LETTER **Communicated by Michael Pfeiffer**

Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras

Zhenjiang Ni
ni@isir.upmc.fr
Institute of Robotics and Intelligent Systems, University Pierre and Marie Curie, 75005 Paris, France

Sio-Hoi Ieng
sio-hoi.ieng@upmc.fr
Christoph Posch
christoph.posch@inserm.fr
Vision Institute, University Pierre and Marie Curie, 75012 Paris, France

Stéphane Régnier
stephane.regnier@upmc.fr
Institute of Robotics and Intelligent Systems, University Pierre and Marie Curie, 75005 Paris, France

Ryad Benosman
ryad.benosman@upmc.fr
Vision Institute, University Pierre and Marie Curie, 75012 Paris, France

This letter presents a novel computationally efficient and robust pattern tracking method based on a time-encoded, frame-free visual data. Recent interdisciplinary developments, combining inputs from engineering and biology, have yielded a novel type of camera that encodes visual information into a continuous stream of asynchronous, temporal events. These events encode temporal contrast and intensity locally in space and time. We show that the sparse yet accurately timed information is well suited as a computational input for object tracking. In this letter, visual data processing is performed for each incoming event at the time it arrives. The method provides a continuous and iterative estimation of the geometric transformation between the model and the events representing the tracked object. It can handle isometry, similarities, and affine distortions and allows for unprecedented real-time performance at equivalent frame rates in the kilohertz range on a standard PC. Furthermore, by using the dimension of time that is currently underexploited by most artificial vision systems, the method we present is able to solve ambiguous cases of object occlusions that classical frame-based techniques handle poorly.

Neural Computation **27, 925–953** (2015)
doi:10.1162/NECO_a_00720

1 Introduction

Real-time pattern-based tracking is a fundamental component of various applications in artificial vision. It deals with the problem of establishing correspondences between a known model of an object consisting of a set of 2D points and a target appearing on the image plane. The goal is to estimate the relative pose between the model and the target. Current pattern tracking methods are designed to work on images, or frames, acquired at a fixed rate. Image acquisition is usually limited to the order of tens of milliseconds in real-time applications. This drastically restricts the ability to track correctly high-velocity objects. Increasing the frame rate is often not a solution as the increasing amount of acquired data sets a limit to real-time computation. This is currently the bottleneck of several mobile robotics applications, where there is always a trade-off to find between the frame rate and the computational load allowed.

A recent and evolving branch of artificial vision exploits the unique characteristics of a novel family of asynchronous frame-free vision sensors whose principles of operation are based on abstractions of the functioning of biological retinas (Lichtsteiner, Posch, & Delbruck, 2008; Posch, Matolin, & Wohlgenannt, 2011; Lenero-Bardallo, Serrano-Gotarredona, & Linares-Barranco, 2011). These cameras acquire the content of scenes asynchronously. Every pixel is independent and autonomously encodes visual information in its field of view into precisely time-stamped spike events. These cameras allow the derivation of a new methodology for computation that operates on pixels or areas of the image at the time they convey relevant (dynamic) information. Processing can be performed on standard digital hardware and takes full advantage of the precise timing of the events. The increasing availability and the improving quality of these sensors open up the potential to introduce a shift in the methodology of acquiring and processing visual information in various demanding machine vision applications (Perez-Carrasco, Serrano, Acha, Serrano-Gotarredona, & Linares-Barranco, 2008; Benosman, Ieng, Clercq, Bartolozzi, & Srinivasan, 2012; Benosman, Ieng, Posch, & Rogister, 2011; Serrano-Gotarredona et al., 2009).

As we will show, this biomimetic asynchronous acquisition allows to introduce a novel computationally efficient and robust visual real-time pattern-based tracking method that relies on the accurate timing of individual pixels' response to visual stimuli. We will further show that asynchronous acquisition allows us to develop a tracking technique that can follow patterns at an equivalent frame rate of several kilohertz, overcoming occlusions at the lowest computational cost. The high temporal precision of the presented methodology does not necessitate motion estimation filters to ensure the smoothness of the motion trajectory or the use of a priori kinematics models.

There has been a lot of research on visual pattern-based tracking. The algorithms differ in three main aspects: the appropriate representation for the

patterns of interest, the transformation model used to transform the source model to match with the target pattern, and the tracking criterion used to estimate the optimal transformation parameters given a pattern representation and a transformation model. Point cloud is the most commonly used pattern representation. Iterative closest point (ICP), as the name suggests, iteratively uses a point set delineating the object contour to match an acquired data point set by searching their nearest correspondences (Besl & McKay, 1992). Basic curve or surface elements can also be used for matching. Robust point matching (RPM) is a method to deal with nonrigid shapes by treating the point matching as an optimization problem to preserve local neighborhood structures (Zheng & Doermann, 2006). The shape contexts method attaches a descriptor to each point during the matching process (Belongie, Malik, & Puzicha, 2002). Moreover, distance functions or distance transforms are applied for shape alignment in Huang, Paragios, and Metaxas (2006). A level-set-based implicit shape representation framework is presented in Cremers, Osher, and Soatto (2006) for human body tracking and object segmentation. Finally, a parametric curve or surface requires explicitly parameterizing shapes, which makes arbitrary form shape representation non-trivial (Abd El Munim & Farag, 2007).

The transformation between the data set and the model set may be separated into two categories. Global tracking is performed between the known model set and the data set to express their overall geometric relations by satisfying some tracking criterion. As an example, a modified ICP relies on affine transformations (Rusinkiewicz & Levoy, 2001). The second category deals with a local deformations issue to provide a more accurate match. Thin plate spline (TPS; Bookstein, 1989) is such a method for parameterizing nonrigid spatial mapping and modeling deformation. The combination of robust point matching and thin-plate splines forms a more general framework known as the TPS-RPM method (Yang, 2011). Other known methods are free-form deformations (FFD; Rueckert et al., 1999) and radial basis functions (RBF; Fornefett, Rohr, & Stiehl, 1999). Complete solutions usually combine a global tracking and a local deformation transform to achieve the best tracking result. Local deformations are beyond the scope of this letter and will be dealt with in future work. In this letter, we focus on introducing a novel methodology for visual computation using the notion of visual events for shape tracking.

Another key characteristic of the proposed approach is its ability to deal with highly dynamic situations in real time. Generally the most time-consuming part of tracking algorithms lies in determining the correspondences between the data and the model. K-dimensional (kd) tree allows the matching procedure in ICP to be reduced from an average complexity of $O(n^2)$ to $O(n \log n)$. A multiscale or coarse-to-fine scheme is illustrated in both Jost and Hugli (2003) and Kim & Kim (2010) to accelerate the process. Hardware implementations (e.g., FPGA) are often employed to overcome performance limitations toward real-time operation (Dandekar & Shekhar,

2007). However, hardware development cycles are usually longer than that of software dealing with the same algorithm complexity, and complex tracking algorithms are difficult to express in a hardware description language such as VHDL. Although many techniques improve the processing speed, most of them are still not adapted for real-time high-speed applications. The frame-based tracking's computations remain bounded by the acquisition frequency of the camera used. Furthermore, most of the developed techniques are power and time-consuming and are mostly used offline. The visual shape tracking technique we describe offers an alternative in performing pattern tracking using the timing of changes of individual pixels' illuminance as a computational input. We will show that this technique is naturally highly robust to occlusions and that it allows real-time processing at a very high temporal accuracy and very low computational cost.

2 Time-Encoded Imaging

Biomimetic, event-based cameras are a novel type of vision devices that, like their biological role models, are driven by events happening within the scene, and not like conventional image sensors by artificially created timing and control signals (e.g., frame clock) that have no relation whatsoever to the source of the visual information (Delbruck, Linares-Barranco, Culurciello, & Posch, 2010). The pattern tracking method that we present is designed to work on the data delivered by such a time-encoding sensor and takes full advantage of the superior characteristics—most important, the ultra-high temporal resolution and the sparse data representation.

The asynchronous time-based image sensor (ATIS) used in this work is a time-domain encoding image sensors with quarter video graphics array resolution (Posch et al., 2011; Posch, Matolin, & Wohlgenannt, 2008). The sensor contains an array of fully autonomous pixels that combine an illuminance change detector circuit and a conditional exposure measurement block.

As shown in the functional diagram of the ATIS pixel in Figure 1, the change detector individually and asynchronously initiates the measurement of an exposure/gray scale value only if—and immediately after—a brightness change of a certain magnitude has been detected in the field of view of the respective pixel. The exposure measurement circuit in each pixel individually encodes the absolute instantaneous pixel illuminance into the timing of asynchronous event pulses, more precisely into interevent intervals.

Since the ATIS is not clocked like conventional cameras, the timing of events can be conveyed with a very accurate temporal resolution on the order of microseconds. The time domain encoding of the intensity information automatically optimizes the exposure time separately for each pixel instead of imposing a fixed integration time for the entire array, resulting in an exceptionally high dynamic range and improved signal-to-noise

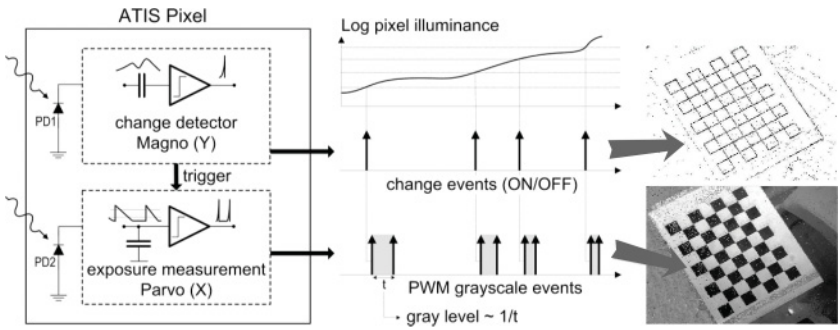

Figure 1: Functional diagram of an ATIS pixel (Posch, 2012). Two types of asynchronous events, encoding change and brightness information, are generated and transmitted individually by each pixel in the imaging array.

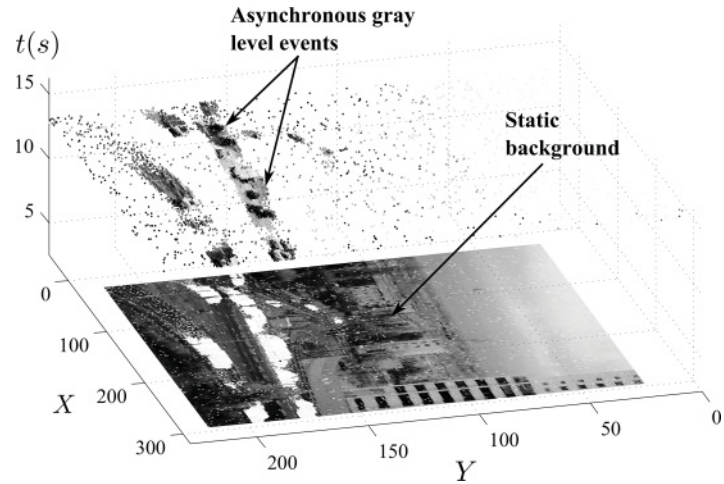

Figure 2: The spatiotemporal space of imaging events. Static objects and scene background are acquired first. Then dynamic objects trigger pixel-individual, asynchronous gray-level events after each change. Frames are absent from this acquisition process. Illustration inspired from Clady et al. (2014).

ratio. The pixel-individual change detector–driven operation yields almost ideal temporal redundancy suppression, resulting in a maximally sparse encoding of the image data.

Figure 2 shows the general principle of asynchronous imaging spaces. Frames are absent from this acquisition process. They can, however, be reconstructed, when needed, at frequencies limited only by the temporal resolution of the pixel circuits (up to hundreds of kiloframes per second).

Static objects and background information, if required, can be recorded as a snapshot at the start of an acquisition, while moving objects in the visual scene describe a spatiotemporal surface at very high temporal resolution.

3 Event-Based Pattern Tracking Algorithm

The event-based pattern tracking updates iteratively the model location and orientation to match as close as possible to the target the 2D image plane based on the arrival of events. The algorithm is tailored to track objects even if they are subject to affine transformations. In a first stage, the novel algorithm making full use of the sparse data will be shown. In a second stage, important spatiotemporal properties are derived from the generated events and will be shown how to use them for multiple object occlusions disambiguation.

3.1 Spatial Matching. Let $e(\mathbf{p}, t)$ a triplet giving the pixel position $\mathbf{p} = (x, y)^T$, the time t of the event. The first step of the pattern tracking relies on the use of a matching function to pair the active pixels of the event-based sensor with the preconstructed model's points. Let $G(t)$ be the set of 2D model points defining the object at time t. $G(t)$ can be continuously updated over time t to the object locations. A matching function for an incoming $e(\mathbf{p}_i, t_i)$ can be defined by

$$\begin{aligned} &match : \mathbb{R}^2 \to G(t) \\ &\qquad \mathbf{p}_i \mapsto \mathbf{p}_k, \text{ with } k = \underset{k \in \{1,\ldots,N_G\}}{\operatorname{argmin}} d(\mathbf{p}_i, \mathbf{p}_k), \end{aligned} \tag{3.1}$$

where $d(\mathbf{p}_i, \mathbf{p_k})$ is the Euclidean distance between two points and N_G is the cardinality of $G(t)$.

3.2 Event-Based Estimation of Geometric Transformation. Once the matching stage has been performed, it becomes possible to update the geometric transformation between the events and the model. $\mathbf{c}$ is defined as the spatial location of the center of gravity of the model (see Figure 3). In the calculation afterward, the transformation is performed with respect to the center of gravity; thus, we define $\mathbf{q}_i = \mathbf{p}_i - \mathbf{c}$ and $match(\mathbf{q}_i) = match(\mathbf{p}_i) - \mathbf{c}$.

For an incoming event $e(\mathbf{p_i}, t_i)$, we define a cost function as

$$f = \| \mathbf{q}_i - R(\theta)(match(\mathbf{q}_i)) - \mathbf{T} \|^2, \tag{3.2}$$

where $R(\theta)$ is the rotation by an angle θ and $\mathbf{T}$ is the translation vector of the geometric transformation that maps $\mathbf{p}_i$ to $match(\mathbf{p}_i)$. The transformation can be summarized as: rotate the model into the same orientation as the pattern and then translate it from its current position to the best matched

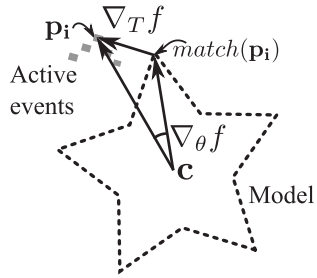

Figure 3: The iterative rigid transformation parameters based on a single event's location. Each iteration induces a small change in the rotation angle (respectively, translation vector) given by the partial derivative of f with respect to θ, $\nabla_\theta f$ (respectively, with respect to $\mathbf{T}$, $\nabla_T f$). The transformation defined by θ and $\mathbf{T}$ according to the model center of gravity $\mathbf{c}$ can be decomposed in two steps: (1) it rotates $match(\mathbf{p}_i)$ of an angle θ along the normal going through $\mathbf{c}$ and (2) then translates it by $\mathbf{T}$.

ones. The goal is then to minimize the least mean square cost function such as,

$$\min_{R(\theta)\in SO(2),\mathbf{T}\in\mathbb{R}^2} \sum f, \tag{3.3}$$

where $SO(2)$ represents the set of all the rotation matrices in $\mathbb{R}^2$.

The event-based visual data are spatially independent and temporally asynchronous. The frame-based processing technique is thus not adapted. An iterative framework responding immediately to the incoming event is more suitable and efficient. We derive an iterative solution from Hersch, Billard, and Bergmann (2012) and update the transformation parameters $\mathbf{T}$ and θ iteratively according to a gradient descent scheme with the terms

$$\Delta\mathbf{T} = \eta_1 \nabla_T f, \tag{3.4}$$

$$\Delta\theta = \eta_2 \nabla_\theta f, \tag{3.5}$$

where η_1 and η_2 are the adjusting parameters and $\nabla_T f$ and $\nabla_\theta f$ are the partial derivatives with respect to $\mathbf{T}$ and θ. Since we are considering only rotations in the focal plane, only the rotation angle θ changes in R (i.e., the rotation axis is constant). The iterations are then performed on arrival of each event so that it drags the shape to a new position in the direction of the gradient.

The translation term is given by the derivative of f with respect to $\mathbf{T}$:

$$\nabla_T f = 2(\mathbf{T} - \mathbf{q}_i + R(\theta)match(\mathbf{q}_i)). \tag{3.6}$$

The rotation angle is given by the partial derivative of f with respect to θ:

$$\nabla_\theta f = 2(\mathbf{q}_i - \mathbf{T})^T R_{\frac{\pi}{2}} R_\theta (-match(\mathbf{q}_i)), \tag{3.7}$$

where $R_{\frac{\pi}{2}}$ is the rotation in the z-axis, about an angle of $\pi/2$. The superscript T in equation 3.7 stands for the transpose operation.

The similar approach can be extended to affine transformations as they contain scaling, rotation, shearing, and translation that can be described efficiently by small perspective changes between successive iterations. They allow the process to be more robust and to cope with more general geometric deformations encountered in real scenes. The general affine transformation is mathematically defined as

$$\mathbf{q}_i = A\,(match(\mathbf{q}_i)) + \mathbf{T}, \tag{3.8}$$

where A is the linear map composed by a scaling and a rotation. The cost function becomes

$$f = \parallel \mathbf{q}_i - A\,(match(\mathbf{q}_i)) - \mathbf{T} \parallel^2 . \tag{3.9}$$

Equation 3.8 can then be rewritten as

$$\mathbf{q}_i = R(\theta) \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} match(\mathbf{q}_i) + \mathbf{T}. \tag{3.10}$$

s_x and s_y are scaling factors—in other words, the change of the size of the object—that applied to the two axes with the following definitions:

- $\mathbf{q}_i = (q_{ix}, q_{iy})^T$.
- $match(\mathbf{q}_i) = (\tilde{q}_{ix}, \tilde{q}_{iy})^T$.
- $\mathbf{s} = (s_x, s_y)^T$.

To eliminate the orientation and retrieve only the scale information, we set $(|q_{ix}|, |q_{iy}|)^T$ to represent the size of the object at a particular moment and $(|\tilde{q}_{ix}|, |\tilde{q}_{iy}|)^T$ as the size of the model along the x- and y-axes.

The difference between these two vectors provides a good estimation of the scale error between consecutive iterations along the two axes. The sign of the difference indicates a need to enlarge or shrink the model set. Parameter $\mathbf{s}$ is then updated as

$$\mathbf{s}_{i+1} = \mathbf{s}_i \left(1 + \eta_3 \left(\begin{pmatrix} |q_{ix}| \\ |q_{iy}| \end{pmatrix} - \begin{pmatrix} |\tilde{q}_{ix}| \\ |\tilde{q}_{iy}| \end{pmatrix} \right) \right), \tag{3.11}$$

where η_3 is the scale adjusting parameter and i is the iteration index.

Remark. It is important to emphasize that the spatiotemporal approach performs an iterative computation for each incoming event rather than recreating images by accumulated events. This event-based approach significantly reduces the computational load as it is not processing previously processed data. It allows computations at temporal resolutions equivalent to frequencies of several kHz, as we show in section 4.

3.3 Occlusions in Multiple Object Tracking. When multiple objects cross and occlude each other, both spatial and temporal constraints can be used, separately or in combination, to disambiguate these overlaps. An ambiguous event is an event that can possibly be attributed to several objects. In actual implementation, the distances between the event and the closest matching point on each interested object are calculated. If only one of the distances is below a threshold (e.g., 3 pixels), then this event is attributed to that object; otherwise it is considered ambiguous.

3.3.1 Spatial Constraints. Considering spatial information, several strategies can potentially be used to assign incoming events to multiple overlapping shapes:

1) *Assign to closest*. An incoming event is assigned to the closest model. In theory, this should work correctly if the incoming event perfectly matches the model shape and if the data contain no or little noise.
2) *Reject All*. This approach simply throws away all ambiguous events. This appears to be a logical solution since a priori, no clear decision can be made. However, the tracking may fail as the true dynamics of shapes is lost.
3) *Update all*. All the models around an ambiguous event are updated,
4) *Weighted update*. An ambiguous event contributes to each model according to a weighted update proportional to its distance to each shape.

3.3.2 Temporal Constraints. The high temporal resolution of event-based acquisition provides crucial additional information to solve ambigous situations. A shape's event rate contains information about the tracked object and is partly encoding the object's dynamics.

Let t_c be the time stamp of the current event $e(\mathbf{p}, t_c)$. A sequence of time stamps of N events within a time window of a given object k can be written as $(t_0^k, t_1^k, \ldots, t_N^k)$, with t_N^k being the last that happens before t_c. These time stamps of each object k are used to calculate a moving average of events' rate r^k, which can be defined by

$$r^k(t_c) = \frac{N}{t_N^k - t_0^k}. \tag{3.12}$$

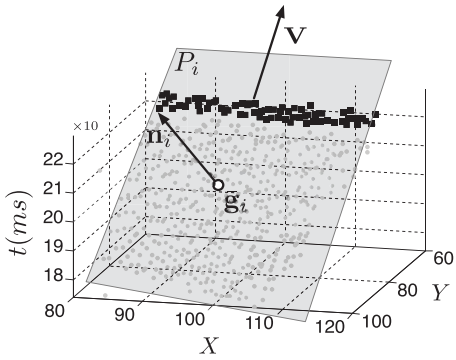

Figure 4: Edge sweeping a surface over time. The surface is approximated by a mean plane P_i, defined by the normal $\mathbf{n}_i$ and $\overline{\mathbf{g}_i}$, the center of gravity of the point cloud used for the plane fitting. The edge is sweeping the plane at velocity $\mathbf{V}$, a 2D vector embedded in P_i.

In a second stage, a score function for each object is defined by

$$C^k = (t_c - t_N^k) r^k(t_c). \tag{3.13}$$

An object indexed by k, moving at a constant speed within a short period, is characterized by an event rate r^k. An event $e(\mathbf{p}, t_c)$ is tested for its temporal consistency with each object k. If t_N^k is the time of the last event assigned to the object k, the duration $(t_c - t_N^k)$ should be close to the rate r^k if the $e(\mathbf{p}, t)$ belongs to the same object k. In short, the event is assigned to the object that leads to the C^k closest to 1.

3.3.3 Spatiotemporal Constraints. The spatial and temporal constraints can be combined to improve the disambiguation process further. For simplification, let us assume a simple straight edge moving in the scene. It will sweep a plane in the spatiotemporal space, as shown in Figure 4. If the object is more complex, multiple planes will be generated by different edges of that object. The essential idea of the spatiotemporal constraint is to adjust a continuous surface on the event cloud to smooth the input events. A mean plane is a reasonable linear fitting of the events and is a simple manifold that is differentiable everywhere on its surface. The angle defined by the plane's (P_i) normal and the t-axis is related to the speed of the object. In practical implementation, to accelerate the averaging process, the mean plane is represented by a matrix that stores only the time stamp of the last spike emitted by each pixel. Each moving object seen by the sensor is associated with a mean plane P_i, summarized by its normal $\mathbf{n}_i$ and a point by which P_i passes through, $\overline{\mathbf{g}_i}$, which is also the center of gravity of the point cloud on which the plane is fitted. For each tracked target, the plane

Algorithm 1: Event-Based Shape Tracking Algorithm.

Require: The object model $G(t = 0)$.

1: **for** every incoming $e(\mathbf{p_i}, t_i)$ **do**

2: Compute $match(\mathbf{p_i})$ using equation 3.1 to find the event's matching point for each model and use a strategy from section 3.3 to assign an event to an object.

3: Compute $(\nabla_\theta, \nabla_\mathbf{T})$ using equations 3.6 and 3.7.

4: Refresh the global tracking parameters $(R(\theta), \mathbf{T}, \mathbf{s})$ using equations 3.4 , 3.5, and 3.11.

5: Update the position of model points in $G(t)$ using equation 3.10.

6: **end for**

is initialized by selecting events that belong to the object; only the spatial constraint is used to select events during the initialization. The plane parameters are then continuously updated by using the spatiotemporal constraint afterward. The plane-fitting process can be achieved by applying a principal component analysis (PCA) to estimate the vector $\mathbf{n}_i$. (Readers who are interested in more detail on the property of spatial temporal space can refer to Benosman, Clercq, Lagorce, Ieng, & Bartolozzi, 2014.) To assign an event to a particular target among N objects, the distances between the incoming event and every fitted plane P_i, $i \in \{1, \ldots, N\}$ of the spatiotemporal surfaces must be computed. An event $e(\mathbf{p}, t)$ is assigned to the target i if

$$i = \underset{j \in \{1,\ldots,N\}}{\operatorname{argmin}} ||(e(\mathbf{p}, t) - \overline{\mathbf{g}_j})^T \mathbf{n}_j||. \tag{3.14}$$

In case of two moving objects that overlap, the set of ambiguous events in the spatiotemporal space is a line. Compared to the large set of events describing the moving objects, occlusions concern only a tiny subset of them, thus providing more robustness to the tracking process.

The complete event-based tracking algorithm is summarized in algorithm 1.

4 Experiments

In the first stage, experiments are carried out in a controlled laboratory environment to prove the method and study its properties. In the second

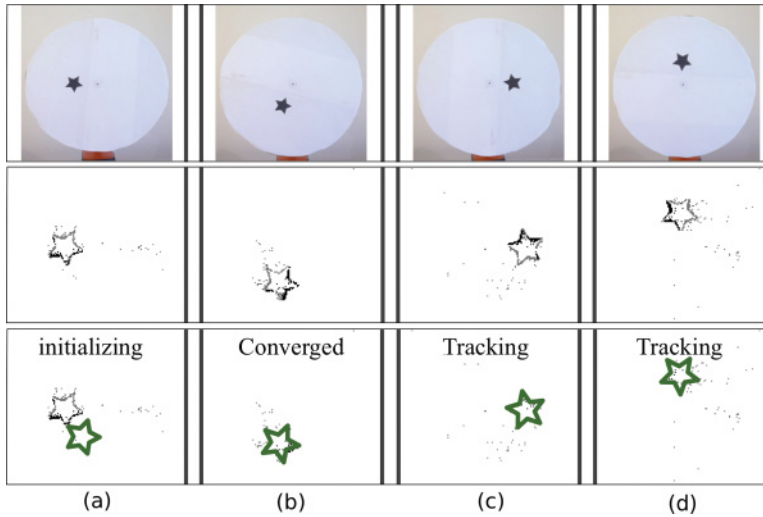

Figure 5: Event-based shape tracking of a predefined star-shaped target. The top row shows a sequence of conventional image frames of a star rotating on a disk, the middle row shows frames from accumulated events acquired by the event-driven camera, and the bottom row provides the result of the event-based tracking method. The three images in the first column (a) provide the initial state when the star starts to rotate. The model represented by solid green lines is trying to match the corresponding closest events. In panel b, the star passes close to the model and thus pulls the model to its position. Finally, after a few iterations, in panels c and d, while the star is still moving, the model succeeds in superimposing with the star by updating its parameters according to the pose of the star.

stage, outdoor multitracking experiments are done to investigate different spatial and temporal constraints for disambiguating occlusions.

4.1 Controlled Environment Experiments. A tracking example on event-based data of a moving star shape is shown in Figure 5. The camera acquires data of the shape fixed on a rotating disc. The camera faces the disc such that the geometric relation between the model and the acquired events is roughly isometric. The disc rotates at a speed of 670 rpm. The model (similar in the following sections) is generated manually by selecting pixels in a frame. It contains 60 points—6 on each edge. A distance threshold of 6 pixels is applied to each incoming event for the matching stage in order to eliminate the impact of noise and reduce the computational load at this stage. As shown in Figure 5, the algorithm succeeds in tracking the rotating shape efficiently despite of the high rotation velocity.

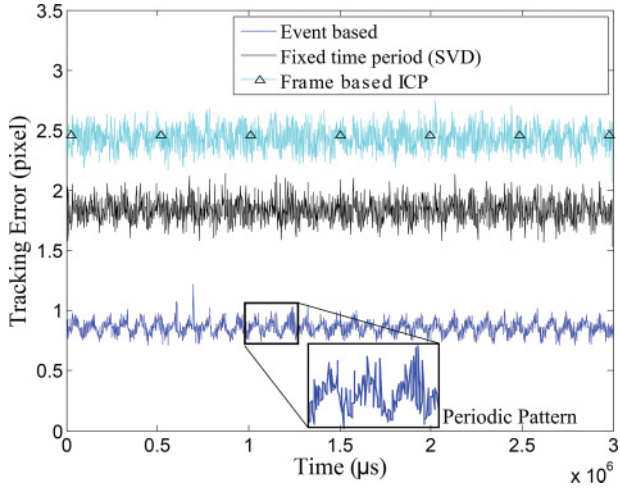

Figure 6: A comparison of the tracking precision between the event-based tracking with iterative event-based isometry transformation, fixed-time-period rigid transformation using SVD, and the framed-based ICP showing mean errors of 0.86, 1.83, and 2.43 pixels with standard deviations of 0.11, 0.19, and 0.20 pixels, respectively. The periodic pattern is due to the rotation periodicity.

In order to provide a comparison with conventional frame-based acquisition and processing, two classic frame-based shape tracking techniques have been tested on artificially created frames. The first relies on a classical approach using singular value decomposition (SVD; Challis, 1995), and the second is the ICP algorithm.

The SVD method requires sufficient measurements to produce correct estimations. We then generate frames from a 50 μs event accumulation period. The 50 μs is determined from experiments.

The frame-based ICP uses images generated by accumulating events during periods of 200 μs. The process simulates a hypothetical high-speed frame-based camera running at 5 kHz. The generated image implies that points can only be processed in batch mode on the whole frame. In order to ensure a fair comparison, only pixels' locations remaining after frame differencing are considered. The 200 μs has been determined experimentally so that the generated frame contains sufficient events in order to have the full shape of the tracked target.

To evaluate the precision of event-based tracking, the points' fitting accuracy is measured by computing the mean distance between the model set and the location of active events every 200 μs. Figure 6 shows the tracking error with respect to time. The event-based tracking with iterative isometry estimation, closed-form rigid transformation based on SVD, and a purely frame-based ICP achieves mean errors of 0.86, 1.83, and 2.43 pixels with

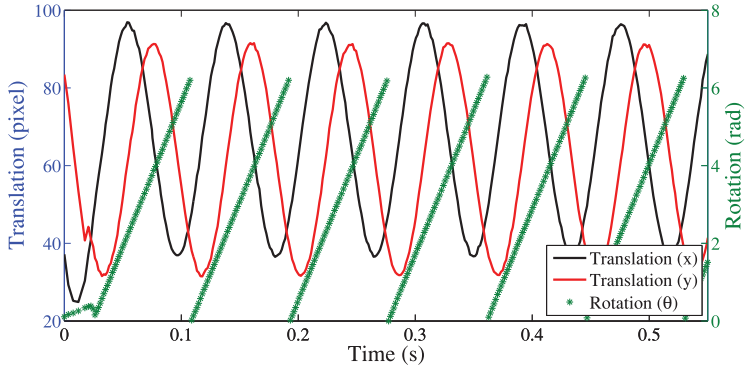

Figure 7: The trajectory of the translation and rotation of the rotating star. The translation in the x- and y-axes are shown in solid lines, while the star-marked curve shows the angular motion. The linearity of the rotation angle and the regular sinusoids outlines the constant angular rotation of the star.

standard deviations of 0.11, 0.19, and 0.20 pixels. Higher temporal resolution leads to more precise tracking. The differences between successive tracking results are less significant when the temporal resolution is finer (Ni, Pacoret, Benosman, Ieng, & Régnier, 2011). A periodic pattern can be observed on the error curve, which originates from the repetitive rotations of the stimulus. The tracking error demonstrates a good reproducibility and reliability of the algorithm. This error is due to the pixels' square array geometry combined with the limited spatial resolution of the sensor that does not provide an isotropic response with respect to the angular position of the shape.

Figure 7 shows the trajectory of the translation and rotation estimates of the rotating star. The solid curves show the translation with respect to the x- and y-axes in the image plane. The sinusoidal curves indicate a repetitive circular motion in the 2D plane. The dotted curve records the rotation (modulo 2π), and the trajectory corresponds logically to a constant angular velocity. In order to provide a measurement of our tracking algorithm, the ground truths are analytically calculated by using the fully calibrated speed of the DC motor that can be compared to the measured tracking data. The errors are shown in Figure 8. The solid lines are errors of translations, and the dotted ones are those of rotation. The mean spatial and angular errors and their standard deviations are, respectively,

- $\overline{T}_x = -0.11$ pixel, $\sigma_x = 0.45$ pixel
- $\overline{T}_y = -0.19$ pixel, $\sigma_y = 0.54$ pixel
- $\bar{\theta} = -0.02$ rad, $\sigma_\theta = 0.023$ rad

One can see that the spatial errors are subpixelic.

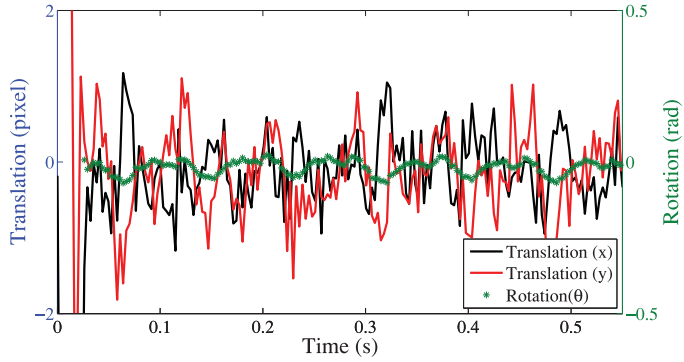

Figure 8: The errors between the estimated motion parameters and the ground truth for the rotating star: translation in solid lines and rotation in star-marked ones.

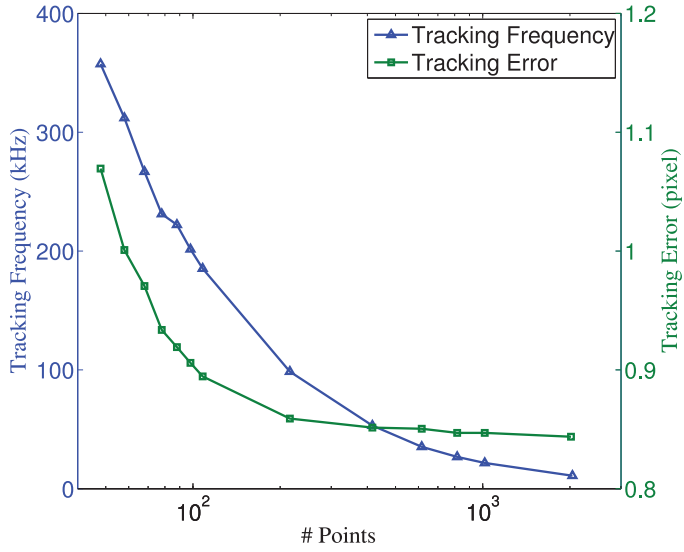

Figure 9: The influence of the size of the model on the tracking frequency in kHz (triangle marker) and the tracking error in pixel (square marker).

4.2 Impact of Number of Points. This section focuses on the impact of the size of the model on computational costs and precision.

Figure 9 shows the link between the size of the model and the maximum frequency that can be achieved. The maximum equivalent frame rate is defined as the inverse of the computational time for processing one event. In our application, a size of 90 model points can provide a detection frequency

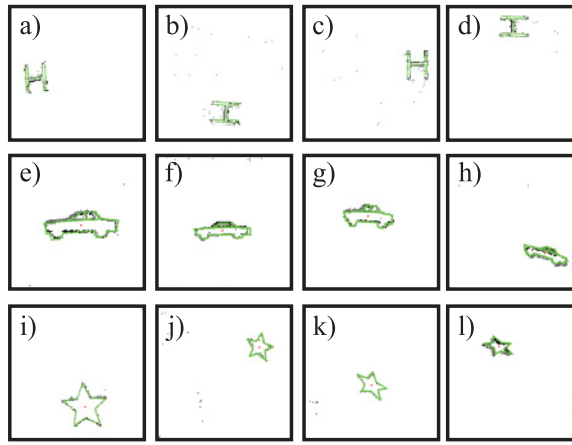

Figure 10: Indoor tracking experiment. Accumulation frames show the tracking of the shape of (a–d) the character H, (e–h) a car, and (i–l) a star. Apparent change of sizes and perspectives are tested in panels e to l. The rotation, the translation, and the scaling in the x, y-dimensions are estimated simultaneously. The tracking sequence can be found in the supplemented video.

of around 200 kHz equivalent frame rate. Up to 2000 model points, the pose can be updated at an equivalent frame rate of 11 kHz. The experiment shows that for a size of 60 to 70 model points, the algorithm is able to track in real time a shape moving at a speed up to 1250 rpm. As a guideline for selecting the size of the model, the corners on the object contour should always be used. On a straight edge, however, the number of points can be reduced, as this will usually not have an influence on the final tracking precision.

The mean tracking error with respect to the size of the model is also illustrated in Figure 9. When the point number increases, the error converges to 0.84 pixel due to the spatial resolution limit of the event-based camera. Logically, larger models achieve better tracking precision but require more computational effort. A size of 60 to 100 points is a good compromise for maintaining a reasonable precision (around 0.90 pixel) and preserving a high tracking frequency (around 200 kHz). The program is running on a computer equipped with Intel Core i5 CPU at 2.8 GHz, taking 25% the CPU occupation.

4.3 Affine Tracking. To examine the algorithm's affine tracking capability, three objects are used as the target objects: a rotating H, a star, and a car (see Figure 10). During the experiment, the distance and the perceptive angle between the object and the event-based camera are continuously changing. The objects are de-formed and resized as shown in Figure 10. In

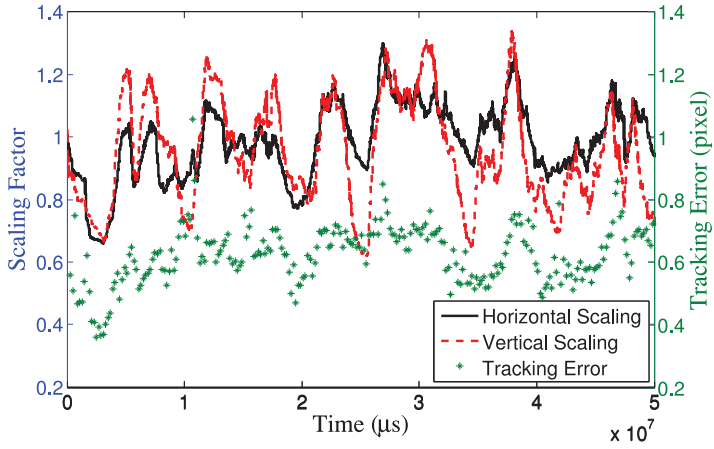

Figure 11: (Top) The scaling factor applied along x-(black) and y-(red) axes for the car tracking sequence. The factor is bigger vertically because the tilt is larger. (Bottom) The tracking error is shown with a mean value of 0.64 and a standard deviation of 0.48 pixels.

Figure 11, the scaling factor with respect to the original size during the car tracking is shown. Different scaling ratios along two axes can be observed. This difference is due to the fact that the changes of perspective are more often performed along the vertical axis during the test. This experiment shows that the algorithm can efficiently track complex planar objects moving in 3D space with a mean tracking error of 0.64 pixels and a standard deviation of 0.48 pixels (see Figure 11). Due to the flexibility added by the scaling factors, the affine estimation appears to have more accurate point matching between the model and the active events than a rigid transform, an expected result.

4.4 Object Tracking in Outdoor Scenes. In this experiment, real-world car traffic data acquired with the asynchronous camera are used. As shown in Figure 12, vehicles are moving simultaneously on different lanes. In order to provide a fair comparison, the same sequence has been used for both the time-oriented algorithm and the classic ICP frame-based method as appearing in Du, Zheng, Ying, and Liu (2010). The gradient adjusting parameters η_1, η_2, and η_3 as described in section 3 are experimentally set to 0.1, 2.10^{-5}, and 10^{-5}, respectively. While the event-based method uses the timing of each incoming pixel's value at a precision of 1 μs for computation, the classic frame-based ICP relies on the gray-level images reconstructed at a frame rate of 100 fps. Each image is preprocessed. Moving edges are obtained by computing the frame difference between successive images followed by a thresholding operation. Frame-based ICP is then applied

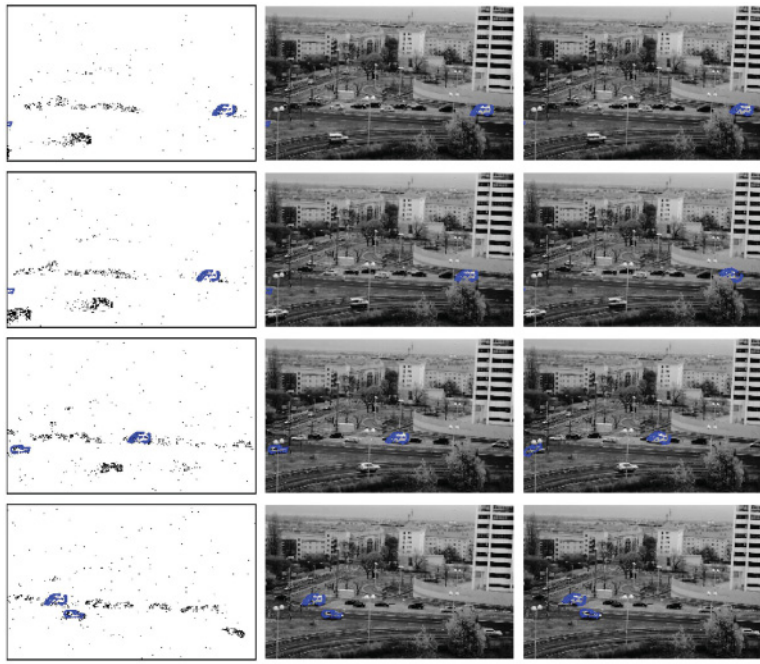

Figure 12: Real-world traffic image sequence and achieved tracking results. (a) Event accumulation frames with event-based tracking. (b) Gray-level images with event-based tracking. (c) Gray-level images with frame-based tracking. Critical instants exhibiting strong occlusions are specially selected. Static occlusions like trees and lamp posts and mobile occlusions like passing cars and trucks constitute significant challenges for robust tracking.

for object tracking. No postprocessing motion filters are supplemented to smooth the motion trajectory.

Figure 12 shows the results of the two algorithms applied to follow multiple targets—in this case, two objects simultaneously. A van and a car have been selected as tracking targets. The car model has 67 points, and the van model has 102 points. Templates are generated by manually clicking points on an acquired image. Column a of Figure 12 shows the event accumulation frame generated by each vehicle superimposed with the matched shape using the event-based algorithm. The same results are superimposed on the corresponding gray-level images in column b. Finally, column c shows the tracking results of the frame-based ICP. The tracking errors for each method are given in Figure 13. The image patches correspond to the critical situations encountered by both techniques. Each image patch corresponds to

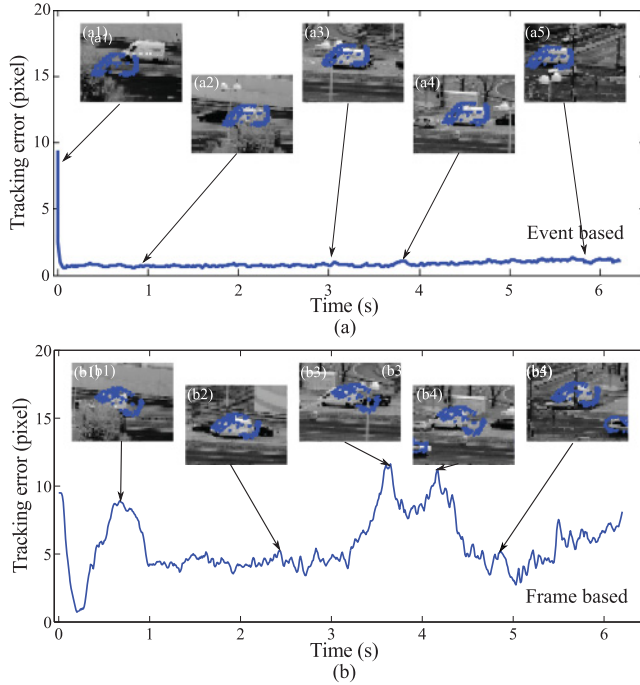

Figure 13: Tracking errors for both (a) event-based and (b) frame-based approaches. The mean tracking errors for event-based and frame-based approaches are, respectively, 0.86 and 5.70 pixels, with variances of 0.19 and 2.11 pixels. Image patches show several difficult situations for the tracking. They correspond to the locations where the mean error is maximal. A supplementary video is available for the comparison at https://drive.google.com/file/d/0B5gzfP0R1VEFdHRUUFFrWTJTSXM/view?usp=sharing.

several local maxima of the error curve. For better readability, we chose to label tracking errors only from the van experiment. Results from other moving objects exhibit the same range of errors. Due to its size and path in the scene, the van encounters several severe occlusions.

Large static and mobile occlusions appear frequently in this sequence. Images shown in Figure 12 are selected for highlighting the difficult situations. The van is entering an occlusion zone due to a tree covering part of the lane. The situation can be seen more clearly in a1, a2, and b1 in Figure 13. The third row shows a case of dynamic occlusion due to several moving cars surrounding the "van". The situation can also be seen in frames a4, b2, and b3 in Figure 13. Finally, the last row shows the van in a combined case of static and dynamic occlusions caused by a light pole and multiple cars

that are moving simultaneously, also observable in frames a2, a5, b3, and b5 in Figure 13.

The results in Figure 12 and their corresponding errors in Figure 13 show that the computation using precise timing is more stable than the frame-based one, which loses the shape's orientation several times. The mean tracking errors for event- and frame-based methods are around 0.86 and 5.70 pixels, respectively, with standard deviations of 0.19 and 2.11 pixels. The frame-based ICP method is not always capable of providing a correct match. Wrong point correspondences lead the minimization process to local minima. The method suffers more from static or mobile occlusions (tree, lamp posts, automobiles, and so on) with errors raising up to more than 10 pixels. A high frame rate acquisition (100 Hz in this case) is not always sufficient to overcome occlusions. The dynamic content of the asynchronous, event-based sensor data provides more stable input. Static obstacles do not generate events; therefore, they have little impact on the tracking process.

4.5 Resolving Occlusions. In this section, we compare different ambiguity-resolving strategies for multiple object tracking. The importance of the concept of dynamics brought by the event-based processing framework and the importance of timing will be studied. The spatial and temporal constraints presented in section 3.3 are examined with the same traffic sequence as above.

A van and a car (as shown in Figure 15) moving in the same direction but with different speeds are tracked simultaneously. During a specified period, the van and the car are superimposed; then the van overtakes the car. Objects other than the van or car are considered as background noise. (Its processing method will be shown in section 4.6.)

In Figure 14a, the different distances between the center of gravity of the two vehicles are shown. Ambiguities arise when these curves have similar values, meaning that both vehicles are spatially overlapping (at time 2.2 s to 2.9 s). It is clear that in that case, the use of spatial information is not sufficient unless the size of the common region is very small. Figure 14b shows the mean event rates of each vehicle. The results prove that both vehicles remain separable during all the sequence, based on event rates of around 3000 events per second for the car and 5000 events per second for the van. Clearly, time and space provide complementary information, and the temporal domain data constitutes a valuable supplement to classic spatial-only methods.

Figures 15a to 15f plot the estimated model speeds (solid lines) of the van (blue) and the car (red) and the ground truth speeds (dashed lines). The ground truth is obtained manually by identifying the corresponding points on successive images and then estimating the speed according to the time differences. The mean speeds for the two automobiles are 42.0 and 24.3 pixels per second, respectively. The image patches in Figure 15 are selected to present the most interesting moment to study the differences between

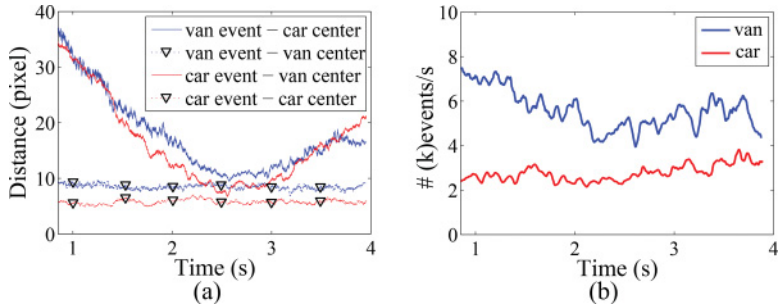

Figure 14: (a) The distances between events and the center of gravity of the two vehicles. The distances are encoded as follows: van event–car center, solid blue; van event–van center, dashed blue; car event–van center, solid red; and car event–car center, dashed red. Ambiguity arises when these curves have similar values. (b) The mean event rates alone allow us to distinguish the two tracked targets in this experiment because the rates are different enough over time. This is, however, too idealistic a case and should not be seen as a sufficient technique for more general situations where multiple tracked targets can have closely similar mean event rates. The temporal constraint must be used in combination with additional constraints such as the spatial one.

each method. The beginning of each curve provides similar results because no occlusion has happened yet. Figure 15, frame a1, shows the beginning when the detection results are equally good for all.

The experimental results of different strategies are analyzed separately:

1) *Assign to closest*. No special processing is performed in this case. In Figure 15, frame a2, events that should belong to the van mistakenly match the car model. It can be observed that the car model moves toward the van's location, and the van model is consequently lost. Different sources of noise may contribute to this failure, with the most important being the model construction noise as the model shape does not exactly match the event contour and the contour varies during time. In Figure 15, frame a3, the car finally returns to the correct position but the van is lost from tracking.
2) *Reject all*. Throws away all the ambiguous events. This method is shown to be unsuccessful. In Figure 15, frame b1, the van starts to overtake the car. In frame b2, the car is lost from its original location and then, in frame b3, is attracted by another black car with the same shape right behind. During this process, the estimated speed of the van is close to the ground truth, although oscillations appear. The speed of the car, however, falls far below the ground truth, because when all the ambiguous events are rejected, the dynamics introduced by the events are not used for computation. Since the van is much

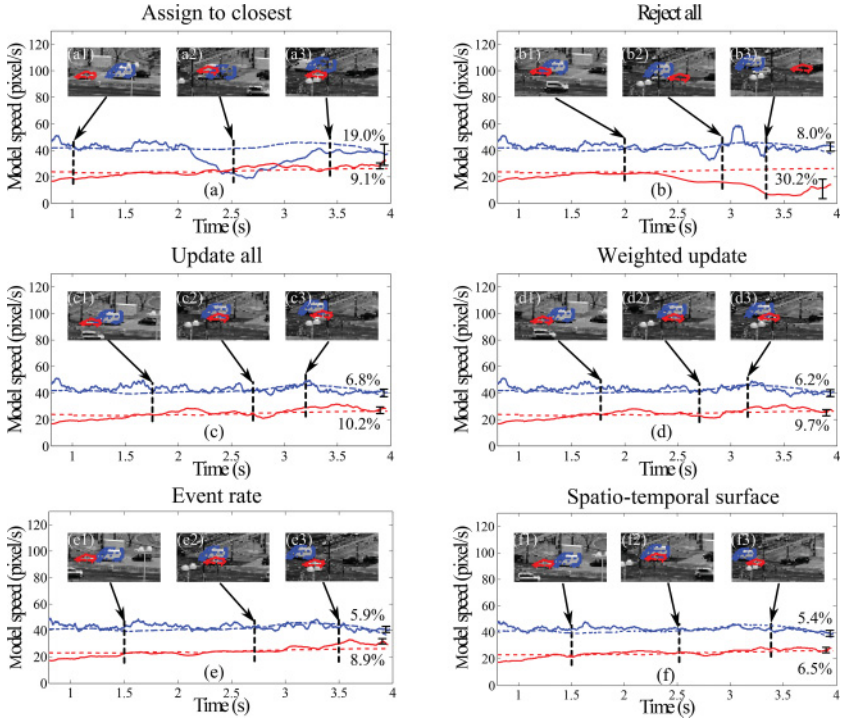

Figure 15: A van and a car are tracked simultaneously. The estimated model speeds (solid lines) of the van (blue) and the car (red) and the ground truth speeds (dashed lines) using four different spatial strategies are depicted in panels a to d. The solution using temporal information is shown in panel e, and the spatiotemporal combination is given in panel f. The black bars at the end of each curve show the mean speed errors with respect to the ground truth.

larger than the car, only a small fraction of its events is rejected. It can then still be tracked efficiently.

3) *Update all*. Updating all the possible models around the ambiguous event shows better results. Both vehicles are tracked and no shape is lost (see Figure 15, frames c1 and c2). At the end of the sequence, the events generated by the van attract the car model, so that the resulting car trajectory deviates slightly toward the van's position and the estimated van speed becomes lower as the car speed becomes higher (see Figure 15, frame c3).
4) *Weighted update*. The dynamics introduced by the ambiguous events are distributed to all the colliding objects with weights adjusted according to the distances. The van and the car are tracked without loss, and the speed curves better match the ground truth (see Figure 15,

frames d1 and d2). However, differences between the estimated car speed and the ground truth are still present at the end of the sequence (see Figure 15, frame d3). The reason is the same as in the "update all" strategy. However, the error is slightly lower than the "update all" strategy. This is the preferred strategy among the spatial solutions when multiple objects cross or collide.

5) *Temporal constraint*. Using time as the only element to segment objects according to equation 3.13 is a strategy that can give acceptable results only in some constrained cases. As expected, it cannot produce reliable results when targets have similar event rates. Even in the cases where the targets have different events rates, the temporal constraint alone should be used with caution. Figure 15 shows reasonable tracking results, and the time constraint achieves correct targets disambiguation. However, this method cannot guarantee tracking stability. This technique, as shown in the results, can drift away from the tracked object, as seen for the car at time 3.5 s, where the error starts to increase. Ignoring the spatial dimension shows its limits. Due to the time pattern contained in the event rates, if an event is assigned to the incorrect target, this will imply irreversible errors in the computation of interevent timings. The temporal constraint provides valuable information for segmenting the tracked targets, but it should be used in combination with the spatial constraint.
6) *Spatiotemporal constraint*. Planes are fitted on the set of events occurring inside a time window for which constant speeds of tracked cars are assumed (3 s in this case). Figure 15, frames f1 to f3, are image patches where correct trackings are achieved. The average errors shown in Figure 15 are 5.4% for the van and 6.5% for the car. This solution combining both space and time constraints provides the best disambiguation between both vehicles as the size of the small object is not affected by the larger one as in the previous cases. Figure 16 shows the events generated by the van and the car in the spatiotemporal space. Two planes, P^1 (Van) and P^2 (Car), with their respective normals $\mathbf{n}_1$ and $\mathbf{n}_2$, are fitted to the events. The spatiotemporal event-to-plane distance adds more constraints, supporting the correct assignment of the event to the tracked objects, especially in case of occlusions. In cases where there are no occlusions, the use of the spatiotemporal constraint allows decreasing the tracking errors (see Figure 15 f). This is due to the fact that local spatiotemporal planes constrain strongly the neighborhood by discarding possible wrong assignments and more robustness to noise. It is interesting to note that errors are more equally shared between the van and the car as a result of the more reliable event attribution.

The exclusive use of spatial data ignoring time logically implies less available information. The dynamic content of visual information is then

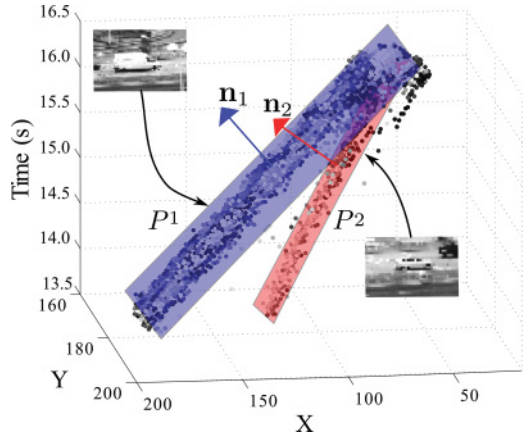

Figure 16: The events generated by the van and the car during 3 s are fitted into two planes, denoted as P^1 in blue and P^2 in red. $\mathbf{n}_1$ and $\mathbf{n}_2$ are the surface normals.

lost. On the other hand, time alone is not sufficient to overcome all ambiguities. The events rate provides incomplete information on the scene. The combination of both space and time then likely leads to more accurate results. A high temporal resolution combined with spatial information allows us to determine when and where things happen. More important, it allows us to define an accurate local space-time subset that necessarily contains the events generated by a target. This valuable geometric constraint is valid on a short temporal interval depending on the velocities of targets.

In principle, the determination of local spatiotemporal subsets using images is possible, but their low temporal resolution will provide less accurate results in local plane fitting, thus producing less accurate estimations. Increasing the frame rate would overcome this limitation but at the cost of losing low computation processing and subsequently at the cost of not being able to reach real-time requirement.

4.6 Pattern-Based Tracking with Camera Motion. When the camera is in motion, events are also generated on the background. One gets computationally free edge detection on both the moving object and the moving background. Figure 17 shows the typical distance distribution of object events to the plane and the distance distribution of background events to the plane. As one can see, the object events' distances satisfy a much narrower distribution than the background events. Therefore, this shows that the mean plane is a sufficiently discriminating criterion to separate tracked object events from the nontracked ones, including those from the

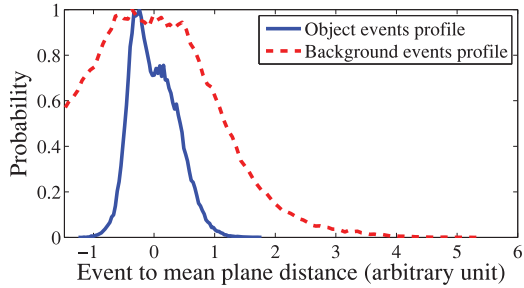

Figure 17: The distribution of distances of the object events to the plane (plain curve) and the distribution of distances of the background events to the plane (dashed curve). Object events are satisfying a much narrower distribution than the background events' one. This can be used to filter out the background events and prevent them from updating the transformation parameters.

background. All events not consistent with the tracked object or equivalently "too far" from the plane are simply discarded.

Two sequences have been tested using the full event-based approach combining both space and time. A "star" target is moved in an indoor environment, while the vision sensor is hand-held and moved simultaneously. Figure 18, frames a1 to a5, shows several snapshots of the scene, with tracking results shown in green. To compare results, the estimated velocity of the star is calculated and shown in Figure 18a. Note that this velocity combines the star's motion and the camera motion. The red dot line represents the ground truth. It is obtained by manually tracking specific portions of the star in each generated event accumulation frame, while the blue curve shows the velocity estimated using the event-based tracking approach. The average error between the tracked and the ground truth data is 6%. Although the scene is filled with complex background edges, the tracker remains robust. The second sequence is an outdoor traffic scene in which a car is tracked by a handheld moving asynchronous vision sensor. Figure 18b shows the velocity estimation results. Although difficult situations are present (e.g., when sidewalks or crosswalks are close to the car or even superimpose with it), the spatial temporal property of the algorithm is capable of solving such ambiguous cases. The average velocity estimation error is close to 15%. This average error is slightly higher because the algorithm is used outside its boundaries. The tracking quality starts to degrade around 3.5 s (see Figure 18, frame b5) when the car starts to have strong perspective changes. This problem will be dealt with in our future work, in which the algorithm should infer 3D information from the scene and from 3D models to compensate for perspective distortions. In summary, this additional experiment shows that the event-based camera, combined with an adequate

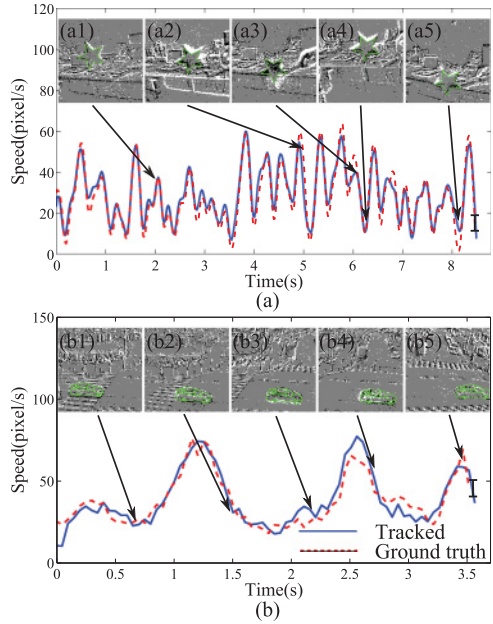

Figure 18: (a) The target, a star, is being observed by a moving asynchronous vision sensor. Insets are event maps showing snapshots of the scene: in black and white, the events according to their polarity, and in green, the target shape and its evolution over time. The estimated focal plane velocity of the tracked shape is shown (solid curve) with the ground truth (dotted). (b) A car is followed by a moving asynchronous vision sensor. The curves show the tracked (solid) and the ground truth (dotted) speed of the car. (A supplemental video is provided at https://drive.google.com/file/d/0B5gzfP0R1VEFdHRUUFFrWTJTSXM/view?usp=sharing.)

spatiotemporal algorithm, is able to achieve robust pattern-based tracking even if the camera and the tracked objects are moving.

5 Discussion and Conclusions

This letter presents a novel methodology to perform pattern-based tracking. It shows that pattern tracking can be performed in real time at a lower cost without the need of the usual trade-offs involved in high frame-rate tracking. Sampling the *x-y-t* space-time volume of pixel data is shown to be an efficient input for pattern tracking. We have shown that high temporal accuracy offers a new alternative to determine objects' velocities that are contained in a plane defined within the *x-y-t* volume of visual information. The sparse acquisition of visual data reduces the redundant

information and retains relevant information of the scene's dynamics. This allows the reduction of computational load, as shown by experiments. The event-based computation we have presented captures the asynchronous and time-continuous computation presented in biological nervous systems and allows shape tracking at an equivalent frame rate of 200 kHz. The process is data driven and iterative and it provides a natural robustness to occlusions.

The asynchronous method is generic enough to track robustly objects even if the camera itself is moving. This implies having prior knowledge of the shapes of the objects to track. The combination of time and space information provides an efficient constraint to disambiguate moving targets. This allows us to discard events that are not consistent with the targets in shape and in motion. This strategy is systematic; it can be applied to all events, from the background and from the targets. This work has produced one of the first applications showing the viability of using an event-based camera for machine vision in cluttered environments, with the camera itself moving in the scene. We believe that this work will open a new route for frame-free machine vision, allowing fast and low-cost computations that are two essential properties of several robotics tasks.

Acknowledgments

We are grateful to both the CapoCaccia Cognitive Neuromorphic Engineering Workshop and the NSF Telluride Neuromorphic Cognition workshops. This work was supported by LABEX LIFESENSES (ANR-10-LABX-65 and ANR-11-IDEX-0004-02)

References

Abd El Munim, H., & Farag, A. (2007). Curve/surface representation and evolution using vector level sets with application to the shape-based segmentation problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(6), 945–958. doi:10.1109/TPAMI.2007.1100

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(4), 509–522. doi:10.1109/34.993558

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. doi:10.1109/TNNLS.2013.2273537

Benosman, R., Ieng, S., Clercq, C., Bartolozzi, C., & Srinivasan, M. (2012). Asynchronous frameless event-based optical flow. *Neural Networks*, *27*, 32–37.

Benosman, R., Ieng, S., Posch, C., & Rogister, P. (2011). Asynchronous event-based Hebbian epipolar geometry. *IEEE Transactions on Neural Networks*, *22*, 1723–1734.

Besl, P., & McKay, H. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(2), 239–256. doi:10.1109/34.121791

Bookstein, F. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(*6*), 567–585. doi:10.1109/34.24792

Challis, J. H. (1995). A procedure for determining rigid body transformation parameters. *Journal of Biomechanics*, *28*(6), 733–737.

Clady, X., Clercq, C., Ieng, S. H., Houseini, F., Randazzo, M., Natale, L., . . . Benosman, R. B. (2014). Asynchronous visual event-based time-to-contact. *Frontiers in Neuroscience, 8*(*9*).

Cremers, D., Osher, S. J., & Soatto, S. (2006). Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *Int. J. Comput. Vision, 69*(*3*), 335–351. doi:10.1007/s11263-006-7533-5

Dandekar, O., & Shekhar, R. (2007). FPGA-accelerated deformable image registration for improved target-delineation during CT-guided interventions. *IEEE Transactions on Biomedical Circuits and Systems*, *1*(2), 116–127.

Delbruck, T., Linares-Barranco, B., Culurciello, E., & Posch, C. (2010). Activity-driven, event-based vision sensors. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (pp. 2426–2429). Piscataway, NJ: IEEE.

Du, S., Zheng, N., Ying, S., & Liu, J. (2010). Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters*, *31*(9), 791–799.

Fornefett, M., Rohr, K., & Stiehl, H. (1999). Elastic registration of medical images using radial basis functions with compact support. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition* (vol. *1*, pp. 1402–1409). Piscataway, NJ: IEEE.

Hersch, M., Billard, A., & Bergmann, S. (2012). Iterative estimation of rigid-body transformations. *Journal of Mathematical Imaging and Vision*, *43*, 1–9.

Huang, X., Paragios, N., & Metaxas, D. N. (2006). Shape registration in implicit spaces using information theory and free form deformations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *28*, 1303–1318.

Jost, T., & Hugli, H. (2003). A multi-resolution ICP with heuristic closest point search for fast and robust 3D registration of range images. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling, 2003* (pp. 427–433). Piscataway, NJ: IEEE.

Kim, D., & Kim, D. (2010). A fast ICP algorithm for 3-D human body motion tracking. *IEEE Signal Processing Letters, 17*(*4*), 402–405. doi:10.1109/LSP.2009.2039888

Lenero-Bardallo, J., Serrano-Gotarredona, T., & Linares-Barranco, B. (2011). A 3.6 μs latency asynchronous frame-free event-driven dynamic-vision-sensor. *IEEE Journal of Solid-State Circuits, 46*(*6*), 1443–1455. doi:10.1109/JSSC.2011.2118490

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.

Ni, Z., Pacoret, C., Benosman, R., Ieng, S., & Régnier, S. (2011). Asynchronous event based high speed vision for micro-particles tracking. *Journal of Microscopy*, *245*, 236–244.

Perez-Carrasco, J., Serrano, C., Acha, B., Serrano-Gotarredona, T., & Linares-Barranco, B. (2008). Event based vision sensing and processing. In *Proceedings of the 15th IEEE International Conference on Image Processing* (pp. 1392–1395). Piscataway, NJ: IEEE. doi:10.1109/ICIP.2008.4712024

Posch, C. (2012). Bio-inspired vision. In *Proceedings of the Topical Workshop on Electronics for Particle Physics*. Philadelphia: IOP Publishing.

Posch, C., Matolin, D., & Wohlgenannt, R. (2008). An asynchronous time-based image sensor. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (pp. 2130–2132). Piscataway, NJ: IEEE.

Posch, C., Matolin, D., & Wohlgenannt, R. (2011). A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits, 46*(*1*), 259–275. doi:10.1109/JSSC.2010.2085952

Rueckert, D., Sonoda, L., Hayes, C., Hill, D., Leach, M., & Hawkes, D. (1999). Non-rigid registration using free-form deformations: Application to breast MR images. *IEEE Transactions on Medical Imaging*, *8*, 712–721.

Rusinkiewicz, S., & Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling*. Piscataway, NJ: IEEE.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gomez-Rodriguez, F., . . . Linares-Barranco, B. (2009). Caviar: A 45K neuron, 5M synapse, 12G connects/s AER hardware sensory system for high-speed visual object recognition and tracking. *IEEE Transactions on Neural Networks, 20*(*9*), 1417–1438. doi:10.1109/TNN.2009.2023653

Yang, J. (2011). The thin plate spline robust point matching (tps-rpm) algorithm: A revisit. *Pattern Recognition Letters, 32*(*7*), 910–918. doi:10.1016/j.patrec.2011.01.015

Zheng, Y., & Doermann, D. (2006). Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*, 643–649.

Received December 12, 2013; accepted July 28, 2014.