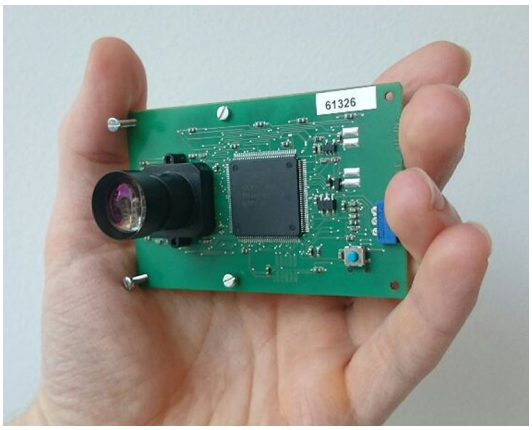


Fig. 1 The event camera "eDVS" produced by iniLabs (https://inilabs.com/products/)

event camera such as the DVS (Lichtsteiner et al. 2008) (see Fig. 1) generates an event $e_k \doteq \langle x_k, y_k, t_k, p_k \rangle$ if the change in logarithmic brightness at pixel $\mathbf{u}_k = (x_k, y_k)^\top$ reaches a threshold C (typically 10–15% relative brightness change):

$$\Delta L(\mathbf{u}_k, t_k) \doteq L(\mathbf{u}_k, t_k) - L(\mathbf{u}_k, t_k - \Delta t) = p_k C, \qquad (1)$$

where t_k is the timestamp of the event, Δt is the time since the previous event at the same pixel $\mathbf{u}_k$, and $p_k = \pm 1$ is the polarity of the event (the sign of the brightness change). A comparison between the outputs of a standard and an event camera is shown in Fig. 2.

Therefore, visual information is no longer acquired based on an external clock (e.g., global shutter); instead, each pixel has its own sampling rate, based on the visual input: event cameras are data-driven sensors. This different paradigm of acquiring visual information, i.e., reporting temporal contrast, offers significant advantages over that of standard cameras, namely redundancy removal, a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, new computer vision algorithms that exploit the high temporal resolution and the asynchronous nature of the sensor are required to cope with this unfamiliar representation of the visual information.

Event cameras find applications in real-time interaction systems such as robotics or wearable electronics (Delbruck 2016), where operation under uncontrolled lighting conditions, latency, and power are important. Event cameras have been used for object tracking (Delbruck and Lichtsteiner 2007; Drazen et al. 2011; Delbruck and Lang 2013), surveillance and monitoring (Litzenberger et al. 2006; Piatkowska et al. 2012), object recognition (Wiesmann et al. 2012; Orchard et al. 2015; Lagorce et al. 2016) and gesture control (Lee et al. 2014). They have also been used for stereo depth estimation (Rogister et al. 2012; Piatkowska et al. 2013) (see also related work in Sect. 3), 3D panoramic imaging (Schraml et al. 2015), structured light 3D scanning (Matsuda

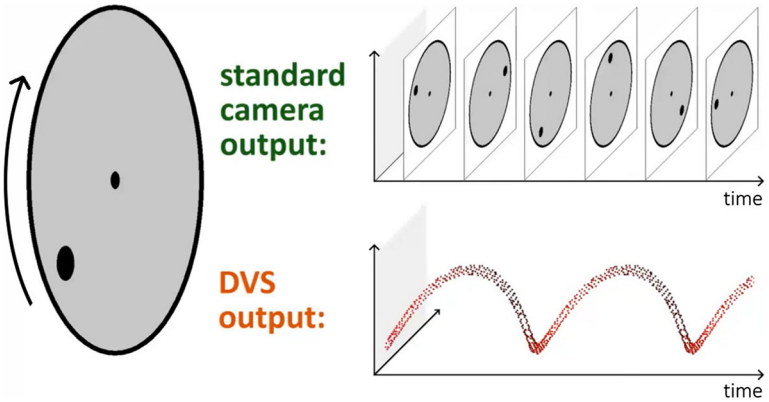


Fig. 2 Comparison of the output of a standard camera and an event camera (DVS) when viewing a spinning disk with a black circle. The standard camera outputs frames at a fixed rate, thus sending redundant information when no motion is present in the scene. In contrast, event cameras are data-driven sensors that output pixel-level brightness changes with microsecond latency. Therefore, they do not suffer from motion blur and produce no output if there is no visual change in the scene. An animated version can be found here: https://youtu.be/LauQ6LWTkxM

et al. 2015), optical flow estimation (Benosman et al. 2012, 2014; Rueckauer and Delbruck 2016; Bardow et al. 2016), high dynamic range (HDR) image reconstruction (Cook et al. 2011; Reinbacher et al. 2016), mosaicing (Kim et al. 2014) and video compression (Brandli et al. 2014a). In ego-motion estimation, event cameras have been used for pose tracking (Weikersdorfer and Conradt 2012; Mueggler et al. 2014; Gallego et al. 2017), and visual odometry and simultaneous localization and mapping (SLAM) (Weikersdorfer et al. 2013; Censi and Scaramuzza 2014; Kueng et al. 2016; Kim et al. 2016; Rebecq et al. 2017). Event-based vision is a growing field of research, and many more applications are expected to appear as event cameras become widely spread.

3 Related Work on Event-Based Depth Estimation

The majority of works on event-based depth estimation tackle the 3D reconstruction problem by using *two or more* event cameras that are rigidly attached (i.e., with a fixed baseline) and share a *common clock*. These methods follow a two-step approach: first they solve the event correspondence problem across image planes and then triangulate the location of the 3D point. Events are matched in two ways: either using traditional stereo methods on artificial frames generated by accumulating events over time (Schraml et al. 2010; Kogler et al. 2011b), or exploiting simultaneity and temporal correlations of the events across sensors (Kogler et al. 2011a; Rogister et al. 2012; Lee et al. 2012; Camunas-Mesa et al. 2014).

The event-based depth estimation problem that we address is entirely different: (1) we consider a *single* camera and (2) we do not require simultaneous event observations.

Depth estimation from a single event camera is more challenging because we cannot exploit temporal correlation between events across multiple image planes. Notwithstanding, we show that a single event camera suffices to estimate depth, and, moreover, that we are able to do it without solving the data association problem, as opposed to event-based stereo-reconstruction methods.

Since the publication of our monocular event-based depth estimation method (Rebecq et al. 2016), another solution has been proposed in (Kim et al. 2016). Their method is part of a pipeline that uses three filters operating in parallel to jointly estimate the motion of the event camera, a 3D map of the scene, and the intensity image. Their depth estimation approach requires using an additional quantity—the intensity image—to solve for data association (events corresponding to the same 3D point have the same image intensity under the Lambertian hypothesis). Intensity estimation and depth regularization are carried out using dedicated hardware (a GPU) to achieve real-time performance. In contrast, our approach (Rebecq et al. 2016) leverages directly the sparsity of the event stream to perform 3D reconstruction (it does not need to recover the intensity image to estimate depth), and is computationally efficient, running in real-time on the CPU. In our most recent article (Rebecq et al. 2017), we address the problem of parallel tracking and mapping with an event camera; notably, we show how the 3D reconstruction method proposed in the present paper can be combined with an event-based pose tracking algorithm to yield both trajectory estimates as well as semi-dense 3D maps.

4 The Event-Based Multi-View Stereo Problem

MVS with traditional cameras addresses the problem of 3D structure estimation from a collection of images taken from known viewpoints (Szeliski 2010) of an intrinsically cali-

brated camera. Our event-based MVS (EMVS) shares the same goal; however, there are some key differences:

1. Traditional MVS algorithms work on full images, so they cannot be applied to the stream of asynchronous events provided by the sensor. EMVS must take into account the *sparse* and *asynchronous* nature of the events.
2. Because event cameras do not output data if both the sensor and the scene are static, any event-driven algorithm, such as EMVS, requires the sensor to be *moved* in order to acquire visual content. In traditional MVS, the camera does not need to be in motion to acquire visual content.
3. Because events are caused by intensity edges, the natural output of EMVS is a *semi-dense* 3D map, as opposed to the dense maps of traditional MVS.

Hence, the EMVS problem consists of obtaining the 3D reconstruction of a scene from the sparse asynchronous streams of events acquired by moving event cameras with known viewpoints. Without loss of generality, it suffices to consider the case of one event camera.

To solve the EMVS problem, classical MVS approaches cannot be directly applied since they work on intensity images. Nevertheless, our event-based approach builds upon previous works on traditional MVS (Seitz et al. 2006). In particular, we follow (in Sect. 5) the solving strategy of scene space MVS methods (Seitz et al. 2006), which consist of two main steps: computing an aggregated consistency score in a discretized volume of interest [the disparity space image (DSI)] by warping image measurements, and then finding 3D structure information in this volume. The term DSI (Szeliski and Golland 1999) is interchangeably used to refer to the projective sampling of the volume (i.e., discretized volume) or to the scalar function defined in it (i.e., the score). Just by considering the way that visual information is provided, we can point out two key differences between the DSI approaches in MVS and EMVS:

1. In classical MVS, the DSI is *densely* populated using pixel intensities. In EMVS, the DSI may have holes (voxels with no score value), since warped events are also *sparse*.
2. In classical MVS, scene objects are obtained by finding an optimal surface in the DSI. By contrast, in EMVS, finding *semi-dense* structures (e.g., points, curves) is a better match to the sparsity of the DSI.

5 Event-Based Space-Sweep Method

Our method to solve the EMVS problem is similar to Collin's space-sweep approach for MVS (Collins 1996), which shows how sparsity can be leveraged to estimate 3D structures without the need for explicit data association or photometric information. We generalize the space-sweep approach for the case of a moving event camera by building a virtual camera's DSI (Szeliski and Golland 1999) containing only geometric information of edges and finding 3D points in it.

First, we review the classical space-sweep method for standard cameras (Sect. 5.1), and then we describe our generalization to a moving event camera (Sect. 5.2), showing that the continuous stream of events produced by the sensor is specially relevant to recover 3D structure.

5.1 Classical Space-Sweep Method

In contrast to most classical MVS methods, which rely on pixel intensity values, the space-sweep method (Collins 1996) relies solely on binary edge images (e.g., Canny) of the scene from different viewpoints.

Thus, it leverages the sparsity or semi-density of the viewpoint dependent edge maps to determine 3D structure.

More specifically, the method consists of three steps: (1) warping (i.e., back-projecting) image features as rays through a DSI, (2) recording the number of rays that pass through each DSI voxel, and (3) determining whether or not a 3D point is present in each voxel. The DSI score measures the geometric consistency of edges in a very simple way: each pixel of a warped edge-map onto the DSI votes for the presence or absence of an edge. Then, the DSI score is thresholded to determine the scene points that most likely explain the image edges.

5.2 Event-Based Space-Sweep Method

In this section, we extend the space-sweep algorithm in Sect. 5.1 to solve EMVS. Notice that the stream of events provided by event cameras is an ideal input to the space-sweep algorithm because (1) event cameras naturally highlight edges in hardware, and (2) edges trigger events from *many* consecutive viewpoints rather than a few sparse ones (cf. Fig. 3).

Next we detail the three steps of the event-based space-sweep method: back-projection (Sect. 5.2.1), ray-counting (Sect. 5.2.2), and determining the presence of scene structure (Sect. 5.2.3). Then, we also discuss how to merge depth maps from multiple viewpoints (Sect. 5.2.4), and how to improve the quality of the reconstruction with simple post-processing techniques (Sect. 5.2.5).

5.2.1 Feature-Viewing Rays by Event Back-Projection

Let us formally define an event $e_k = (x_k, y_k, t_k, p_k)$ as a tuple containing the pixel position (x_k, y_k), timestamp t_k, and polarity p_k (i.e., sign) of the brightness change. We extend the space-sweep method to the event-based paradigm by using

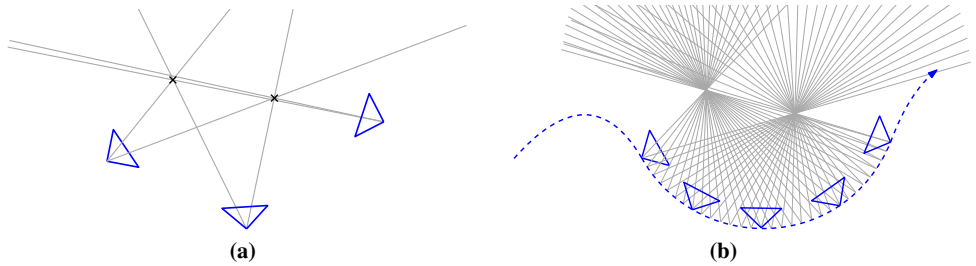


Fig. 3 Comparison of the back-projection step in classical space-sweep and event-based space-sweep. This is a 2D illustration with the scene consisting of two points. **a** Classical (frame-based) Space-Sweep: only a fixed number of views is available. Two points of an edge map are visible in each image. The intersections of rays obtained by backprojecting the image points are used as evidence for detection of scene features (object points). **b** Event-Based Space-Sweep: as the event sensor moves, events are triggered on the sensor. To each observed event corresponds a ray (through back-projection), that spans the possible 3D-structure locations. The areas of high ray density correspond to the locations of the two points, and are progressively discovered as the sensor moves.

the event stream $\{e_k\}$ output by the event camera as the input point-like features that are warped into the DSI. Each event e_k is back-projected according to the viewpoint of the event camera at time t_k, which is known according to the assumptions of MVS.

From a geometric point of view, we compare the back-projection step in the classical frame-based and the event-based settings using Fig. 3. Observe that in frame-based MVS the number of viewpoints is small compared to that in the highly sampled trajectory of the event camera (at times $\{t_k\}$). This higher abundance of measurements and viewpoints in the event-based setting generates many more viewing rays than in frame-based MVS, and therefore, it facilitates the detection of scene points by analyzing the regions of high ray density.

A major advantage of our method is that no explicit data association is needed. This is the main difference between our method and existing event-based depth estimation methods (Sect. 3). While other works essentially attempt to estimate depth by first solving the stereo correspondence problem in the image plane (using frames of accumulated events (Schraml et al. 2010; Kogler et al. 2011a), reconstructed intensity (Kim et al. 2016), temporal correlation of events (Kogler et al. 2011b; Rogister et al. 2012; Lee et al. 2012; Camunas-Mesa et al. 2014), etc.), our method works directly in 3D space.

This is illustrated in Fig. 3b: there is no need to associate an event to a particular 3D point to be able to recover its 3D location.

5.2.2 *Volumetric Ray Counting. Creating the Disparity Space Image (DSI)*

In the second step of space-sweep, we discretize the volume containing the 3D scene and count the number of viewing rays passing through each voxel using a DSI. To allow for the reconstruction of large scenes in a scalable way, we split the 3D volume containing the scene into smaller 3D volumes along the trajectory of the event camera, compute local 3D reconstructions, and then merge them, as will be explained in Sect. 5.2.4.

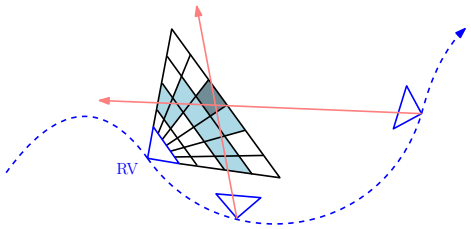
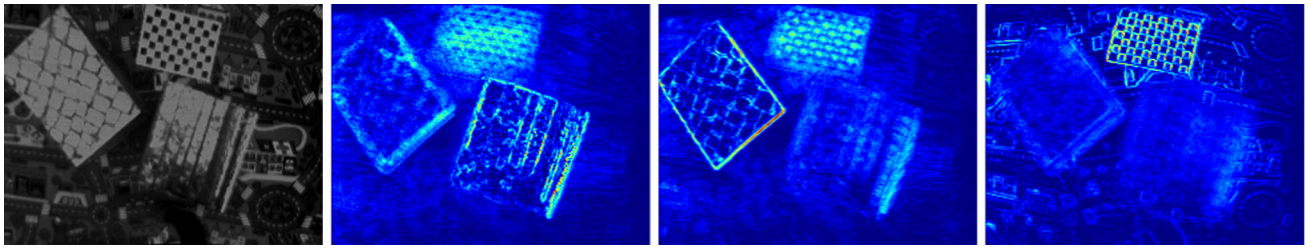


Fig. 4 The DSI ray counter is centered at a virtual camera in a reference viewpoint (RV) and its shape is adapted to the perspective projection. Every incoming viewing ray from a back-projected event (in red) votes for all the DSI voxels (in light blue) which it traverses (Color figure online)

For now, let us focus on computing a local 3D reconstruction of the scene from a subset of events. For this task, we create a virtual camera located at a reference viewpoint that is chosen among those event camera viewpoints associated to the subset of events, and then define a DSI in a volume V adapted to the field of view and perspective projection of the event camera, as illustrated in Fig. 4 [see (Szeliski and Golland 1999)]. The DSI is defined by the event camera pixels and a number N_z of depth planes $\{Z_i\}_{i=1}^{N_z}$, i.e., it has size $w \times h \times N_z$, where w and h are the width and height of the event camera, respectively. The score stored in the DSI

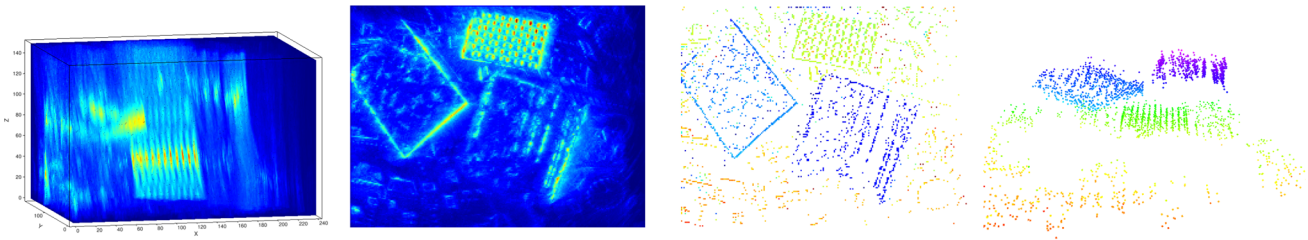
$$f(\mathbf{X}) : V \subset \mathbb{R}^3 \to \mathbb{R}^+ \tag{2}$$

is the number of back-projected viewing rays passing through each voxel with center $\mathbf{X} = (X, Y, Z)^\top$, as shown in Fig. 4. We show in Sect. 7.1 how to efficiently compute the ray-voxel intersections using a two-step approach, allowing for real-time performance on a single CPU.



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

Fig. 5 **a** Scene with the event camera moving above three textured planes located at different depths (close, middle, far). We build the ray density DSI $f(\mathbf{X})$ as described in Sect. 5.2.2 and show the effect of slicing it at different depths, **b–d**, as simulating a plane sweeping through the DSI. When the sweeping plane coincides with an object plane, the latter appears very sharp while the rest of the scene is "out of focus". **a** Image at virtual camera. **b** DSI slice at *close* depth. **c** DSI slice at *middle* depth. **d** DSI slice at *far* depth



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

Fig. 6 Our method builds the ray density DSI (**a**), from which a confidence map (**b**) and a semi-dense depth map (**c**) are extracted in a virtual camera. The semi-dense depth map gives a point cloud of scene edges (**d**). Same dataset as in Fig. 5. **a** Ray density DSI $f(\mathbf{X})$. **b** Confidence map. **c** Semi-dense depth map. **d** 3D point cloud

5.2.3 *Detection of Scene Structure by Maximization of Ray Density*

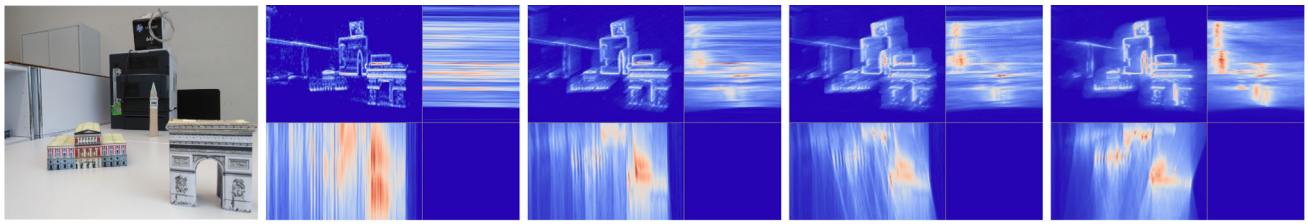
In the third step of space-sweep, we obtain a semi-dense depth map in the virtual camera by determining whether or not a 3D point is present in each DSI voxel. The decision is taken based on the ray density function stored in the DSI, $f(\mathbf{X})$.

Rephrasing the assumption of the space-sweep method (Collins 1996), scene points are likely to occur at regions where several viewing rays nearly intersect (see Fig. 3b), which correspond to regions of high ray density. Hence, scene points are likely to occur at *local maxima* of the ray density function. Figure 5 shows an example of slicing the DSI in Fig. 6a, from a real dataset, at different depth planes; the presence of local maxima of the ray density function is evidenced by the in-focus areas. Additionally, Fig. 7 shows the emergence of high ray-density regions in the DSI as the sensor moves and more events are observed.

We detect the local maxima of the DSI $f(\mathbf{X})$ following a two-step procedure: we first generate a dense depth map $Z^*(x, y)$ in the virtual camera and an associated confidence map $c(x, y)$ by recording the location and magnitude of the best local maximum of the DSI $f(X(x), Y(y), Z^*) =: c(x, y)$ along the row of voxels in the viewing ray of each pixel (x, y). Then, we select the most confident pixels in the depth map by thresholding the confidence map, yielding a semi-dense depth map (Fig. 6c). We use Adaptive Gaussian Thresholding: a pixel (x, y) is selected if $c(x, y) > T(x, y)$, with $T(x, y) = c(x, y) * G_\sigma(x, y) - C$. In practice, we use a 5×5 neighborhood in G_σ and $C = -10$. The adaptive approach yields better results than global thresholding (Collins 1996). A summary of the main elements of our DSI approach is given in Fig. 6.

5.2.4 *Merging Depth Maps from Multiple Reference Viewpoints*

So far, we have shown how to reconstruct the structure of scene corresponding to a subset of the events around a reference view. As pointed out in Sect. 5.2.2, motivated by a scalable design, this operation is carried out on subsets of the event stream, thus recovering semi-dense depth maps of the scene at multiple *key* reference views. More specifically, we select a new *key* reference view as soon as the distance to the previous *key* reference view exceeds a certain percentage of the mean scene depth (typically a number between 15 and 40%), and use the subset of events until the next *key* reference view to estimate the corresponding semi-dense depth map of the scene. The depth maps are then converted to point clouds, cleaned from isolated points (those whose number of neighbors within a given radius is less than a threshold) and



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

Fig. 7 Evolution of the DSI as the event camera moves. Figure **a** shows a preview of the scene, while figures **b**–**e** show the successive projections of the DSI along its three axes (top-left inset: front view, top-right inset: side-view, bottom-left inset: top-view). As more events are observed, areas of high ray density (in red) start appearing and the uncertainty in depth decreases in all directions. In this example, the DSI is sampled uniformly in inverse depth (Color figure online)

merged into a global point cloud using the known positions of the virtual cameras. Other depth map fusion strategies could be implemented. However, such a research topic is out of the scope of this paper. In practice, our approach shows compelling large-scale 3D reconstruction results even without the need for complex fusion methods or regularization.

5.2.5 Map Cleaning

To further enhance the quality of the 3D reconstruction, we use a median filter on the semi-dense depth maps obtained in Sect. 5.2.3. Specifically, we consider only the converged pixels, i.e., the remaining pixels after the Adaptive Gaussian Thresholding, as input to the median filter. This allows removing outliers while preserving depth discontinuities.

Additionally, we also apply a radius filter (Rusu and Cousins 2011) to the final point cloud, which discards the points whose number of neighbors within a given radius is less than a threshold. This helps remove isolated points, which are most likely outliers.

6 Sampling the DSI: Uniform Versus Projective

In this section we justify our choice of using a projective sampling of the DSI volume, i.e., a projective voxel grid, instead of using a uniform sampling [as originally proposed in (Collins 1996)]. The reader who is not interested in this explanation can jump to Sect. 7.

We compare both sampling strategies (uniform and projective) by means of a simple experiment in 2D, illustrated in Fig. 8, and support the comparison by means of well-grounded mathematical results.

Let us consider a 2D scene consisting of a moving event camera and a few set of points with large contrast so that they generate events (Fig. 8a).

For simplicity, and since our method does need the event polarity, we model the event camera as a sensor that outputs a binary value describing whether a scene point is visible by a specific camera pixel. This is only an approximate model; for example, an event camera moving forward towards a point in the center of the image plane would not trigger events (since the brightness of this pixel does not change), but in this model we consider that for every visible scene point an event is generated at each camera pose. Nevertheless, this is a good geometric model that provides insight into the EMVS problem and our proposed solution. We use this model to compute the DSI ray density function on a region of the XZ space, and sample it in two different ways: (1) using a uniform grid along both X and depth Z axes (i.e., on a Cartesian grid), as shown in Fig. 8b, and (2) using a projective grid (as in Fig. 4) that mimics the perspective operation of a camera located somewhere along the event camera trajectory, as shown in Fig. 8a. Approach (1) corresponds to the one originally proposed in (Collins 1996). We are interested in comparing the effect of both sampling strategies on the shape and size of the rays, more correctly "cones", obtained by back-projecting events, that is, we consider that pixels are not just points but have a finite extent.

6.1 Shape of the Back-Projected Rays

First, let us analyze the shape, i.e., ignoring the finite extent of the pixel. Later, we will analyze the effect of the finite pixel size on the back-projection operation to create the DSI. The ray back-projected from a point $\mathbf{u}$ in the camera is a line in Euclidean space. Using calibrated coordinates, and assuming that $\mathtt{P} = (\mathtt{R}|\mathbf{t})$ is the projection matrix of the camera, the ray is given by the line joining two points (Hartley and Zisserman 2003, p.162): the optical center of the camera $\mathbf{C} = -\mathtt{R}^\top \mathbf{t}$ and the point at infinity $(\mathbf{D}^\top, 0)^\top$, with $\mathbf{D} = \mathtt{R}^\top \mathbf{u}$, projecting on $\mathbf{u}$. A point on the ray has Euclidean coordinates

$$\mathbf{X} = \rho \mathbf{D} + \mathbf{C}. \tag{3}$$

These are the parametric equations of the line, with depth parameter ρ. The uniform sampling strategy preserves the

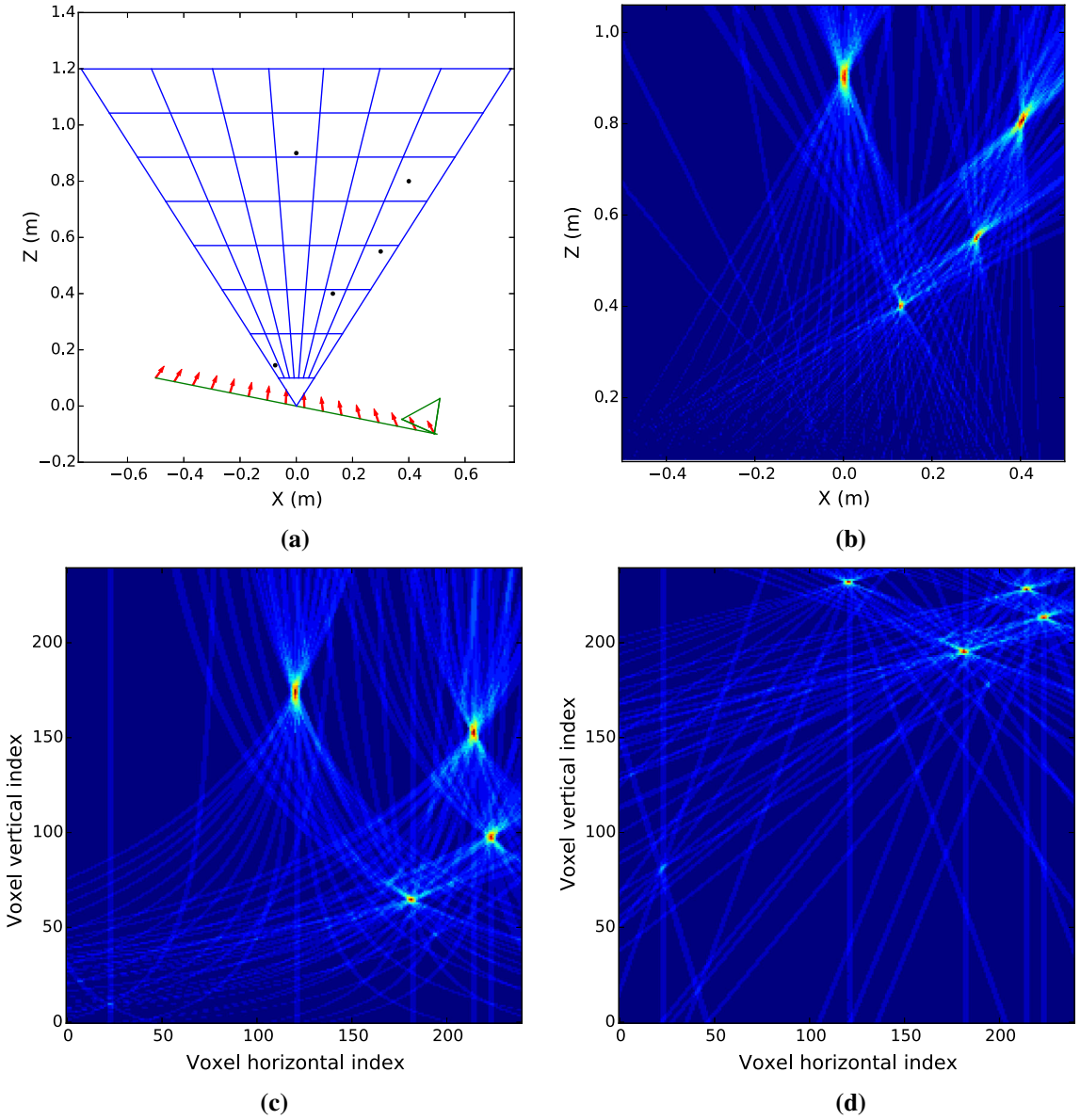


Fig. 8 Illustration of uniform versus projective sampling of the DSI using a 2D example. Ray density plots are pseudo-colored, from dark blue (small density) to red (high density). Figure generated with 50 camera poses, a camera FOV of 75°, and image resolution of 100 pixels (along the camera's X axis). The voxel grid has a resolution of 240 pixels (along the X axis) and 240 depth planes. **a** 2D scene geometry featuring five points, the camera trajectory (in green) and optical axis direction (in red), and the projective voxel grid (in blue). **b** Ray density in Euclidean space (uniform voxel grid). The width of each ray grows with the depth. **c** Ray density using projective voxel grid, equispaced in depth (voxel vertical index). The width of each ray is constant along the depth. **d** Ray density using projective voxel grid, equispaced in inverse depth (voxel vertical index). As in Fig. 8c, the width of each ray is constant along the depth (Color figure online)

straight nature of the back-projected rays, as shown in Fig. 8b. In contrast, the rays are no longer straight in the case of the projective sampling (Fig. 8c). In the projectively sampled space, a Euclidean point $\mathbf{X} \doteq (X, Y, Z)^\top$ is described by coordinates

$$\mathbf{X}_p = \left(\frac{X}{Z}, \frac{Y}{Z}, Z\right)^\top \doteq (x, y, Z)^\top. \tag{4}$$

Letting $\mathbf{C} = (C_i)$ and $\mathbf{D} = (D_i)$, $i = 1, \ldots, 3$, we combine (3) and (4) to obtain the parametric equations

$$\mathbf{X}_p = \left(\frac{\rho D_1 + C_1}{\rho D_3 + C_3}, \frac{\rho D_2 + C_2}{\rho D_3 + C_3}, \rho D_3 + C_3\right)^\top. \tag{5}$$

Let us show that (5) explains the curved shapes of back-projected rays observed in Fig. 8c. For depth values $\rho \gg 1$,

the points on the ray follow the curve $\mathbf{X}_p \approx (D_1/D_3, D_2/D_3, 0)^\top + \rho\,(0, 0, D_3)^\top$, which is a line with direction vector $(0, 0, D_3)$, i.e., a line parallel to the Z-axis. This is observed in the top part of Fig. 8c. For small depth values ($\rho \to 0$), the points on the ray approach the optical center of the camera, as expected, $\mathbf{X}_p \approx (C_1/C_3, C_2/C_3, C_3)^\top$, so we look at the way that they approach this point by computing the tangent:

$$\frac{d\mathbf{X}_p}{d\rho} \overset{(5)}{=} \left(\frac{D_1C_3 - D_3C_1}{(\rho D_3 + C_3)^2}, \frac{D_2C_3 - D_3C_2}{(\rho D_3 + C_3)^2}, D_3 \right)^\top. \tag{6}$$

The plots in Fig. 8c where generated with a moving camera with $C_3 \ll D_3$, and so, for small depth values, $d\mathbf{X}_p/d\rho \approx ((-C_1/(D_3\rho^2), -C_2/(D_3\rho^2), D_3)^\top$. In the 2D example (only considering X and Z coordinates), as $\rho \to 0$ the tangent is dominantly along the X axis, which agrees with Fig. 8c. In summary, when going from zero to infinite depth, the tangent changes from being parallel to the X axis to being parallel to the Z axis, and so, the tangent varies (smoothly) between these two directions, as shown in the curved shapes of Fig. 8c.

Finally, consider what happens when the DSI is sampled projectively and equispaced in inverse depth instead of depth: the curved shapes analyzed in Fig. 8c become almost straight, as shown in Fig. 8d. This is similar to the effect of representing the function $y = e^x$ in logarithmic scale: $\log(y)$ becomes a line. The curve represented by the X and Z coordinates of (5) is the parametric curve $(x(r), r)$, with $x(r) = D_1/D_3 + (C_1 - (C_3D_1)/D_3)r^{-1}$ and the change of variables $r = \rho D_3 + C_3$. Thus, $x(r)$ is a line when using the parameter $r^{-1} = (\rho D_3 + C_3)^{-1}$, which is approximately inverse depth. Fig. 8d was generated with $C_3 \ll D_3$, and so the ray $(x(r), r)$ is indeed almost straight when the Z axis is given in inverse depth.

6.2 Size of the Back-Projected Cones

We now consider that pixels have a non-zero area and study how a back-projected event contributes to the DSI depending on the sampling scheme.

A pixel collects the light in a fixed, small angle around a given direction. This angle correspond to different object sizes depending on the distance of the object to the camera. This idea is roughly expressed by the formula of the area A of a sphere patch seen by a central solid angle Ω: $A = \Omega r^2$, where r is the radius of the sphere. Thus, the same pixel angle Ω covers an area A at a distance r and an area four-times larger $4A$ at double the distance $2r$. Hence, the back-projection of a pixel into space generates a cone whose base area A grows quadratically with the distance to the camera.

In a uniform sampling of the DSI, where all voxels have the same size, a pixel back-projects into a cone that will cover more voxels the farther they are from the camera. In contrast, using a projective sampling of the DSI, we compensate for the perspective effect of the camera by making the size of the voxel increase with the distance of the voxel to the virtual camera defining the projective grid, so that a pixel back-projected into space will cover always roughly the same number of voxels: one. This comparison can be observed in Fig. 8b, c. In Fig. 8b we can identify the cones, whose apexes lie on the event camera trajectory. In Fig. 8c, the cones are represented by curves of approximately constant width (perpendicular to the depth axis). This constant width is also appreciated in Fig. 8d, where the cones become "cylinders".

Let us mathematically support the previous statements. Fig. 8a illustrates the geometry of the projective sampling considered. The projective DSI is defined by a virtual camera with projection matrix $\mathtt{P}_v = (\mathtt{I}|0)$, in calibrated coordinates. At the time of the current event $e = (u, v, t, p)$, the event camera is described by projection matrix $\mathtt{P}_e = (\mathtt{R}|\mathbf{t})$. The pixel where the event has been triggered is back-projected into points of the form (4) in the projective DSI. Each depth plane $Z = Z_i$ induces a planar homography between the image plane of the event camera and the image plane of the virtual camera, by mapping the event coordinates $(u, v)^\top$ to the first two coordinates of (4), $(x, y)^\top$. We use this planar homography to measure the area in the virtual camera (i.e., the area perpendicular to the depth axis in the projective grid) that is due to the pixel that triggered the event. The relation between the area elements in both cameras is given by the determinant of the Jacobian of the homography:

$$dxdy = \det\left(\frac{\partial(x, y)}{\partial(u, v)}\right) dudv. \tag{7}$$

The planar homography $\mathtt{H}_{Z_i} : (u, v) \mapsto (x, y)$ from the event camera to the virtual camera, induced by the plane $Z = Z_i$ (with coordinates $\boldsymbol{\pi} = (\mathbf{e}_3^\top, -Z_i)^\top$, $\mathbf{e}_3 = (0, 0, 1)^\top$), is given by the inverse of the homogeneous matrix [see (24)]

$$\mathtt{H}_{Z_i}^{-1} \sim \mathtt{R} + \frac{1}{Z_i}\mathbf{t}\mathbf{e}_3^\top. \tag{8}$$

The Jacobian in (7) can be computed applying Result 2 in the Appendix to (8) and the fact that the Jacobian of $\mathtt{H}_{Z_i}$ is the inverse of the Jacobian of $\mathtt{H}_{Z_i}^{-1}$:

$$\det\left(\frac{\partial(x, y)}{\partial(u, v)}\right) = \left(\frac{Z_i'}{Z_i}\right)^3 \left(1 - \frac{C_z}{Z_i}\right)^{-1}, \tag{9}$$

where Z_i' is the depth of the point $\mathbf{X} \in \boldsymbol{\pi}$ with respect to the event camera $\mathtt{P}_e$, and C_z is the third coordinate of the optical center of $\mathtt{P}_e$. Therefore, the conversion factor between areas in the image planes is a function of the ratio of depths of the scene point with respect to both cameras and the ratio

of depths C_z/Z_i. Assuming that the scene point is equally far away from both cameras (i.e., $Z_i' \approx Z_i$) and that the amount of forward motion of the event camera is negligible compared to Z_i, $C_z \ll Z_i$, the conversion factor (9) in (7) becomes approximately 1, that is, a pixel maps to an area (perpendicular to the depth axis) of 1 pixel in the projective grid; this is the area of a cross-section of a voxel, hence for each depth plane $Z = Z_i$, a pixel in the event camera votes for 1 voxel in the projective grid.

To summarize, we have shown that the projective sampling of the DSI is a better choice than the uniform sampling because a back-projected event will vote for approximately one grid cell per depth plane instead of multiple cells (in case of uniform sampling) whose number would grow quadratically with depth. This property (area conversion factor ≈ 1) is not only advantageous when creating the DSI (only one vote needed per depth plane), but also when extracting the scene edges from it. Indeed, areas at different depth planes of the virtual camera are comparable when using the projective DSI, thus enabling the use of a fixed-size adaptive-threshold mask in all depth planes to extract clusters of high ray density along the viewing rays of the virtual camera. In contrast, with a uniform voxel grid, the size of the clusters depends on the depth, which means that the mask size of the adaptive threshold itself would have to be dependent on the depth plane.

Remark The previous analysis used calibrated coordinates. If, instead, we use pixel coordinates, with $\mathtt{K}_v$ and $\mathtt{K}_e$ being the intrinsic parameter matrices of the DSI virtual camera and the event camera, respectively, it is easy to show, using an argument on how area elements transform (7), that (9) will become

$$\det\left(\frac{\partial(x,y)_{\text{pixel}}}{\partial(u,v)_{\text{pixel}}}\right) = \frac{\det(\mathtt{K}_v)}{\det(\mathtt{K}_e)}\left(\frac{Z_i'}{Z_i}\right)^3\left(1-\frac{C_z}{Z_i}\right)^{-1}, \quad (10)$$

that is, the ratio of the focal lengths of the cameras can be used to modify the number of voxels that each event votes for. However, a typical design choice is $\det(\mathtt{K}_v) = \det(\mathtt{K}_e)$ so that such a number is 1, as analyzed above.

Other compelling reasons to choose a local projective DSI over a global, uniform DSI are that: (1) for a given amount of memory, it is better to maintain a local map since it allows for higher resolution, and (2) for some applications, such as visual odometry (without loop closure), it suffices to provide a local 3D map.

7 Algorithmic Considerations for Real-Time Performance

The goal of this section is twofold: (1) describe the two-step approach that is used to accelerate computations and (2) quantitatively measure the computational performance of the method (e.g., in number of events processed per second).

Algorithm 1 Efficient event back-projection

Goal: back-project events positions $\{(u_j, v_j)\}$ to the projective DSI.
Input: a projective DSI defined by a virtual camera $\mathtt{P} = (\mathtt{I}|\mathbf{0})$ and N_z depth planes $Z = Z_i$; points $\{(u_j, v_j)\}$ at the current location of the event camera $\mathtt{P}_e = (\mathtt{R}|\mathbf{t})$. Procedure:

1. Map points from the event camera to the virtual camera via a canonical plane $Z = Z_0$, according to homography $\mathtt{H}_{Z_0}$ (see (8)), and store the transferred points $\{(x_j(Z_0), y_j(Z_0))\}$ with full precision.
2. For each depth plane $Z = Z_i$:
   (a) Map points from the event camera to the virtual camera via the plane $Z = Z_i$ using the homography $h_{i0} \equiv \mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ on the stored points: $(x_j(Z_i), y_j(Z_i)) = h_{i0}((x_j(Z_0), y_j(Z_0)))$. See (15).
   (b) Vote for the DSI voxels at positions $\{(x_j(Z_i), y_j(Z_i), Z_i)\}$.

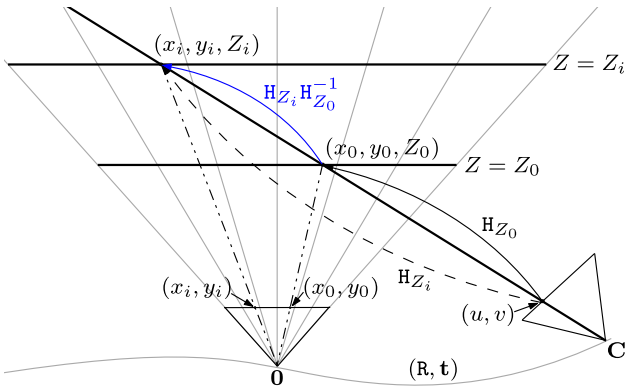


Fig. 9 Efficient event back-projection in Algorithm 1. An event with coordinates (u, v) is mapped onto the depth plane $Z = Z_i$ of the projective DSI in two steps: first, it is mapped to the depth plane $Z = Z_0$ via $\mathtt{H}_{Z_0}$ and then it is mapped to $Z = Z_i$ via the similarity $\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ in (15). In the figure, the notation $x_i = x(Z_i)$ and $y_i = y(Z_i)$ is used for brevity

7.1 Efficient Event Back-Projection onto the DSI

Following (Collins 1996), we populate the DSI using a space-sweep strategy. However, our approach differs from his in the fact that we use a projective DSI instead of a uniform one and we keep the entire DSI in memory, not just a slice of it, for later processing.

The approach is summarized in Algorithm 1. The main idea behind the approach is that to compute the back-projection locations corresponding to the depth plane $Z = Z_i$ it is more efficient to do it in two steps (back-projecting via a depth plane $Z = Z_0$ and then modifying the point locations to take into account the change in Z value) than it is to apply the homography to the original points. This is illustrated in Fig. 9.

The homography to transfer points from the event camera to points on the virtual camera of the DSI via a plane Z_0 is

$\mathtt{H}_{Z_0}$, used in step 1 of Algorithm 1:

$$(x(Z_0), y(Z_0), 1)^\top \sim \mathtt{H}_{Z_0}(u, v, 1)^\top, \tag{11}$$

where we explicitly wrote the dependency of the transferred point $(x(Z_0), y(Z_0))$ with respect to the plane used $Z = Z_0$. Points transferred via another plane, $Z = Z_i$, can be written in terms of the points transferred using $Z = Z_0$ as follows:

$$(x(Z_i), y(Z_i), 1)^\top \sim \mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}\,(x(Z_0), y(Z_0), 1)^\top, \tag{12}$$

where the homography $\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ has a very simple structure: a similarity without rotation. Let us show this. Using the matrix inversion lemma on the first term of

$$\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1} \overset{(8)}{=} \left(\mathtt{R} + \frac{1}{Z_i}\mathbf{t}\mathbf{e}_3^\top\right)^{-1}\left(\mathtt{R} + \frac{1}{Z_0}\mathbf{t}\mathbf{e}_3^\top\right), \tag{13}$$

and the equation of the optical center of the event camera, $(C_x, C_y, C_z)^\top \doteq \mathbf{C} = -\mathtt{R}^\top\mathbf{t}$, we obtain

$$\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1} \sim \mathtt{I} + \frac{Z_0 - Z_i}{Z_0(Z_i - C_z)}\mathbf{C}\mathbf{e}_3^\top. \tag{14}$$

Dividing the homogeneous matrix (14) by its last entry and writing (12) in expanded form gives

$$\begin{aligned} x(Z_i) &= \frac{Z_0}{Z_i}\delta\; x(Z_0) + \frac{1}{Z_i}(1-\delta)C_x, \\ y(Z_i) &= \frac{Z_0}{Z_i}\delta\; y(Z_0) + \frac{1}{Z_i}(1-\delta)C_y, \end{aligned} \tag{15}$$

where $\delta = (Z_i - Z_0)/(Z_0 - C_z)$. Hence, the transformation $\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ in (15) is very simple and fast to compute. This is the advantage of the two-step approach. These equations are similar to the equations in (Collins 1996), except for the additional multiplicative factors Z_0/Z_i and $1/Z_i$.

Accumulating votes in the DSI (line 2b of Algorithm 1) is a process known as forward mapping in image processing (Wolberg 1990, ch. 3), and it can be done in different ways. The simplest one is nearest neighbor: point $(x_j(Z_i), y_j(Z_i))$ votes for a single cell of the depth plane $Z = Z_i$. A better strategy because it mitigates the grid discretization effect is bilinear voting: point $(x_j(Z_i), y_j(Z_i))$ votes for its four nearest cells on the depth plane $Z = Z_i$, splitting the vote according to the distances of $(x_j(Z_i), y_j(Z_i))$ to the integer cell locations, similarly to bilinear interpolation.

7.2 Computational Performance of the Method

The algorithm can be parallelized in a multi-core architecture by making each thread work on a different group of depth planes so that there are no race conditions during voting.

The two-step approach in Algorithm 1 is efficient if events are processed in groups or batches. Theoretically, each event has a different camera pose $\mathtt{P}_e(t)$, but using a different pose to process each event would make any algorithm terribly inefficient. For example, just the simple operation of pose interpolation along the camera trajectory becomes an expensive operation when it is done at the event rate (in the order of 10^5–10^6 events/s). In practice, it is sensible to assume that events, which have microsecond resolution, can be grouped in time so that they are assigned the same camera pose and processed together (i.e., they share the same homography $\mathtt{H}_{Z_0}$, which is the most expensive part to compute). We typically use batches containing a small, fixed number of events (typically, 256 events). The corresponding time interval depends on the event rate (hence the camera motion), but it is typically very small (in the order of 1ms or less).

The number of operations required to compute the DSI grows linearly with the number of depth planes in the voxel grid. Moreover, as explained in Sect. 6.2, for the choice $\det(\mathtt{K}_v) = \det(\mathtt{K}_e)$, the complexity does not depend on the spatial resolution of the depth planes, because in that case only one vote is necessary per depth plane.

Finally, for an efficient implementation with real cameras, it is a good practice to use a look-up-table of undistorted calibrated coordinates (u, v) of the event camera and to use SIMD instructions for matrix multiplications in Algorithm 1.

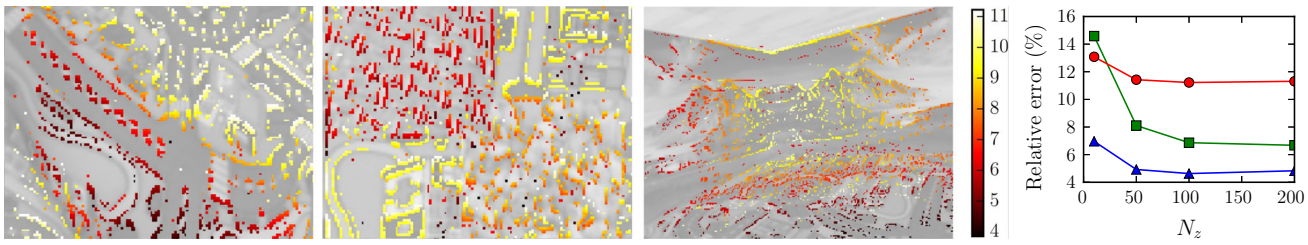
Quantitative Evaluation We measured the speed of our implementation on a Lenovo W541 laptop computer containing an Intel Core i7-4810MQ @2.80 GHz quad-core processor, and a scene recorded in a typical office environment (similar to the first row in Fig. 15) with the DAVIS camera (240 × 180 resolution). The event rate in the scene varied between 250,000 and 900,000 events/s. We used 100 depth planes in the voxel grid, and a batch size of 256 events. On a single core, our implementation can process on average 1.2 million events/s (which is higher than the maximum event rate in the scene, thus running faster than real-time), and on average 4.7 million events/s with the multi-core implementation (using 4 cores).

8 Experiments

We now evaluate the performance of our event-based space sweep method, on both synthetic and real datasets.

8.1 Synthetic Data

We generated three synthetic datasets with ground truth information by means of an event camera simulator (Mueggler et al. 2017). We set the spatial resolution to 240 × 180 pixels, as that of commercial event sensors. The datasets



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

Fig. 10 Synthetic experiments: estimated semi-dense depth maps overlayed over screenshots of the scene, in three datasets **a**–**c**. Depth is colored, from close (red) to far (yellow). Our EMVS algorithm successfully recovers most edges, even without regularization or outlier filtering. **d** Relative depth error as a number of depth planes N_z, in all three datasets: Dunes (blue), 3 planes (red), and 3 walls (green). **a** Dunes. **b** 3 planes. **c** 3 walls. **d** Depth error (Color figure online)

Table 1 Depth estimation accuracy in the synthetic datasets ($N_z = 100$)

	Dunes	3 Planes	3 Walls
Depth range (m)	3.00	1.30	7.60
Mean error (m)	0.14	0.15	0.52
Relative error (%)	4.63	11.31	6.86

also contain intensity images along the event camera viewpoints. However, these are not used in our EMVS algorithm; they are solely shown to aid the visualization of the semi-dense depth maps obtained with our method. The datasets exhibit various depth profiles and motions: *Dunes* consists of a smooth surface (two dunes) and a translating and rotating camera in two degrees of freedom (DOF), *3 planes* shows three planes at different depths (i.e., discontinuous depth profile with occlusions) and a linear camera motion; finally, *3 walls* shows a room with three walls (i.e., a smooth depth profile with sharp transitions) and a general, 6-DOF camera motion.

Our EMVS algorithm was executed on each dataset. First, we evaluated the sensitivity of our method with respect to the number of depth planes N_z used to sample the DSI. In this experiment, the planes in the DSI were equispaced in depth (as opposed to inverse depth) since it provided better results in scenes with finite depth variations. Fig. 10d shows, as a function of N_z, the relative depth error, which is defined as the mean depth error (between the estimated depth map and the ground truth) divided by the depth range of the scene. As expected, the error decreases with N_z, but it stagnates for moderate values of N_z. Hence, from then on, we fixed $N_z = 100$ depth planes. Table 1 reports the mean depth error of the estimated 3D points, as well as the relative depth error for all three datasets. Depth errors are small, in the order of 10% or less, showing the good performance of our EMVS algorithm and its ability to handle occlusions and a variety of surfaces and camera motions.

8.2 Real Data

We also evaluated the performance of our EMVS algorithm on datasets from a DAVIS sensor (Brandli et al. 2014b). The DAVIS outputs, in addition to the event stream, intensity frames as those of a standard camera, at low frame rate (24 Hz).[1] However, our EMVS algorithm does not use the frames; they are displayed here only to illustrate the semi-dense results of the method.

We considered two methods to provide our EMVS algorithm with camera pose information: a motorized linear slider or a visual odometry algorithm on the DAVIS frames. We used the motorized slider to analyze the performance in controlled experiments (since it guarantees very accurate pose information) and a visual odometry algorithm [SVO (Forster et al. 2014)] to show the applicability of our method in handheld (i.e., unconstrained) 6-DOF motions.

8.2.1 High Dynamic Range and High-Speed Experiments

In this section, we show that our EMVS algorithm is able to recover accurate semi-dense structure in two challenging scenarios, namely (1) high-dynamic-range (HDR) illumination conditions and (2) high-speed motion. For this, we place the DAVIS on the motorized linear slider, facing a textured wall at a known constant depth from the sensor. In both experiments, we measure the accuracy of our semi-dense maps against ground truth and demonstrate compelling depth estimation accuracy, in the order of 5% of relative error, which is very good, especially considering the low resolution of the sensor (only 240 × 180 pixels). In order to provide a fair measurement of the raw accuracy of our approach, we did

[1] The DAVIS comprises both a frame camera and an event sensor (DVS) in the same pixel array of size 240 × 180. The frames may be used to simplify intrinsic camera calibration, by applying standard algorithms (Zhang 2000). Otherwise, tailored event-based algorithms, such as (Mueggler et al. 2014), may be applied.

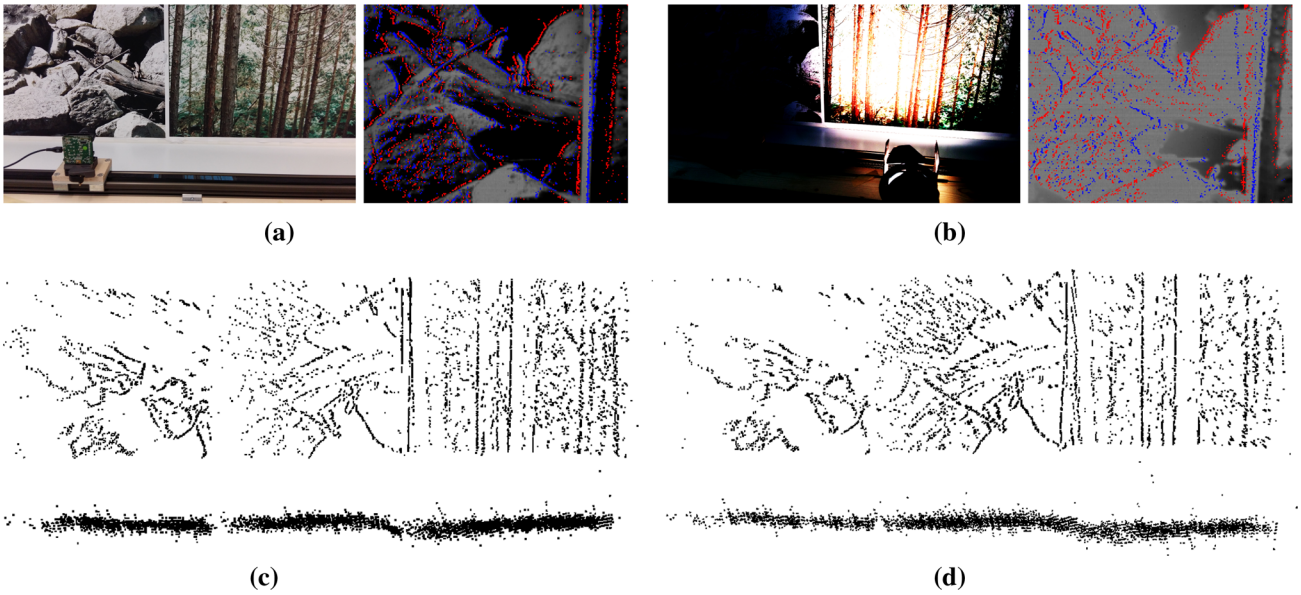


Fig. 11 HDR experiment: top: scene and illumination setups, with the DAVIS on the motorized linear slider (**a**) and a lamp (**b**). Sample frames show under- and over-exposed levels in HDR illumination (**b**). By contrast, the events (overlayed on the frames) are unaffected, due to the high dynamic range of the event sensor. Bottom: reconstructed point clouds. **a** Constant illumination setup. Events on a frame. **b** HDR illumination setup. Events on a frame. **c** Constant illum. 3D points: front and top views. **d** HDR illum. 3D points: front and top views

Table 2 Depth estimation accuracy in the HDR experiment (no post-processing)

Illumination	Close (distance: 23.1 cm)		Far (distance: 58.5 cm)	
	Mean error (cm)	Relative error (%)	Mean error (cm)	Relative error (%)
Constant	1.22	5.29	2.01	4.33
HDR	1.21	5.25	1.87	3.44

not perform any additional post-processing or map cleaning (Sect. 5.2.5) for these quantitative experiments.

High Dynamic Range Experiment We recorded two datasets under the same acquisition conditions except for illumination (Fig. 11): first with constant illumination throughout the scene and, second, with a powerful lamp illuminating only half of the scene. In the latter case, a standard camera cannot cope with the wide intensity variation in the middle of the scene since some areas of the images are under-exposed while others are over-exposed. We performed the HDR experiment with two different wall distances (close and far).

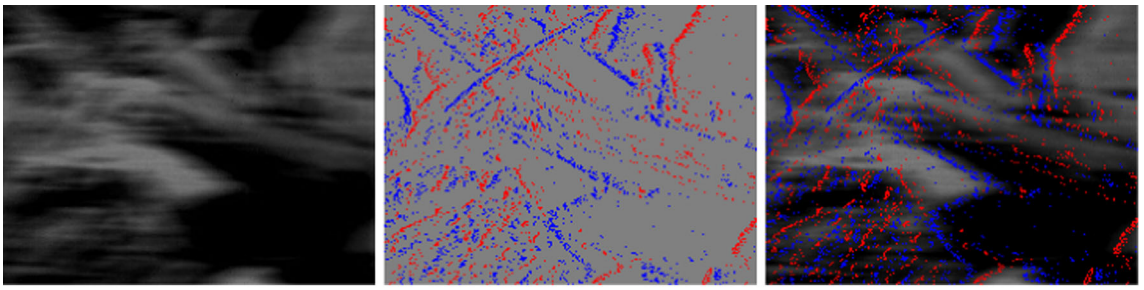
The results of our EMVS algorithm are given in Fig. 11 and Table 2. Observe that the quality of the reconstruction is unaffected by the illumination conditions. In both cases, the EMVS method has a very high accuracy (mean relative error $\approx 5\%$), and also in spite of the low spatial resolution of the sensor or the lack of regularization. Moreover, observe that the accuracy is not affected by the illumination conditions. Hence, we unlocked the high-dynamic range capabilities of the sensor to demonstrate successful HDR depth estimation.

High-Speed Experiment To show that we can exploit the high-speed capabilities of the event sensor for 3D reconstruction, we recorded a dataset with the DAVIS at 40.5 cm from the wall and moving at 0.45 ms. This translated into an apparent speed of 376 pixels/s in the image plane, which caused motion blur in the DAVIS frames (Fig. 12). The motion blur makes the images unintelligible. By contrast, the high temporal resolution of the event stream still accurately captures the edge information of the scene. Our EMVS method produced a 3D reconstruction with a mean depth error of 1.26 cm and a relative error of 4.84 %. The accuracy is consistent with that of previous experiments ($\approx 5\%$), thus supporting the remarkable performance of our method and its capability to exploit the high-speed characteristics of the event sensor.

8.2.2 Three-Dimensional Scenes

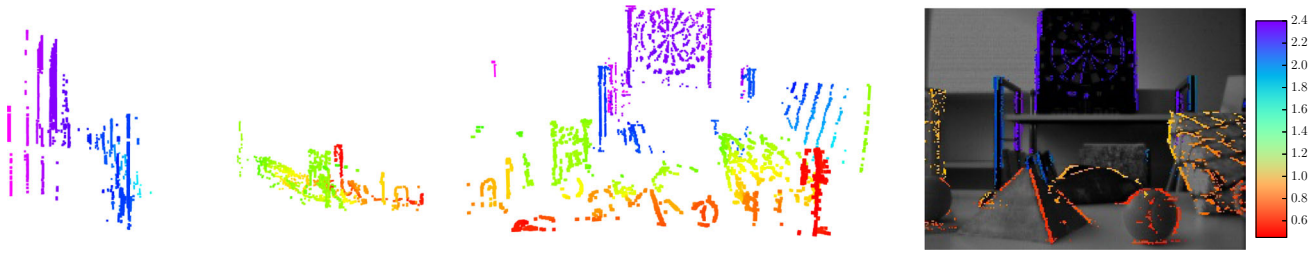
All previous experiments were carried out with nearest-neighbor DSI voting (Sect. 7.1), and lacked structure post-processing (no median or radius filters were applied). The following experiments were performed with bilinear DSI voting and structure post-processing (Sect. 5.2.5).

Figures 13 and 14 show some results obtained by our EMVS method on non-flat scenes. We show both the semi-dense point cloud and its projection on a frame (for better understanding). To ease the visualization, depth is colored from red (close) to blue (far).



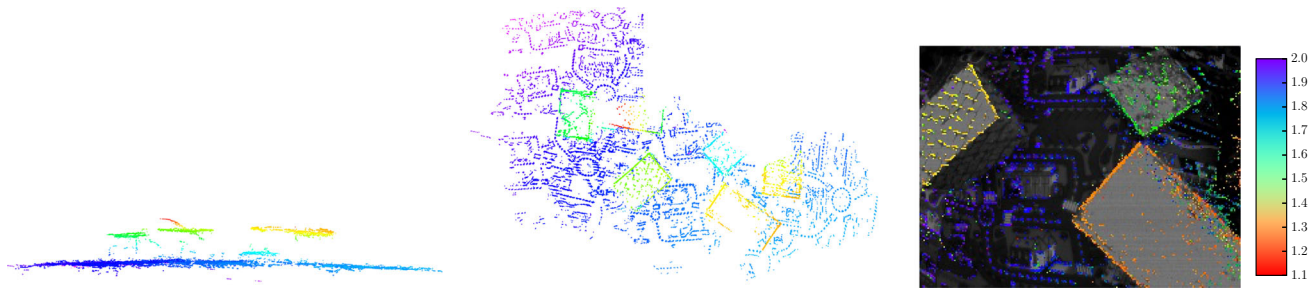
(a) **(b)** **(c)**

Fig. 12 High-speed experiment. Frame and the events from the DAVIS at 376 pixels/s. The frame suffers from motion blur, while the events do not, thus preserving the visual content. **a** Frame (motion blur). **b** Events ($\Delta t = 2ms$). **c** Frame and events



(a) **(b)** **(c)**

Fig. 13 Desk dataset: scene with objects and occlusions. **a** Side view. **b** Front view. **c** Projection on a frame



(a) **(b)** **(c)**

Fig. 14 Boxes dataset: large-scale semi-dense 3D reconstruction with a hand-held DAVIS. **a** Side view. **b** Top view. **c** Projection on a frame

In Fig. 13, the DAVIS was moved in front of a scene containing various objects with different shapes and at different depths. In spite of the large occlusions of the distant objects, generated by the foreground objects, our EMVS algorithm was able to recover the structure of the scene reliably. Figure 14 shows the result of our EMVS algorithm on a larger scale dataset. The sensor was hand-held moved in a big room featuring various textured boxes. Multiple local point clouds were estimated along the trajectory, which were then merged into a global, large-scale 3D reconstruction.

Finally, Fig. 15 shows qualitative results of our approach in various natural environments (both indoors and outdoors) and depth ranges. For each scene, we moved the event camera in a circular fashion, in order to generate events from edges in all directions. We used a visual odometry algorithm (Forster et al. 2014) on the DAVIS frames to estimate the camera motion, and used linear interpolation to provide the camera pose for each event. The DSI was sampled uniformly in inverse depth (as in Fig. 8d) to cope with large depth variations, using between 100 and 150 depth planes. The minimum and maximum depth values were set manually, differently for each experiment to adapt better to the depth range in the scene. We used a median filter of size 15 pixels in the semi-dense depth maps. Then, in the point clouds, we used a radius filter of size equal to 5% of the mean scene depth, and a minimum number of neighbors of $N = 4$ to remove isolated points.

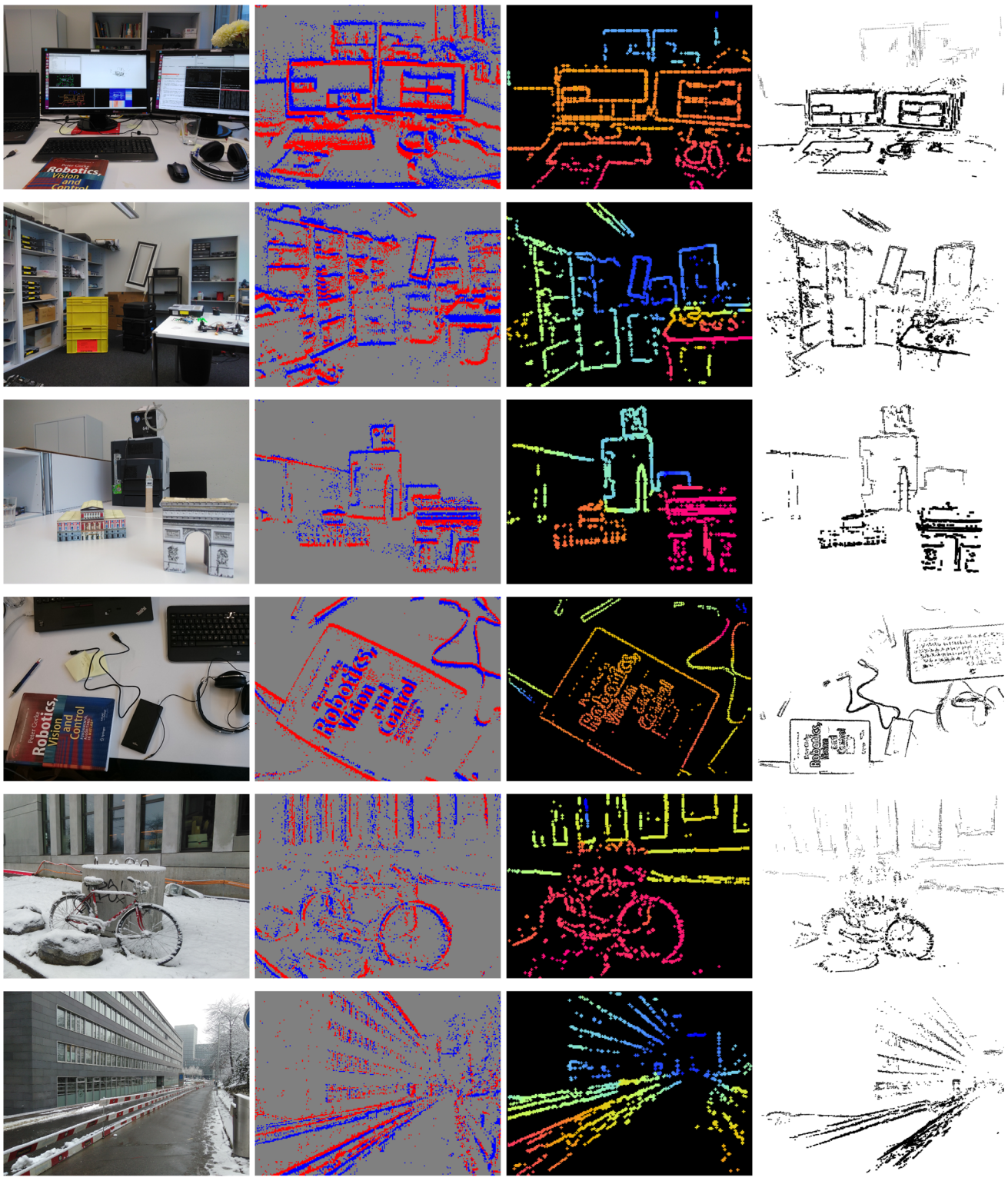


Fig. 15 Semi-dense 3D reconstructions of several scenes with a hand-held DAVIS. **a** Scene, **b** events (positive and negative), **c** semi-dense depth map, pseudo-colored from red (close) to blue (far), **d** point cloud (Color figure online)

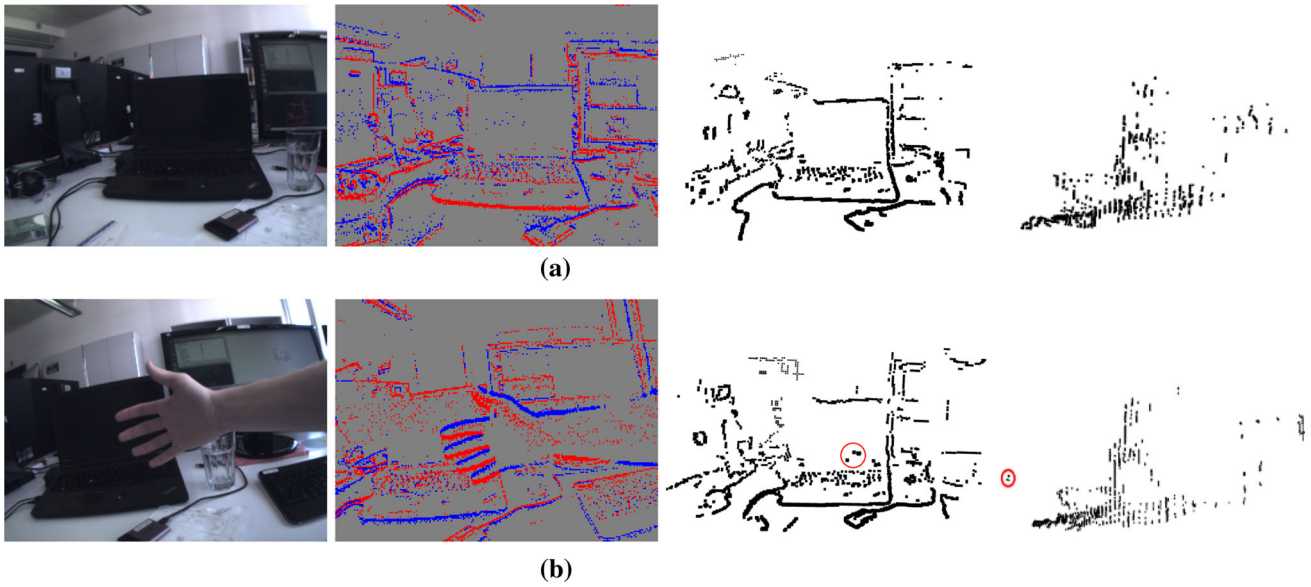


Fig. 16 Effect of a dynamic scene: the same scene and camera motions were used to create the 3D reconstructions shown in Fig. 16a, b. However, in Fig. 16b, a hand was continuously waived in front of the sensor, generating a large number of outlier events. Nonetheless, our algorithm is barely affected and both 3D reconstructions are similarly good. **a** Static scene. From left to right: Preview image; Preview of the events; 3D reconstruction (front view and side view). **b** Dynamic scene with hand continuously waving in front of the sensor. Apart from a small number of outlier 3D points generated by the moving hand (circled in red), our algorithm is able to reconstruct the scene as well as in the static case

8.2.3 Effect of Dynamic Objects

In this section, we show that the proposed method is robust to the presence of moving objects in the scene. In Fig. 16, we compare two 3D reconstructions obtained by our method, with and without the presence of a moving, occluding object in front of the sensor, and show that they are qualitatively equivalent. Indeed, the moving object does not generate votes with a spatial persistence in the DSI, and so the votes are treated as noise and are filtered out by the Adaptive Gaussian Thresholding. In both cases, the length of the sequence of events used for reconstruction was the same, and the camera motion was very similar.

8.2.4 Effect of Light Changes

Due to the fact that the event camera reacts to light changes, one might think that strong temporal light changes would perturb the performance of the algorithm. In Fig. 17, we show that this is not the case, e.g., the proposed approach is robust to strong light changes. The reason of this robustness is twofold: (1) the sensor itself, thanks to its high dynamic range, is to a large extent invariant to illumination conditions (Fig. 17b), and (2) strong light changes generate a burst of events across the whole sensor, which results in simply adding a constant offset to the DSI, which does not affect the adaptive thresholding step.

9 Discussion

This work has focused on multi view stereo with a single moving event camera. Our goal was to show that 3D reconstruction with a single event camera is possible, and that we do not need to solve the data association problem or estimate image intensity. The results showed that (1) the method provides accurate results, being able to unlock the capabilities of the sensor in challenging scenarios (HDR and high-speed) where standard cameras fail, (2) the method can handle inaccurate poses (the experiments with poses provided by a frame-based visual odometry algorithm show visually appealing results, which suggests that the method is robust to pose uncertainty), and (3) the method is computationally efficient and can run on the CPU, without additional dedicated hardware.

The applicability of multi view stereo depends on the availability of pose information, which in the experiments was provided by an external tracking algorithm or system. However, this is not a limitation, since the method can be used in combination with an event-based motion estimation algorithm, as shown in (Rebecq et al. 2017), thus removing the need for an external pose estimator.

The major limitation of the proposed approach is that it provides depth values on a discrete set, thus the resolution is limited by the number of depth planes used, N_z. The computational complexity of the method is linear in the number of depth planes, $O(N_z)$, while the discretization error is propor-

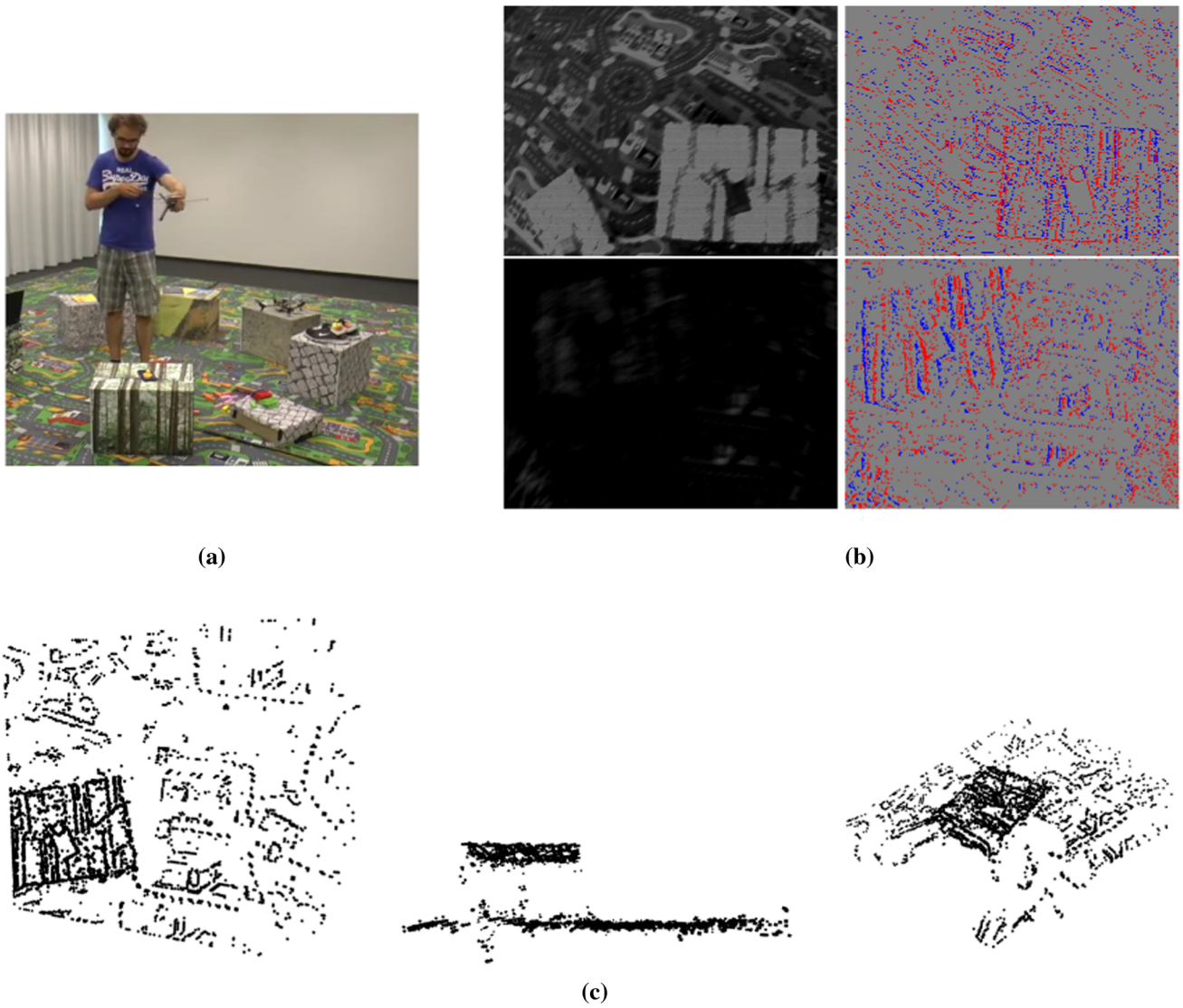


Fig. 17 Effect of strong light changes: despite switching off the light in the middle of the sequence (Fig. 17b), the obtained 3D reconstruction remains unaffected and of high quality (Fig. 17c). **a** Preview of the scene. **b** Visualization of frames from a standard camera, compared to the events. Top row: light ON; Bottom row: light OFF. The events are unaffected by the strong light change. **c** From left to right: Top view, side view, and perspective view of the reconstructed 3D scene

tional to $1/N_z$. Hence, there is an accuracy vs. computation effort tradeoff. However, increasing N_z does not improve the total accuracy, as shown in Fig. 10d, since the accuracy also depends on the triangulation uncertainty. The discretization effect has also an undesirable influence when merging point clouds from different keyframes: the same 3D point may be extracted from two different DSIs, but the 3D positions may not agree since they are rounded to the position of the center of a voxel. A continuous formulation, in the form of depth filters (Vogiatzis and Hernández 2011; Pizzoli et al. 2014), where depth can have any positive real value, would be more desirable, and it is a line of future work.

Investigating methods to regularize semi-dense depth maps is also interesting and of large applicability since semi-dense depth maps are used not only in our method but also in state-of-the-art visual odometry algorithms for standard cameras, such as LSD-SLAM (Engel et al. 2014) and DSO (Engel et al. 2017). We showed how simple processing techniques, such as median filtering, are effective tools to improve the quality of the reconstructions, but more principled methods would also be desirable.

10 Conclusion

We introduced the EMVS problem, and provided a simple and elegant solution to it that exploits the natural strengths of the sensor, and runs in real-time on a CPU. We validated our algorithm on both synthetic and real data, for various motions and scenes, showing very accurate 3D reconstructions (rela-

tive depth error of 5%) in spite of the low resolution of the sensor and the high amount of noise typical of event cameras. We believe this work is a major step towards building 3D reconstruction algorithms robust to speed (the events do not suffer from motion blur), and HDR illumination. This paper further highlights the potential of event cameras and the astounding possibilities it opens to computer vision.

Acknowledgements This research was funded by the DARPA FLA Program, the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant.

A Relation of Area Elements due to a 2D Homography

This section provides a useful result on how a 2D transformation given by a homography affects the area element.

Result 1 (Jacobian of a Homography) *Let* $\mathtt{H}$ *be a 2D homography transforming points* $\mathbf{x} \doteq (x, y, 1)^\top$ *to points* $\mathbf{x}' \doteq (x', y', 1)^\top$ *in homogeneous coordinates:* $\mathbf{x}' \sim \mathtt{H}\mathbf{x}$*, where* $\sim$ *means equality up to a non-zero scale factor. The determinant of the Jacobian of the transformation* $(x, y) \overset{\mathtt{H}}{\mapsto} (x', y')$ *(in Euclidean coordinates),*

$$\mathtt{J} \doteq \frac{\partial(x', y')}{\partial(x, y)} = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{pmatrix} \tag{16}$$

is

$$\det(\mathtt{J}) = \frac{\det(\mathtt{H})}{(\mathbf{e}_3^\top \mathtt{H}\mathbf{x})^3}, \tag{17}$$

where $\mathbf{e}_3 = (0, 0, 1)^\top$ *is the 3-rd vector of the canonical basis in* $\mathbb{R}^3$.

The determinant of the Jacobian (17) provides the relation between the area elements in (x, y) and in (x', y') according to the geometric transformation given by the homography $\mathtt{H}$,

$$dA' \doteq dx'dy' = \det(\mathtt{J})\, dxdy = \det(\mathtt{J})\, dA, \tag{18}$$

as illustrated in Fig. 18.

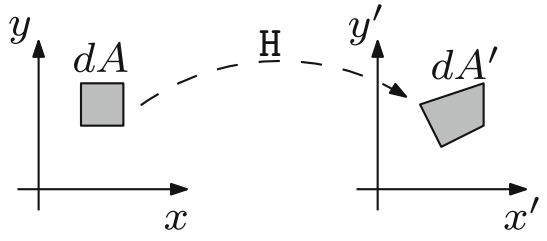


Fig. 18 Result 1. A homography $\mathtt{H}$ maps points to points and lines to lines. Area elements are transformed according to $dA' = |\mathtt{J}|dA$, where $\mathtt{J}$ is the Jacobian of the homography $\mathtt{H}$

Proof Let $\mathtt{H} = (h_{ij})$ be the homogeneous matrix of the homography, and let $\mathbf{h}_3^\top \doteq \mathbf{e}_3^\top \mathtt{H}$ be its third row. Writing out explicitly the transformed variables

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}, \tag{19}$$

we may compute the four elements of the Jacobian matrix (17):

$$\mathtt{J} = \frac{1}{\mathbf{h}_3^\top \mathbf{x}} \begin{pmatrix} h_{11} - x'h_{31} & h_{12} - x'h_{32} \\ h_{21} - y'h_{31} & h_{22} - y'h_{32} \end{pmatrix} \tag{20}$$

Next, we compute the determinant of this matrix. Noting that $(h_{11} - x'h_{31})(h_{22} - y'h_{32}) - (h_{12} - x'h_{32})(h_{21} - y'h_{31}) = \mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2))$ is a mixed product in terms of the first two columns of $\mathtt{H}$, with $\mathbf{e}_1 = (1, 0, 0)^\top$ and $\mathbf{e}_2 = (0, 1, 0)^\top$, gives

$$\det(\mathtt{J}) = \frac{1}{(\mathbf{h}_3^\top \mathbf{x})^2}\, \mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2)). \tag{21}$$

Substituting $\mathbf{x}' = \mathtt{H}\mathbf{x}/(\mathbf{h}_3^\top \mathbf{x})$ in the mixed product $\mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2)) = \det(\mathbf{x}', \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2)$ and using the properties of the determinant, $\det(\mathtt{H}\mathbf{x}, \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2) = \det(\mathtt{H}) \det(\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2) = \det(\mathtt{H})$, gives the desired result (17):

$$\det(\mathtt{J}) \overset{(21)}{=} \frac{\det(\mathtt{H}\mathbf{x}, \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2)}{(\mathbf{h}_3^\top \mathbf{x})^3} = \frac{\det(\mathtt{H})}{(\mathbf{e}_3^\top \mathtt{H}\mathbf{x})^3}. \tag{22}$$

□

A.1 Planar Homography

Next, we particularize the previous general Result 1 to the case of a planar homography induced by a plane in space.

Let us consider (1) two finite cameras (i.e., whose optical centers are not at infinity) with projection matrices given by $\mathtt{P} = (\mathtt{I}|\mathbf{0})$ and $\mathtt{P}' = (\mathtt{R}|\mathbf{t})$ in calibrated coordinates, and (2) a plane not passing through the optical centers of the cameras, with homogeneous coordinates $\boldsymbol{\pi} = (a, b, c, d)^\top = (\mathbf{n}^\top, d)^\top$, where $\mathbf{n}$ is the unit normal to the plane. The optical centers of $\mathtt{P}$ and $\mathtt{P}'$ are $\mathbf{0}$ and $\mathbf{C} = -\mathtt{R}^\top \mathbf{t}$, respectively. The planar homography from the image plane of $\mathtt{P}$ to the image plane of $\mathtt{P}'$ via the plane $\boldsymbol{\pi}$, such that $\mathbf{x}' \sim \mathtt{H}\mathbf{x}$, is

$$\mathtt{H}_{\boldsymbol{\pi}}(\mathtt{P}, \mathtt{P}') \sim \mathtt{R} - \frac{1}{d}\mathbf{t}\mathbf{n}^\top = \mathtt{R}\left(\mathtt{I} + \frac{1}{d}\mathbf{C}\mathbf{n}^\top\right), \tag{23}$$

where $\mathtt{I}$ is the identity matrix.

The planar homography from $\mathtt{P}'$ to $\mathtt{P}$ via the plane $\boldsymbol{\pi}$ is given by the inverse of (23):

$$\mathtt{H}_{\boldsymbol{\pi}}(\mathtt{P}', \mathtt{P}) = \mathtt{H}_{\boldsymbol{\pi}}^{-1}(\mathtt{P}, \mathtt{P}') \sim \left(\mathtt{I} - \frac{1}{d + \mathbf{n}^\top \mathbf{C}}\mathbf{C}\mathbf{n}^\top\right)\mathtt{R}^\top. \tag{24}$$

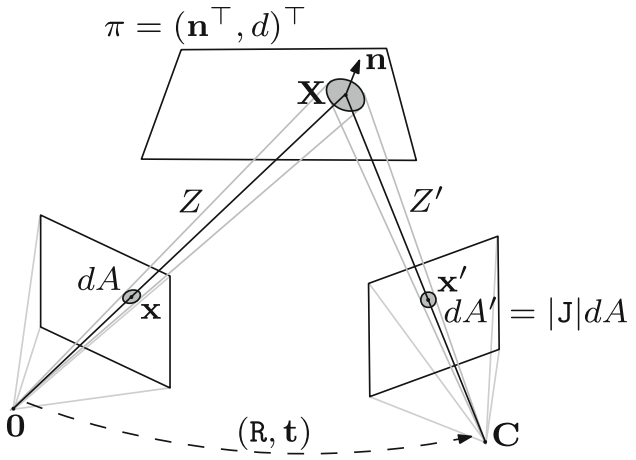


Fig. 19 Result 2. Relation of area elements induced by a planar homography: $dA' = |\mathtt{J}|dA$, where $\mathtt{J}$ is the Jacobian of the planar homography, and Z, Z' are the depths of the scene point $\mathbf{X}$ with respect to the two cameras, respectively

Result 2 (Jacobian of a Planar Homography) *For a planar homography* (23)*, Result* 1 *becomes*

$$\det(\mathtt{J}) = \left(\frac{Z}{Z'}\right)^3 \left(1 + \frac{\mathbf{C}\cdot\mathbf{n}}{d}\right), \tag{25}$$

where Z and Z' are the depths of the point $\mathbf{X} \in \boldsymbol{\pi}$, projecting on $\mathbf{x}$ and $\mathbf{x}'$, with respect to cameras $\mathtt{P}$ and $\mathtt{P}'$, respectively. This is illustrated in Fig. 19*.*

Proof Let us compute the numerator and denominator of (17). Applying $\det(\mathtt{R}) = 1 = \det(\mathtt{I})$ and the matrix determinant lemma to (23) gives

$$\det(\mathtt{H}) = \det(\mathtt{R}) \det\left(\mathtt{I} + \frac{1}{d}\mathbf{C}\mathbf{n}^\top\right) = 1 + \frac{1}{d}\mathbf{n}^\top\mathbf{C}. \tag{26}$$

A point $\mathbf{X} \doteq (X, Y, Z)^\top$ lies on the plane $\boldsymbol{\pi}$ if it satisfies

$$\mathbf{n}^\top\mathbf{X} + d = 0. \tag{27}$$

The point $\mathbf{X}$ expressed in the frame of $\mathtt{P}'$ becomes

$$(X', Y', Z')^\top \doteq \mathbf{X}' = \mathtt{R}\mathbf{X} + \mathbf{t} = \mathtt{R}(\mathbf{X} - \mathbf{C}). \tag{28}$$

Since $\mathbf{x}Z = (x, y, 1)^\top Z = \mathbf{X}$ and

$$\mathtt{H}\mathbf{X} \overset{(23)}{=} \mathtt{R}\left(\mathbf{X} + \frac{\mathbf{X}\cdot\mathbf{n}}{d}\mathbf{C}\right) \overset{(27)}{=} \mathtt{R}(\mathbf{X} - \mathbf{C}) \overset{(28)}{=} \mathbf{X}', \tag{29}$$

the denominator of (17) is given in terms of $\mathbf{e}_3^\top\mathtt{H}\mathbf{x} \overset{(29)}{=} \mathbf{e}_3^\top\mathbf{X}'/Z$. Substituting this result and (26) in (17) gives (25). □

References

Bardow, P., Davison, A. J., & Leutenegger, S. (2016). Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.102.

Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H., & Bartolozzi, C. (2014). Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(2), 407–417. https://doi.org/10.1109/TNNLS.2013.2273537.

Benosman, R., Ieng, S.-H., Clercq, C., Bartolozzi, C., & Srinivasan, M. (2012). Asynchronous frameless event-based optical flow. *Neural Networks*, *27*, 32–37. https://doi.org/10.1016/j.neunet.2011.11.001.

Brandli, C., Muller, L., & Delbruck, T. (2014a) Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor. In *International Symposium Circuits and Systems (ISCAS)* (pp. 686–689). https://doi.org/10.1109/ISCAS.2014.6865228.

Brandli, C., Berner, R., Yang, M., Liu, S.-C., & Delbruck, T. (2014b). A 240 × 180 130 dB 3us latency global shutter spatiotemporal vision sensor. *IEEE Jorunal of Solid-State Circuits*, *49*(10), 2333–2341. https://doi.org/10.1109/JSSC.2014.2342715.

Camunas-Mesa, L. A., Serrano-Gotarredona, T., Ieng, S. H., Benosman, R. B., & Linares-Barranco, B. (2014). On the use of orientation filters for 3D reconstruction in event-driven stereo vision. *Frontiers in Neuroscience*, *8*, 48. https://doi.org/10.3389/fnins.2014.00048.

Censi, A., & Scaramuzza, D. (2014). Low-latency event-based visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/IROS.2016.7758089.

Collins, R. T. (1996). A space-sweep approach to true multi-image matching. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition* (pp. 358–363). https://doi.org/10.1109/CVPR.1996.517097.

Cook, M., Gugelmann, L., Jug, F., Krautz, C., & Steger, A. (2011). Interacting maps for fast visual interpretation. In *International Joint Conference Neural Networks (IJCNN)* (pp. 770–776). https://doi.org/10.1109/IJCNN.2011.6033299.

Delbruck, T. (2016). Neuromorophic vision sensing and processing. In *European Solid-State Device Research Conferernce (ESSDERC)* (pp. 7–14). https://doi.org/10.1109/ESSDERC.2016.7599576.

Delbruck, T., & Lichtsteiner, P. (2007). Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 845–848). https://doi.org/10.1109/ISCAS.2007.378038.

Delbruck, T., & Lang, M. (2013). Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*,. https://doi.org/10.3389/fnins.2013.00223.

Drazen, D., Lichtsteiner, P., Hafliger, P., Delbruck, T., & Jensen, A. (2011). Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids*, *51*(5), 1465–1469. https://doi.org/10.1007/s00348-011-1207-y.

Engel, J., Schöps, J., & Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*. https://doi.org/10.1007/978-3-319-10605-2_54.

Engel, J., Koltun, V., & Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PP*(99), 1. https://doi.org/10.1109/TPAMI.2017.2658577.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 15–22). https://doi.org/10.1109/ICRA.2014.6906584.

Gallego, G, Lund, E. A., Mueggler, E., Rebecq, H., Delbruck, T., & Scaramuzza, D. (2017). Event-based, 6-DOF camera tracking from photometric depth maps. In *IEEE Transactions on Pattern Analysis*

and Machine Intelligence, 2017. https://doi.org/10.1109/TPAMI.2017.2769655.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press.

Kim, H., Handa, A., Benosman, R., Ieng, S.-H., & Davison, A. J. (2014). Simultaneous mosaicing and tracking with an event camera. In *British Machine Vision Conference (BMVC)*. https://doi.org/10.5244/C.28.26.

Kim, H., Leutenegger, S., & Davison, A. J. (2016). Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *European Conference on Computer Vision (ECCV)* (pp. 349–364). https://doi.org/10.1007/978-3-319-46466-4_21.

Kogler, J., Humenberger, M., & Sulzbachner, C. (2011a). Event-based stereo matching approaches for frameless address event stereo data. In *International Symposium on Advances in Visual Computing (ISVC)* (pp. 674–685). https://doi.org/10.1007/978-3-642-24028-7_62.

Kogler, J., Sulzbachner, C., Humenberger, M., & Eibensteiner, F. (2011b). Address-event based stereo vision with bio-inspired silicon retina imagers. In *Advances in Theory and Applications of Stereo Vision* (pp. 165–188). InTech. https://doi.org/10.5772/12941.

Kueng, B., Mueggler, E., Gallego, G., & Scaramuzza, D. (2016). Low-latency visual odometry using event-based feature tracks. In *IEEE/RSJ International Conference on IIntelligent Robots and Systems (IROS)* (pp. 16–23). Daejeon, Korea. https://doi.org/10.1109/IROS.2016.7758089.

Lagorce, X., Orchard, G., Gallupi, F., Shi, B. E., & Benosman, R. (2016). HOTS: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,. https://doi.org/10.1109/TPAMI.2016.2574707.

Lee, J., Delbruck, T., Park, P. K. J., Pfeiffer, M., Shin, C.-W., Ryu, H., & Kang, B. C. (2012). Live demonstration: Gesture-based remote control using stereo pair of dynamic vision sensors. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. https://doi.org/10.1109/ISCAS.2012.6272144.

Lee, J. H., Delbruck, T., Pfeiffer, M., Park, P. K. J., Shin, C.-W., Ryu, H., et al. (2014). Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(12), 2250–2263. https://doi.org/10.1109/TNNLS.2014.2308551.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, *43*(2), 566–576. https://doi.org/10.1109/JSSC.2007.914337.

Litzenberger, M., Belbachir, A. N., Donath, N., Gritsch, G., Garn, H., Kohn, B., Posch, C., & Schraml, S. (2006). Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor. In *IEEE Intelligent Transportation Systems Conference* (pp. 653–658). https://doi.org/10.1109/ITSC.2006.1706816.

Matsuda, N., Cossairt, O., & Gupta. M. (2015). MC3D: Motion contrast 3D scanning. In *IEEE International Conference on Computational Photography (ICCP)* (pp. 1–10). https://doi.org/10.1109/ICCPHOT.2015.7168370.

Mueggler, E., Huber, B., & Scaramuzza, D. (2014). Event-based, 6-DOF pose tracking for high-speed maneuvers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2761–2768). https://doi.org/10.1109/IROS.2014.6942940.

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., & Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *International Journal of Robotics Research*, *36*, 142–149. https://doi.org/10.1177/0278364917691115.

Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., & Benosman, R. (2015). HFirst: A temporal approach to object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(10), 2028–2040. https://doi.org/10.1109/TPAMI.2015.2392947.

Piatkowska, E., Belbachir, A. N., & Gelautz, M. (2013). Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach. In *International Conference on Computer Vision Workshops (ICCVW)* (pp. 45–50). https://doi.org/10.1109/ICCVW.2013.13.

Piatkowska, E., Belbachir, A. N., Schraml, S., & Gelautz, M. (2012). Spatiotemporal multiple persons tracking using dynamic vision sensor. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshop* (pp. 35–40). https://doi.org/10.1109/CVPRW.2012.6238892.

Pizzoli, M., Forster, C., & Scaramuzza, D. (2014). REMODE: Probabilistic, monocular dense reconstruction in real time. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2609–2616). https://doi.org/10.1109/ICRA.2014.6907233.

Rebecq, H., Gallego, G., & Scaramuzza, D. (2016). EMVS: Event-based multi-view stereo. In *British Machine Vision Conference (BMVC)*. https://doi.org/10.5244/C.30.63.

Rebecq, H., Horstschäfer, T., Gallego, G., & Scaramuzza, D. (2017). EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time. *IEEE Robotics and Automation Letters*, *2*, 593–600. https://doi.org/10.1109/LRA.2016.2645143.

Reinbacher, C., Graber, G., & Pock, T. (2016). Real-time intensity-image reconstruction for event cameras using manifold regularisation. In *British Machine Vision Conference (BMVC)*. https://doi.org/10.5244/C.30.9.

Rogister, P., Benosman, R., Ieng, S.-H., Lichtsteiner, P., & Delbruck, T. (2012). Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(2), 347–353. https://doi.org/10.1109/TNNLS.2011.2180025.

Rueckauer, B., & Delbruck, T. (2016). Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in Neuroscience*,. https://doi.org/10.3389/fnins.2016.00176.

Rusu, R. B.,& Cousins, S. (2011). 3D is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China. https://doi.org/10.1109/ICRA.2011.5980567.

Schraml, S., Belbachir, A. N., & Bischof, H. (2015). Event-driven stereo matching for real-time 3D panoramic vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 466–474). https://doi.org/10.1109/CVPR.2015.7298644.

Schraml, S., Belbachir, A. N., Milosevic, N., & Schön, P. (2010). Dynamic stereo vision system for real-time tracking. In *IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1409–1412). https://doi.org/10.1109/ISCAS.2010.5537289.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2006.19.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*., Texts in computer science London: Springer.

Szeliski, R., & Golland, P. (1999). Stereo matching with transparency and matting. *International Journal of Computer Vision*, *32*(1), 45–61. https://doi.org/10.1023/A:1008192912624.

Vogiatzis, G., & Hernández, C. (2011). Video-based, real-time multi view stereo. *Image and Vision Computing*, *29*(7), 434–441. https://doi.org/10.1016/j.imavis.2011.01.006.

Weikersdorfer, D., & Conradt, J. (2012). Event-based particle filtering for robot self-localization. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 866–870). https://doi.org/10.1109/ROBIO.2012.6491077.

Weikersdorfer, D., Hoffmann, R., & Conradt, J. (2013). Simultaneous localization and mapping for event-based vision systems. In

International Conference on Computer Vision Systems (ICVS) (pp. 133–142). https://doi.org/10.1007/978-3-642-39402-7_14.

Wiesmann, G., Schraml, S., Litzenberger, M., Belbachir, A. N., Hofstatter, M., & Bartolozzi, C. (2012). Event-driven embodied system for feature extraction and object recognition in robotic applications. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshop* (pp. 76–82). https://doi.org/10.1109/CVPRW.2012.6238898.

Wolberg, G. (1990). *Digital image warping*. California: Wiley-IEEE Computer Society Press.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334. https://doi.org/10.1109/34.888718. ISSN 0162-8828.