Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor

Timo Stoffregen[*†], **Lindsay Kleeman**[*]
[*]Dept. of Electrical and Computer Systems Engineering
Monash University, Australia
[†]Australian Centre of Excellence for Robotic Vision
timo.stoffregen@monash.edu

Abstract

We present an algorithm (SOFAS) to estimate the optical flow of events generated by a dynamic vision sensor (DVS). Where traditional cameras produce frames at a fixed rate, DVSs produce asynchronous events in response to intensity changes with a high temporal resolution. Our algorithm uses the fact that events are generated by edges in the scene to not only estimate the optical flow but also to simultaneously segment the image into objects which are travelling at the same velocity. This way it is able to avoid the aperture problem which affects other implementations such as Lucas-Kanade. Finally, we show that SOFAS produces more accurate results than traditional optic flow algorithms.

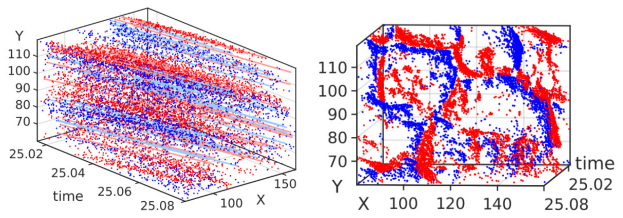
1 Introduction

Dynamic Vision Sensors (DVS), also known as Event Cameras or Neuromorphic Cameras avail robots of an entirely new class of visual information. Where traditional cameras produce image frames at a fixed rate, the pixels of a DVS produce events asynchronously in response to illumination intensity changes, analogously to the way a biological retina generates neural spikes in response to intensity change [Lichtsteiner *et al.*, 2008]. The data generated by such a sensor is inherently sparse, since elements of the scene which do not change much in the image such as backgrounds produce little or no data. At the same time elements in the scene which do change are able to receive corresponding "framerates" in the hundreds of kilohertz [Ni, 2013; Delbruck and Lang, 2013; Delbruck *et al.*, 2015] . This vastly reduces the need for computational power in many classic computer vision applications [Brandli *et al.*, 2014].

In computer vision, optic flow has long been a key task in object tracking, image registration, visual odometry, SLAM and other robot navigation techniques [Aires *et al.*, 2008]. Computing optic flow is not a trivial task and has been the theme of many research papers in computer vision. While many techniques for optic flow detection exist, the most common implementations are still the original Horn-Schunck [Horn and Schunck, 1980] and Lucas-Kanade [Lucas and Kanade, 1981] algorithms. However, since most optic flow estimations are based on local computations on a small patch of the image they tend to suffer from the aperture problem [Binder *et al.*, 2009]. This states that the motion of one dimensional structures in the image cannot be unambiguously determined if the ends of the stimulus are not visible. Our algorithm avoids the aperture problem by examining the entirety of the structure rather than performing a local estimate.

While the suppression of redundancy and high temporal resolution of these data-driven, frameless sensors has presented the possibility of performing demanding visual processing in real time, it also requires a rethink in how the data is processed, since the majority of computer vision algorithms cannot be directly used on event streams. Simply translating existing algorithms to the event-based paradigm often misses the opportunities presented by DVSs. Further, many important recent algorithms accumulate past events to detect image gradients [Ni *et al.*, 2012; Benosman *et al.*, 2012; 2014; Kim *et al.*, 2014]. While these algorithms allow events to asynchronously alter the current state, by buffering they lose much of the advantage of the high resolution timestamps.

Our contribution is to introduce a novel way of computing the optic flow of events produced by a DVS, which exploits the properties of event based data. Our approach is demonstrated to produce more accurate results than traditional optic flow techniques on such data and is able to adapt to evolving velocities and form changes of the underlying structures. Further, our approach is able to segment events by the structures generating them, in the process avoiding the aperture problem entirely.



(a) Events (dots) and the trajectories that they follow

(b) Events visualized along the trajectories

Figure 1: Visualization of the events (positive (blue dots) and negative (red dots)) in the image plane vs. time (50 ms) [Gallego and Scaramuzza, 2017]

2 Related Work

Optical flow using event based sensors is a topic that has been visited frequently, resulting in many different algorithms. A thorough review and evaluation of these was performed in [Rueckauer and Delbruck, 2016]. Notably all of the dense flow algorithms require the computation of the first derivatives of some image quantity, be it gradients of the luminance intensities or gradients of the timestamps of buffered events. Our method does not require the use of any derivatives.

In [Gallego and Scaramuzza, 2017] the angular velocity of an event camera was estimated based on contrast maximisation by transforming events along their trajectories on the time-image plane space. Essentially, the authors realised that if the events were viewed along their trajectories, they would produce a crisp image in which the polarities of lined-up events would sum to maximise the contrast (Fig 1). That paper focused on estimating angular velocity by this method under the assumption that during a camera rotation the majority of events would conform to trajectories represented by one rotation transformation. Our work follows a similar thought, except that instead of estimating camera angular velocity we seek to estimate image plane velocities. The difficulty here, of course, is that we cannot assume that all event trajectories will conform to one particular transformation.

Finally, we would like to mention the work of Mueggler *et al.* [Mueggler *et al.*, 2015], in which a solution to the event buffering for the generation of image gradients was proposed. In that work the optical flow, calculated by locally fitting a plane to the *Surface of Active Events*, was used to estimate the lifetime of events. A small part of our work involves estimating the lifetime of events via the current optical flow velocity estimate, whereby we adopt a similar approach to that proposed by Mueggler *et al.*

3 Methodology

3.1 Structure Detection

Events generated by a DVS are represented by a vector and a sign $e = (u, v, t)^T, s$ in which u, v represent the event's position on the image plane, t represents the event timestamp and s the event polarity (+1 for an increase in intensity, -1 for a decrease). Suppose an object generates events on it's visual structures, such as internal and external edges as it moves. Let the set of the x, y coordinates of the events generated by the object's progression on the image plane by one pixel be called the 2D cross-section C (Fig. 7). Supposing the object moves at a velocity $\vec{v} = (\vec{v}_u, \vec{v}_v)^T$ across the image plane. Then at time t an event $e_t = (u_t, v_t, t_t)^T$ can be generated where

$$u_t = u_c + \vec{v}_u t \tag{1}$$

$$v_t = v_c + \vec{v}_v t \tag{2}$$

$$u_c, v_c \in C \tag{3}$$

This set of events forms a point cloud which resembles an extrusion of the original contour C. Thus this set of events in 3D pixel/time space will be referred to as the Event Contour Extrusion (ECE) or $E = \{e_0, e_1 ... e_k\}$ (Fig. 2). The principle axis of this extrusion runs parallel to the parametrically defined line l

$$u = \vec{v}_u t \tag{4}$$

$$v = \vec{v}_u t \tag{5}$$

$$t = t \tag{6}$$

Therefore $l = (u, v, t)^T = t(\vec{v}_u, \vec{v}_v, 1)^T = t\vec{n}$. By projecting the events in E along $t\vec{n}$ the time dimension can thus be collapsed for the ECE generated by a structure moving with constant velocity with respect to the DVS, revealing the original contour C. This projection is defined as

$$proj_{\vec{n}}(e) = (u - \vec{v}_u t, v - \vec{v}_v t, 0) \tag{7}$$

$$= \begin{bmatrix} 1 & 0 & -\vec{v}_u \\ 0 & 1 & -\vec{v}_v \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ t \end{bmatrix} \tag{8}$$

If the events are now projected onto a horizontal plane spanning u, v using $proj_{\vec{n}}(e)$, events generated by the same feature of the object will project onto the same parts of the plane. Summing these accumulated events gives us

$$f(x, y) = \sum_{i=0}^{k} \begin{cases} 0 & proj_{\vec{n}}(e_k) \neq \vec{q} \\ s_k & proj_{\vec{n}}(e_k) = \vec{q} \end{cases} \tag{9}$$

for every location $\vec{q} = (x, y)^T$ on the plane. Summing the squares of these sums gives us a metric

$$m = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f(i, j)^2 \tag{10}$$

If one now considers the set of n projections $F_p = \{f_{p1}...f_{pn}\}$ slightly perturbed relative to the true projection f_{p0}, it becomes apparent that the corresponding metric $m_{p0}...m_{pn}$ will change as the accumulated sums change. Further, as k grows large, the metric of the most representative projection m_0 will become greater than that of any of the other projections $m_{p1}...m_{pn}$. This is obvious, since any projection which is inclined to the principle axis will eventually begin to have the projected events drift away, in the equivalent phenomenon of motion blur [Gallego and Scaramuzza, 2017], while the true projection will continue to gather accumulations. Thus, finding the projection with the maximum value of m given sufficient events is equivalent to finding the optic flow of a given structure moving across the image plane (Fig. 3).

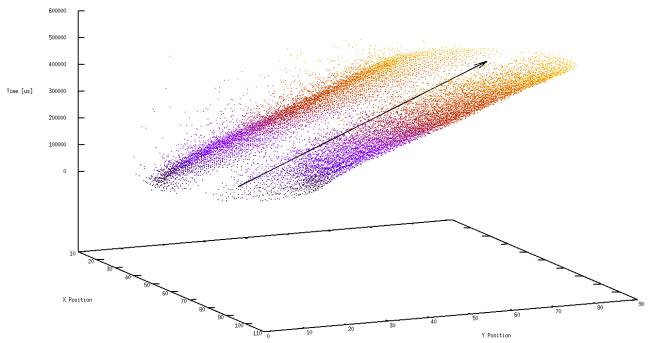
In fact, this property remains even with multiple structures with different velocities. In practice, it requires more events for the best projections to become apparent, however given a sufficient number of events the number of local maxima in the set of projections will match the number of structures with a distinct optic flow velocity and those maxima will correspond to the best approximations to the optic flow of each structure respectively. This can be seen clearly in Fig. 4.

3.2 Structure Tracking

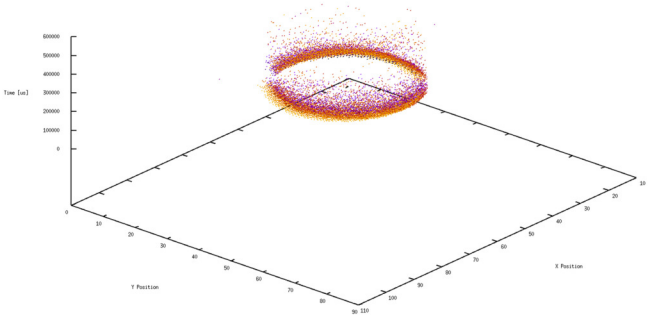
Clearly, simple projections of events as defined above can only work under the assumption that the structures generating events are travelling with a constant velocity. A more complicated motion model that allows for acceleration could be developed, by adding a further dimension to the projections. Further it is possible for the contours of structures in the scene to change, rendering previous accumulations of events a liability. Therefore this concept of projection metrics is only used in the initialisation stage of our algorithm. Since both the velocity and the contours of structures generating events can be assumed constant over short periods of time, we introduce an iteratively improving tracking mechanism. The method described in 3.1 provides a good estimate of the initial flow vector as well as an outline of the edges that generated the events. This outline together with the corresponding projection vector is then used to predict the location of incoming events. The accuracy of this prediction is then used to both update the inclination of the projection as well as the outline of the structure.

4 Algorithm and Implementation

The algorithm for optical flow detection is split into two separate modules (Fig. 5) as described in section 3. As events are added they are first matched against existing structures or *Track Planes* in the Track Plane stack. If matching fails, it is assumed that this event belongs to



(a) Events form a cylinder in space and time. The principle axis is shown as a black arrow



(b) Events as viewed along the principle axis or along visual flow vector

Figure 2: Visualisation of the events generated by a rolling ball (coloured by timestamp

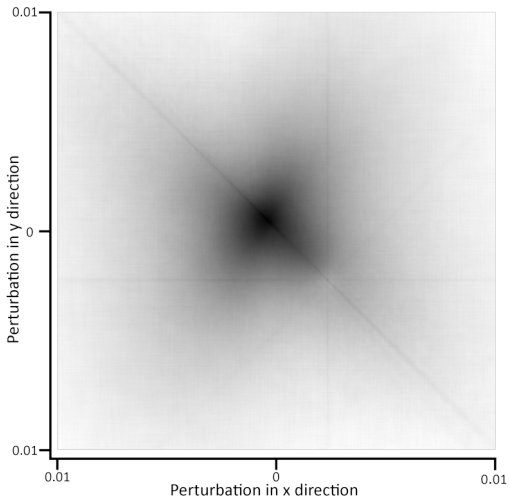


Figure 3: A map of the metrics m computed across a 400x400 array of projections, whose normal $\vec{n}$ is perturbed in the x axis and y axis respectively. The events projected are from the data in Fig. 2. Darker colors indicate a higher metric, the lines visible are due to discretisation of events. The darkest spot represents the closest projection to the true flow vector

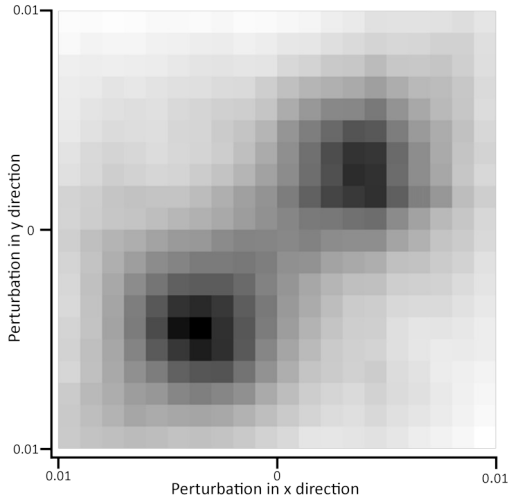


Figure 4: A map of the metrics m computed across a 20x20 array of projections, whose normal $\vec{n}$ is perturbed in the x axis and y axis respectively. The events projected were generated by two structures with opposite velocities moving across the image plane - this is clearly visible in the two maxima. Darker colors indicate a higher metric. The darkest spots represent the closest projections to the true flow vectors

a previously undiscovered structure moving across the image plane and is added to the *Flow Plane* module, which seeks to discover structures with a particular flow velocity in the scene and generate new Track Planes accordingly. When an event is allocated a Track Plane, it is allocated the corresponding flow vector estimate of that Track Plane.

In some cases, the same structure will be segmented into several distinct Track Planes (for example in the case of an object entering the frame). Thus the algorithm also has a mechanism to merge several Track Planes belonging to the same structure. Track Planes that receive less than $n\%$ of the events expected (based on their velocity and the number of accumulator cells) are removed from the stack.

4.1 Flow Plane

The Flow Plane module consists of an array ($M(x, y)$) of nxn flow projections. Each of these is perturbed evenly by some factor, such that the angle between corner planes is some value r. (While this value is arbitrary, we typically perturb the planes such that $r = \pi$. Since any projection that is tilted at angle $\frac{\pi}{2}$ from $(0, 0, 1)^T$ has infinite velocity, this covers the full range of possible velocities). Incoming events are now projected to each flow plane, where accumulating values are summed, and each such sum is squared and summed, forming the metric space $M(x, y) = m, 0 \leq x, y \leq n$. The

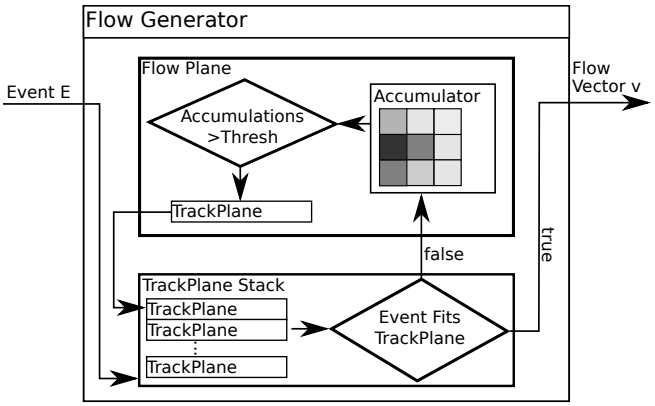


Figure 5: Structure of the algorithm. Incoming events are matched to Track Planes and failing a match added to the initialisation or Flow Plane module. On being allocated to a plane they are given the corresponding flow vector.

corresponding metrics thus change value with each incoming event. After each event the global maximum in $m = (x, y)^T, M(x, y) = \max(M)$ is found. In early stages the location of this maximum changes frequently, however as more events are added it converges. Thus, after the maximum has remained constant for p events, it is considered to contain a structure(s) and valid flow vector estimate.

This estimate however, tends to be poor, especially if the value of n is low. Therefore, in the next stage, the events that correspond well to the chosen projection are taken and separately projected by a new nxn array of projections, with a perturbation range of $\frac{r}{q}$ for some factor q (Fig. 6). This has two effects: for one the new estimate is much closer to the true velocity of the underlying structure since the resolution is improved, for another the new estimate is much more able to represent the structure since only events corresponding to the chosen flow projection are chosen for this refining step.

The way the most associated events are chosen is by inspecting $f(x, y) \forall x, y \in \mathbb{Z}$. If $f(x, y) > \mu_f + \sigma_f w$, where μ_f and σ_f are the average value and standard deviation of the values of $f(x, y)$ respectively and w is a thresholding factor, then the events accumulated at that location are taken to be part of the structure located. All events at locations connected to the locations found are also added to the set of associated events E_{assoc} akin to flood-fill.

After several such refining cycles, E_{assoc} and the located projection $\vec{n}$ are bundled into a new Track Plane, which is then added to the Track Plane stack.

The Flow Plane is now cleared and the set of remaining events $E_{\overline{assoc}}$ is projected onto it again. Since this set of events is no longer polluted by the structure de-

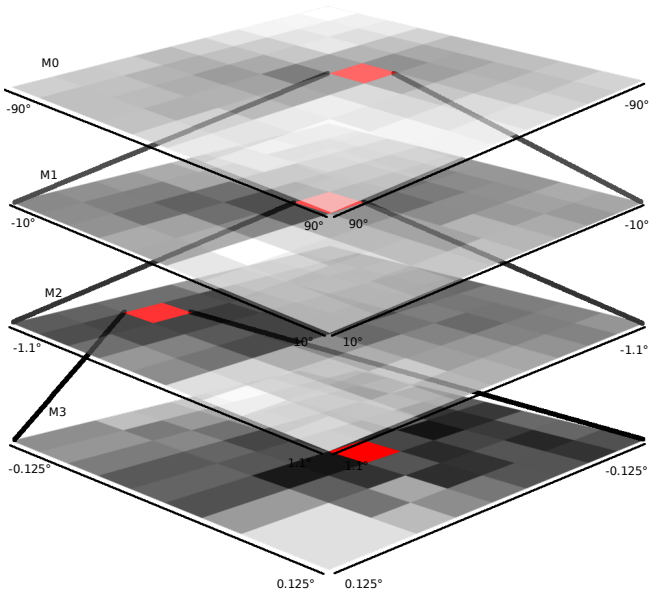


Figure 6: Stack of projection metric arrays $M(x, y)$. The top metric array M_0 has a low resolution (180°), but is expanded into M_1 (20°) when a best projection is found. Darker values indicate a higher value. Global maximum of each M is shown in red.

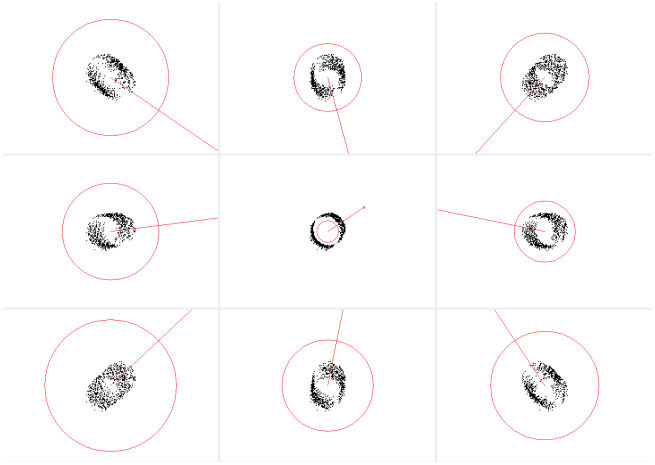


Figure 7: A 3x3 array of projections created with the data from Fig. 2. The centre projection best represents the optic flow, as seen by the crisp outline of the accumulator cells. The surrounding projections are perturbed by $h = 0.02°$ and are worse estimates, as shown by the "motion blur" of viewing the cylinder from a side-on angle. The circles depict a scaled version of the magnitude and the red arrows the unscaled flow vector for each respective projection.

tected in E_{assoc}, it is much more likely to converge on any remaining structures fast. Finally, events which are due to noise and should not be associated to any structure are eventually flushed out of the Flow Plane. This is done with a simple lifespan threshold.

4.2 Track Plane

Each Track Plane consists of m^2 projections arranged in an mxm array (Fig.7). The projection in the centre of this array represents the current best prediction of the optical flow, with the surrounding projection perturbed by a factor h. These all project the current set of associated events and those coordinates (x, y) to which they are transformed become the set of accumulator cells $A = \{(x_0, y_0)^T, (x_1, y_1)^T \cdots (x_n, y_n)^T\}$.

As events enter the algorithm (Fig. 5), they are matched against existing Track Planes. This happens by projecting the given event by the current best estimate. If the projection returns a location $(u, v)^T$ in A the event is considered to be associated with the plane and accordingly added to A. If it misses, it raises the value of the accumulator at (u, v). This way, if the value of the accumulator at (u, v) passes a threshold it can be considered to be a valid accumulator cell and thus the contour can evolve.

A projection in the array is considered to be the best projection if the number of hits surpasses a given percentage of the number of accumulator cells (10% typically works well). When this occurs, the projections are recalculated, centred around the new best estimate and the accumulators are regenerated based on the set of associated events from the new best projections accumulator. If the best projection is the previous best projection, the perturbation factor is halved and conversely if the new best projection was one of the surrounding projections the perturbation factor is doubled. This way the Track Plane is able to iteratively improve the estimate if the current estimate is good and conversely improve the rate of exploration if it is bad (generally because of acceleration).

Clearly, the Track Plane would be highly sensitive to changes in the velocity of the tracked structure if it kept the events in A indefinitely. To this purpose a lifetime for events is defined, after which events are removed from A. Similarly to [Mueggler *et al.*, 2015] this lifetime is calculated by taking the current flow velocity and calculating how long it would take for the structure to cross over p pixels (we have found $p = 3$ to work well). Thus if the structure is moving very slowly or even standing still, the lifetime is very long and vice versa.

5 Results

The data used to validate the algorithm was generated with a DAVIS-240C DVS from Inilabs. The algorithm was written in Java using the jAER toolbox [Delbruck,

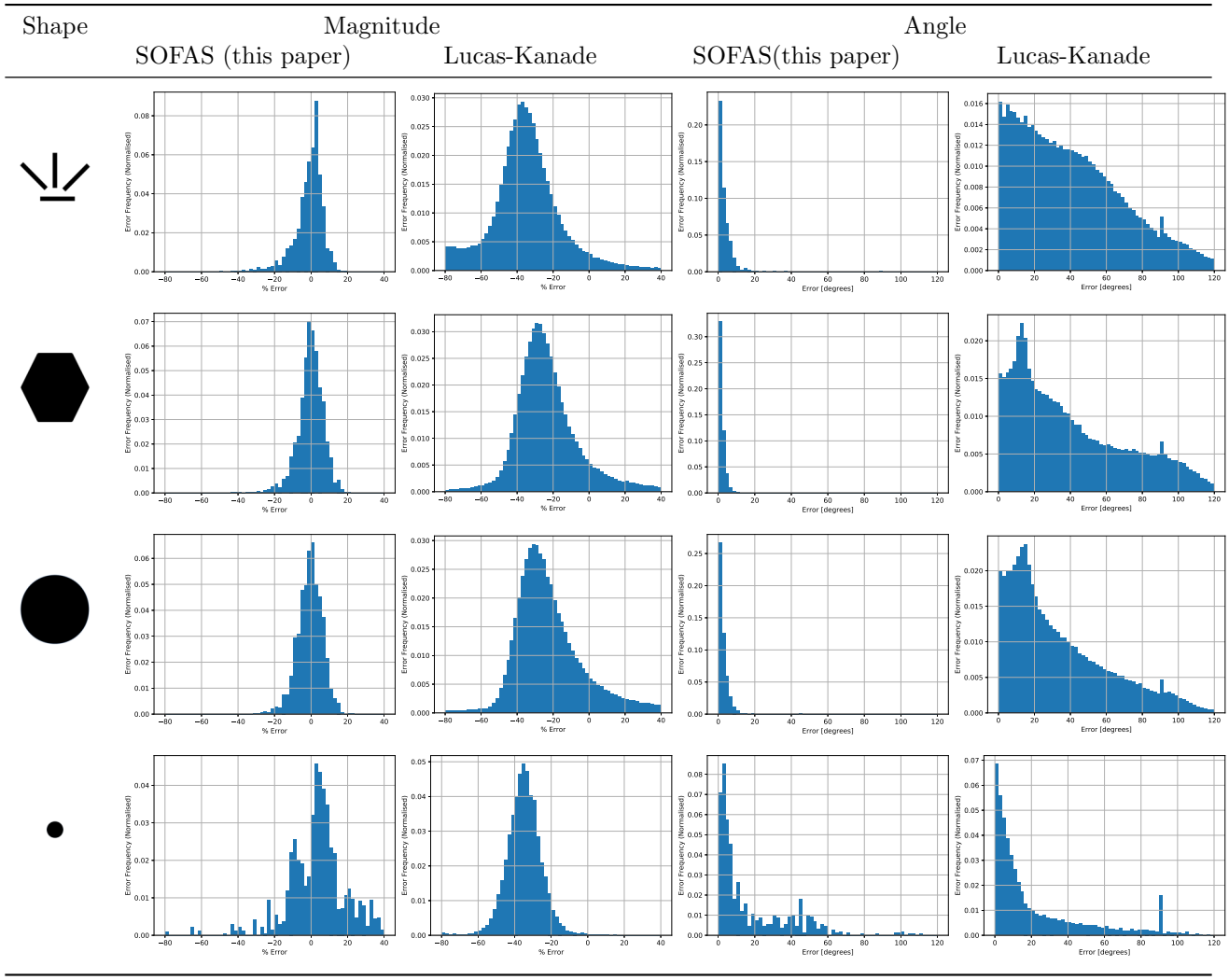


Figure 8: Comparison to Lucas-Kanade via % errors of flow vector magnitudes and errors of flow vector angles. Flow velocity = 58 pix/s

2017] and compared to an event based Lucas-Kanade implemented by Bodo Rueckauer [Rueckauer and Delbruck, 2016] and contained in the jAER toolbox. Since the algorithm performs several tasks, the tests are broken up into several sections. First the accuracy of the flow vectors estimated by algorithm is compared to that of Lucas-Kanade on a ground-truth dataset we generated. We chose to benchmark our algorithm against Lucas-Kanade since it returned the lowest errors of flow vectors of any of the optic flow implementations in the jAER toolbox. Having established the competitiveness of our algorithm in comparison to traditional optical flow estimation algorithms, we then examine the performance of the algorithm at different velocities and with a rapidly accelerating dataset when compared to ground truth. Having tested the accuracy, we then show the rate at which the estimate converges and finally we show qualitative results in more complex scenes for which we were unable to generate ground truth data, but which show the ability of the algorithm to successfully segment the scene into structures with distinct flow velocities.

Ground truth optical flow data was generated by using a UR5/CB3 robot arm [Uni, 2017]. Magnets were mounted on the end effector of the robot, so that the robot was able to move simple structures with magnets attached across a background panel without generating events itself. Since the velocity of the robot end effector and the field of view of the DVS were known, the optical flow in the horizontal direction could be determined with ease (flow vector $\vec{v} = w_c \frac{\vec{v}_r}{w_{fov}}$) where w_c is the width of the camera plane in pixels (240 on the DAVIS 240C), $\vec{v}_r$ is the end effector velocity and w_{fov} is the width of the field of view.

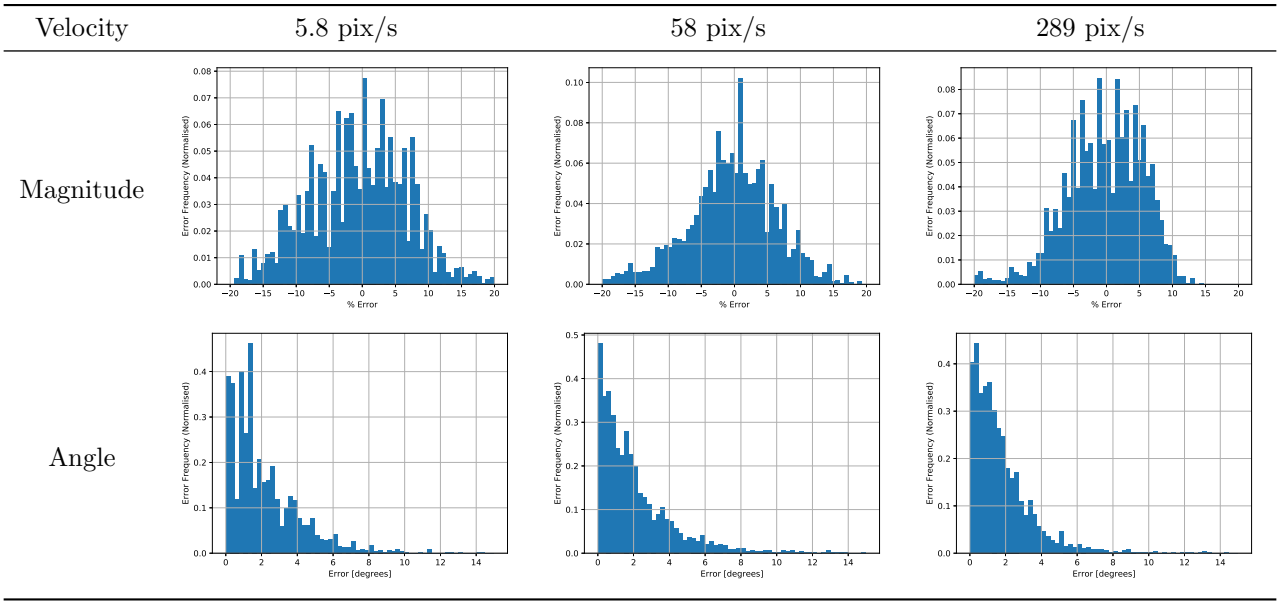
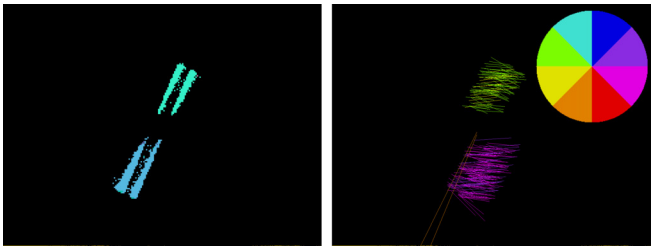


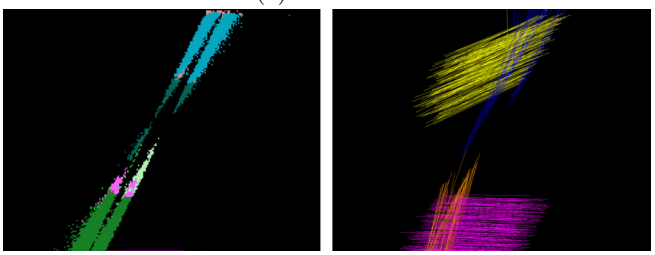
Figure 9: % error of flow vector magnitudes and flow vector angle error at a range of velocities computed using SOFAS

5.1 Accuracy Comparison

Here we contrast the percentage error to ground truth of the flow vector magnitudes and the error of the flow vector directions. The data used four different shapes with a flow velocity of 58 pix/s (a collection of rectangles (width=120 pixels), a hexagon (width=65 pixels), a large circle (width=70 pixels) and a small circle (width=7 pixels)). The results (Tab. 8) show that our algorithm not only had a lower standard deviation of errors, but also was able to avoid the aperture problem. Results from Lukas-Kanade under-estimated the flow magnitude by about 35%. This is because Lucas-Kanade, being a local estimation algorithm (in this case using a 7x7 pixel window), tends to estimate the flow vectors normal to the image gradients rather than in the direction of optical flow. Therefore, our method was also far superior in estimating the direction. The exception was for the small circle, where the methods were roughly even, since most of the object is able to fit into the kernel window in Lucas Kanade and the aperture problem is therefore avoided.



(a) Short bar

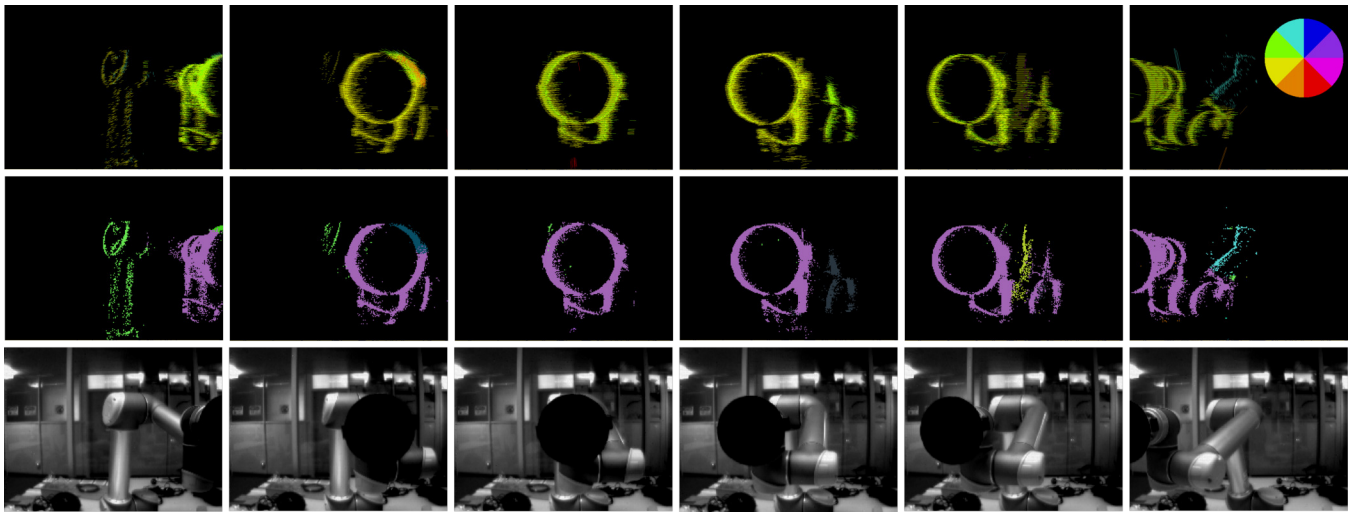


(b) Long bar (filling entire frame)

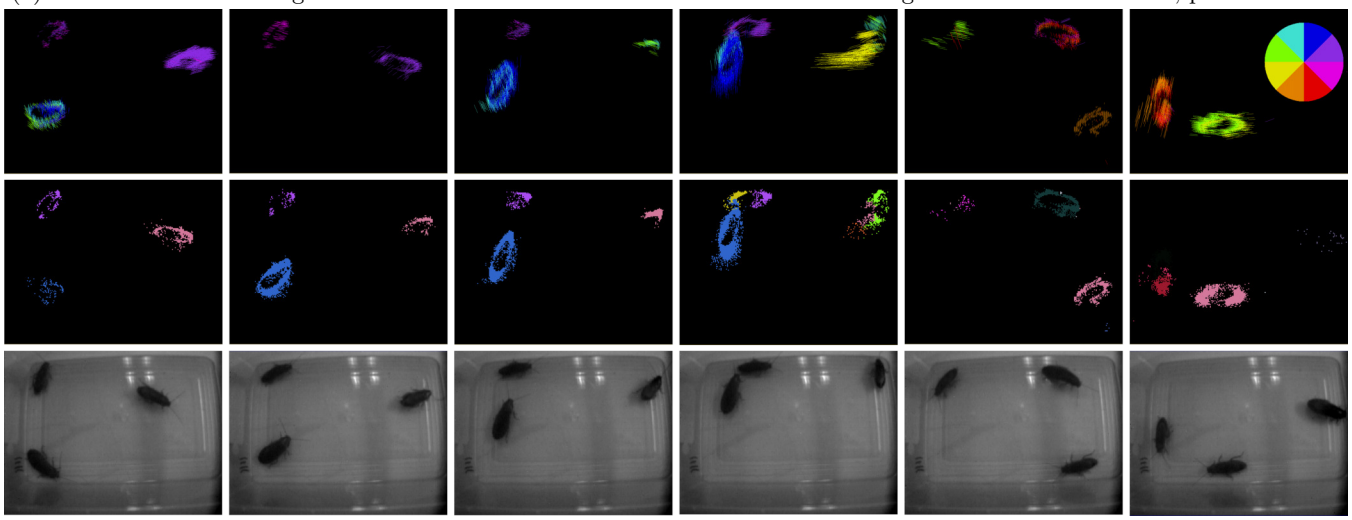
Figure 10: Two bars rotating counterclockwise at 1rps. On the left, the segmentation the algorithm performs, on the right the velocity vectors of the optic flow (see key in top right for disambiguating the direction)

5.2 Performance at Different Velocities

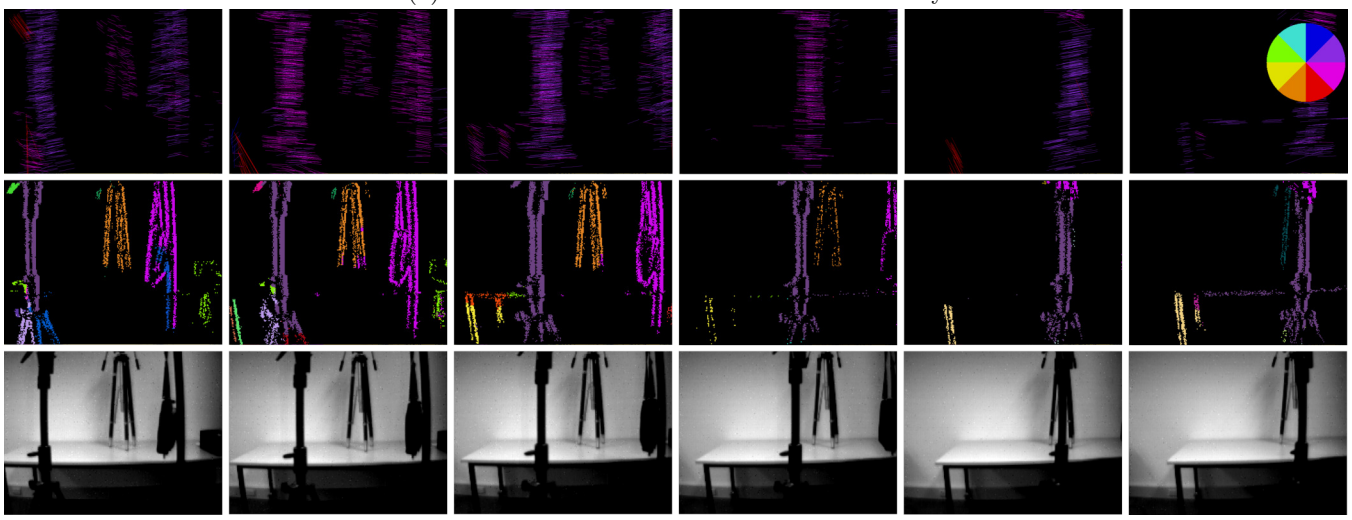
Here we test our algorithm over a range of velocities using the large hexagon dataset. It shows that the algorithm performs slightly better at higher velocities. This is likely to do with the fact that the DVS used tends to generate more events over the same path at a higher velocity.



(a) UR5 robot arm bearing a black cardboard circle moves the end effector from right to left in a horizontal, planar motion



(b) Cockroaches in a box scuttle around aimlessly



(c) The camera pans past objects at different distances from the DVS. The tripod in the background (orange segmentation) is occluded and immediately recognised as a separate structure upon reappearance (green segmentation)

Figure 11: Snapshots of the algorithm. The top row shows the calculated flow vectors (see color wheel for direction), the middle row shows events coloured by segmentation, the bottom row shows grayscale stills of the scene

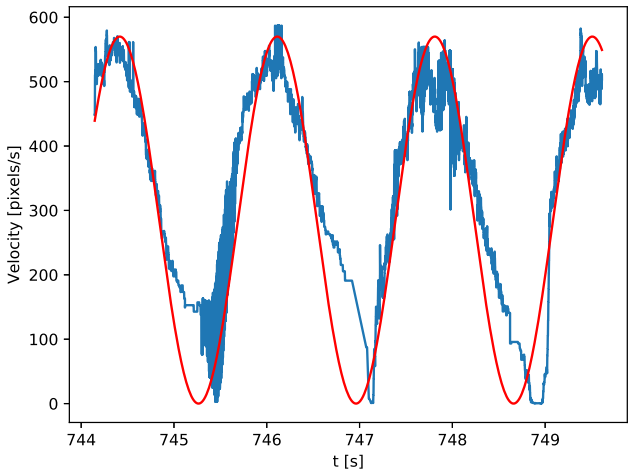


Figure 12: Estimated optic flow magnitude (blue) vs calculated ground truth (red) for the events generated by a pendulum (Length $L = 0.72$m, starting angle $\theta_{max} = 23°$)

5.3 Performance with Acceleration

The ability of the algorithm to deal with acceleration (which is not explicitly modelled in the projections, which assume locally constant velocity) was tested by comparing the computed flow magnitude of a pendulum to the calculated velocity. Since the maximum velocity of a pendulum is given by $v_{max} = \sqrt{2gL(1 - cos(\theta_{max}))}$ and the period by $T = 2\pi\sqrt{\frac{L}{g}}$ where $g = 9.82ms^{-2}$, the pendulum length $L = 0.72m$ and the maximum starting angle of the pendulum $\theta_{max} = 23°$, we were able to fully characterise the velocity and frequency of the pendulum, assuming a lossless sinusoidal motion model. Since the size of the field of view of the camera was also known, we were able to get ground truth for the optic flow magnitude. The phase shift was estimated from the footage. As can be seen in Fig. 12, the algorithm generates a relatively noisy estimate in cases where there are no constant velocities, but where the structures are under constant acceleration. Nevertheless, the algorithm is able to adapt to such circumstances to generate reasonable estimates.

5.4 Rotation

In this experiment, two bars were set to rotate in front of the DVS. Predictably, SOFAS performs poorly in such circumstances, since the projection does not model rotation (Fig. 10). Especially when the rotating bar fills the entire screen, the algorithm suffers somewhat from the aperture problem as indicated by the grossly incorrect flow predictions near the centre (blue and orange vectors, Fig. 10b). Nevertheless, the algorithm does attempt to segment the bar lengthwise and for the short bar at least generates reasonable optical flow. This is because over short enough timeframes it is able to assume the velocity as constant.

5.5 Segmentation

Here we aim to showcase the algorithms ability to segment the scene into structures with distinct velocities. To do this, we show the contours of the Track Planes generated on three datasets; one in which the UR5/CB3 robot arm moves the circle horizontally, one in which three cockroaches are moving randomly in a box and one in which the algorithm pans across a simple scene with four vertical bars at different distances to the DVS. As is clear (Fig. 11), SOFAS is able to clearly segment objects by their flow velocities, and event when the same object is segmented into multiple sections (see frame two of Fig. 11a), it is able to merge these together again. Because of the nature of the algorithm, there is some quantisation to the segmentation, though the extent of this is not easily quantifiable. In particular the result from Fig. 11c is important, since the effective segmentation of structures at different distances makes this algorithm a potential candidate for visual odometry in the future. Indeed, it should be noted how well the segmentation in Fig. 11c performs, separating the objects in the scene perfectly. When the tripod in the background is occluded, SOFAS is able to re-segment it as a structure immediately after reappearance. At the same time in this scene, the edges of the table are segmented into separate entities as they have distinct optic flow velocities.

6 Conclusion and Future Work

This paper presents an algorithm for detecting the optical flow of events generated by a DVS in a novel way that is better suited to the event-paradigm than traditional optical flow techniques. Because of the nature of the algorithm, segmentation by flow velocity comes "free". In the course of this paper, SOFAS has been demonstrated to be robust and to be superior in accuracy when compared to a traditional optical flow algorithm.

We plan in future work to produce an optimised implementation of this algorithm. Since the computations involved are relatively basic and the overall structure of the algorithm (Fig. 5) is modular and inherently parallel, we feel that it should be possible to produce a real-time capable implementation. For the implementation presented here, the hexagon dataset from Fig. 8 required 13,602 ms to process (81,700 events) and the robot arm dataset from Fig. 11 took 46,461 ms to process (257,571 events). The algorithm was run on a PC with a 3.4 GHz CPU.

Clearly, fail cases for this algorithm are those in which most of the structures generating events on the image plane are accelerating or rotating rapidly. This problem

could be solved, or at least improved by adding a dimension to the projection used, so that the acceleration is modelled also, a compelling direction for future work. Nevertheless, the adaptive nature of our algorithm enabled it to give reasonable estimates even under such non-ideal conditions. Other fail cases are likely to occur in situations where fast global lighting changes (such as a flickering light source) cause large numbers of events to be generated. This however, is an intrinsic weakness of DVSs and not a particular failing of our algorithm.

6.1 Acknowledgements

A short video showing the output of this algorithm can be found here: https://youtu.be/JVkQOW_iUqs

This work was supported by the Australian Research Council Centre of Excellence for Robot Vision, project number CE140100016 (www.roboticvision.org).

References

[Aires *et al.*, 2008] Kelson R. T. Aires, Andre M. Santana, and Adelardo A. D. Medeiros. Optical flow using color information: Preliminary results. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1607–1611, New York, NY, USA, 2008. ACM.

[Benosman *et al.*, 2012] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32 – 37, 2012.

[Benosman *et al.*, 2014] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi. Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, Feb 2014.

[Binder *et al.*, 2009] Marc D. Binder, Nobutaka Hirokawa, and Uwe Windhorst, editors. *Aperture Problem*, pages 159–159. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Brandli *et al.*, 2014] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[Delbruck and Lang, 2013] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, 7:223, 2013.

[Delbruck *et al.*, 2015] T. Delbruck, M. Pfeiffer, R. Juston, G. Orchard, E. Muggler, A. Linares-Barranco, and M. W. Tilden. Human vs. computer slot car racing using an event and frame-based DAVIS vision sensor. *Proceedings - IEEE International Symposium on Circuits and Systems*, 2015-July:2409–2412, 2015.

[Delbruck, 2017] T. Delbruck. jaer open source project. [Online], 2017.

[Gallego and Scaramuzza, 2017] G. Gallego and D. Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632–639, April 2017.

[Horn and Schunck, 1980] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. Technical report, Cambridge, MA, USA, 1980.

[Kim *et al.*, 2014] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew Davison. Simultaneous mosaicing and tracking with an event camera. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[Lichtsteiner *et al.*, 2008] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 dB 15us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2), feb 2008.

[Lucas and Kanade, 1981] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[Mueggler *et al.*, 2015] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015.

[Ni *et al.*, 2012] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier. Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Transactions on Robotics*, 28(5):1081–1089, Oct 2012.

[Ni, 2013] Zhenjiang Ni. Asynchronous Event Based Vision : Algorithms and Applications to Microrobotics. 2013.

[Rueckauer and Delbruck, 2016] Bodo Rueckauer and Tobi Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in Neuroscience*, 10:176, 2016.

[Uni, 2017] Universal Robots. *Universal Robots UR5 Technical Details*, 2017.