Event-Driven Stereo Visual Tracking Algorithm to Solve Object Occlusion

Luis A. Camuñas-Mesa, Teresa Serrano-Gotarredona, *Member, IEEE*, Sio-Hoi Ieng, Ryad Benosman, and Bernabé Linares-Barranco, *Fellow, IEEE*

***Abstract*— Object tracking is a major problem for many computer vision applications, but it continues to be computationally expensive. The use of bio-inspired neuromorphic event-driven dynamic vision sensors (DVSs) has heralded new methods for vision processing, exploiting reduced amount of data and very precise timing resolutions. Previous studies have shown these neural spiking sensors to be well suited to implementing single-sensor object tracking systems, although they experience difficulties when solving ambiguities caused by object occlusion. DVSs have also performed well in 3-D reconstruction in which event matching techniques are applied in stereo setups. In this paper, we propose a new event-driven stereo object tracking algorithm that simultaneously integrates 3-D reconstruction and cluster tracking, introducing feedback information in both tasks to improve their respective performances. This algorithm, inspired by human vision, identifies objects and learns their position and size in order to solve ambiguities. This strategy has been validated in four different experiments where the 3-D positions of two objects were tracked in a stereo setup even when occlusion occurred. The objects studied in the experiments were: 1) two swinging pens, the distance between which during movement was measured with an error of less than 0.5%; 2) a pen and a box, to confirm the correctness of the results obtained with a more complex object; 3) two straws attached to a fan and rotating at 6 revolutions per second, to demonstrate the high-speed capabilities of this approach; and 4) two people walking in a real-world environment.**

***Index Terms*— Address event representation (AER), event-driven processing, neuromorphic vision, object occlusion, object tracking, stereo vision.**

I. Introduction

REAL-TIME object tracking is a fundamental task in many computer vision applications, such as motion-based recognition [1], automated surveillance [2], [3], human–computer interaction [4]–[6], traffic monitoring [7], vehicle navigation [8], or augmented reality [9]. However, tracking fast objects in nonstructured environments is still a highly demanding, computationally expensive task that imposes a critical tradeoff between frame rate and computational load.

In recent years, the emergence of bio-inspired event-driven neuromorphic dynamic vision sensors (DVSs) [10]–[12] has introduced new methodologies for addressing problems like object tracking [13]. In a DVS artificial retina, each pixel operates autonomously and sends an output spike (event) whenever it senses a change of light greater than a preset threshold. This alternative approach exploits high temporal resolution and sparsity of spiking events to develop new algorithms capable of processing visual information more efficiently. Different studies have demonstrated the suitability of single-sensor event-driven vision for object tracking, both in hardware [14], [15] and software implementations [16]–[18]. Most approaches encountered difficulties when dealing with object occlusions, due to ambiguous events generated in overlapping regions. For example, Ni *et al.* [16] used the velocity of tracked objects over time to address occlusion, although a prior knowledge of the objects' geometry was needed to initialize tracking.

Stereo vision is another task where event-based neuromorphic processing offers great advantages when extracting 3-D information from setups with two or more DVSs. Exploitation of the high temporal resolution in DVSs was proposed in [19] and [20] for stereo matching. However, both matching techniques presented in these works were negatively impacted by occlusions. When an occlusion occurs, the moving structure is not seen by one of the sensors, and events are wrongly paired, resulting in wrongly reconstructed 3-D points.

Considering the problem caused for object tracking and stereo matching algorithms by occlusion scenarios, we propose a new procedure for processing both subtasks simultaneously, providing each subalgorithm with extra information that can be used dynamically by the other to improve its performance. Thus, while the tracking subalgorithm attaches clustering information to individual events, which can be used as an additional restriction to enhance stereo matching, the matching subalgorithm learns 3-D information about the events that can be used to improve tracking. The high temporal resolution and sparsity of data associated with DVSs also make this method very well suited to high-speed stimuli. In this

This work was supported in part by the Spanish research under Grant TEC2012-37868-C04-01 (BIOSENSE) and Grant TEC2015-63884-C2-1-P (COGNET) (with support from the European Regional Development Fund), and in part by the Andalusian research under Grant TIC-6091 (NANONEURO). The authors benefited by the interaction provided by the CapoCaccia Cognitive Neuromorphic Engineering Workshop, Sardinia, Italy, and by the Telluride Neuromorphic Cognition Engineering Workshop, Telluride, Colorado. The work of L. A. Camuñas-Mesa was supported by the Spanish research fellowship "Juan de la Cierva." *(Corresponding author: Luis A. Camuñas-Mesa.)*

L. A. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco are with the Instituto de Microelectrónica de Sevilla IMSE-CNM, CSIC y la Universidad de Sevilla, 41092 Sevilla, Spain (e-mail: camunas@imse-cnm.csic.es).

S.-H. Ieng and R. Benosman are with UMR S968 Inserm/UPMC/CNRS 7210, Institut de la Vision, Université de Pierre et Marie Curie, 75005 Paris, France.

paper, we demonstrate how this novel procedure can track the position of two different objects, one of which crosses in front of the other several times at high speed, by reconstructing the 3-D positions of both objects.

This has traditionally constituted a problem for conventional frame-based cameras, and multiple camera structures are actually largely explored. The basic technique is the same: depth information is used to identify which tracked object a pixel belongs to. However, these algorithms track objects across frames by building models that are refined/stabilized using techniques based on Bayesian inference (Kalman filter, more complex Markovian models) [21]–[23]. This is done by focusing on clusters of pixels in each frame, frame per frame, i.e., packets of pixels are assumed to be seen simultaneously. This approach is not directly translatable to DVSs unless by building frames that would ruin all the benefits provided by this kind of sensor (i.e., high equivalent acquisition frequency). With our method, we managed to track and reconstruct 3-D objects without resorting to complex costly Bayesian inference techniques.

Some algorithms have recently been developed to perform object recognition based on feature extraction [24]–[26]. The aim of those works was to classify objects (like handwritten digits, characters, or symbols) by extracting a relatively large number of features. The goal of the work presented in this paper, however, is to identify objects and track them, even under occlusion. Classification tasks are not included, so complex feature extraction strategies are not needed. The proposed algorithm only extracts and learns basic features like the position and size of objects, in order to track them even when they are not visible.

This paper is organized as follows. After a brief description of the neuromorphic DVS and its main properties in Section II, Section III concisely describes the previous event-based tracking algorithm that inspired this paper. Section IV depicts the fundamentals of stereo matching and Section V describes in detail the proposed algorithm that integrates stereo matching and object tracking. Section VI presents the results obtained to validate this work, and, finally, some conclusions are given.

II. Dynamic Vision Sensor

The DVS used in this paper is an address event representation (AER) neuromorphic silicon retina with 128×128 pixels and increased contrast sensitivity, capable of detecting contrasts as low as 1.5% [12]. The DVS output consists of asynchronous AER events representing changes in the sensed relative light intensity. Each pixel independently detects the changes in log intensity larger than a programmable threshold that have taken place since the last emitted event. A new event is generated whenever it reaches $\theta_{\text{ev}} = |I(t) - I(t_{\text{prev}})|/I(t)$, where θ_{ev} represents the minimum detectable temporal contrast, given by the relative change between the present value of the photocurrent $I(t)$ and its value at the previous event time $I(t_{\text{prev}})$.

The most important property of these sensors is that pixel information is obtained not synchronously at a fixed frame rate δt, but asynchronously driven by data at fixed relative light increments θ_{ev}, as shown in Fig. 1. Fig. 1 represents the photocurrent transduced by a single pixel in a DVS, and the generated train of events. Every time the log of the photocurrent increases or decreases by a fixed amount θ_{ev}, a positive or negative event is generated, where the polarity of the event represents the sign of this variation. These events are transmitted off-chip, timestamped, and sent to a computer using a standard USB connection. In a stereo setup, it would be possible to consider two pixels in two different sensors, configured so that both pixels could sense equivalent activity. Even with this arrangement, however, the trains of events generated by the two pixels would be slightly different due to mismatch between the two (caused by intradie and interdie variations) and the different initial conditions in the pixels' integrator [12].

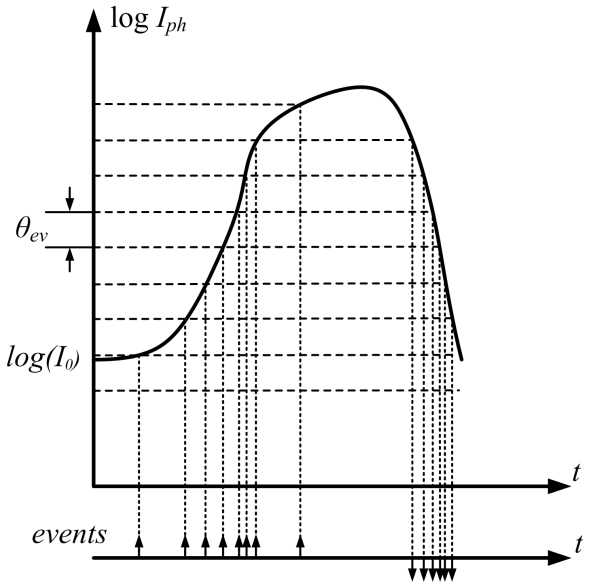

Fig. 1. Data-driven asynchronous event generation for a given pixel in a DVS. Asynchronous events are generated every time the photocurrent increments by θ_{ev}, with initial condition I_0.

This DVS has two important properties that make it especially suitable for stereo processing (as will be shown in this paper): a very high temporal resolution of events and sparse output. Since each pixel works independently, the temporal resolution of this sensor would be equivalent to a sampling rate higher than 100 kframes/s. This fine time resolution is exploited by event matching algorithms, as described in Section IV. The sparsity of the DVS's output means that no redundant information is produced: only those pixels that detect a certain change in light intensity generate events, resulting in a more efficient information flow. Sparse data also reduce the computational cost of processing algorithms, making them more efficient.

III. Event-Based Cluster Tracking

The cluster tracking algorithm used in this paper was inspired by the one presented in [14] for a single DVS. This algorithm is based on a distance criterion between incoming events and a dynamic list of clusters (storing the size and center coordinates), as illustrated in Fig. 2. An incoming event appears in location (x_1, y_1), and the algorithm finds one single

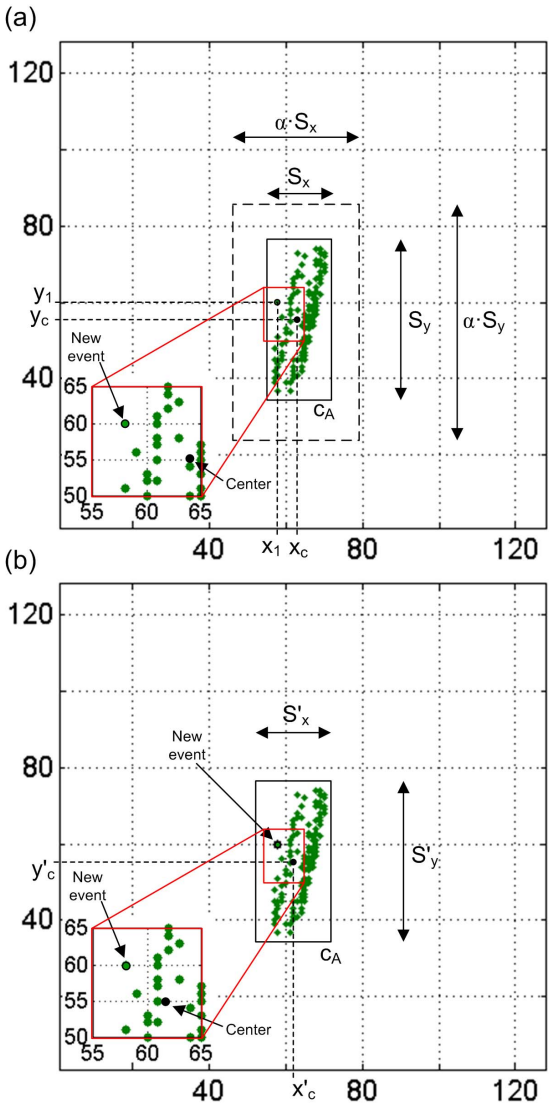

Fig. 2. Illustration of event-based cluster tracking algorithm. (a) New event with coordinates (x_1, y_1) lies within the seek range of cluster c_A. (b) Cluster c_A is updated (new center position and new size) to include event located at (x_1, y_1). The insets show closeups of the central area of the object to highlight how the center of the cluster updates its position.

cluster c_A in the list of available clusters. This cluster c_A, modeled as a rectangle in Fig. 2(a), is represented by center coordinates (x_c, y_c) and horizontal and vertical dimensions (S_x, S_y). The incoming event with coordinates (x_1, y_1) lies within the seek range of cluster c_A, given by $(\alpha \cdot S_x, \alpha \cdot S_y)$, so the algorithm includes the event in cluster c_A and updates the cluster [the seek range is defined by parameter α, as shown in Fig. 2(a)]. The updated cluster is shown in Fig. 2(b), where the new center coordinates are (x'_c, y'_c) and the new cluster size is given by (S'_x, S'_y). If the incoming event with coordinates (x_1, y_1) had been outside the range of cluster c_A, a new cluster c_B would have been created for it.

The list of clusters is stored sorted by creation time, so older clusters will be preferred for assignation to incoming events (although excessively old, inactive clusters are removed periodically, since they may correspond to objects that are no longer present). This prevents the formation of multiple clusters from a single object when edge events fall out of the seek range. We denote this algorithm as cluster tracking with priority. As the algorithm has to make new calculations only when a new event is generated, the DVS's output sparsity considerably reduces computational costs. The periodic removal of old inactive clusters represents a very small fraction of computation resources in comparison with event processing.

IV. Event-Based Stereo Matching

In stereo vision systems, a 3-D point in space $\boldsymbol{M}$ is projected onto the focal planes of two sensors in pixels $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$, generating events $\boldsymbol{e}(\boldsymbol{m_1^i}, t)$ and $\boldsymbol{e}(\boldsymbol{m_2^i}, t)$, where $i = 1, 2, \ldots$ represents the event number. To reconstruct the original 3-D coordinates, it is necessary to match each pair of events produced by point $\boldsymbol{M}$ at time t [27]. The stereo matching algorithm proposed in [19] is based on a list of restrictions that are applied to each event generated in one sensor in order to find its matching pair in the other. If only one single event satisfied all the restrictions, it could be considered to form a matched pair. The usual constraints are time window, distance to the epipolar line, event polarity, uniqueness constraint, ordering constraint [19], and edge orientation [20] (this list is not exhaustive, but it contains the most frequently used constraints). The high temporal accuracy of the DVS is crucial for event matching algorithms, especially to apply small time windows of only a few milliseconds [20].

V. Proposed Stereo Cluster Tracking Algorithm

If the event-based stereo matching strategies mentioned in Section IV are intuitively compared with human vision, it seems reasonable to assume that our brain knows which object an event belongs to and uses this information for 3-D reconstruction. This could be included as an extra restriction in a stereo matching algorithm, considering that the disparity of a matched pair of events must be the same as that measured for the object to which those events belong. Using the center coordinates of the cluster given by a 2-D cluster tracking algorithm [14], and including the given disparity in the event-matching algorithm described in [20], we observe an improvement in the quality of the 3-D reconstruction, especially caused by a reduction in wrongly matched event pairs. However, this strategy works only for a single object, or for several objects between which there is no interaction or occlusion. Returning to human vision, when two objects are being observed and one of them is occluded by the other, the brain uses learned knowledge about the hidden object to guess its new position after occlusion. By assimilating this intuitive trait of human vision and trying different empirical approaches, we were able to define and organize a series of steps that would enable 3-D cluster tracking.

As a result, we propose in this paper a novel procedure in which cluster tracking and stereo matching are integrated in a single algorithm, forcing each subalgorithm dynamically to use the information given by the other. Sometimes, for example, in the case of occlusion between objects, the 2-D cluster tracking algorithm is not able to determine which cluster an

incoming event belongs to. When this happens, we propose that the algorithm should consider previously learned information related to all the possible clusters, assuming successively that the incoming event belongs to each cluster, and then validate each assumption by means of stereo matching and 3-D reconstruction, to decide which initial assignment is more likely to be true. In this way, there is feedback between 2-D clustering and 3-D reconstruction, with each subalgorithm improving the behavior of the other. Although cluster tracking and stereo matching algorithms have been proposed before, what we are proposing in this paper is not just the combination of two existing methods, but a new procedure where both subalgorithms are enhanced through interaction with the other. This new algorithm is capable of solving problems that the previous subalgorithms could not solve by themselves. There now follows a detailed description of the proposed algorithm, which is summarized in Algorithm 1.

Algorithm 1 Proposed Stereo Cluster Tracking Algorithm

1: **for all** events $\boldsymbol{e}(\boldsymbol{m_r^i}, t)$ **do**
2: Detect occlusion between 2-D clusters
3: **if** There is no occlusion **then**
4: Apply 2-D tracking with priority (assign cluster c_r^n)
5: Apply event matching with disparity (Algorithm 2)
6: **end if**
7: **if** There is occlusion **then**
8: Apply 2-D tracking with no priority (assign clusters candidates $c_r^{n_1}, c_r^{n_2}, \ldots$)
9: **if** There is one candidate c_r^n **then** Algorithm 2 **end if**
10: **if** There are several candidates $c_r^{n_1}, c_r^{n_2}, \ldots$ **then**
11: **for** Each candidate $c_r^{n_i}$ **then** Algorithm 2 **end for**
12: **if** One candidate gives a valid 3-D event **then** Take it **end if**
13: **if** Several candidates give valid 3-D events **then** Discard them **end if**
14: **end if**
15: **end if**
16: **end for**

Two sensors were used, R_1 and R_2. A pair of 2-D events were denoted by $\boldsymbol{e}(\boldsymbol{m_1^i}, t)$ and $\boldsymbol{e}(\boldsymbol{m_2^i}, t)$, where $\boldsymbol{m}$ represents the events' coordinates $\boldsymbol{m} = [x\ y]^T$ in the sensor (subscript 1 or 2 indicates the sensor, and superscript i indicates the event number), and t the time. 3-D events were denoted by $\boldsymbol{e}(\boldsymbol{M^i}, t)$, where $\boldsymbol{M}$ represents the events' spatial coordinates $\boldsymbol{M} = [X\ Y\ Z]^T$ [19] (superscript i indicates the event number). $\boldsymbol{M}$ was described in an arbitrarily chosen Euclidean world coordinate system. In our study, the system used was the one defined by the calibration structure. 2-D clusters of events were denoted by c_r^n, where r indicates the sensor (1 or 2) and n indicates the cluster index. 3-D clusters of events were denoted by C^n. The algorithm consists in applying a number of steps to each incoming event $\boldsymbol{e}(\boldsymbol{m_r^i}, t)$ (line 1 in Algorithm 1, with subscript r representing the sensor and superscript i indicating the event number), so the data sparsity of the DVSs reduces the computational cost. It first has to be decided if there is currently occlusion between previously matched 2-D clusters, because the steps to be applied will differ depending on the existence or not of occlusion (line 2 in Algorithm 1). Occlusion is considered to occur if the seek ranges of at least two clusters overlap in either of the sensors. That is checked using the centers and sizes of those clusters that have already been matched. If no occlusion is observed (line 3 in Algorithm 1), the problem is easier to solve and the steps described in Section V-A are applied. This is shown in Fig. 3, where Fig. 3(a) represents a scenario with no occlusion and Fig. 3(b) a scenario with occlusion. In both cases, two 3-D objects are observed by two DVSs R_1 and R_2, each object n projecting cluster c_r^n onto sensor r. Each cluster is surrounded by a dotted rectangle representing its seek range. In Fig. 3(a), the two pairs of seek ranges do not overlap, so no occlusion is detected by the algorithm. In R_2 in Fig. 3(b), they overlap, indicating occlusion.

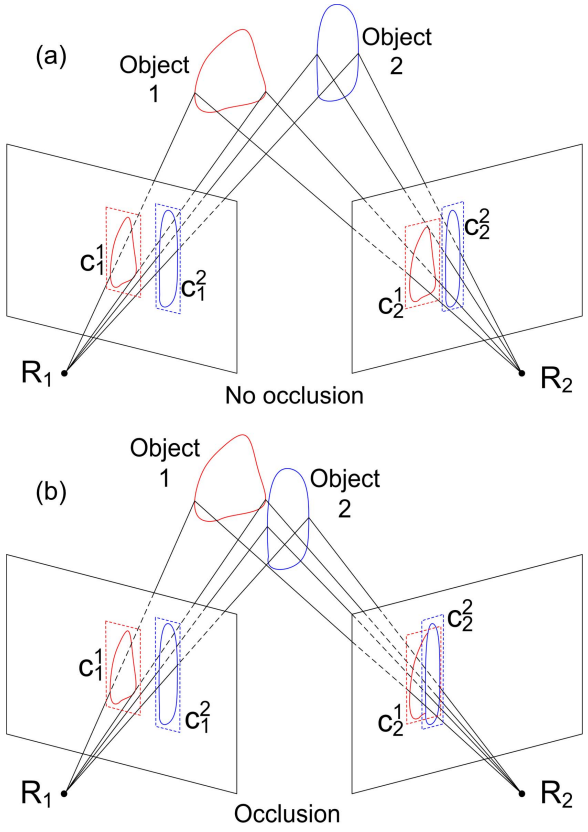

Fig. 3. Diagrams illustrating different occlusion conditions. (a) There is no occlusion between objects 1 and 2, because the seek ranges (dotted rectangles) around the clusters projected in both DVSs do not overlap. (b) There is occlusion in sensor R_2, because seek ranges for clusters c_2^1 and c_2^2 overlap.

A. Steps to Follow When There Is No Occlusion

First, the basic 2-D cluster tracking algorithm with priority list is applied to the incoming event (line 4 in Algorithm 1), and the first cluster on the list (higher priority) that includes the event in its seek range, c_r^n, is assigned to it. If none of the clusters on the list fulfill this condition, a new cluster is created (see Section III). The address of the center of the cluster

Algorithm 2 Apply Event Matching Algorithm With Disparity and Validate the 3-D Reconstructed Event

1: **if** The assigned cluster c_r^n is not matched **then** Apply cluster matching to obtain a pair $c_1^n - c_2^n$ **end if**
2: **if** cluster c_r^n is matched **then** Calculate the disparity vector $\boldsymbol{disp^n}$ and apply event matching with disparity **end if**
3: **if** There is one event candidate **then** Create a match $\{\boldsymbol{e}(\boldsymbol{m_1^i}, t), \boldsymbol{e}(\boldsymbol{m_2^i}, t)\}$ and reconstruct the 3-D event $\boldsymbol{e}(\boldsymbol{M^i}, t)$ **end if**
4: Apply 3-D cluster tracking and assign a 3-D cluster C^n (associated to a pair of 2-D clusters $c_1^n - c_2^n$)
5: Check if the 3-D cluster corresponds to the pair of 2-D clusters assigned to the 2-D events

and its size are updated with this new event. The following steps, grouped into Algorithm 2 (line 5 in Algorithm 1), are then applied.

1) If the assigned cluster c_r^n is not yet matched to a cluster in the other sensor, it means it is a new cluster. For new clusters, it is necessary to wait until they have a certain number of events (in the range of 1000, although this parameter is adjustable), and only when that threshold is reached a cluster matching algorithm is applied to create a pair $c_1^n - c_2^n$ (line 1 in Algorithm 2). The cluster matching algorithm estimates a velocity vector [28] for different clusters with a similar number of events. It is assumed that the velocity vectors of the object projections in both sensors will be similar. This assumption is valid when the sensors are aligned roughly vertically with a small horizontal vergence [20]. In this way, a pair of clusters is obtained, which correspond to the projections of a single 3-D object in both sensors. After this step, it can be considered that the algorithm has learned a new object, represented by a pair of clusters. Note that this step is applied to each cluster only once.
2) If the assigned cluster c_r^n is already matched, disparity vector $\boldsymbol{disp^n}$ is calculated as the difference between the centers of the corresponding clusters in both DVSs. Knowing the coordinates of an event in one sensor, the disparity vector thus estimates the area where that event's matching event must be in the other sensor, based on learned information about the position of each object. This information can be used to determine the region in which to seek the matching event, adding one more important restriction to the event matching algorithm described in Section IV. The full list of constraints applied in this algorithm is: 1) time window; 2) epipolar line; 3) disparity area; 4) polarity; 5) uniqueness; and 6) ordering constraint (line 2 in Algorithm 2). This step exploits the sensors' high temporal accuracy.
3) Application of the event matching algorithm with disparity gives several possible results.
   a) No event satisfies all the restrictions, so this incoming event is discarded.
   b) Several events satisfy the restrictions, so the incoming event is also discarded.
   c) There is only one candidate event, so we continue with the matching pair $\{\boldsymbol{e}(\boldsymbol{m_1^i}, t), \boldsymbol{e}(\boldsymbol{m_2^i}, t)\}$.

   The projection matrices of the stereo system are used to reconstruct the coordinates of the 3-D event $\boldsymbol{e}(\boldsymbol{M^i}, t)$, as described in [20] (line 3 in Algorithm 2).
4) To assign this 3-D event to a 3-D cluster C^n, the 3-D cluster tracking algorithm is used. This tracking algorithm behaves just like the one described above for 2-D events with priority list, the only difference being that it is not allowed to create more 3-D clusters than the number of matched pairs of 2-D clusters (this limitation is applied to avoid overclustering and to reduce wrong matches). If the algorithm reconstructs a 3-D event that is far from all existing 3-D clusters, we assume that the reconstruction was wrong and the event is discarded (line 4 in Algorithm 2). This processing is based on the 3-D features learned by the algorithm, which indicate the position and size of objects.
5) Finally, we also check that the assigned 3-D cluster corresponds to the same object projected onto the pair of 2-D clusters assigned to the 2-D events. This correspondence is based on a decision taken when a certain number of events (an adjustable parameter, usually in the range of a few tens or hundreds) have been assigned to the 3-D cluster after its creation, assuming an initial condition where the objects' shapes are clearly stabilized. If this correspondence is not right, the event is discarded (line 5 in Algorithm 2).

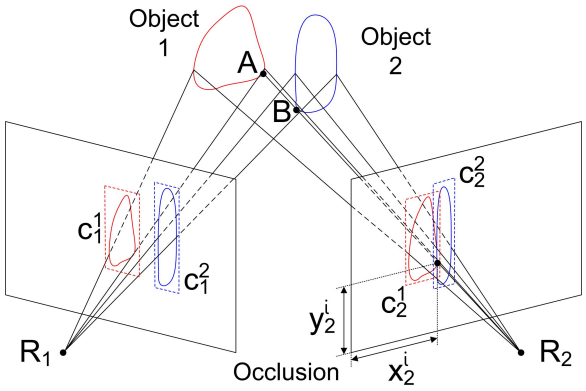

Fig. 4. Illustration of the difficulties involved in assigning a cluster to an incoming event under occlusion. The event projected as (x_2^i, y_2^i) in R_2 might correspond to 3-D point A in object 1 or point B in object 2.

B. Steps to Follow When There Is Occlusion

When at least two clusters in one of the DVSs intersect in a certain area, the procedure described above is modified (line 7 in Algorithm 1). Fig. 4 illustrates this situation, with occlusion occurring between objects 1 and 2 in R_2. Here, it would be impossible to know *a priori* whether an event with x-coordinate x_2^i in the region where both clusters intersect belongs to object 1 (3-D point A) or to object 2 (3-D point B), so the following strategy is used to solve the ambiguity.

An alternative version of the 2-D cluster tracking algorithm is applied, with no priority list. In this case, the algorithm

searches for all possible clusters that have the incoming event in their seek range. There are two possible outcomes: either there is only one cluster candidate c_r^n or there are several $c_r^{n1}, c_r^{n2}, \ldots$ (line 8 in Algorithm 1).

If there is only one cluster candidate (line 9 in Algorithm 1), it is necessary to follow all the steps described above for a no-occlusion scenario (Algorithm 2). This includes checking if the assigned 2-D cluster is already matched (and applying cluster matching if the number of events reaches the threshold), calculating the disparity vector, and applying the event matching algorithm. If a single matching event is obtained, the 3-D event is reconstructed, the 3-D cluster tracking algorithm is applied, and the correspondence with the assigned 2-D cluster is checked.

If the incoming event may belong to several 2-D clusters (line 10 in Algorithm 1), the same steps as indicated in the previous paragraph (Algorithm 2) are applied for each cluster candidate (line 11 in Algorithm 1). If, for example, cluster c_r^{n1} is the right cluster, we try to reconstruct the 3-D event and assign a 3-D cluster. If the result is correct, that means that this assumption may be true. After repeating this procedure for all the candidates, it is then necessary to check how many of them gave valid 3-D events which match the initial assumption. If it was only one, that one can be identified as the good 2-D cluster (line 12 in Algorithm 1). If not, the event is discarded (line 13 in Algorithm 1). This procedure is based on the 3-D information learned by the algorithm about the position and size of the objects.

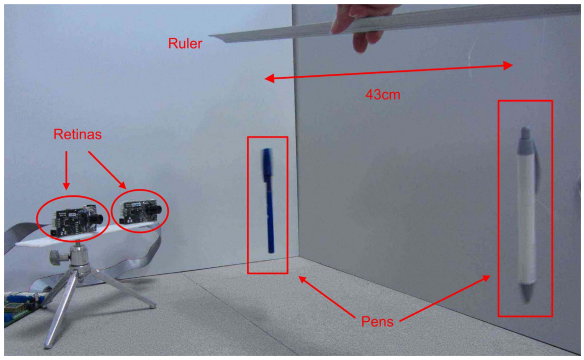

Fig. 5. Experimental setup 1. Two pens hanging from each side of a ruler swing in front of two DVSs (we also call them retinas). The activity captured by the sensors is processed by the algorithm. The distance between the pens is 43 cm.

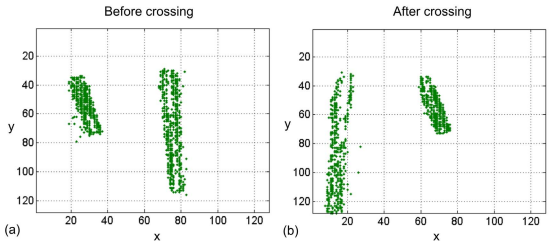

Fig. 6. Visualization of two different frames of the recording from experiment 1 (a) before and (b) after the two pens crossed one another's path.

VI. Results

To evaluate the proposed stereo cluster tracking algorithm experimentally, we used four different setups. In the first one, described in Section VI-A, two very simple objects (pens) were used to analyze the algorithm's behavior in detail. Section VI-B used a box and a pen to validate the algorithm when both objects are different and one of them is more complex. Section VI-C used two rotating straws attached to a fan to show the algorithm's capability to process high-speed data. Section VI-D demonstrated the algorithm's capability to process a real-world situation with two people walking in front of the DVSs. Finally, Section VI-E compares the computational cost of the proposed event-based algorithm with conventional frame-based approaches.

A. Experiment 1: Simple Objects

In this first experiment, we recorded the movement of two pens swinging in front of two DVSs using the setup shown in Fig. 5. These pens were suspended from both sides of a ruler, to which they were attached at a distance of 43 cm from each other, as shown in Fig. 5. In the recording, one of the pens is closer to the sensors with the other always behind it, in such a manner that, from the sensors' point of view, they cross one in front of another as they swing. Fig. 6 displays two different time frames captured in sensor R_1 while the pens were swinging. As can be seen, one of the pens looks larger than the other, indicating that it was closer to the sensor (since both pens were the same size). In the first snapshot shown in Fig. 6(a), the nearer pen (the large one) can be seen to be on the right-hand side, while in the second snapshot in Fig. 6(b), it is on the left-hand side. This poses a difficult problem for conventional cluster tracking algorithms [14], because there is an instant when both clusters merge together (one pen is right behind the other): when the pens separate again, the algorithm does not know which new cluster corresponds with each previous cluster. This causes cluster identification errors, as shown in Fig. 7.

Fig. 7 shows the results of applying a 2-D cluster tracking algorithm [14], [18] to the recordings obtained independently in each sensor [Fig. 7(a) and (b)]. Each dot in Fig. 7 (red or blue) represents an event, with the x-coordinate plotted against time. Red and blue indicate clusters assigned by the algorithm, while yellow and green lines represent the ground truth for objects 1 and 2, respectively. The ground truth was constructed by visual inspection of the input recording. The vertical lines **A**–**E** represent important time instants, and the insets show snapshots from each sensor at those instants, indicating the clustering results in red or blue. At the beginning ($t = 0$), both pens are identified in both sensors (the larger pen in red, the smaller pen in blue). After 3 s (instant **A**), the first crossing occurs, causing the first important confusion. When the tracking algorithm was applied to sensor R_1 [Fig. 7(a)], only one cluster was detected, and the wrong color was assigned to it (the inset shows only blue events, but the large pen that is occluding the small one should be red). In sensor R_2, where there was no occlusion, the algorithm was able to distinguish both pens easily [Fig. 7(b)].

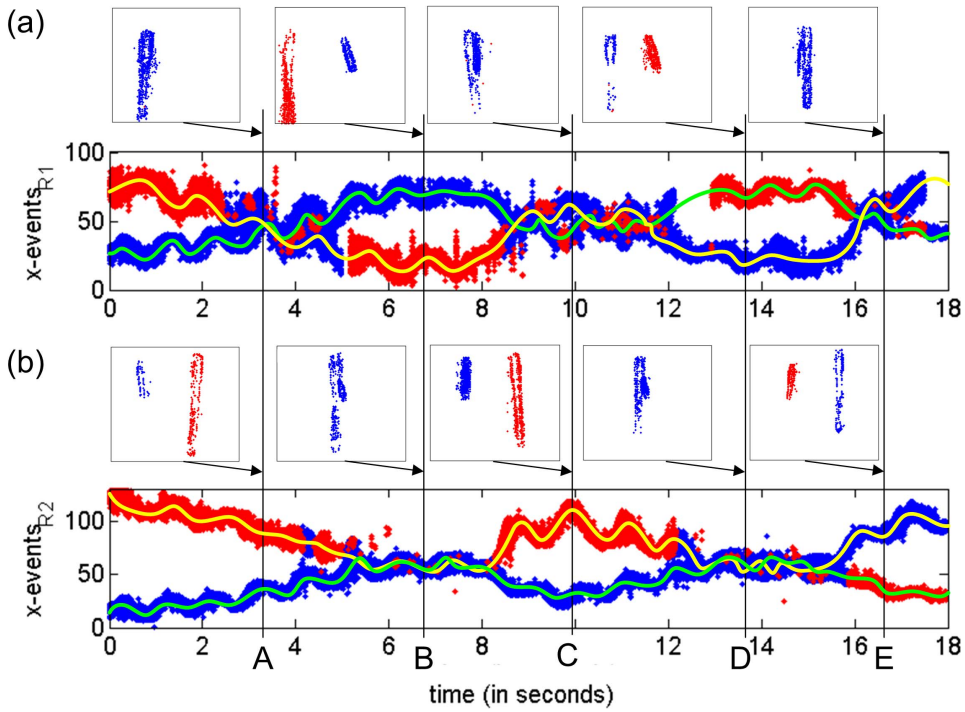

Fig. 7. Results of applying conventional cluster tracking algorithm to the recording from experiment 1. Each plot corresponds to a different sensor. (a) R_1. (b) R_2. Each dot represents an event, showing the x-coordinates versus time. Red and blue indicate the clusters assigned to each event by the algorithm. Yellow and green lines indicate the ground truth for objects 1 and 2, respectively. The vertical lines **A**-**E** represent important time instants, and the insets show snapshots from each sensor at those instants. As can be seen, the assigned cluster (red or blue) is not consistent with the ground truth (yellow or green), especially after the second crossing.

After 6 s of recording, it is sensor R_2 that observes both pens intersecting (instant **B**). In this case, the algorithm detects one single cluster in sensor R_2, again assigning it the wrong color [Fig. 7(b)]. However, both clusters are properly identified in sensor R_1, despite having been lost during the previous occlusion [Fig. 7(a)]. After 10 s of recording (instant **C**), the algorithm produces a correct clustering with sensor R_2 (no occlusion), solving the confusion observed during the previous occlusion, but it obtains one single cluster in sensor R_1, again with the wrong color (it is clustered in blue, but it was the red pen that was occluding the blue one).

After 13 s of recording, a new occlusion occurs in sensor R_2 (instant **D**), and the same problem is observed once again. The algorithm detects only one cluster, and assigns the wrong color to it [Fig. 7(b)]. In this case, the algorithm also identifies two clusters in sensor R_1, but these two are assigned the wrong colors, resulting in tracking confusion between the two pens [Fig. 7(a)]. Finally, after 16 s, the last crossing takes place in sensor R_1 (instant **E**). Again, one cluster is detected in sensor R_1 with the wrong color [Fig. 7(a)], and two clusters are identified in sensor R_2 also with the wrong colors [Fig. 7(b)]. In short, the algorithm completely lost track of the objects after two crossings.

Fig. 8 shows the results of applying the proposed stereo cluster tracking algorithm to the same recording of the two swinging pens. When occlusion is detected between clusters, the algorithm obtains more information by reconstructing 3-D events using information obtained by integrating the two sensors. This 3-D information is introduced as feedback in the 2-D cluster tracking, correcting the errors observed when using a conventional tracking algorithm (see Fig. 7). Fig. 8 shows how the 2-D clusters are properly identified throughout the recording [see the ground truth in yellow and green in Fig. 8(a) and (b)] by incorporating the 3-D reconstruction information [see the x and y coordinates of the 3-D events on Fig. 8(c) and (d)]. The vertical lines **A**–**E** represent the same time instants as Fig. 7, with the corresponding insets showing the results obtained with the clustering algorithm.

In Fig. 8, the first crossing occurs at instant **A** in R_1. The inset shows how the algorithm detects both objects (with the large pen, in red, occluding the small pen, in blue), although the small pen is barely visible. At instant **B**, something similar happens in R_2, where the algorithm again detects and properly identifies both objects. The same result is shown at instants **C**–**E**, where the algorithm keeps track of both objects even during occlusion. The small gaps in Fig. 8(c) and (d) are caused by the lack of 3-D events obtained by the algorithm over certain periods, generally because the event matching algorithm cannot obtain one single candidate for each input event. However, this method is sufficiently robust to allow both clusters to appear again without causing identification errors, even after such periods of "darkness."

The number of events captured by the sensors was around 150K for R_1 and 180K for R_2, while the number of reconstructed 3-D events was around 30K. This corresponds to a matching rate of approximately 20%—a good result considering that all the events had already been validated by the algorithm. In earlier comparable studies,

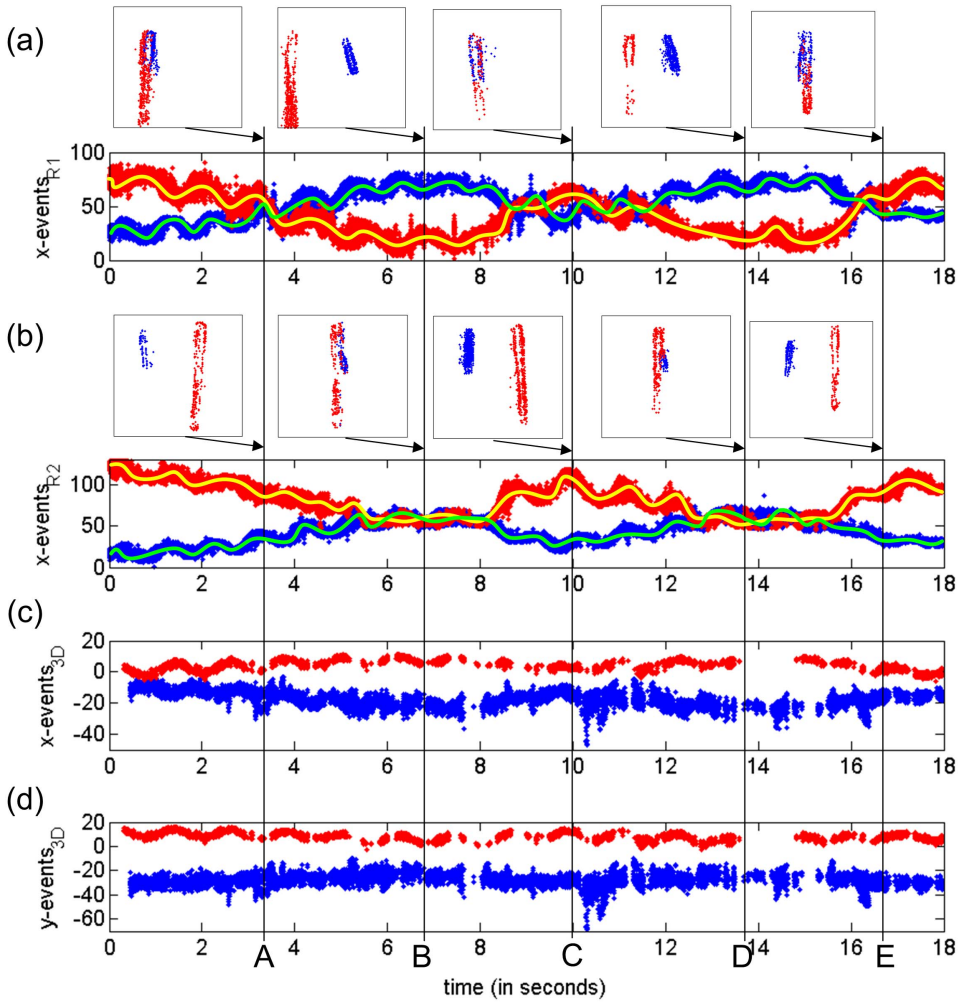

Fig. 8. Results of applying the proposed stereo cluster tracking algorithm to the recording from experiment 1. (a) and (b) x-coordinates of the events recorded by the DVSs [R_1 in (a) and R_2 in (b)], with red and blue indicating the cluster assigned by the algorithm, and yellow and green indicating the ground truth for each object. The vertical lines **A**–**E** represent important time instants, and the insets show snapshots from each sensor at those instants. (c) and (d) Reconstruction of 3-D events (x and y coordinates, respectively).

Camuñas-Mesa *et al.* [20] obtained a matching rate of up to 28% for a single pen and up to 15% for a ring. Assuming that the complexity of the current recording with two pens lies somewhere between these two cases (a scenario with two pens is more complex than a scenario with one pen, but less complex than a scenario with a ring, which includes more different orientations), 20% can be considered a positive result. To characterize the precision of the 3-D reconstruction, we dynamically computed 3-D coordinates of the center of each object, and measured the distance between both objects, obtaining a mean distance of 42.79 cm with a standard deviation of 3.38 cm, to represent how the distance between the objects changes while they are swinging. It should be remembered that the pens were tied to the ruler at a distance of 43 cm from each other. The error for the mean distance was less than 0.5%. Fig. 9(a) and (b) shows two snapshots with 40 ms reconstructions of the final 3-D sequence. Each image shows the 3-D coordinates of the reconstructed events, indicating the assigned cluster in red and blue. The position of the sensors is indicated by the black dots, while the black arrows represent the direction in which the pens are moving.

To measure the success rate, we focused on the algorithm's ability to solve a situation with occlusion in one of the sensors. For a single DVS, an occlusion is correctly processed if the cluster identification after the occlusion is correct (i.e., it matches the ground truth). Otherwise, it is wrong. We therefore measured the success rate as the number of correct occlusions over the total number of occlusions in both sensors. Analyzing the results for the conventional tracking algorithm shown in Fig. 7, we found three occlusions in sensor R_1 (one of which was solved correctly), and two occlusions in sensor R_2 (one of which was solved correctly). This gave a success rate of 40%. In contrast, the results obtained for the proposed stereo tracking algorithm gave a success rate of 100%, as shown in Fig. 8.

This experiment demonstrated that the proposed method does not require any prior information about the observed objects, but that the tracking of moving objects can be enhanced by adding 3-D information. This approach is different from that adopted in [16]. In that work, recognition tasks were also carried out, whereas the scope of our study was limited to tracking moving objects. In [16], occlusion

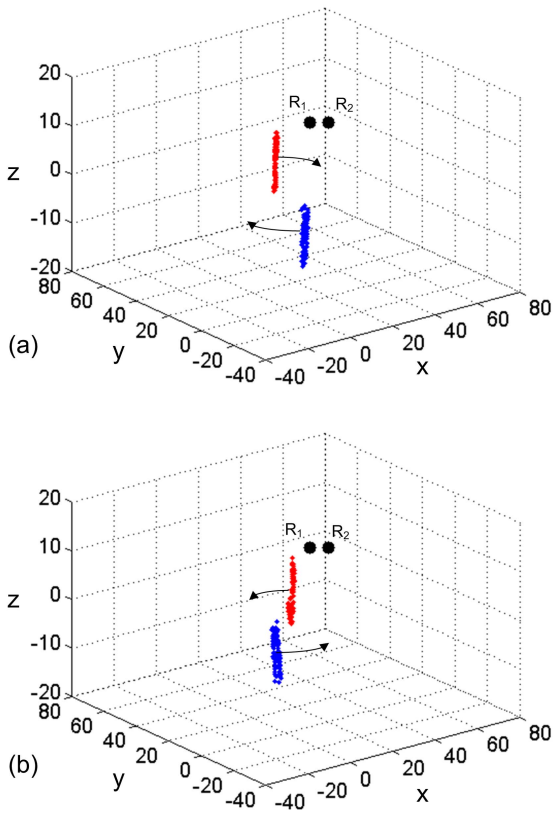

Fig. 9. (a) and (b) Two snapshots of the 3-D coordinates of the reconstructed events. The two pens are represented in blue and red, depending on the cluster assigned to each event. The position of the two sensors is shown by the black dots. The black arrows represent the swinging movement of the pens.

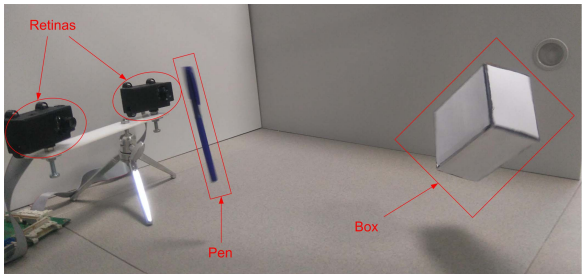

Fig. 10. Experimental setup 2. Two objects, a pen and a box, swing back and forth in front of two DVSs (we also call them retinas). The activity captured by the sensors is processed by the algorithm.

disambiguation was performed using a combination of pattern recognition and velocity estimation of moving targets.

B. Experiment 2: Complex Object

In a second experiment, one of the pens in experiment 1 was replaced by a small box, as shown in Fig. 10. This new object introduced a higher level of complexity, especially for the stereo matching tasks. Since a box has several parallel lines, each event it generates in a DVS has a higher probability of finding more than one candidate event around the corresponding epipolar line in the other DVS. Moreover, only part of the front portion of the box's total volume can be observed by both sensors simultaneously, and this is the only portion that can be reconstructed in 3-D using stereo matching. This in turn limits the amount of information related to the 3-D shape of the box that can be used by the algorithm to track the object.

Fig. 11 shows the results of applying the proposed stereo cluster tracking algorithm to the recording obtained from the setup in Fig. 10, while the pen was passing in front of the box. The 2-D clusters are properly identified and tracked throughout the recording, as can be seen in Fig. 11(a) and (b), where the x-coordinates of the events are plotted versus time with red and blue indicating the cluster assigned by the algorithm to each event and yellow and green lines representing the ground truth. The 3-D reconstruction information used to track the objects during occlusion is shown in Fig. 11(c) and (d), where the x and y coordinates of the 3-D events are plotted versus time (again, red and blue indicate the assigned cluster). The vertical lines **A**–**C** represent specific time instants for which the output of both sensors is shown in the insets.

At the beginning of the recording (around $t = 1s$, instant **A**), both objects are identified correctly in both sensors. The insets in Fig. 11(a) and (b) show the pen's events in red and the box's events in blue. After 6 s of recording, the box is partially occluded by the pen, first in R_2, and immediately afterward in R_1. Instant **B** shows how this occlusion has just finished in R_2 while it is still occurring in R_1. The inset for R_1 shows how the algorithm uses the 3-D information to distinguish between the events generated by either the pen or the box, even though they are superimposed, to produce a correct clustering output.

After 9 s of recording (instant **C**), the box is again partially occluded by the pen (the crossing is detected first by R_1 and immediately afterward by R_2). Initially, the algorithm is confused by the reduced amount of information it is receiving, and it loses the box for a small time window just before instant **C**. This is illustrated by the large amount of red events in Fig. 11(c) and (d), surrounded by blue events around $t = 9\ s$. However, as soon as R_1 observes both objects separately again, the algorithm solves the confusion at instant **C**, as shown in the insets. There, both clusters are clearly identified by R_1, while R_2 shows them superimposed but with the correct colors assigned to the events.

This experiment showed how the algorithm also works with more complex objects, even though the 3-D information reconstructed was not as clear as it was in experiment 1. Fig. 11(c) and (d) shows how the 3-D events obtained for the box (the blue events) are more discontinuous, with the edges observed by the sensors appearing and disappearing while the object was moving. Fig. 11(c) and (d) also shows how the 3-D events associated with each object are also superimposed in the x and y projections, making it more difficult to track the 3-D clusters. Even in this situation, the algorithm was robust enough to provide good tracking results. Using the success rate measurement method described in Section VI-A, we obtain a success rate of 100% for this example (four occlusions solved correctly).

C. Experiment 3: High Speed

In this experiment, we designed a different setup that could be used to validate the algorithm's capability to process

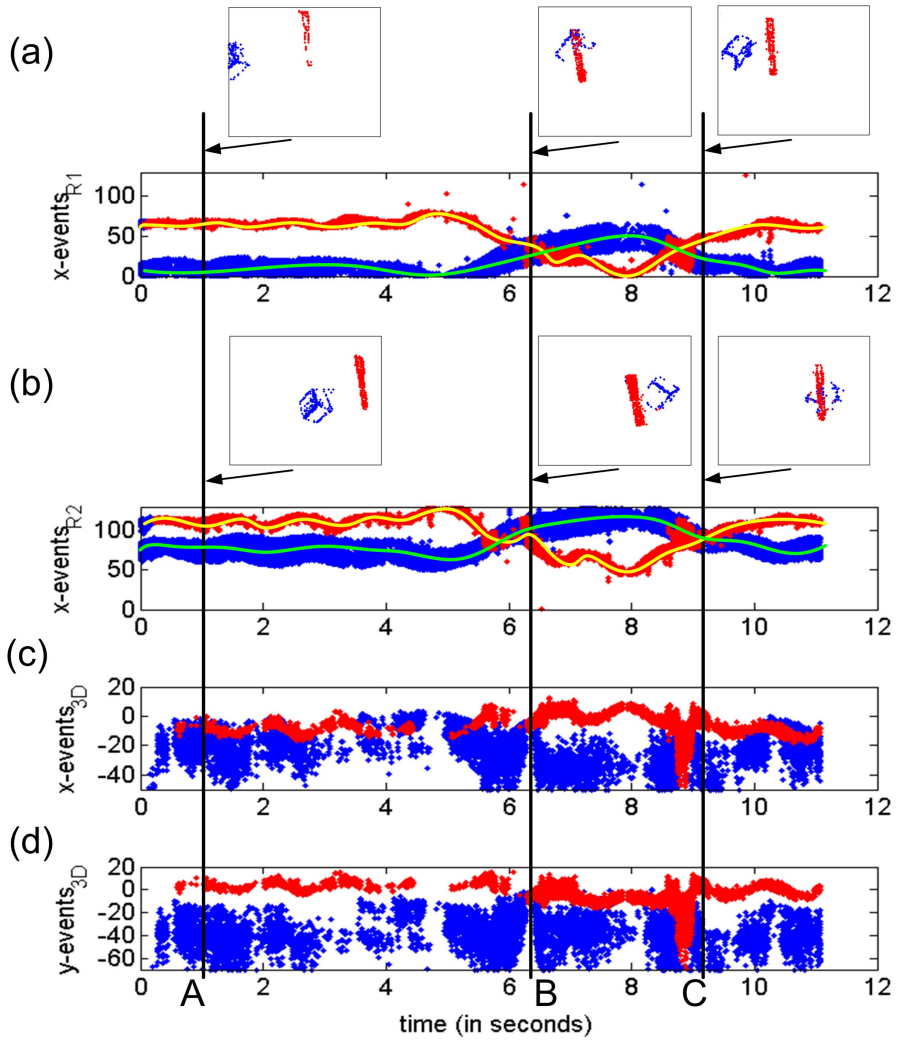

Fig. 11. Results of applying the proposed stereo cluster tracking algorithm to the recording from experiment 2. (a) and (b) x-coordinates of the events recorded by the DVSs [R_1 in (a) and R_2 in (b)], with red and blue indicating the cluster assigned by the algorithm, and yellow and green lines representing the ground truth for each object. The vertical lines **A**–**C** represent important time instants, and the insets show snapshots from each sensor at those instants. (c) and (d) 3-D events' reconstruction (x and y coordinates, respectively).

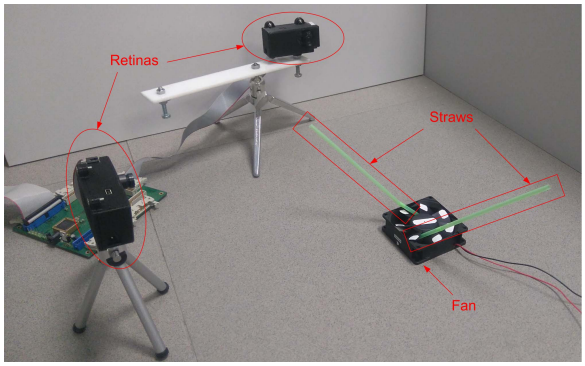

Fig. 12. Experimental setup 3. Two straws attached to a fan rotate in front of two DVSs. The activity captured by the sensors is processed by the algorithm.

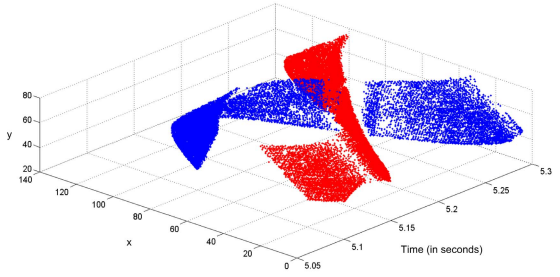

Fig. 13. Events captured by a DVS while the two straws are rotating, represented in 3-D space (x, y, t) and with the different straws shown in red and blue.

high-speed visual information. Two straws were glued to two blades of a small fan to make them rotate while being observed by two DVSs, as shown in Fig. 12. In this setup, the sensors were positioned further away from each other, so that they could both observe the wider movement of the rotating straws. The rotation of the fan produced two relative occlusions between the straws per cycle. The activity recorded by the sensors during this rotation is illustrated in Fig. 13, which shows a subset of events captured by one of the sensors in (x, y, t) space while the two objects were rotating. The two

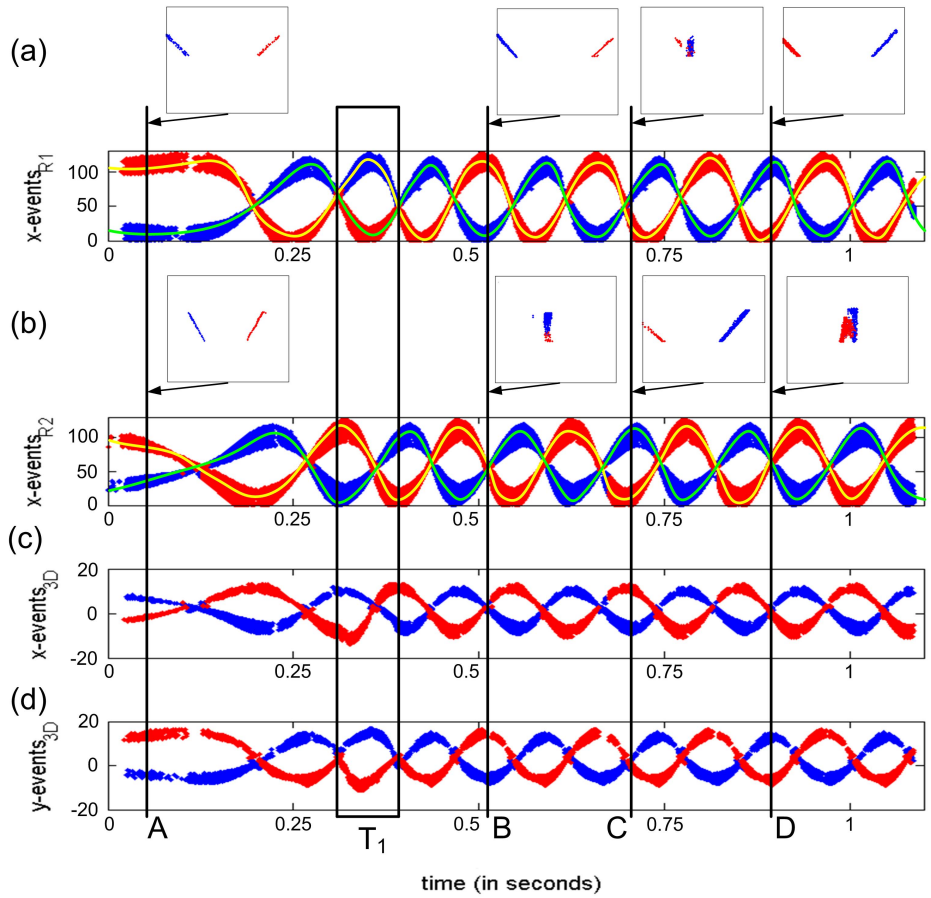

Fig. 14. Results of applying the proposed stereo cluster tracking algorithm to the recording from experiment 3. (a) and (b) x-coordinates of the events recorded by the DVSs [R_1 in (a) and R_2 in (b)], with red and blue indicating the cluster assigned by the algorithm, and yellow and green lines representing the ground truth for each object. The vertical lines **A**–**D** represent important time instants, and the insets show snapshots from each sensor at those instants. (c) and (d) 3-D events' reconstruction (x and y coordinates, respectively). Box T_1 marks a specific time slot.

straws are represented in red or blue each, with an occlusion in the middle of the plot.

Fig. 14 shows the results of applying the proposed algorithm to a recording obtained using this high-speed experimental setup. Each cycle in all the traces in Fig. 14 corresponds to a complete rotation of the fan, so an approximate speed of 6 revolutions per second (r/s) can be estimated from the figure (slightly more than 3 cycles are observed in half a second). When the ground truth [yellow and green lines in Fig. 14(a) and (b)] is compared with the cluster identification (red and blue dots), an error can be observed in time slot T_1, where the clusters assigned to the events in R_1 do not match the ground truth. The reason for this error is to be found in the algorithm's strategy for validating its own results. Every time a cluster is assigned to an input event in one sensor, the algorithm tries to match it with an event in the corresponding cluster in the other sensor, and a 3-D event is obtained. This 3-D event is then assigned to a 3-D cluster, and the algorithm validates whether this 3-D cluster matches the 2-D cluster assigned initially. If it does not, the algorithm detects an error in the initial assignment and discards it. The problem in this case is that a wrong initial assignment gives a valid 3-D reconstruction. Looking at box T_1 in Fig. 14, it can be seen that it corresponds to half a cycle where the clusters are correct for R_2 and incorrect for R_1. This

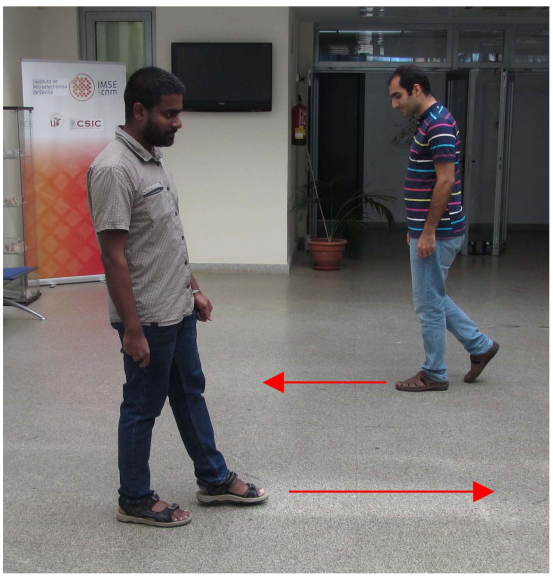

Fig. 15. Experimental setup 4. Two people walking in front of the stereo setup. The activity captured by the DVSs is processed by the algorithm.

resulted in the 3-D reconstruction shown in Fig. 14(c) and (d), which indicates that both objects rotated backward for half a cycle. We know this was not true, because we controlled the experiment, but the answer would be valid assuming that the

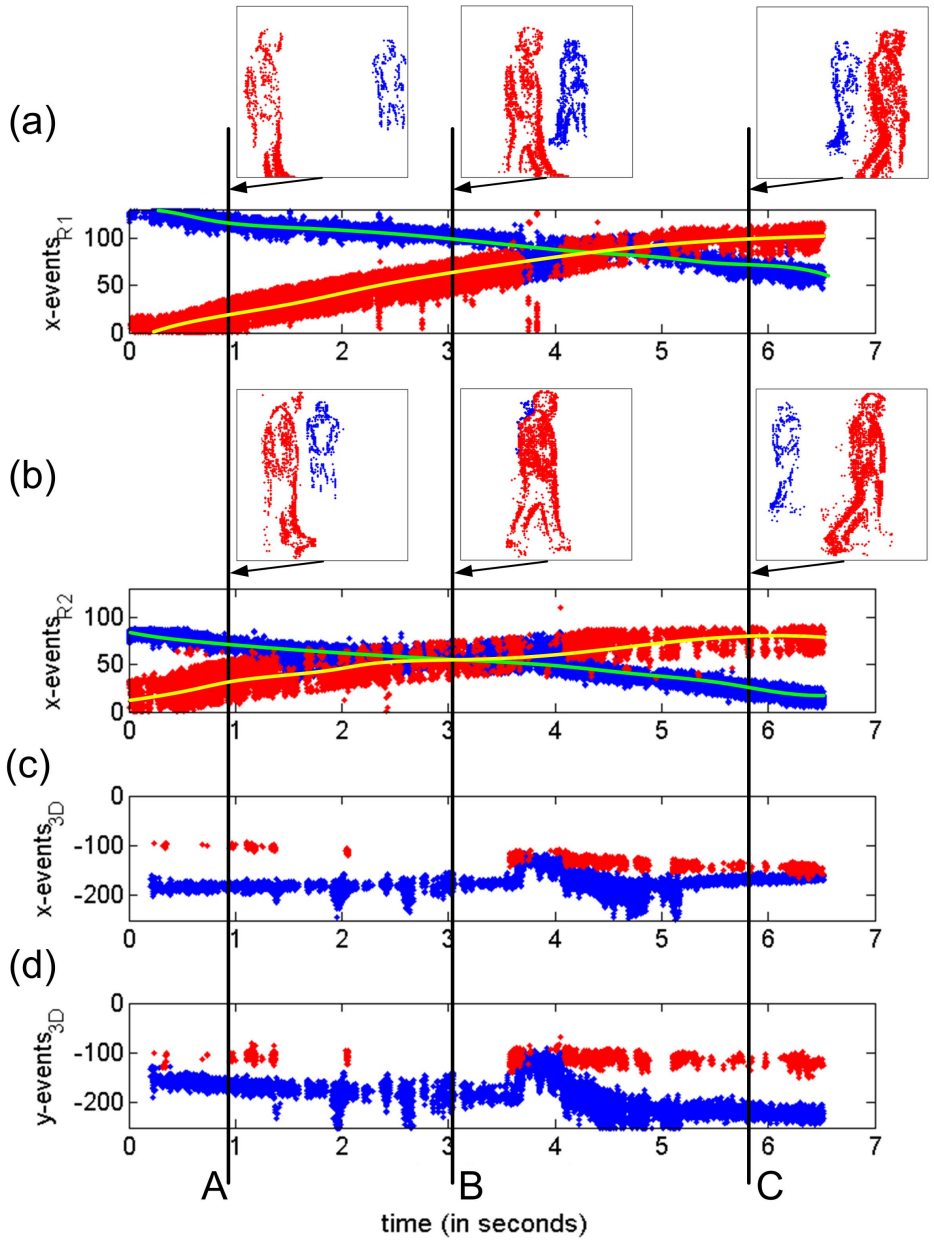

Fig. 16. Results of applying the proposed stereo cluster tracking algorithm to the recording from experiment 4. (a) and (b) x-coordinates of the events recorded by the DVSs [R_1 in (a) and R_2 in (b)], with red and blue indicating the cluster assigned by the algorithm, and yellow and green lines representing the ground truth for each object. The vertical lines **A**–**C** represent specific time instants, and the insets show snapshots from each sensor at those instants. (c) and (d) 3-D events' reconstruction (x and y coordinates, respectively).

algorithm had no prior knowledge about the recording, because the continuity of the movement is maintained. We therefore consider that this error is attributable not to the algorithm, but to uncertainty with regard to the recording. A human observer looking at both 2-D recordings would not be able to say for sure in which direction the objects were rotating.

Besides the confusion in time slot T_1, the algorithm gives correct results for the rest of the recording, as shown in Fig. 14. At the beginning (around $t = 75$ ms, instant **A**), before the fan starts rotating, both objects are clearly separated for sensors R_1 and R_2, as can be seen in the corresponding insets. As soon as rotation begins, the x-coordinates represented in Fig. 14(a) and (b) show that the objects are passing one in front of the other twice per cycle. Time instant **B** (around $t = 500$ ms) therefore represents a situation where for R_2, the blue straw is occluding the red straw [see inset in Fig. 14(b)], while for R_1, the objects are completely separated from each other [see the inset in Fig. 14(a)]. At around $t = 700$ ms (instant **C**), we see the complementary situation where for R_1, the blue straw is occluding the red straw [see the inset in Fig. 14(a)], while the two straws are properly identified by R_2 [see the inset in Fig. 14(b)]. Finally, at around $t = 900$ ms (instant **D**), the situation observed for R_2 is similar to that seen at instant **B**, while the situation observed by R_1 is just the opposite, indicating that it is the red straw that is occluding the blue straw.

This experiment validated the algorithm's behavior when tracking objects moving at high speed, highlighting its main

advantage in comparison with frame-based vision systems. Considering a rotation speed of 6 r/s, and with a frame-based system estimated to need around 20 frames per rotation to track the movement of the objects properly (this is equivalent to around one frame per 20° rotation), a frame rate of some 120 frames/s would be necessary to match the performance of the event-based algorithm proposed in this paper. Furthermore, by applying the success rate measurement method described in Section VI-A to this example, we obtain a success rate of 95.83% (23 correctly solved occlusions out of 24).

D. Experiment 4: Real-World Experiment

As a last experiment, and to evaluate the performance of our method qualitatively with a real-world example, we applied the algorithm to a more realistic scenario: two people walking in front of the stereo setup. One of the people walked from left to right, closer to the DVSs, and the other walked from right to left, farther away from the sensors. The scenario observed by the stereo setup is shown in Fig. 15.

Fig. 16 shows the results of applying the proposed stereo cluster tracking algorithm to the recording obtained from the setup in Fig. 15, with the nearer person passing in front of the one farther away. The 2-D clusters were correctly identified and tracked throughout the recording, as shown in Fig. 16(a) and (b), where the x-coordinates of the events are plotted versus time (red and blue represent the cluster assigned to each event by the algorithm, while the yellow and green lines indicate the ground truth). The 3-D reconstruction information used to track the people during occlusion is shown in Fig. 16(c) and (d), where the x and y coordinates of the 3-D events are plotted versus time and red and blue once again indicate the assigned cluster. The vertical lines **A**–**C** represent specific time instants for which the output of both sensors is shown in the insets.

Toward the beginning of the recording (at around $t = 1s$, instant **A**), both people are correctly identified in both sensors, as shown in the insets in Fig. 16(a) and (b). After 3 s of recording, the nearer person occludes the one farther away in R_2, while both people are still visible in R_1 (instant **B**). The inset for R_1 shows how the algorithm was able to identify both people properly. After 6 s of recording, occlusion has finished in both sensors and the two people can once again be observed separately (instant **C**). The insets in Fig. 16(a) and (b) show how the algorithm identified both people correctly after occlusion: the nearer person is shown in red, and the one farther away in blue. By applying the success rate measurement method described in Section VI-A to this example, we obtained a success rate of 100% (two correctly solved occlusions).

E. Computational Cost Analysis

One of the main advantages of the neuromorphic event-based processing strategy described in this paper is its reduced computational cost. To analyze this advantage, we compared the computational cost of our approach with an equivalent frame-based strategy. We first took one of the experimental recordings used in this paper and extracted the number of events over time, using a time bin of 1 ms. Multiplying this information by the average number of operations performed for each event in our algorithm, we estimated the number of operations per millisecond. This is represented in the blue trace in Fig. 17, with an approximate average of around $2k$ operations per millisecond.

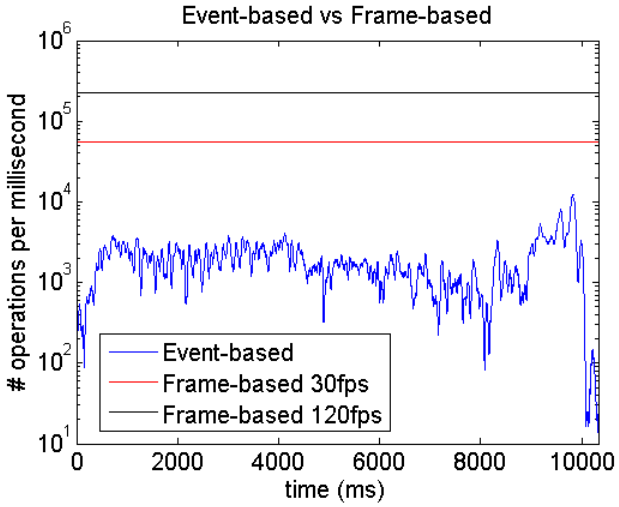

Fig. 17. Comparison between the computational cost of the event-based approach and two frame-based equivalent situations, with frame rates of 30 and 120 frames/s and showing the number of operations per millisecond versus time.

For the frame-based comparison, we considered a resolution of 128×128 pixels, and estimated an equivalent average activity of 1 event/pixel. Based on this estimate, we obtained a number of around $1.8M$ operations per frame, with different frame rates giving different numbers of operations per millisecond (using again a time bin of 1 ms). The red and black traces in Fig. 17 represent the equivalent numbers of operations per millisecond for frame rates of 30 and 120 frames/s, respectively. Based on this estimate, we found that the mean number of operations per millisecond obtained for the event-based approach corresponds to 3.42% of that obtained for a frame-based strategy with 30 frames/s, and 0.85% of that obtained for 120 frames/s.

In this paper, we show how the addition of 3-D information introduces more robustness, especially when dealing with occlusions, as depth adds more constraints with regard to the position of moving objects. The work presented also makes it possible to obtain a 3-D reconstruction, allowing sizes and distances between objects to be measured at no extra computational cost. It was shown that event-based 3-D reconstruction can be implemented using the coincident firing activities of pixels in two different sensors, demonstrating how the merging of both 2-D and 3-D approaches results in more accurate tracking algorithms. This is especially useful when tracking objects moving at high speed, as was shown in experiment 3. In comparison with an equivalent frame-based implementation, this event-based strategy offers a considerable reduction in terms of computational cost, with an average number of operations per millisecond representing only 0.85% of those needed by a frame-based implementation with 120 frames/s.

VII. Conclusion

A new bio-inspired algorithm for event-based stereo object tracking has been proposed. The algorithm exploits the sparse activity generated by the neuromorphic DVSs and their high temporal resolution. Compared with conventional frame-based processing systems, event-based algorithms do not need to use complex costly Bayesian inference techniques. So far, several cluster tracking algorithms have been developed using a single DVS, but they typically encounter problems when dealing with object occlusions. In this paper, a stereo setup was used to extract 3-D information and thereby improve tracking task behavior. Object tracking and stereo matching were integrated in a single algorithm to solve the problem of object occlusion using learned information about the position and size of objects. The algorithm was validated with four different experiments, with two objects moving in front of the stereo setup. When one of the objects was repeatedly occluded by the other, the proposed algorithm was able to identify each object in 3-D space and track their positions. The first experiment used two slow-moving pens, giving an event matching rate of 20%, and the main distance between the objects was measured with an error of 0.5%. The second experiment used a pen and a box, validating the algorithm with a more complex object. The third experiment used two straws rotating at 6 r/s, demonstrating the high-speed capabilities of the proposed method. The last experiment tracked the positions of two people walking in a real-world environment.

References

[1] C. Cédras and M. Shah, "Motion-based recognition a survey," *Image Vis. Comput.*, vol. 13, no. 2, pp. 129–155, Mar. 1995.

[2] G. L. Foresti, "Object recognition and tracking for remote video surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 1045–1062, Oct. 1999.

[3] I. Cohen and G. Medioni, "Detecting and tracking moving objects for video surveillance," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Fort Collins, CO, USA, Jun. 1999, p. 325.

[4] J. M. Rehg and T. Kanade, "DigitEyes: Vision-based hand tracking for human-computer interaction," in *Proc. IEEE Workshop Motion Non-Rigid Articulated Objects*, Nov. 1994, pp. 16–22.

[5] C. von Hardenberg and F. Bérard, "Bare-hand human-computer interaction," in *Proc. Workshop Perceptive User Interfaces (PUI)*, New York, NY, USA, 2001, pp. 1–8.

[6] R. J. K. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," *Mind*, vol. 2, no. 3, p. 4, 2003.

[7] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transp. Res. C, Emerg. Technol.*, vol. 6, no. 4, pp. 271–288, 1998.

[8] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen, "Computer vision for driver assistance systems," *Proc. SPIE*, vol. 3364, pp. 136–147, Jul. 1998.

[9] U. Neumann and S. You, "Natural feature tracking for augmented reality," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 35–64, Mar. 1999.

[10] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 × 128 120 dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[11] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[12] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128 × 128 1.5% contrast sensitivity 0.9% FPN 3μs latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance peramplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, Mar. 2013.

[13] T. Delbrück and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Front. Neurosci.*, vol. 7, p. 223, Nov. 2013.

[14] M. Litzenberger *et al.*, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Toronto, ON, Canada, Sep. 2006, pp. 653–658.

[15] F. Gómez-Rodríguez, L. Miró-Amarante, F. Díaz-del-Rio, A. Linares-Barranco, and G. Jimenez, "Real time multiple objects tracking based on a bio-inspired processing cascade architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2010, pp. 1399–1402.

[16] Z. Ni, S.-H. Ieng, C. Posch, S. Régnier, and R. Benosman, "Visual tracking using neuromorphic asynchronous event-based cameras," *Neural Comput.*, vol. 27, no. 4, pp. 925–953, 2015.

[17] D. R. Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S.-H. Ieng, and R. Benosman, "An asynchronous neuromorphic event-driven visual part-based shape tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3045–3059, Mar. 2015.

[18] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous event-based multikernel algorithm for high-speed visual features tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1710–1720, Aug. 2014.

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

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

[21] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *Proc. IEEE Workshop Motion Video Comput.*, Dec. 2002, pp. 169–174.

[22] I. Mikic, S. Santini, and R. Jain, "Video processing and integration from multiple cameras," in *Proc. Image Understand. Workshop*, San Francisco, CA, USA, 1998, pp. 1–5.

[23] K. Otsuka and N. Mukawa, "Multiview occlusion analysis for tracking densely populated objects based on 2-D visual angles," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun./Jul. 2004, pp. 1–8.

[24] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward categorization on AER motion events using cortex-like features in a spiking neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 1963–1978, Sep. 2015.

[25] X. Peng, B. Zhao, R. Yan, H. Tang, and Z. Yi, "Bag of events: An efficient probability-based feature extraction method for AER image sensors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 791–803, Apr. 2017.

[26] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "HFirst: A temporal approach to object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2028–2040, Oct. 2015.

[27] J. Carneiro, S.-H. Ieng, C. Posch, and R. Benosman, "Asynchronous event-based 3D reconstruction from neuromorphic retinas," *Neural Netw.*, vol. 45, pp. 27–38, Sep. 2013.

[28] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, Feb. 2014.