Optical Flow Estimation Using Event-Based Cameras

Improving Optical Flow Estimation Accuracy Using Space-Aware De-Flickering

Per Magnus Skullerud

Supervisors: Nergis Tömen, Hesam Araghi

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Per Magnus Skullerud
Final project course: CSE3000 Research Project
Thesis committee: Nergis Tömen, Hesam Araghi, Guohao Lan

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Event cameras are novel sensors whose high temporal resolution and bandwidth motivate their use for the *optical flow* estimation problem. However, the properties of event cameras also introduce a vulnerability to *flickering*. Flickering hurts the perceptibility of motion by overwhelming event data with unrelated information. The single existing event de-flicker method (EFR) is built for scenarios where the relative position of the camera and the flickering object is constant, which is uncommon in motion-heavy optical flow estimation scenarios. Our contribution is a new de-flickering method that incorporates spatial awareness of nearby pixels. We hypothesize this feature to increase robustness to movement, and thus to better improve optical flow accuracy. Compared to EFR our method falters at filtering intensely flickering surfaces, but better preserves the spatial coherence of edges. However, we observe that both de-flickering methods remove much geometric information, especially given slow motion or weak ambient illumination. Our benchmarking shows that neither our method nor EFR significantly affects optical flow estimation accuracy, despite reducing event counts by $50 - 65\%$. Overall, we conclude that the niche benefits of spatial filtering are nullified by the result that filtering hardly affects optical flow estimation.

Code: https://github.com/per1234567/spatial-event-flicker-filtering

1 Introduction

Optical flow estimation is the problem of predicting near-future motion using earlier motion. More precisely, optical flow is the displacement of pixels corresponding to the same object in two adjacent video frames [3]. Predicting optical flow is relevant to systems whose functioning incorporates the movement of their surroundings (e.g. drones, robots, etc.). Conventional (shutter-based) cameras are suitable for computing optical flow if motion is slow, but details are lost at high speeds due to motion blur and insufficient frame rates [2]. These problems are alleviated by *event cameras* - novel sensors featuring high temporal resolution and bandwidth, as well as high dynamic range. However, event data is largely incompatible with optical flow estimation algorithms designed for standard cameras. That is because standard cameras produce periodic frames containing absolute brightness and colour information, whereas event cameras record asynchronous per-pixel brightness changes (*events*). In spite of event cameras' challenging and unfamiliar output format, their great potential for optical flow estimation has spurred ample research [4].

A broad challenge in using event cameras for optical flow estimation is accounting for brightness variations. That is because events produced by brightness variations are uncorrelated to motion, and thus obstruct the purity of motion information. This work focuses on a type of brightness variation known as *flickering*. Flickering is a faint, high-frequency brightness oscillation exhibited by some light bulb designs when connected to an AC power grid. While flickers are usually too faint to be observed by the naked eye or even a standard camera, they are easily picked up by event cameras due to their high dynamic range and temporal bandwidth [6]. Flickering can overwhelm an event stream, increasing processing costs while distorting the scene's geometric features. Explicitly mitigating flickers is not essential when using learning-based methods, as training can embed a resistance to the adverse effects of flickering [7]. However, powerful model-based methods built on the *Contrast Maximization* (CM) [5] framework often assume *brightness constancy* (that scene illumination is invariant); the methods' unavoidable reliance on this assumption is listed as a limitation [3; 5; 8; 10]. Some of these methods explicitly highlight bad accuracy attained on the popular DSEC dataset's [6] single heavily flickering test sequence *zurich_city_12_a*. To the best of our knowledge, the adverse effects of flickering on optical flow estimation have never been explicitly investigated.

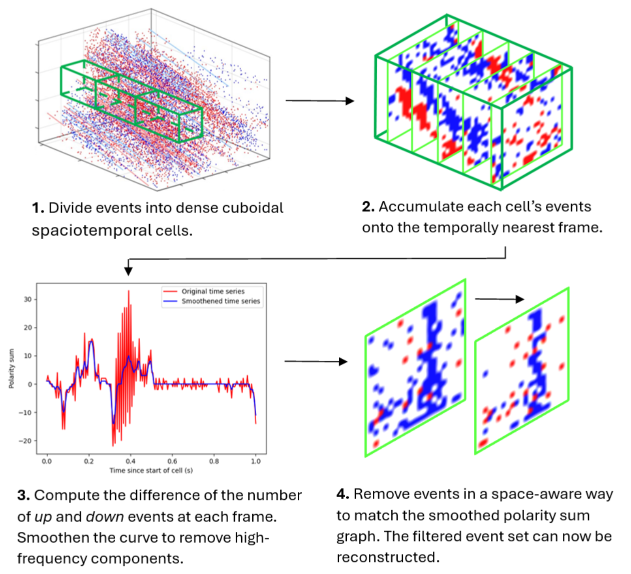
1. Divide events into dense cuboidal spaciotemporal cells.
2. Accumulate each cell's events onto the temporally nearest frame.
Original time series
Smoothened time series
Polarity sum
Time since start of cell (s)
3. Compute the difference of the number of *up* and *down* events at each frame. Smoothen the curve to remove high-frequency components.
4. Remove events in a space-aware way to match the smoothed polarity sum graph. The filtered event set can now be reconstructed.

Figure 1: Our algorithm's event data de-flicker pipeline. Step 1 image adapted from Gallego et. al.'s survey [4].

Our research follows from *EFR* [11], which is (to the best of our knowledge) the only algorithm removing flickering from event data. This method produces remarkable results in static conditions but is not designed (nor tested) for motion-rich scenarios such as optical flow estimation, where the relative position of the camera and the flickering object is not constant. That is because the algorithm performs filtering per pixel, but the temporal distribution of events reaching a single pixel is likely distorted by motion. **We hypothesize that integrating spatial awareness into event-based de-flicker filters improves subsequent optical flow estimation accuracy**. We further hypothesize that the effect is more positive for CM-based methods due to them frequently assuming brightness constancy. The overarching objective is to derive an algorithm filtering out flickers that is robust to motion, or to demonstrate that such an improvement is not beneficial.

To investigate our hypothesis, we design a motion-resistant algorithm that pre-processes event datasets to remove events identified as caused by flickering. We achieve this by grouping events into small spatial regions, wherein distortions caused by small movements are less noticeable. We then remove events responsible for rapidly changing ratios of brightness *up* and *down* events in narrow temporally adjacent slices. The process is visualized in Figure 1. We test our algorithm against EFR in scenarios with and without flickering, using CM-based and supervised learning-based optical flow estimators. In short, our contributions are:

- A hypothetically motion-resistant algorithm to remove flickering from event data.
- An evaluation of our method relative to EFR for optical flow estimation tasks.
- A deeper understanding of the impact of flickering on event data used for optical flow estimation.

2 Previous works

2.1 The Effects of Brightness Variations in Optical Flow Estimation

Model-based optical flow estimators are predominantly built on the Contrast Maximization (CM) framework, which assumes *Brightness constancy* [5]. This restrictive assumption is therefore inherently embedded into CM-based optical flow estimators, but they are nevertheless promising due to their great accuracy in favourable lighting conditions. A notable example is MultiCM [10], which builds upon the initial formulation of CM to mitigate its dominant aperture and event collapse problems [9]. MultiCM has been empirically shown to outperform many learning-based methods in the widely used MVSEC [13] dataset. This result suggests that accurate optical flow can be computed without network-based methods, which is relevant in scenarios where GPU acceleration is impractical. Brightness constancy is also regularly assumed by self-supervised learning (SSL) methods that use CM as an objective function. TamingCM [8] is one such method; it integrates CM by continuously adjusting contrast-maximizing motion directions upon receiving new events. TamingCM performs well if lighting is constant, but benchmarks on the single flicker-heavy *zurich_city_12_a* sequence from the DSEC dataset [6] suggest that intense brightness variations heavily hurt accuracy. TamingCM's EPE (average endpoint error) metric on this sequence is 50.4% worse than the average of all test sequences, as seen in the publicly available optical flow estimation benchmark on the DSEC dataset [1]. Therefore, it is both theorized and evidenced that CM-based methods are adversely sensitive to brightness variations.

Supervised learning (SL) methods can discard the brightness constancy assumption because training them may embed an ability to account for brightness variations. However, training SL methods is difficult due to the immense challenge of acquiring ground-truth optical flow vectors. Furthermore, encoding event data into tensors typically leads to losing information about the temporal distribution of events [7; 12; 14]. An early SL method is EV-Flownet [14], which encodes events as 4-channel 2D images containing the count and most recent timestamp of up and $down$ events at each pixel. Later methods, such as E-RAFT [7] and IDNet [12], encode events into 3D voxel grids, effectively preserving more temporal detail. This data representation is more likely than EV-Flownet's to capture intricate temporal patterns such as flickering. In the DSEC benchmark [1], both E-RAFT's and IDNet's EPE metric is $\sim$20% better in the flicker-heavy *zurich_city_12_a* sequence compared to their respective averages over all sequences. This is in stark contrast to TamingCM's significant decrease in accuracy. It is worth noting that SL methods consistently outperform all other benchmarked methods, though it has been posited that SL methods overfit to DSEC's largely forward-facing motion [10]. Nevertheless, an accuracy improvement observed on a flicker-heavy sequence suggests that SL methods may be resistant to brightness variations due to their inherent properties.

2.2 Event Data De-Flickering

To the best of our knowledge, EFR [11] is the only algorithm that attempts to solve the problem of removing flickering from event data. EFR incorporates the observation that the brightness signal of flickering lights is not sinusoidal, but is composed of infinitely many harmonics of decreasing magnitude, with peaks at multiples of the power grid frequency. This observation motivated EFR to use a feed-forward comb filter with a feedback mechanism, which encodes an ability to attenuate the recurring frequency peaks. Figure 2 shows the power spectral density of a light bulb, and the comb filter's corresponding shape. EFR produces good results when the relative position of the light source and camera is constant. However, the method performs filtering per pixel, where each pixel is exposed to the same composition of harmonics over a long time frame. These conditions are likely to be violated in many optical flow estimation scenarios, as both the camera and the flickering objects may move independently. We are not aware of any methods preprocessing event data to reduce flickering adapted to motion-heavy situations.

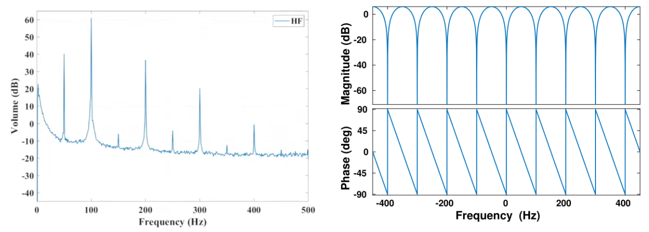

Figure 2: **Left:** The power spectral density of a flickering light bulb's intensity. **Right:** The frequency response of the comb filter used by EFR; damping occurs at the frequency peaks of flickering. Images taken from the EFR paper and video demonstration [11].

3 Method

We describe the process of designing our algorithm to remove events caused by flickering in motion-heavy scenarios. We begin by describing event cameras and CM, whose properties have motivated our decision to improve optical flow estimation via preprocessing. We then discuss our assumptions made to ease modeling flickers, and finally, we explain our algorithm's steps.

3.1 Event Cameras & Contrast Maximization

Event cameras are light sensors built upon an operation principle different to conventional (frame or shutter-based) cameras. Instead of employing a shutter mechanism to capture frames at a constant frequency, pixels of event cameras individually record asynchronous events triggered whenever the change in brightness (log intensity) since the last emitted event exceeds a certain threshold [4]. The output of an event camera is thus a stream of data points characterized by four parameters: pixel x and y coordinates, timestamp t, and polarity $p \in \{-1, +1\}$ (whether brightness decreased or increased respectively). Event cameras boast high temporal resolution and high pixel bandwidth, as well as low power consumption and high dynamic range. However, event cameras do not measure absolute brightness nor color; event streams are difficult to integrate with multimedia processing algorithms designed for classical cameras.

CM is one of the first methods operating natively on event data [5], optical flow estimators built upon it show good performance if lighting is constant [8; 10]. CM's reliance on brightness constancy follows from its modeling of all events as caused by moving edges, whose movement leaves a planar trail of events in spacetime. CM finds the orientation of the planar trail by warping its constituent events along a trajectory, iteratively adjusting it to make it better resemble a likely direction of motion (Figure 3). The objective function of this iterative process is the variance (contrast) of a 2D image whose pixel intensity is the count of events accumulated along the trajectory ending at that pixel. However, the presence of brightness events uncorrelated to motion can reduce image contrast at any warp direction. The consequence can be a slower convergence speed of optimization or an incorrect final direction, especially in cases of structured noise such as flickering. Furthermore, warp trajectories are typically assumed to be linear; regions containing motion modellable as linear are small and thus lack context from surrounding events that may help identify large edges whose observability is less impacted by noise. CM's demonstrated weakness to flickering, alongside its indispensable reliance on brightness constancy, has motivated our choice to filter out flickers via preprocessing as opposed to integrating flicker resistance directly into optical flow estimators.

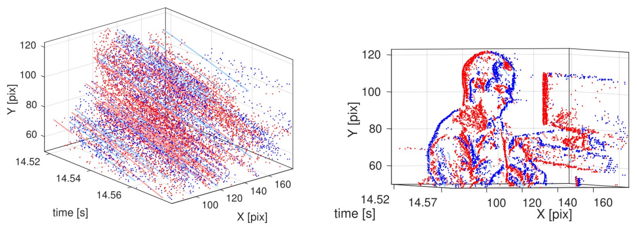

Figure 3: **Left:** A 3D scatter plot representing events in spacetime. **Right:** When viewed from the angle corresponding to the direction of motion, the events form a sharp image. Contrast Maximization finds this direction. Images from Gallego et. al.'s survey [4].

3.2 Problem Modelling

To make the problem of identifying flickering-induced events tractable, we replace brightness constancy with several new less restrictive assumptions, not as likely to be violated in a real-world scenario. The assumptions, while not universally describing brightness inconsistencies, provide a framework for identifying events triggered by flickering. These ideas follow primarily from observations we have made while exploring event data.

1. *Events caused by mild brightness changes are indistinguishable from noise.* Mild brightness changes are characterized by a relatively small total number of events and presence across the whole scene possibly over a long time period. This characterization is almost identical to that of noise (sporadic, isolated motion events), possibly differing solely by illumination events' near-uniform polarity in a short time slice. However, harnessing this distinction is probably unrealistic given the large number of total events produced under normal conditions.
2. *The presence of brightness events does not affect the observability of motion events (and vice versa).* It follows that each event can be classified as uniquely produced by a motion or a brightness change, an idea underpinning the process of filtering out events identified as unwanted. However, the assumption may be violated in cases where motion and illumination changes affect a scene point simultaneously. For example, equal positive and negative brightness changes produced by motion and lighting changes would cancel out, effectively producing no events at the affected spatiotemporal region. Mitigating this would require generating artificial events to fill the gap, which is likely overcomplicated for fixing these edge cases. Realistically, however, an overlap of motion and brightness event groups in a region maintains many of their distinguishable features.
3. *Motion events are spatially close to other motion events of opposite polarity.* Brightness at a point does not rise nor fall indefinitely, so the presence of many $+1$ events must be followed by a presence of -1 events (and vice versa). In the case of motion events, this can usually be observed as edges of opposite polarity passing through pixels sequentially as seen in Figure 4. On the other hand, flickers result in close proximity between events of opposite polarity *temporally* because flickering is an oscillation of illumination intensity over time. This observation is important for modeling the way events caused by different effects are positioned relative to each other.

The three assumptions imply the use of temporal filtering in small spatial regions to be beneficial. The first assumption suggests that for brightness events to be distinguishable, the illumination changes should be intense, i.e. many events can be attributed to them. The third assumption implies that the sum of event polarities in a spatiotemporal region mildly fluctuates around zero when only motion events are present, since the presence of an edge results in a similar number of events of opposite polarity. Whereas these fluctuations are chaotic and aperiodic, flickering is likely to appear mostly sinusoidal. Consequently, a time series of event polarity sums

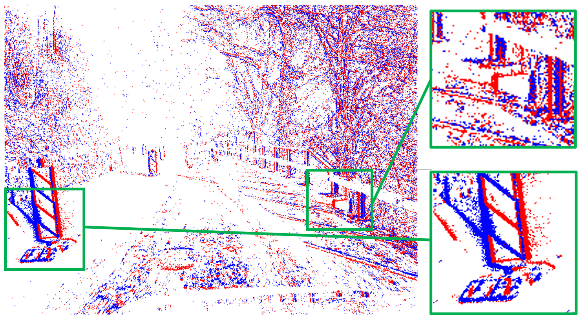

Figure 4: A common spatial feature of event data is edges being mirrored in the direction of motion by edges of opposite polarity. Visualization frame from sequence *interlaken_00_c* of the DSEC dataset [6]. The frame is produced by grouping events by pixels in a temporal slice of width $1/200s$, and color-coding pixels by the dominant event polarity (blue if $+1$, red if -1, white if equal).

(the count of $+1$ events minus the count of -1 events) will appear as fluctuations around zero superimposed onto a high-magnitude sinusoid. We can remove it with a smoothing filter while maintaining the motion component of the time-series. The presence of the sinusoid is unlikely to be affected by small movements, as translating event clouds spatially retains largely the same distribution of dominant event polarities over time. This will not hold under fast motion due to many events entering and leaving the boundaries of a fixed spatiotemporal region; this is a limitation of our method. Nevertheless, in most scenarios, as hypothesized, we expect that incorporating information from events spatially close to those being filtered increases robustness to motion and thus result in more accurate optical flow estimation.

3.3 Algorithm Description

We integrate the results of previous works with our observations to derive an algorithm for solving the problem at hand. We provide, motivate, and evaluate the steps of the algorithm. We choose hyperparameters that work favourably with the DSEC dataset, which our implementation is designed around. The steps of our algorithm are:

1. **Group all events into cells of a dense 3D lattice**. This is done to group events into clouds of a size appropriate for identifying flickering behaviour. Specifically, we first divide the entire event space into temporal slices of width 1 second, then divide each slice spatially into 80×60 cells of dimensions $8px \times 8px$. Temporal slicing is essential to avoid loading in the entire dataset simultaneously, assuming that events are pre-sorted by time. The cell dimensions $8px \times 8px$ were chosen because they evenly divide the pixel dimensions of the camera used to record the DSEC dataset ($640px \times 480px$) [6]. A cell too small would be unable to capture oscillating behavior due to insufficient data, whereas one too big may contain different lighting effects or lead to heavy discretization artifacts in the filtered event set. In our testing, cell dimensions of size $1s \times 8px \times 8px$ produce visualizations with desirable features.

2. **Compute the event polarity sum time series in each cell**. This step is what fundamentally highlights the presence of brightness oscillations by sampling the dominant event polarities at many time steps within a cell. To achieve this, we subdivide the cell temporally into 200 equally spaced frames and group events into their respective temporally closest frames by common pixel coordinates. An illustration of how we perform the event splitting is given in Figure 5. The hyperparameter 200 is a multiple of the frequency of the AC power grid of Switzerland (50Hz), where DSEC was recorded [6]. We sum the event polarities of each frame to obtain a 1-dimensional time-series of length 200.

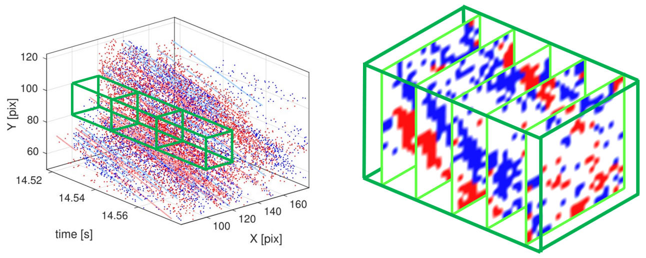

Figure 5: **Left:** Example of event grouping showing three $0.02s \times 20px \times 20px$ cells in an event space, adapted from [4]. **Right:** A division of a cell's events into 5 frames. The images are formed using parameters with values chosen to ease visualization.

3. **Apply a Gaussian filter onto the event polarity sum time-series**. By applying a Gaussian filter onto the time-series obtained in the previous step, we remove the high-frequency sinusoidal component as shown in Figure 6. We use σ=1, as this smoothens out adjacent peaks of the high-frequency flickering while preserving the low-frequency general shape of the series. We opt for a Gaussian filter due to its generality; a filter derived from the power spectral density of a flickering light bulb (Figure 2) may suffer from its dominant frequencies being distorted by motion. With respect to our second assumption, this step corresponds to removing the event field caused by flickering while maintaining the natural fluctuations resulting from motion.

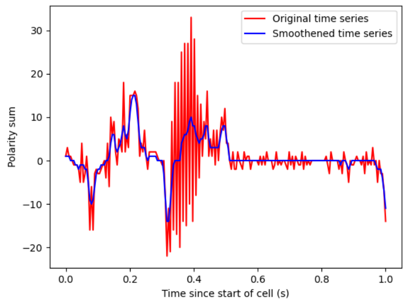

Figure 6: The original and the dampened event polarity sum time-series in a cell; our method attenuates flickering visible at 0.3-0.4s.

4. **Delete events such that the event polarity sum time-series matches the smoothed one**. This means deleting $+1$ events in frames whose time-series entry was decreased by smoothing, or -1 events where smoothing increased it. The challenging part of this step is removing events in a way that preserves well-defined edges while removing the formless event structures attributable to flickering. To solve this problem, we group events from adjacent 2×2 pixel squares into buckets and repeatedly remove one event from each bucket until the expected number of events in the frame is reached. Our approach removes a smaller proportion of events from dense clusters compared to sparse noise-like fields as exemplified by Figure 7. Conversely, a simpler method like randomly selecting events for deletion would remove the same proportion from edges and flickers. This step concludes the algorithm.

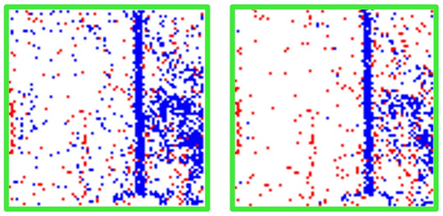
Figure 7: **Left:** The dominant event polarity at each pixel in a frame. **Right:** The frame after removing 636 events of polarity $+1$. Sparse & noisy regions are affected more heavily as desired; the clear edge is preserved.

4 Evaluation

We begin this section by discussing the implementation of our described algorithm and presenting the experimental setup. Our results begin with an analysis of the filtering itself, which we use to support the subsequent analysis of the optical flow estimation results.

4.1 Implementation

We have implemented and published the described algorithm in Python. Since we have not implemented GPU acceleration, our algorithm is too slow to be practically applicable - filtering a dataset with 10^9 events on an Intel i5-7500 CPU takes around 2 hours. The performance is mostly limited by the process of removing events, as we were unable to find a fast method of grouping events by common coordinates; this part of our algorithm is implemented in pure Python. As such, our code is primarily a proof of concept to be used for research purposes.

Our development was guided by visualizing the effects of possible adjustments using the *zurich_city_10_a* sequence from the DSEC dataset [6]. This sequence was chosen as it is characterized by a strong flickering due to overhead streetlights in a relatively dark evening setting, as seen in Figure 8. As a result, the implementation may not generalize without tweaking our chosen parameters. Notably, the hyperparameter denoting the number of frames per cell (200 in our case) is chosen with knowledge of the lights' flickering frequency. We specifically use the sequence *zurich_city_10_a* because it captures the effects of flickering while involving few other severe challenges.

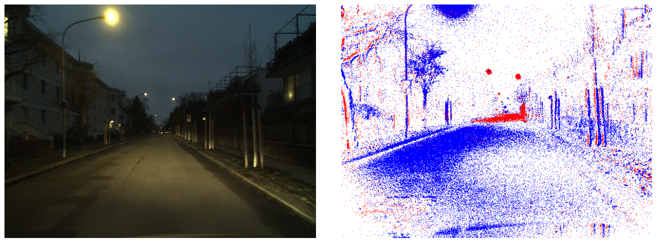
Figure 8: **Left:** A frame recorded using a standard camera from DSEC's *zurich_city_10_a* sequence. **Right:** A frame at the same timestamp constructed from event data; flickering events on the street are abundant. The frames appear misaligned due to absent rectification.

4.2 Procedure

Given that our algorithm is a proof-of-concept solution to a scarcely studied problem, we aim to test on cases we know to be favourable. Therefore, we test on a reduced version (first 10^9 events) of the flicker-heavy sequence *zurich_city_10_a*, which was also used to guide development. To evaluate the ability of our algorithm to distinguish flickers from other effects, we also test on the sequence *zurich_city_02_d* which is filmed at a brighter time of day and displays no flickering. Both sequences are shot facing forward from a car driving along a street, the vehicle's speed is faster in *zurich_city_02_d*. We aim to provide complete qualitative and quantitative evaluation for both filtering and optical flow estimation. We examine the original sequences, as well as ones filtered by our algorithm and by EFR, to determine if our spatial awareness delivers meaningful benefits.

To support the analysis of optical flow estimation results, we begin by identifying noteworthy characteristics immediately displayed by the filtering process itself. We visualize the original event set alongside the filtered one and qualitatively highlight notable phenomena. Our visualization technique is a video made of frames using the same method as described in Figure 4. We focus on our chosen flicker-heavy sequence *zurich_city_10_a* because we are primarily interested in the effect of filtering on large flickering blobs, where a large number of flicker events is concentrated. We also care about the clarity of affected edges, as they are heavily exploited by optical flow estimators to discern motion [7; 8]. Overall, we seek to determine whether both de-flickering algorithms' results correspond to expectations before attempting to predict optical flow.

Ultimately, we are interested in the effect of filtering on optical flow estimation accuracy. To validate our hypothesis that spatial filtering is an improvement, we benchmark our two chosen sequences on their original forms, as well as versions produced by filtering using our algorithm and EFR. As optical flow estimation algorithms, we use TamingCM [8] since it is the best-ranked CM-based algorithm on the public

DSEC benchmark (as of the writing of this paper) [1]. To contextualize the relevance of flickers to CM, we additionally benchmark E-RAFT as a representative supervised learning (SL) method [7]. We use a checkpoint, pre-trained on DSEC, provided by each method's authors; we do not train the networks ourselves. We record two often used metrics: the average Euclidean distance between predicted and ground truth optical flow vectors (*EPE*), and the percentage of vectors off by more than 3 pixels (*3PE%*). As also hypothesized, we expect our filtering's advantages to be best defined when using TamingCM on *zurich_city_10_a* because of CM's sensitivity to flickers. To better justify the validity of our results, we qualitatively highlight noteworthy variations of errors by spatiotemporal region.

It is worth noting that the benchmarking was not performed under perfectly identical conditions. A problem arises from TamingCM using reformatted versions of DSEC sequences, whereas E-RAFT accepts them in the same format as they are downloaded from the official repository [1]. To mend this problem we had to edit TamingCM's data loader. Similarly, since EFR loads events stored in a *.txt* format, we modified its data loader to support the *.h5* format used natively by DSEC sequences. We also use our own code to compute metrics. These circumstances affect the reliability of our work; we seek to mitigate this by providing all supplementary code used alongside our filtering algorithm itself.

4.3 Filtering

Just like EFR, our algorithm successfully removes the bulk of flickering events. Figure 9 demonstrates a decrease in the overwhelming number of flicker events below the streetlight in the sequence *zurich_city_10_a*. However, as seen in Table 1, EFR removes 12.8% more events than our algorithm. The images suggest that our algorithm struggles at the boundary of the flickering blob; this faltering is likely caused by a quickly changing flickering intensity at this spatial region over time due to motion. This is an example of a discretization problem caused by our algorithm dividing event space into $8px \times 8px$ cells, with related artifacts evident at the boundaries between cells containing starkly different flickering intensities. It seems that EFR's per-pixel filtering is more effective on large textureless surfaces such as the street, since the distribution of events upon it appears near-uniform spatially due to the absence of edges. Naturally, spatial awareness has no reason to be beneficial in this case, and can only lead to discretization problems. However, EFR filters the flicker-free sequence *zurich_city_02_d* more aggressively, removing almost twice as many events as our algorithm. Overall, our algorithm satisfies its most basic expectations; while our removal of bulk flickers is not as effective as EFR's, we are less likely to mischaracterize motion events as caused by flickering.

A big downside of both filtering methods is that edges, particularly minuscule ones, lose much of their definition. Concerningly, this appears to be a consequence of the algorithms working as intended. Given the dark ambient lighting, the flickering street light is a dominant part of the total illumination received by many edges. Removing flickering events from them thus means removing much information

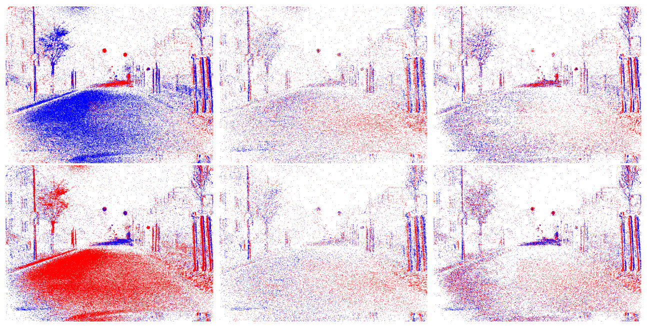
Figure 9: Filtering removes most flickering events. Three pairs of subsequent frames ($5ms$ apart) at the same timestamp, constructed using event data of the sequence *zurich_city_10_a*. **Left:** No filtering. **Center:** EFR. **Right:** Our filtering.

Filtering	Sequence	Event count	
		Count	*Reduction*
None	*zurich_city_10_a*	$1.000 \cdot 10^9$	n/a
	zurich_city_02_d	$0.795 \cdot 10^9$	n/a
EFR	*zurich_city_10_a*	$0.368 \cdot 10^9$	63.2%
	zurich_city_02_d	$0.506 \cdot 10^9$	36.4%
Ours	*zurich_city_10_a*	$0.496 \cdot 10^9$	50.4%
	zurich_city_02_d	$0.640 \cdot 10^9$	19.5%

Table 1: The decrease in event count as a result of filtering.

about their presence. As seen in Figure 10, our algorithm reduces the thickness of edges by discarding events outside of the dense central region, whereas EFR leaves thick but noisy edges by removing events evenly. As anticipated, movement results in EFR filtering adjacent pixels inconsistently. The distinction showcases the effect of recognizing the spatial coherence of edges, though it is unclear whether our algorithm's behaviour is better due to its reduction of edge thickness. Both algorithms show a clearly detrimental effect to small edges, which produce few motion events. This suggests that flickering may actually highlight edges by repeatedly creating events upon them at a much greater pace than motion under constant lighting would. It may also be the case that the intense flickering obstructs the creation of motion events. In either case, it is clearly evidenced that filtering removes some useful information about the scene's geometric features.

4.4 Optical Flow Estimation

The most striking finding, presented in Table 2, is that filtering hardly affects optical flow estimation accuracy in any case. The most significant change is on the sequence *zurich_city_10_a* when using E-RAFT, with the $3PE\%$ metric increasing by 27-30%, though this is likely because of its small absolute value. We observe no cases where filtering increases one of the two metrics but decreases the other. Filtering always hurts E-RAFT accuracy whereas TamingCM sees some improvements, though the changes are $\leq 5\%$ for either metric. Curiously, the flicker-free *zurich_city_02_d* sequence shows mild improvements despite losing non-flicker events, implying that any high-frequency variations may be detrimental. One of the few consistent results is that our filtering

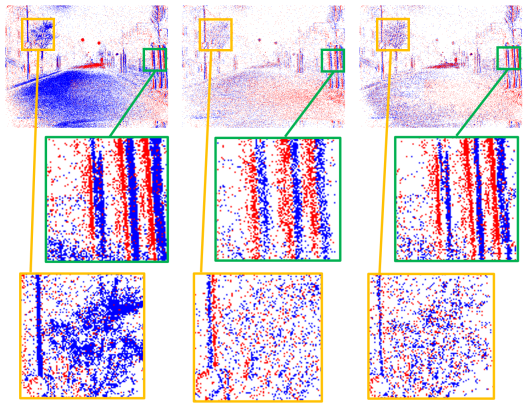
Figure 10: De-flickering causes a harmful reduction to the clarity of edges. The fine tree branches are nearly eliminated, the tree support structure is diversely affected. **Left:** No filtering. **Center:** EFR. **Right:** Our filtering.

gives better results than EFR when using E-RAFT, whereas the opposite is true for TamingCM. It must be remarked that TamingCM's results are much worse than E-RAFT's; it is clearly easier to increase accuracy (or not decrease it as much) the worse it is initially. Overall, neither our algorithm nor EFR meaningfully affects optical flow estimation despite removing a significant portion of events. Both methods' benchmarks are hardly different and do not strictly favour either algorithm; we cannot conclude which one is better.

Filtering	**TamingCM**		**E-RAFT**	
	EPE↓	*3PE%*↓	*EPE*↓	*3PE%*↓
None	3.36	34.4	0.44	1.05
EFR	3.22	33.6	0.63	1.33
Ours	3.45	36.1	0.54	1.31

(a) *zurich_city_10_a* results (heavy flickering)

Filtering	**TamingCM**		**E-RAFT**	
	EPE↓	*3PE%*↓	*EPE*↓	*3PE%*↓
None	3.25	25.7	0.72	2.98
EFR	3.17	24.6	0.88	3.88
Ours	3.20	25.5	0.75	3.12

(b) *zurich_city_02_d* results (no flickering)

Table 2: Benchmarked optical flow estimation metrics. **a)** Results obtained on the sequence *zurich_city_10_a*. **b)** Results obtained on the sequence *zurich_city_02_d*.

A big limitation of the results from Table 2 is that ground truth optical flow vectors are sparse. Due to the peculiarities of the process used to obtain them, they are generally only available for pixels corresponding to edges as opposed to textureless surfaces [6]. The consequences are illustrated in Figure 11, showing how ground truth optical flow vectors are concentrated at the scene's sides as opposed to the street. As a result, we could not quantitatively evaluate filtering on the street below the lamps, where we have observed the most dominant desirable elimination of flicker events. Another explanation for the neutrality of the results is that TamingCM, and to a larger extent E-RAFT, may both be overfitted to the original sequences' overwhelmingly forward-facing motion [10]. Upon filtering, the sequences may become less familiar to both algorithms due to a reduced density of event clusters. These effects imply that to a large extent, quantitative evaluation is insufficient to conclusively evaluate the effects of filtering.

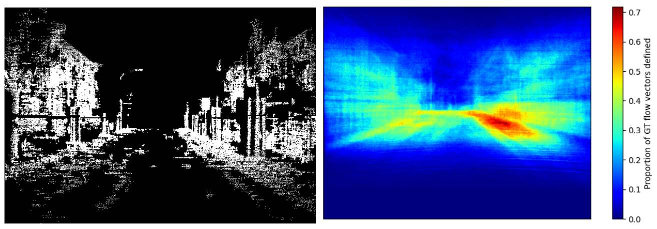

Figure 11: The partial availability of ground truth flow vectors on the first ground truth frame of the sequence *zurich_city_10_a*. **Left:** A frame from the sequence; pixels for which GT is defined are white. **Right:** The proportion of frames for which a GT flow vector is defined at each pixel. Both frames are rectified.

It is worth noting that each entry in Table 2 is computed as the average of comparing around 10^7 predicted flow vectors with their ground truth counterparts. Naturally, much information about the spatiotemporal distribution of errors is lost. One interesting temporal observation is that TamingCM's (unlike E-RAFT's) predicted flow vectors are consistently shorter than the ground truth's, as seen in Figure 12. As for many CM-based approaches [10], recognizing the correct speed of fast motion appears to challenge TamingCM. Filtering further worsens this problem, which is likely caused by edge thinning (Figure 10) since fewer events generally indicate slower motion. On the other hand, improvements to TamingCM's behaviour caused by filtering are clearly visible when analyzed spatially. As seen in Figure 13, the event cloud formed below the streetlight (Figure 9) results in TamingCM predicting optical flow at a direction roughly perpendicular to the true motion. Eliminating these events results in zero flow predicted, which while also incorrect, is unsurprising given the noise-like cloud remaining on the flat textureless street. Flow predicted by TamingCM is also more intricate, possibly explaining why its accuracy is worse when using our filtering (Table 2) - we reshape edges more strongly than EFR. E-RAFT still correctly predicts flow on the street irrespective of the (lack of) information present before and after filtering, which implies overfitting. Overall, a valuable takeaway is that TamingCM (unlike E-RAFT) reacts as expected to the identified positive and negative aspects of filtering.

5 Responsible Research

We go to a great extent to maximize the reproducibility of our research. We provide all of our code and the files from other projects we modified to facilitate our experiments. We have maintained as many variables as possible equal to the original authors' default settings, all deviations are documented both in this paper and the public repository. The code written by

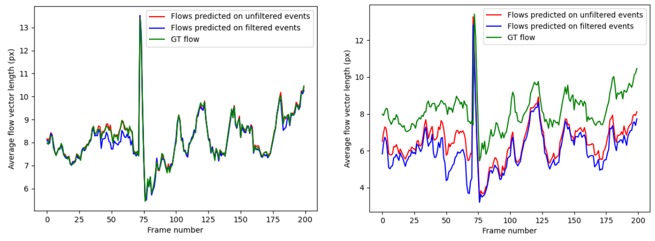

Figure 12: The average lengths of predicted and ground truth flow vectors per frame of the sequence *zurich_city_10_a*. **Left:** The case of E-RAFT. **Right:** The case of TamingCM. Only the first 200 frames are used for clarity of the figure.

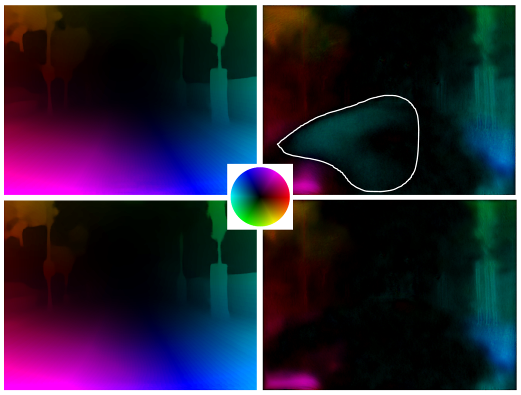

Figure 13: Visualizations of optical flow prediction frames at the same timestamp. **Center:** The used encoding of flow vector directions as color; a color at the circle's edge indicates a flow vector of magnitude 90px. **Top left:** E-RAFT, no filtering. **Bottom left:** E-RAFT, our filtering. **Top right:** TamingCM, no filtering; a region of erroneous flow is outlined in white. **Bottom right:** TamingCM, our filtering; filtering eliminates the erroneous flow predicted.

us is minimal and well-documented to permit simple analysis, tweaking, and reproduction of results. All of our used datasets are publicly available, figures are labeled such that the data used to generate them can be easily extracted. Our implementation slightly randomizes which particular events to delete, though it is unlikely that this effect leads to significantly different results.

With the tremendous amount of predicted flow vectors obtained, it is possible to intentionally choose unusual ways of analyzing them that would illustrate a disproportionately positive outcome. This is particularly enticing given that the near-universally used quantitative optical flow evaluation methods (Table 2) show neutral results. However, we firmly believe that this evaluation discards much useful information about the effects of filtering due to the unfavourable positions where ground truth is defined. As such, it is misleading to not provide a qualitative analysis of what goes wrong. The particular examples in figures 12&13 were chosen to showcase how the main identified positive and negative aspects of filtering translate into optical flow estimation. To avoid cherry picking favourable frames, we use ones at the same timestamp in all of section 4's figures (9, 10, 11 (left), 13). Given the difficult-to-compare results we obtained (many of which are qualitative), we avoid making definitive conclusions about the properties of filtering. We hope that our observations will lead to future interest; we believe this is appropriate given the virtually unexamined state of this paper's niche subject.

An important ethical concern stemming from the development of event cameras is that their various properties make them highly suitable for surveillance tasks. To disable such a system or simply hurt its performance, intentionally generated flickering could be used. Therefore, removing flickers may represent a conflict of interest between owners of the surveillance system and those it is used to monitor. This problem is linked to the broader societal issue of exchanging personal freedoms for security. Nevertheless, we believe that a significant number of proposed sincere uses of event cameras may be held back by flickering, such as self-driving cars at night. Nevertheless, it must be kept in mind that the malicious applicability of event cameras (and/or flicker removal) may prove to outweigh the benefits once more research becomes available.

6 Discussion

We obtain a more nuanced view of flickering by analyzing how removing it affects the scene at diverse regions. Flicker events are most visible in bulk on large edgeless surfaces, but they are present in non-negligible quantities on edges as well (Figure 10). Filtering successfully removes flickers on the flat street below the lamps (Figure 9), though the benefits of this are scant. That is because what remains is mostly noise, not easier-to-discern motion. On the other hand, events produced by edges form well-defined geometric clouds evolving in measurable ways over time. While removing flickers from edges may seem beneficial, it has the side effect of reducing the edge's clarity. This means, to a large extent, removing information about an edge's presence that may be useful to an appropriately designed optical flow estimation algorithm. The effect of thinning an edge is similar to a decrease in illumination intensity or motion speed, both of which are correlated with the density of events corresponding to an edge (Figure 12). Overall, flicker removal increases the uniformity of edges' event polarities, but hurts the observability of the scene's well-defined geometrical features.

The only clearly identified advantage of filtering is that it greatly reduces the number of events in a dataset at a minimal impact on accuracy. This is true for both our algorithm and for EFR. However, their subtle differences result in our filtering giving better performance when using E-RAFT, unlike TamingCM, which prefers EFR (Table 2). Filtering seems to always decrease E-RAFT's performance, so one might assume that our algorithm is better because it filters out fewer events (Table 1). This is unlikely, however, as many of the extra events removed by EFR are on the street, where GT flow is unavailable. A more probable explanation is that E-RAFT is less sensitive to our filtering's reduction of edge thickness (Figure 10) since the flow it predicts appears less spatially intricate than TamingCM's (Figure 13). Quantifying these effects objectively would require correcting our work's limitations: using datasets designed to isolate flickering, retraining models on filtered datasets, or using synthetic datasets with complete ground truth flow. Regardless, all evidence supports rejecting our hypotheses - the advantages of space-aware filtering are too situational, and are further undermined by de-flickering barely impacting optical flow estimation.

7 Conclusion & Future Work

Event cameras are novel sensors whose properties motivate their use for the motion estimation problem of optical flow but also incur susceptibility to flickering lights. The goal of this project was to enhance optical flow estimation accuracy by integrating a spatial component into a de-flicker filter, which we hypothesized to increase robustness to motion. This is in contrast to EFR, the only existing event de-flicker algorithm that filters purely temporally on a per-pixel basis. We have designed and implemented a solution that measures the polarity sum of events in spatiotemporal regions and deletes events detected as constituting intense fluctuations. Qualitatively, edges de-flickered by our method are thinner and denser, whereas EFR's are thick but sparse; neither alternative is obviously superior. Also, EFR is more likely to delete events regardless of whether they were caused by flickering. Quantitative evidence shows that both algorithms' filtering removes $50 - 65\%$ of events from a flicker-heavy dataset. However, this barely impacts subsequently computed optical flow accuracy, even using a Contrast Maximization-based estimator known to suffer from flickering. We conclude that our algorithm is on par with EFR; the advantages of space-aware filtering are niche and undermined by de-flickering hardly affecting optical flow accuracy in the first place.

Our qualitative analysis has revealed that removing flickers does not always positively affect an event set's usefulness. A positive aspect is that removing events on edgeless surfaces can reduce erroneous flow predictions at those regions, but removing flickers on protruding edges harms their clarity. The adverse effect on edges appears more relevant, as edges are more useful for identifying the direction of motion than textureless surfaces. We posit that flickering produces useful information on edges by repeatedly highlighting their presence, whereas they may appear faint under slow motion or constant lighting. Therefore, eliminating events caused by flickering may not be a sound way to distill motion information.

Instead of deleting flickers, we suggest that optical flow estimators incorporate these artifacts into the process of deducing the direction of motion. One option is to simply ignore event polarities. The reason is that a moving edge forms a planar structure in event space; the presence of flickering would result in oscillating event polarities inside it but would not distort the plane's general shape. One may consider using a variation to Contrast Maximization where event counts are accumulated along the warp trajectory as opposed to polarities. A different lucrative direction to investigate is removing any high-frequency event patterns, motivated by our finding that filtering a flicker-free sequence improves accuracy. This result motivates using de-flicker algorithms as general low-pass filters, reducing event dataset sizes at a minimal penalty to accuracy while improving computation speed.

References

[1] Dsec-flow: Optical flow benchmark. https://dsec.ifi.uzh.ch/uzh/dsec-flow-optical-flow-benchmark/. Accessed: 2024-05-27.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56:221–255, 2004.

[3] Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022.

[5] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021.

[7] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *2021 International Conference on 3D Vision (3DV)*, pages 197–206. IEEE, 2021.

[8] Federico Paredes-Vallés, Kirk Y. W. Scheper, Christophe De Wagter, and Guido C. H. E. de Croon. Taming contrast maximization for learning sequential, low-latency, event-based optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9695–9705, October 2023.

[9] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Event collapse in contrast maximization frameworks. *Sensors*, 22(14):5190, 2022.

[10] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of event-based optical flow. In *European Conference on Computer Vision*, pages 628–645. Springer, 2022.

[11] Ziwei Wang, Dingran Yuan, Yonhon Ng, and Robert Mahony. A linear comb filter for event flicker removal. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 398–404. IEEE, 2022.

[12] Yilun Wu, Federico Paredes-Vallés, and Guido C. H. E. de Croon. Lightweight event-based optical flow estimation via iterative deblurring. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'24)*, May 2024. To Appear.

[13] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

[14] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018.