Table tennis ball spin estimation with an event camera

Supplementary Material

6. Camera settings

The Prophesee EVK4 has five bias settings. The name 'bias' comes from the electronics domain. In electronics, 'biasing' usually refers to a fixed DC voltage or current applied to an electronic component, in order to establish proper operating conditions for the component. However, despite the name, bias settings do not have to directly correlate with any voltage or current but is rather just a numerical value to adjust the property of the event camera. The bias settings of the Prophesee EVK4 are listed in Tab. 3. On all the corresponding plots, the green line indicates our chosen setting.

bias_on = 40	Contrast threshold for triggering on events
bias_off = 40	Contrast threshold for triggering off events
bias_fo = 55	Cutoff frequency for the low-pass filter
bias_hpf = 0	Cutoff frequency for the high-pass filter
bias_refr = 80	Pixels' refractory period

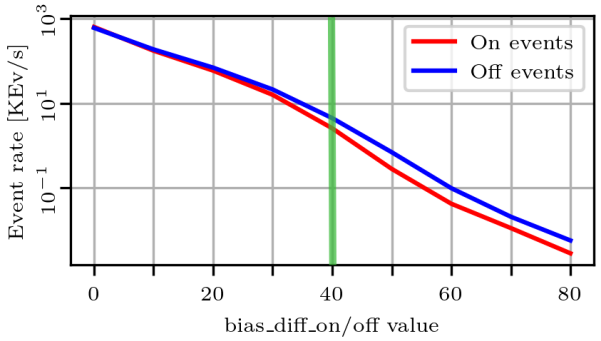
Table 3. List of the biases/settings of the event camera we used. Their tuned values (default values are 0) and what they control (this is the naming convention used for Prophesee cameras)

To make the most out of event cameras, tuning the biases is essential to capture all the necessary events while maintaining a low level of noise. This task is quite involved as the biases are interdependent. Estimating the signal-to-noise ratio in a dynamic scene is also challenging since recreating the same scene for an objective comparison of the biases is not trivial. Moreover, optimal biases are often task-specific and cannot be generalized. Automatic bias tuning was investigated in [13]. The authors used the event rate (ER) as the metric for finding the optimal biases, by tuning the biases independently. With our setup, we have a static background and the only moving object in the scene is the table tennis ball. This makes the ER a reliable metric for tuning the biases. To recreate identical observations for different settings, we relied on a ball thrower to have the same ball trajectory every time. However, the ball thrower can not guarantee the same logo position. For this reason, we used a robot arm to move the ball exactly the same way, with the required ball orientation. This method, however, was only used for qualitative results (Figs. 13 and 18), as the moving robot arm influenced the event rate.

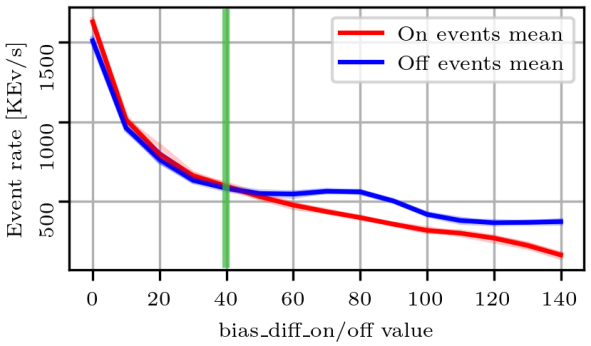
6.0.1 *bias_on* and *bias_off*

Tuning the pixel sensitivity of the event camera with the biases *bias_on* and *bias_off* is a trade-off between having as little noise as possible while still capturing relevant events for our task. Increasing *bias_on* and *bias_off* will increase the contrast threshold and, therefore, decreases the sensor's sensitivity.

Figure 12a displays the event-rate for a static scene with regard to *bias_on* and *bias_off*. As the scene is completely



(a) Event-rate for different *bias_on* and *bias_off* values while observing a static scene, where events can be considered to be noise. The y-axis is in log-scale.



(b) Event-rate for different *bias_on* and *bias_off* values while observing a flying ball (averaged over 5 samples)

Figure 12. Event-rate for different pixel sensitivity (*bias_on* = *bias_off*)

static, the generated events can be considered noise. The noise is decreased to an acceptable level starting when *bias_on/off* goes over 40.

We also measured the event rate when a ball was flying in front of the camera. We noticed that increasing *bias_on/off* further led to an unequal number of ON/OFF events when observing the ball, as shown in Fig. 12b. This is due to

the none symetrical behavior of the event pixels for OFF and ON events. To avoid this behaviour, *bias_on/off* higher than 60 are to be avoided. As such, we decided to set the *bias_on/off* to 40.

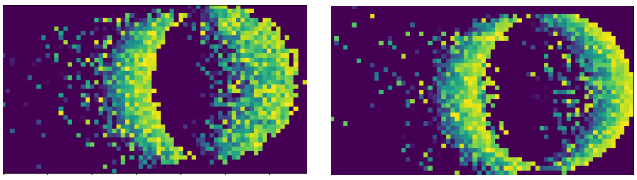
In Fig. 13, we show the effects of different *bias_on/off* on the accumulated event frame. We can clearly see a decrease in the number of generated events for increasing *bias_on/off*.

(a) *bias_on/off* = 40 (b) *bias_on/off* = 90 (c) *bias_on/off* = 140

Figure 13. Accumulated event frames of the moving ball with the logo on the edge for different bias_on/off values (with an accumulation time of 3ms)

6.0.2 *bias_fo*

The *bias_fo* controls the pixel's low-pass cut-off frequency. It can filter out events generated by fast motions and flickering. Increasing *bias_fo* will also increase the cut-off frequency. There is a trade-off between noise and latency. Indeed, a low-pass filter with a lower cut-off frequency will also introduce higher latency to the events as they will be triggered with some delay. This can be observed in Fig. 14a where more new events are being triggered inside the ball. These events are "late" events generated by the edge of the



(a) *bias_fo* = −35 (b) *bias_fo* = 55

Figure 14. Linear time-surfaces ($\tau = 1ms$) of a flying ball with different *bias_fo*.

ball. This can also be observed by the less distinct edge of the ball for $bias\_fo = -35$. To avoid such "late" events being mixed up with events from the logo, the *bias_fo* was set to 55, which increased the cut-off frequency to the maximum. However, from the event rate for different *bias_fo* values shown in Fig. 15, we see that this setting only affects the event rate for negative *bias_fo* values. So any value between 0 and 55 would also work.

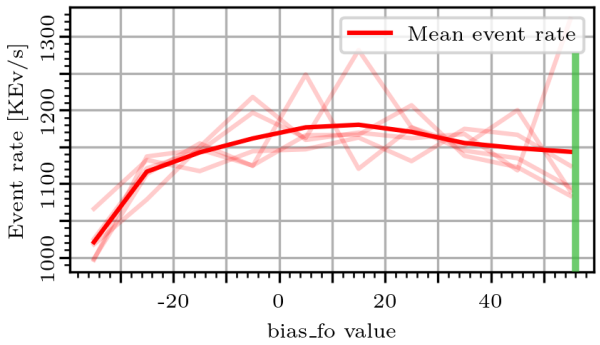


Figure 15. Event-rate for different *bias_fo* values while observing a flying ball (averaged over 5 samples)

6.0.3 *bias_hpf*

The *bias_hpf* controls the pixel's high-pass cut-off frequency. It allows filtering out low-frequency events such as noise and events generated from slow motions. The event rate for different *bias_hpf* is shown in Fig. 16. We set *bias_hpf* to the minimum of 0 for filtering out the least number of events. All in all, the settings for *bias_fo* and *bias_hpf* were chosen to filter out the minimum amount of events.

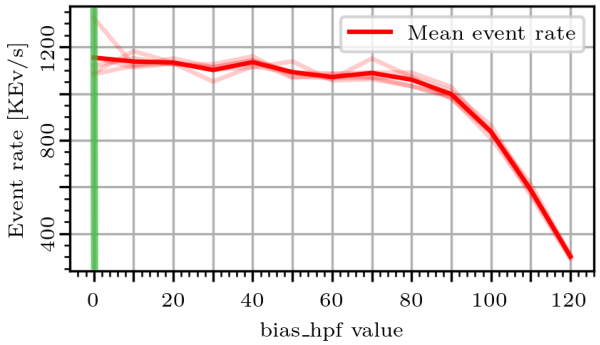
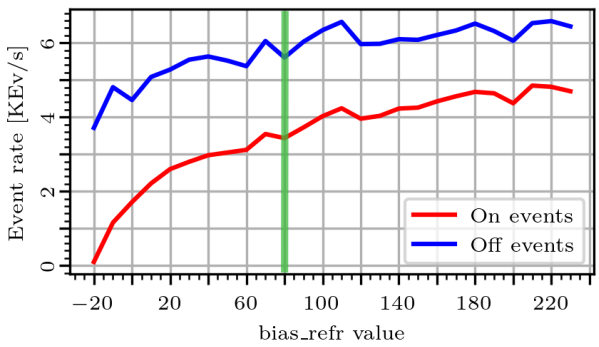


Figure 16. Event-rate for different *bias_hpf* values while observing a flying ball (averaged over 5 samples)

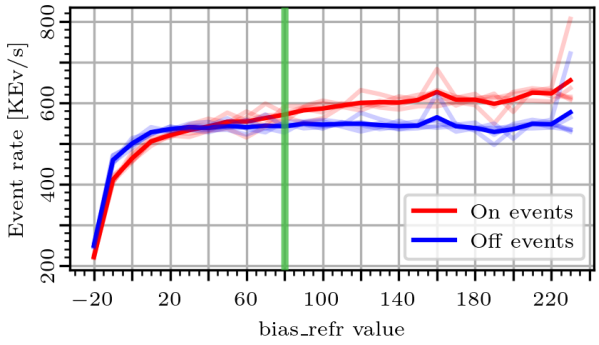
6.0.4 *bias_refr*

The *bias_refr* controls the pixel's refractory time, which is the time during which a pixel does not detect any change in illumination after it emitted an event. Decreasing the pixel's refractory period will generate more events for a large illumination change. A shorter refractory period will also lead to a higher event rate. The event-rate for different *bias_refr* is shown in Fig. 17.

We increased *bias_refr* to 80 to not miss any events,



(a) Event-rate for different *bias_refr* values while observing a static scene, where events can be considered to be noise



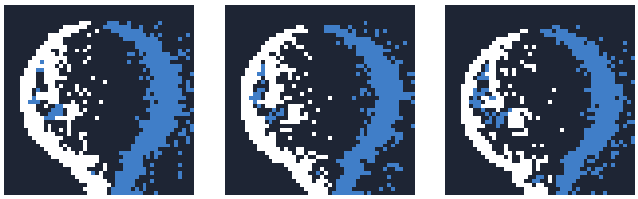
(b) Event-rate for different *bias_refr* values while observing a flying ball (averaged over 5 samples)

Figure 17. Event-rate for different *bias_refr* values

which is also suggested by the camera's documentation. We do not want the pixel to be "dead" at the ball's edge when the logo comes into view. Thus, as soon as the edge moves to another pixel, that pixel should be able to trigger new events. Since the nominal velocity of the flying ball is around 6000 pixels/s, the edge will move to another pixel in approximately every 0.1ms. Increasing *bias_refr* to values higher than 80, i.e., shortening the refractory period results in an increase in noise, as shown in Fig. 17 where the ON event rate keeps increasing with *bias_refr*. On the other hand, a higher refractory period results in too few events for the spin estimation to work accurately Fig. 18c.

7. Filters

In addition to the camera's bias, filters can be applied to the event stream to cancel redundant information or filter out noise. The two options we considered are the Spatio-Temporal-Contrast (STC) filter and the TRAIL filter, as well as their combination. The STC filter filters out isolated events that are not followed by other events of the same polarity. It does so by only retaining the second event from a burst of events. The time window during which the second



(a) *bias_refr* = 120 (b) *bias_refr* = 80 (c) *bias_refr* = −20

Figure 18. Accumulated event frames of the moving ball with the logo on the edge for different *bias_refr* values (with an accumulation time of 2ms). A shorter refractory period of 120 results in too many events/noise on the edge. On the other hand, a longer refractory period of −20 results in too few events for the spin estimation to work accurately. Therefore, we used a value of 80, as a trade-off.

event is waited for can be tuned.

The TRAIL filter, as its name implies, gets rid of events that happen "behind" a moving edge with a certain time window, except if the event is of the opposite polarity. These filters help clean the event stream by reducing noise and redundant information. Both filters can be combined depending on the objective, only leaving one event out of a burst, giving much cleaner edges. It should be noted that the filter's time window parameter should be tuned depending on the ball's velocity.

In Fig. 19 we show how increasing the filter threshold affects the event rate of a flying ball. For the STC filter,

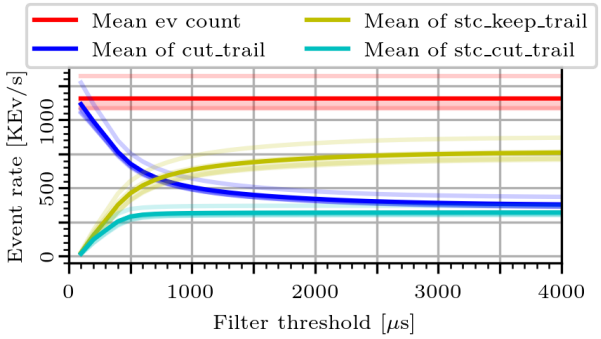


Figure 19. Effect of filter threshold on the event rate of a flying ball for different filter options

we keep very few of the observed events at low threshold values, as there are not enough events that are generated in quick enough succession to count as one burst. As the filter threshold increases, we still only keep a portion of the overall events, as for each burst (successive events at the same location), the filter discards the first one. For the TRAIL filter, the opposite is true. For each burst, only the first event is kept, and subsequent events within the threshold are discarded if they have the same polarity. Thus, with a higher

threshold, fewer events are kept. Applying the STC and then the TRAIL filter (stc-cut-trail) keeps only one event of the same polarity for each burst. Due to this, stc-cut-trail discards the most events but leads to the cleanest edges.

Fig. 20 shows the events that the different filters keep depending on their threshold parameter, with the unfiltered version at the top. For the STC filter, on low threshold values, very few events are being kept, as not enough events fall into the threshold time. Especially for the front edge (top in image), we can see that it takes a higher threshold value before enough events are kept to make it out clearly. It is worth noting that when the logo follows the front edge closely (as in the figure), the front edge will sometimes disappear. It is not entirely clear why this happens. However, we believe it may be because not enough events from the front edge are generated before the logo's events are triggered. Due to this, the STC filter removes the thin line of ON-events from the ball's edge, as it is treated as the start of the burst of the logo edge. As clearly visible, the STC filter removes a lot of noise from the recording, as noise events are usually isolated and not part of a burst of events generated by a moving edge. However, behind the initial edge, there are still a lot of trailing events.

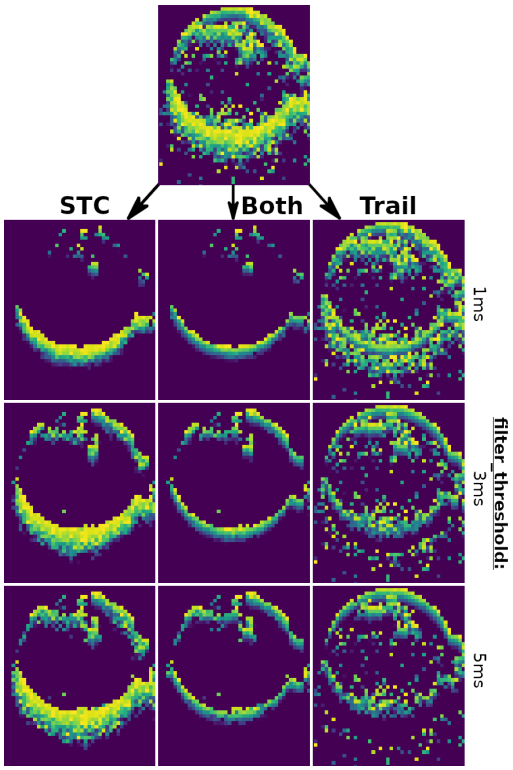


Figure 20. Time surfaces (positive and negative events, 5ms accumulation time), showing the effect of the filter threshold on events kept by the different filter combinations. Top row: no filter.

The TRAIL filter's effects are very clearly visible as the threshold value increases, especially at the back (bottom) of the ball. After an event from the edge is recorded, subsequent events of the same polarity that fall into the threshold time, are removed. This means that many trailing events generated by the edge are cut out. We can clearly see the gap behind the back edge increase as the filter threshold increases. Note, however, that the logo is not being removed behind the front edge, as its polarity change resets the filter window.

Combining the STC and trail filter leaves us with a very clear edge and little remaining noise. Increasing the threshold over 5000μs changed very little about the number and quality of events kept, so we use this as our $filter\_threshold$ when applying STC-cut-trail filter.

8. Optical flow estimation

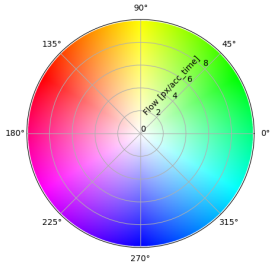


Figure 21. HSV color wheel used to represent the optical flow

We represent the flow using the HSV color scheme for all the optical flow estimations in Fig. 21, where high saturation indicates high flow, and the hue indicates the flow direction.

In Fig. 22, we show examples of accumulated event frames and flow estimates of sidespin for different spin values. As clearly visible, higher spin values lead to more events, allowing more accurate flow estimation.

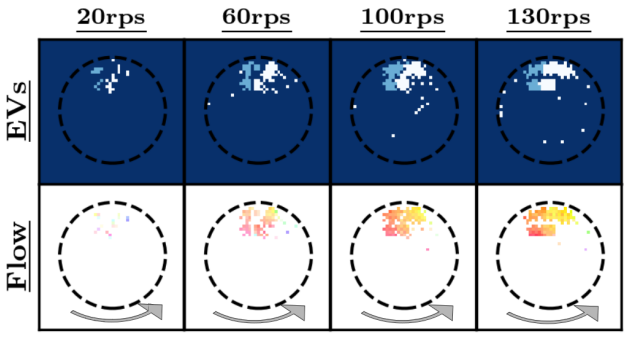


Figure 22. Optical flow for sidespin with $t_{acc} = 0.715ms$. The arrows indicate the direction of the spin.

Examples of accumulated event frames and corresponding flow estimates of backspin/topspin for different spin values are visualized in Fig. 23. The same as for the sidespin,

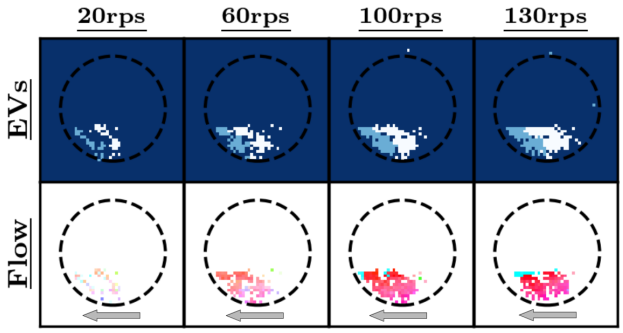


Figure 23. Optical flow for backspin/topspin with $t_{acc} = 0.715ms$. The arrows indicate the direction of the spin.

higher spin values result in more events and, thus, more accurate optical flow.

9. Cleaning the EROS time surface

Even with the STC filter, noise makes it through to the EROS time surface. Not only noise can pollute the time surface, but also trail events, as seen in Fig. 24a. To avoid misdetecting circles with the Hough Transform, we eliminate these isolated events with a hit-or-miss detection. The EROS time surface without cleaning is shown in Fig. 24a and with cleaning in Fig. 24b.

(a) No cleaning

(b) With cleaning

Figure 24. The EROS time surface (a) without cleaning and (b) with cleaning

The kernel we run for the hit-and-miss is defined as

$$k = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix}. \quad (7)$$

10. Generating ground truth for the ball detector benchmark

We rely on event accumulation frames and blob detection to automatically generate ground truth for the ball detection. To do so, we use a large accumulation time of $t_{acc} = 10$ms to generate accumulated event frames. This makes the ball clearly visible for blob detection, as shown in Fig. 25.

Figure 25. Example of ball position labeling with blob detection. The red circle is the ball detected by the blob detection.

11. Ball thrower benchmark

To capture the same ball trajectories, the event and frame cameras were installed next to each other. Despite trying to align them as best as possible, it is not possible for them to exactly share the same field of view. We thus needed to calculate the transformation from the event camera to the frame camera for the benchmark. The transformation between both cameras was calculated in the same fashion as for a stereo camera setup. A chessboard pattern displayed on a LCD screen was used for that purpose. The screen was set to blink for the event camera and to static display for the frame camera. Thanks to this, the spin calculated with the event camera could be transformed to match the spin calculated with the frame camera.

Regarding the generation of the ground truth, SpinDOE [23] was used. It can indeed estimate the table tennis ball spin with high accuracy. It has a relative error of 1% for the spin magnitude and a spin axis error of $2.4°$. However, the ball thrower used has some stochasticity to its behaviour: even though the settings are the same, the ball will not have the same trajectory. To compensate for this, dot-patterned balls were shot 5 times with the same settings for the ball thrower and the ground truth is assumed to be the mean spin vector.

In Tab. 4, we list all the ground truth values for the different velocity and spin settings. The ground thruth values for the higher spins could not always be calculated because of the motion blur due to high spins. It should be noted

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
25	137.81	124.60	107.42	103.23	98.22	81.83	63.69	38.87	35.72	41.38	62.41	-	-
20	127.10	110.70	92.45	93.09	88.47	72.36	60.61	45.53	45.40	53.64	64.80	-	-
15	119.51	98.76	81.94	81.89	72.69	66.75	63.80	57.20	57.48	63.80	68.69	-	-
10	92.31	74.41	60.74	61.23	58.74	57.85	61.46	61.80	62.27	65.54	66.11	-	-

(a) Sidespin

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
25	111.81	90.41	64.34	65.46	53.42	34.13	17.01	16.61	35.10	54.20	74.14	-	
20	99.99	78.42	55.22	53.91	39.42	13.30	2.05	22.39	39.94	54.05	68.90	-	-
15	93.11	71.01	47.27	39.65	22.80	5.65	4.77	25.30	40.17	54.37	62.67	95.77	-
10	72.33	54.58	36.46	23.41	9.21	0.91	7.41	17.10	25.04	34.94	38.67	59.75	88.76

(b) Back/top-spin

Table 4. Ground truth values of the spin magnitude [rps] calculated with SpinDOE for the ball thrower. The rows represent the different velocity settings and the columns the spin settings.

that with the ball thrower used, the ball spin magnitude also increased with higher ball velocity settings.