Real-time high speed motion prediction using fast aperture-robust event-driven visual flow

Himanshu Akolkar, Sio Hoi Ieng, Ryad Benosman

Real-Time High Speed Motion Prediction Using Fast Aperture-Robust Event-Driven Visual Flow

Himanshu Akolkar, Sio-Hoi Ieng, and Ryad Benosman

Abstract—Optical flow is a crucial component of the feature space for early visual processing of dynamic scenes especially in new applications such as self-driving vehicles, drones and autonomous robots. The dynamic vision sensors are well suited for such applications because of their asynchronous, sparse and temporally precise representation of the visual dynamics. Many algorithms proposed for computing visual flow for these sensors suffer from the aperture problem as the direction of the estimated flow is governed by the curvature of the object rather than the true motion direction. Some methods that do overcome this problem by temporal windowing under-utilize the true precise temporal nature of the dynamic sensors. In this paper, we propose a novel multi-scale plane fitting based visual flow algorithm that is robust to the aperture problem and also computationally fast and efficient. Our algorithm performs well in many scenarios ranging from fixed camera recording simple geometric shapes to real world scenarios such as camera mounted on a moving car and can successfully perform event-by-event motion estimation of objects in the scene to allow for predictions of upto 500 ms i.e., equivalent to 10 to 25 frames with traditional cameras.

Index Terms—Event driven, neuromorphic, optical flow, motion prediction

1 Introduction

Optical flow is the measure of motion of an object projected on to the image plane of a camera. It is one of the fundamental steps needed for understanding a dynamic visual scene and has taken an even important role with newer applications such as autonomous driving vehicles [16], drones, action perception during user interactions [14] in robots and traditional applications like video editing [15] and stabilization. Because the visual sensing has traditionally been based on image acquisition at fixed time intervals, the computation of optical flow has been based on finding features that move across two or more consecutive images. Since the intensity of light received on the sensor is the most basic feature, the first principle approach for measurement of optical flow is given by the 'brightness constancy assumption' that assumes that the brightness of an object moving across the camera remains constant over short internal of time. Ideally this time interval should be infinitesimal, but practically, for the traditional cameras, this means the time between two recorded frames. This constant instantaneous brightness assumption forms the basis for the earliest algorithms such as those proposed by Horn and Schunk [4] and the Lucas-Kanade (LK) algorithm [5]. This has been further expanded to 'constant feature assumption' where complex features or descriptors are extracted [6] and tracked over multiple spatial scales [19]. With the advancements in convolution and deep neural networks, a number of new algorithms using these approaches have been proposed to compute visual flow [17], [18], [20]. Some of these methods even propose tackling optical flow computation as a learning problem [21]. While these approaches intend to achieve high accuracy using the improving computational power of GPUs and FPGAs, the fundamental problem of fast sensing and image processing still poses a hinderance towards using such techniques as part of a larger perceptive autonomous system.

The new generation of dynamic visual sensors [1], [2], [3] might be able to fill in this niche application space by virtue of their fast, accurate sensing of light with high temporal precision. In this paper, we propose an algorithm designed for use with one such type of sensor [3]. Event-driven sensors have evolved over the last few years as possible successors to frame based classical cameras, especially for visual sensing in research areas that require high precision over a large temporal dynamics range like robotics [22], [23], [24], [25], autonomous vehicles [26] and navigation in drones [27]. As these sensors provide precise motion information due to the inherent design of the pixels, they are ideal for fast visual flow computations.

A number of methods have been proposed to compute visual flow using event based sensors. As events in the event-driven sensors are essentially encoding the light intensity captured by the pixels, algorithms based on the original image based Lucas-Kanade method have been proposed [7]. While these event-driven derivatives are fast, they cannot achieve the same accuracies as the frame-based variants due to the loss of information in conversion from intensity to events.

Several algorithms are designed specifically to take advantage of the temporal nature of event-driven paradigm

- *Himanshu Akolkar is with the University of Pittsburgh, Pittsburgh, PA 15260 USA. E-mail: akolkar@pitt.edu.*
- *Sio-Hoi Ieng is with the Sorbonne Universite, INSERM, CNRS, Institut de la Vision, 75012 Paris, France. E-mail: sio-hoi.ieng@upmc.fr.*
- *Ryad Benosman is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, the University of Pittsburgh, Pittsburgh, PA 15260, USA, and also with the Sorbonne Universite, INSERM, CNRS, Institut de la Vision, 75012 Paris, France. E-mail: ryad.benosman@upmc.fr.*

Manuscript received 15 Aug. 2019; revised 15 June 2020; accepted 9 July 2020. Date of publication 20 July 2020; date of current version 3 Dec. 2021.
(Corresponding author: Himanshu Akolkar.)
Recommended for acceptance by V. Lepetit.
Digital Object Identifier no. 10.1109/TPAMI.2020.3010468

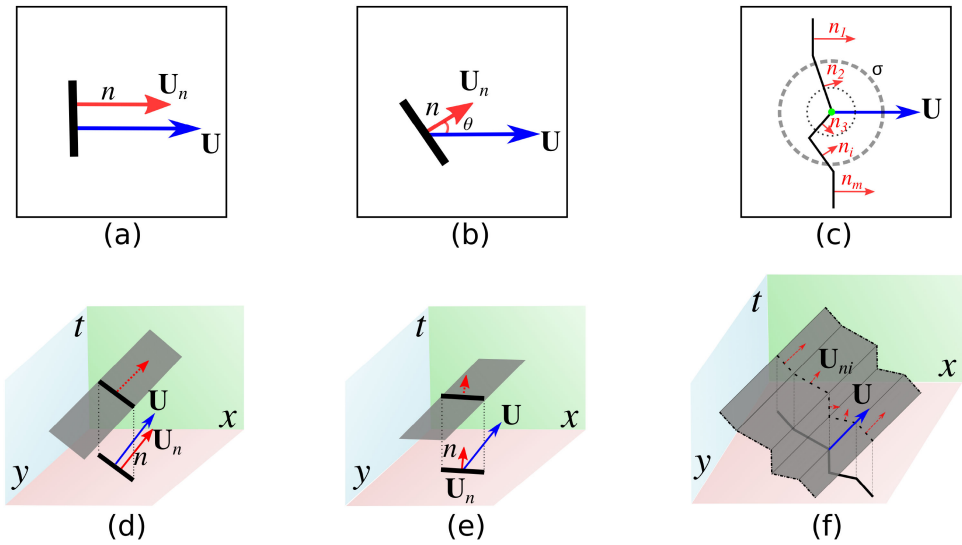


Fig. 1. *(a)* and *(b)* show an oriented edge moving across the sensor in true direction $\mathbf{U}$ and the predicted local flow $\mathbf{U}^T\mathbf{n}$ by fitting the plane over events in [x, y, t] space as in *(d)* and *(e)*. The magnitude of the normal velocity component estimated by the plane fitting method is related to the orientation of the edge and true motion direction as $\mathbf{U}^T\mathbf{n} = |\mathbf{U}|\cos(\theta)$. This relationship can be extended to a larger complex shaped object by linearizing it using multiple small edges *(c)* over small spatial region and performing plane fitting over each local edge *(f)*. *(c)* The best estimate of true flow direction can be estimated by finding the correct spatial size σ corresponding to the maximum mean magnitude $|\overline{\mathbf{U}_n}|$.

[8], [9]. These algorithms use the spatio-temporal structure of events to estimate the flow by fitting a surface (usually a plane) and compute the normal of this surface as flow estimate. These algorithms maybe classified under the label of 'plane-fitting algorithms'. While these algorithms have improved accuracy of event flow, they are limited to computations of local dense flow. Further, the flow obtained is always computed as orthogonal to the edge irrespective of the direction of true motion. Thus, the flow computation is susceptible to the gradient of the edge. This problem is referred to as the aperture problem. The only way to tackle the aperture problem with a traditional plane-fitting method is to increase the size of the spatial neighborhood around the events when fitting the plane but this can lead to errors as the true size of the object is unknown and the shape of object might not remain linear.

A recent algorithm has been able to avoid this problem using constrained statistical properties of the object but it is computationally too intense to be used in real time and is only valid for object with closed form [10]. Another recent method for computing event-driven visual flow uses a spatio-temporal window of events and performs histogram matching of the event clusters to estimate the direction and speed of object. Thus, the current state-of-the-art algorithms lose the temporal dynamics of the input sensor events as they require pooling of events over a temporal and spatial window to avoid aperture problem.

Here we propose a new event-driven algorithm to solve aperture problem using multi-scale spatial pooling that uses the local erroneous flows computed at the lowest scales and corrects their direction towards the true direction of motion of the object. We mathematically prove that because of the specific properties of the plane fitting algorithms, pooling the fast but erroneous local flows over an appropriate spatial scale can correctly estimate the real direction of the object. Further, the estimation of this spatial scale can be computed at every event independently without any a-priori knowledge about the shape and size of the objects in the scene and is independent of any global motion of the camera. The proposed algorithm can perform in myriad of scenarios. Finally, this flow rectification allows us to perform very low-level event predictions i.e., when and where should new events appear according to the observations. We show via experiments that we can estimate on the fly, locations and velocities of moving objects of up to 500 ms ahead in the future. Such prediction can be implemented for solving visual tasks such as collision avoidance and tracking.

2 Methods

The algorithm proposed in this paper uses multi-scale pooling found in biological visual system in higher animals for hierarchical object recognition. The basic idea here is to perform local, fast flow measurements which might be incorrect in their direction estimations but are relatively reliable in amplitude estimates and then correct the direction estimates using global amplitude information.

2.1 Multiscale Pooling

Fig. 1 shows the principle idea motivating the correction procedure explained in the next section. Let us assume the most ideal case suited for the plane fitting method: a single bar moving in front of the camera generating a perfect event plane in the [x, y, t] space. If the bar is oriented orthogonal to its direction of motion (Fig. 1a), the estimate of the velocity computed using the plane fitting method [8] (Fig. 1d) would be equal to the true velocity $\mathbf{U}$. But, if the bar is now rotated (b) by an angle θ, the velocity estimate of the flow from plane fitting is $\mathbf{U}_n = \mathbf{U}^T\mathbf{n}.\mathbf{n}$, $\mathbf{n}$ being the unit normal to the bar. The signed magnitude of this flow can be given by:

$$\mathbf{U}^T\mathbf{n} = |\mathbf{U}|\cos(\theta). \tag{1}$$

This shows that the plane-fitting based estimated flow is equal to the true direction of motion when the magnitude of

the estimate is maximum, i.e., the cosine is maximum in Eq. (1).

It is important to note that the normal $\mathbf{n}$, without additional assumption, can have two directions - namely either one of the two directions along the line orthogonal to the bar. However, if we are considering the temporal surface defined in [8] (as shown in Figs. 1d, 1e, and 1f) as a bi-dimensional function t of (x, y), where the gradient of t allows us to define $\mathbf{n}$ as its unit direction vector then t is always increasing in the direction of the motion (i.e., the directional derivative of t along $\mathbf{U}$ is increasing) and we always have $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ or equivalently $\mathbf{U}^t.\mathbf{n} \geq 0$.

We can generalize the observation in Eq. (1) to more complex objects using this property of the plane-fitting flow computation. Figs. 1c and 1f shows one such example case: let us consider a contour of a random shape moving with velocity $\mathbf{U}$. We can approximate this shape as a set of line segments. For each pixel/event of each segment, the plane fitting method is estimating $\mathbf{U}_n$. If we consider a spatial neighborhood σ around a random pixel (example : green dot in (c)) - for which we have estimated its normal velocity, the mean speed (i.e., the amplitude of the mean velocity) computed within σ is defined as

$$|\overline{\mathbf{U}_n}| = \frac{\sum_{i\in\sigma} K_i \mathbf{U}^T \mathbf{n_i}}{\sum_{i\in\sigma} K_i}, \tag{2}$$

where K_i is the length of the ith segment in pixels within σ and with the assumption that all the pixels are contributing in the mean flow estimation.

If we assume that within this spatial neighborhood there lies a line segment j such that it is oriented relatively closest to the true motion direction (i.e., θ is minimal and ideally $\theta_j = 0$ when it is oriented orthogonal to the true velocity), we can find an upper bound for $|\overline{\mathbf{U}_n}|$

$$|\overline{\mathbf{U}_n}| \leq \frac{\sum_{i\in\sigma} K_i \mathbf{U}^T \mathbf{n_j}}{\sum_{i\in\sigma} K_i} = \mathbf{U}^T \mathbf{n_j} = U_m. \tag{3}$$

Since the local mean speed is upper bounded by the amplitude of the velocity that is "most" colinear to $\mathbf{U}$, the larger the $|\overline{\mathbf{U}_n}|$ we get from a given σ, the closer we are to $\mathbf{U}$ i.e., the σ leading to the largest $|\overline{\mathbf{U}_n}|$ is the "right spatial scale" for which $|\mathbf{U} - \overline{\mathbf{U}_n}|$ is minimized. As we do not know the true velocity $\mathbf{U}$, U_m becomes our next best reference for $\mathbf{U}$.

According to this observation, for a given flow estimate, we define the problem of correction as the minimization problem of finding the neighborhood scale, σ, for which the cluster of flow estimates whose mean magnitude is close to the theoretical maximum $|\mathbf{U}| \approx U_m$ as described previously. Thus, for given neighborhood σ, we define the error function as

$$E_\sigma = \frac{\sum_{i\in\sigma} K_i (U_m - \mathbf{U}^T \mathbf{n}_i)}{\sum_{i\in\sigma} K_i}. \tag{4}$$

We then have according to (3)

$$E_\sigma = U_m - |\overline{\mathbf{U}_n}|. \tag{5}$$

Since $0 \leq |\mathbf{U}_n| \leq U_m$, the problem of finding the right σ is equivalent to the minimization problem

$$\begin{aligned} \arg\min_\sigma(E) &= \arg\min_\sigma(U_m - |\overline{\mathbf{U}_n}|) \\ &\equiv \arg\max_\sigma(|\overline{\mathbf{U}_n}|). \\ &\equiv \arg\min_\sigma(|\overline{\boldsymbol{\theta}}|). \end{aligned} \tag{6}$$

The above equations show that finding the scale with maximum mean magnitude is equivalent to finding the scale which best estimates the direction of true global flow. Eqs. (5) and (6) combine to give the new estimated flow, magnitude and direction, from the optimal spatial scale. Thus, we only require to compute the flow over the smallest scale once, and perform the above maximization over larger spatial scales to get the best estimate of the true global motion direction. Since, the above method is an optimization problem, the resulting estimate of the direction depends on the available line segments directions in the scene. While the method can give us the "true" direction of the object, in non-ideal conditions when the line segment that is orthogonal to the true direction is missing, the method can only provide the closest best estimate of the true direction. Further, as the estimated flow is given by the average local flows in the optimal spatial scale, it is possible that the flow is slightly biased by the flow values from the incorrect orientations. Since the incorrect flow values decrease with cosine of the angle, this bias is generally very small - nonetheless, this means that rather than getting the exact true flow, we will get very small errors in the flow. The proposed algorithm can therefore be divided into three steps. First, we compute local flow for each event using plane fitting. Second, we search for a spatial scale for which the mean magnitude of these local flows is maximized. Third, we calculate the mean direction for the flows in this scale and assign the direction to all the local flow events within this scale.

Interestingly, while we define Eq. (1), in regards to the plane fitting algorithm, this property holds true for many other methods of computing local flow such as the Lucas-Kanade flow [5] or their event based versions [7]. This means that while this paper in the following section uses plane-fitting to compute and correct local flow, the proof below shows that any existing flow methods that adhere to the relationship in Eq. (1) can be corrected using the multi-scale correction method.

3 Algorithm Implementation

The steps involved in the implementation of the flow are described in Algorithm 1. The local flow was computed using an iterative implementation of the plane fitting flow as in [12]. Some minor changes are introduced to the original implementation in [12] to improve performance. First, to improve the accuracy of the flow and remove noise, we add an error correction step to ensure better accuracy of the plane fitting by computing the number of inliers (events that are within a certain distance from the fitted plane). If the number of inliers is more than half the total points used to fit the plane, we consider the fitting to be good and the flow estimate to be reliable. This improves overall efficiency and noise robustness as the rectification is only performed on valid flow events.

To further avoid older events from corrupting flow estimates, we added a temporal history limit such that the correction was performed using events that occurred within a

TABLE 1
Algorithm Parameters

Parameter	Value
Local flow	
Filter size N	5 pixels
Inlier percentage	50%
Rectification	
Spatial range σ	0 to 100 pixels in steps of 10
Temporal limit t_{past}	5 msec

certain time (t_{past}) from current event. Table 1 lists the parameters values used to estimate flow for datasets used the experiments and results in Section 4.

Algorithm 1. Multi-Scale Aperture Robust Optical Flow

for each event x, y, t **do**
  1. COMPUTE LOCAL FLOW (EDL):
  Apply the plane fitting [8] to estimate the plane parameters [a, b, c] within a neighborhood (5x5) of (x, y, t).
  Set $\hat{U} = ||(a, b)||$ and Inliers_count = 0
  $\hat{z} = \sqrt{a^2 + b^2}$
  for each event (x_i, y_i, t_i) in neighborhood N=5 **do**
    $\hat{t} = (ax_i - x) + (by_i - y)$
    if $|t_i - \hat{t}| < \frac{\hat{z}}{2}$ **then**
      Inliers_count = Inliers_count+1
    end if
  end for
  if Inliers_count $\geq 0.5 * N^2$ **then**
    Set $\theta = arctan(a/b)$ and $\mathbf{U}_n = (\hat{U}, \theta)^T$
  else
    $\mathbf{U}_n = (0, 0)^T$.
  end if
  2. MULTI-SPATIAL SCALE MAX-POOLING:
  Define $S = \{\sigma_k\}$, the set of neighborhoods, centered on (x, y, t), σ_k with increasing radius and δ t(σ_k) $\leq t_{past}$ (t_{past} is temporal cut-off delta)
  if $\mathbf{U}_n \neq (0, 0)^T$ **then**
    for each $\sigma_k \in S$ **then**
      $\overline{\mathbf{U}}_{n,\sigma_k} = \underset{j \in \sigma_k}{\text{mean}}(\mathbf{U}_{n_j}) = (\overline{\hat{U}}_k, \overline{\theta}_k)^T$
    end for
    $\sigma_{max} = \underset{\sigma_k \in S}{\arg\max}(\overline{\hat{U}}_k)$
  end if
  3. UPDATE FLOW:
  Flow (x,y) = $= \underset{j \in \sigma_{max}}{\text{mean}} \begin{pmatrix} \hat{U}_j \cos\theta_j \\ \hat{U}_j \sin\theta_j \end{pmatrix}$
end for
Refer Table 1 for parameter values.

4 Experiments and Results

The performance of the algorithm was measured in different scenarios to test its effectiveness over the plane fitting method. The results are divided into two sections - first, we show in four different scenarios how our algorithm corrects the direction errors over the plane fitting algorithm. In the later section we show how this corrected flow can be used to implement event-by-event predictions of moving objects. For the sake of brevity, in the rest of the text, we abbreviate the local plane fitting flow as Event Driven Local (EDL) flow while the corrected flow estimates using our algorithm as Aperture Robust MultiScale (ARMS) flow.

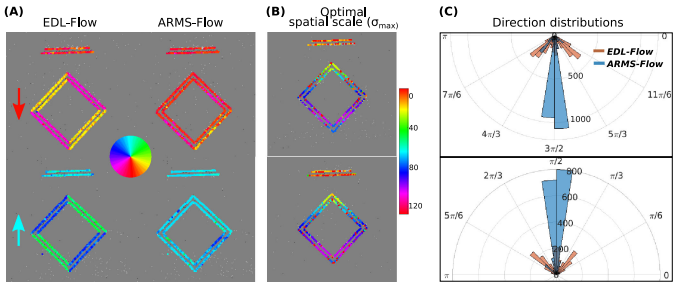


Fig. 2. The figure shows the output of the algorithm for trivial case of bars and squares moving up and down. (A) The direction of EDL flow estimates is normal to the edge orientations which is corrected by ARMS. (B) shows events color coded by the size of optimal window. (C) represents the direction distributions showing how the EDL gives three distinct peaks for each of the orientations which ARMS corrects towards a single peak representing vertical motion.

4.1 Flow Correction

4.1.1 Camera Fixed, Trivial Pattern

We used a simple geometric pattern of bars and squares moving up and down in front of the sensor. Fig. 2 A[EDL Flow] shows the flow computed using just plane fitting algorithm on a given slice of events. As evident from the figure, while most of the events have correct flow direction on the bars, the flow directions of the edges of the square are incorrectly pointing towards the normal of the edges. Fig. 2 A[ARMS Flow] shows the output of our algorithm. The directions of the edges are corrected uniformly towards the true direction of motion. The quantification of these results are shown in the histograms of Fig. 2[right]. The graphs in red show the distribution of directions (in radian) estimated by the plane fitting algorithm. The graph indicates tri-modal distribution for downward/upward ($\pi/2$, $3\pi/2$) and the directions along the normal to the edges ($\pi/4$, $3\pi/4$ for up and $5\pi/4$, $7\pi/4$) while the distribution of the corrected flow directions (blue) largely make up a single peak in the direction of real motion. Fig. 2 B shows the size of optimal scale detected by the correction step for each event. The events on bars have small spatial scale size as they represent the correct direction. For the pixels on the square, however, since the bar represents the best flow, the size of window increases as the events get farther away from the bar and a larger scale which would include the bar is needed to find the best direction. This also means what while the optimal scale sizes for the square are symmetric vertically, the presence of the bar means that the horizontal symmetry is lost. The size of the optimal spatial scales are still independent of direction of motion.

4.1.2 Camera Fixed, Multiple Objects

Next, we tested the robustness of the multi-scale pooling in case of more than one object moving in front of the camera. To do this, we recorded two simple objects (two squares) moving across the camera in opposite directions. We also have a stationary object in the scene that may lead to noisy events. The experiment shows that the spatial pooling is not affected by multiple objects and the algorithm can find the correct scales for each object independently. Further, when the objects cross each other close by, the algorithm is robust enough to recover the correct directions. Fig. 3 shows EDL and ARMS flow output for the two objects and the corresponding direction

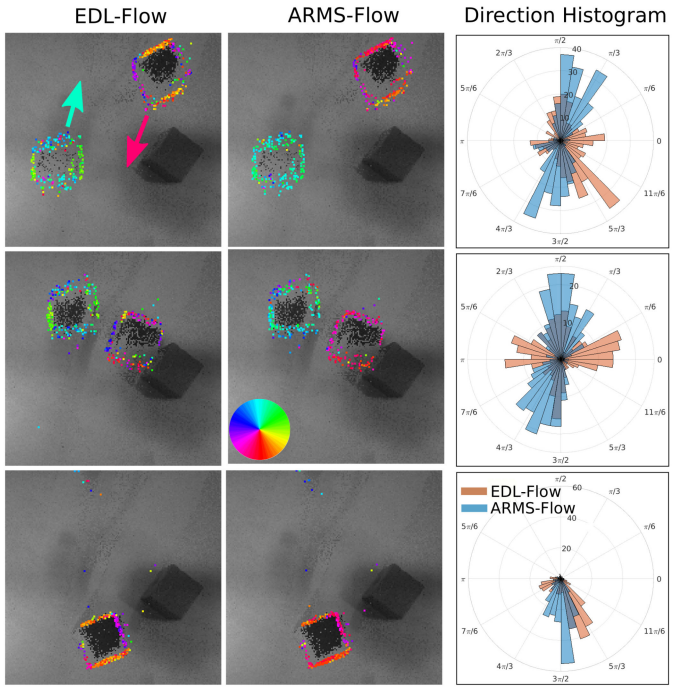


Fig. 3. Comparison of EDL and ARMS for two moving objects. The three rows show the direction outputs of the EDL and ARMS flow at three time points. The algorithm works well even when the two objects cross each other closely. Direction histograms show bimodal distribution from ARMS (blue) for the direction of the two individual shapes. The EDL flow (red) however leads to a larger variance and almost a uniform distribution. Even with only one object in the scene (bottom row), EDL flow gives rise to two modalities but ARMS gives a single peak at $3\pi/2$.

distribution of events over events in a time window of 100 ms. The left column shows the EDL outputs color coded by the direction of flow estimates. As expected, the estimated directions are normal to the edge directions for each of the objects, leading to an almost uniform distribution of event directions. This, is corrected by ARMS so that we get two distinct peaks in the direction histograms. As the objects collide and cross each other, the EDL becomes slightly worse and the peaks shift whereas the ARMS flow distribution remains invariant (Fig. 3 [middle row]). Finally, as the objects move further, and we only have one object, the distributions becomes similar to the one in Exp 4.1.1 with single moving object. Again, while EDL gives two peaks for each of the edge orientations, we get a single peak from ARMS indicating the global motion direction.

4.1.3 *Camera Fixed, Objects Occluding*

Next, we tested the robustness of the multi-scale pooling in case when more than one object moving in front of the camera overlap and occlude each other. We setup two pendulums of the same length and size but placed at different depths from the sensor. The pendulums were left to oscillate at out of phase positions such that while both the pendulums are visible to the sensor, there are moments when the two pendulum overlap each other. The setup and qualitative results are shown in Fig. 4. Figure shows the output of the ARMS flow for a sequence of about 200 msec during which the two objects overlap and then pass each other. We also show how the optimal spatial scale found by the correction step of the algorithm changes over time as different parts of the objects move over a given pixel represented by the black dot on each panel. The spatial scale is represented by the black rectangle. As the pendulums pass by, the optimal scale selected by the correction step changes depending on which parts of the pendulums are passing through the pixel. The ARMS flow provide the directions close to the actual directions when the pendulums are far apart as in the panels 0 to 10 msec. As the pendulums move closer, the flow direction of the rightmost edge of the smaller pendulum gets corrupted by the higher magnitudes of the larger pendulum. At 102 msec, panel we

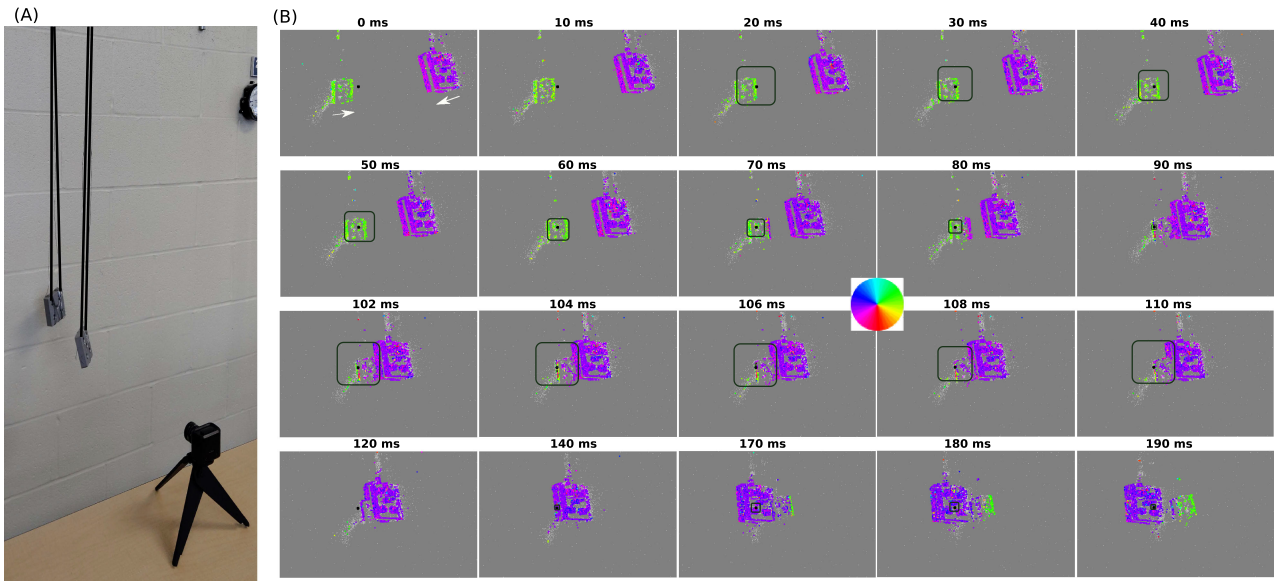


Fig. 4. Figure shows the setup (A) and ARMS flow directions (B) for a scenario with two pendulums of same length and size but at different depths from the sensor. The figure also shows, as black rectangle, the size of optimal window found by ARMS flow for a given pixel (black dot). We can see that as the two pendulums get close around 70 msec time point, the front edge of the smaller pendulum starts to get erroneous direction estimate due to the higher magnitudes of the closer pendulum. The direction estimates are quickly corrected though, as soon as the two pendulums start to move away (170-190 msec).

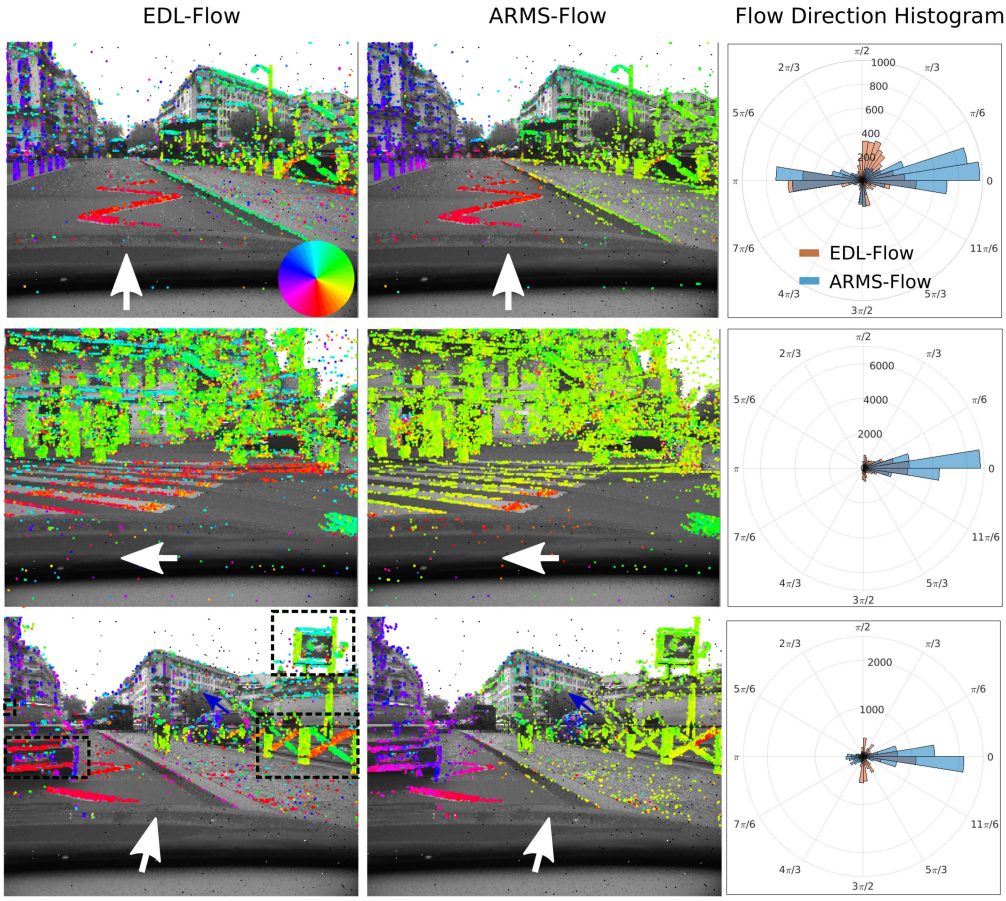


Fig. 5. Figure shows the flow directions for an ATIS mounted on a car moving straight ahead (top), taking a left turn (middle) and navigating around another car (bottom). EDL flow is normal to the edges on most events which is reflected in the histograms by the small incorrect peaks at $\pi/2$ and $3\pi/2$. ARMS-Flow corrects these local abnormalities giving rise to correct direction dependent flow reflected in the two distinct peaks during straight motion and a single large peak around $0\ deg$ when the car is turning left. The bottom row provides shows how well the ARMS flow works in a cluttered dynamic case. The black rectangles show the interesting regions in the scene where the normal directions are improved towards the true global flow while still maintaining the directions of independent moving object like the car on the right which has a relative motion indicated in the forward direction (blue arrow).

start to see the impact of the larger pendulum on the spatial scale detection. Even though the observed pixel is on the smaller pendulum, the higher magnitudes of the larger pendulum lead to larger spatial scales to be selected as optimal, as represented at 110 msec by the larger rectangle. Finally, as the two pendulums get farther apart at 170 msec, we can see that the flow for the leading edge of the smaller pendulum are quickly corrected and the correct flow values are recovered. This shows that while the algorithm can get affected instantaneously due to overlapping object, the error remains only for a very short duration of the overlap and can be quickly recovered.

4.1.4 Real World Scene - Camera Mounted on Moving Car

The flow rectification is also assessed through a real world scene in which the event-based camera was mounted on a car moving through traffic along the streets of Paris. The flow obtained from the algorithm corrects the local perpendicular flow to provide a better global flow especially when the car is making turns, where the whole scene should have the same global flow direction. The optical flow corrections can improve the flow directions when the car is turning, making all events predicting the apparent global direction of turn. Further, the combination of speed and flow directions can easily segment objects moving independently from the car. Fig. 5 shows the EDL and ARMS flow for different traffic conditions. The bottom row shows interesting points (marked by black rectangles) in the scene where the ARMS flow successfully corrects erroneous directions of the EDL. These show that the spatial scale estimation works correctly even in a cluttered environment and large motion events. Further, direction estimates of independent moving objects such as cars is not affected by the global motion.

4.2 DAVIS Dataset With Ground Truth

4.2.1 Application to the Event-Camera Dataset

To test the efficacy of our method on a benchmark dataset, we chose to implement it on events and images recorded with a DAVIS which also recorded motion of the camera using an inertial measurement unit (IMU) at 1,000 Hz [28]. This provides us with not only images of the scene but also the angular velocity of the camera as the scene is recorded. We implemented the ARMS flow on dynamic rotation scene where an office scene is recorded with camera primarily rotating around its axes. The recording involves different speeds of rotations. Fig. 8 shows the output flow directions

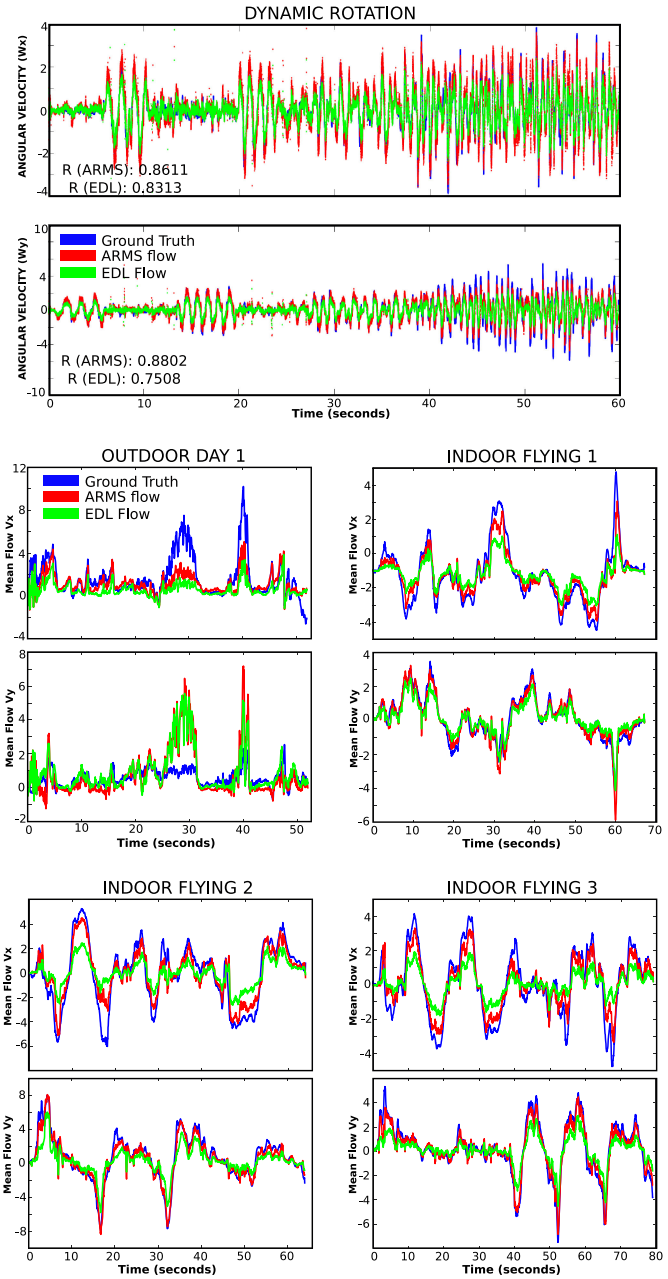


Fig. 6. Comparison of ARMS flow and EDL flow based velocity estimates against the ground truth velocities recorded with an IMU for the rotational dynamics data from DAVIS benchmark data and the Outdoor Driving and Indoor Flying conditions from the MVSEC data over the duration of the recordings.

for the recorded data. Fig. 6 shows the actual imu recordings of the angular velocities and the angular velocities estimated using the EDL and our flow. The graphs show that the predicted velocity can follow the real velocities well as indicated by the high correlation scores which also show improvements over the EDL flow predictions. The flow fails somewhat for the W_y when the camera moves at higher speeds as seen in the 40 to 60 second mark in bottom graph.

4.2.2 *Application to MVSEC Dataset*

Finally, we also have applied the optical flow rectification algorithm on the MVSEC dataset [13] as shown in Fig. 9.

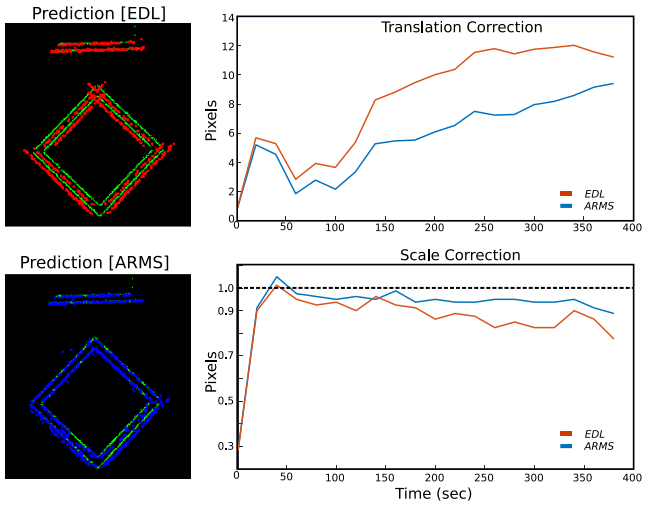


Fig. 7. The figure shows the predicted events based on EDL and ARMS flow at 250ms in future. Green dots indicate the actual future events while the red and blue dots indicate events predicted by the two flows. The scaling and translation error show how well the ARMS flow keeps the affinity of the object events. The ARMS flow has required scaling closer to 1 and translation error lower than the EDL flow error indicating that all events point to the true direction of motion.

Fig. 6 shows the ground truth velocities over the duration of recordings for the several conditions and the corresponding mean EDL amd ARMS flow. The MVSEC dataset ground truth combines several sensor data along with the framed

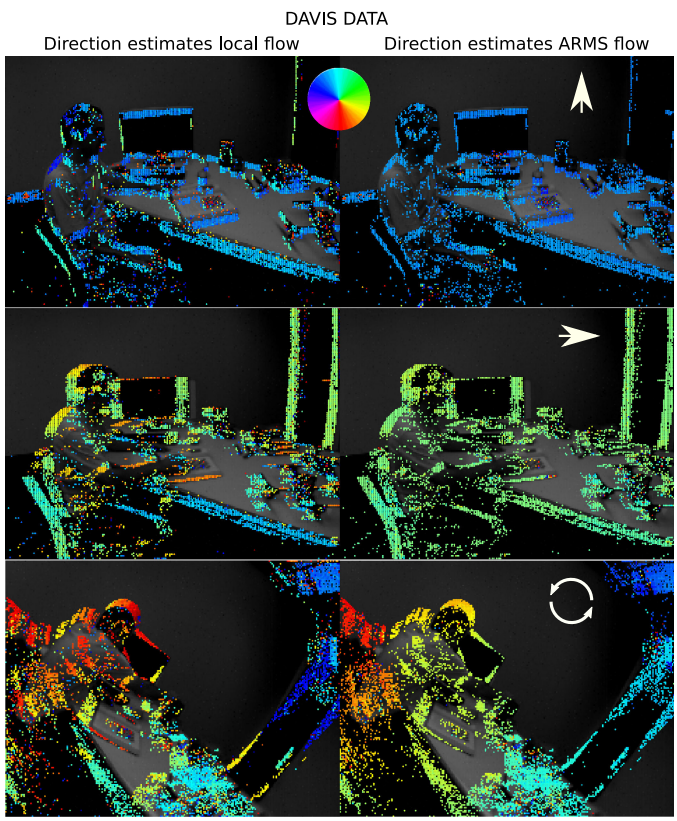


Fig. 8. Figure shows the flow results for two different available set of data recorded with DAVIS. The panels show the EDL and ARMS flow directions computed for data recorded for scenes recorded with camera moving freely while simultaneously recording the events and the motion of the camera with an IMU.

TABLE 2
Average Endpoint Error (AEE) in Pixels for Five MVSEC Data Set

Method	In_Fly1	In_Fly2	In_Fly3	Out_Day	Out_Night
EV-flow Net	1.03	1.72	1.53	N/A	N/A
EV-flow (EDL)	2.45	2.42	5.35	3.87	5.53
ARMS flow	1.52	1.59	1.89	2.75	4.47

The EV FLow-Net paper does not provide any error performance for the outdoor sequences.

images obtained from the camera to create dense flow for every image taken by the camera. Since our algorithm produces flow for every event, to compare with the ground truth, for every pixel we averaged the flow produced between the two images. Fig. 9 shows qualitative comparisons between the EDL, ARMS and ground truth directions from snapshots of the data taken in different conditions. We find that the ARMS flow vastly improves the EDL flow and generally performs well in different conditions. There are also conditions where the flow does perform poorly as shown in the bottom panels for the conditions - indoor flying 1, indoor flying 3 and outdoor driving. Such situations arise when the camera/scene are moving too slow for the EDL flow to provide good flow outputs or when the direction changes are too large for the flow to correct quickly creating discontinuity in the event plane. We must also note that the ground truth is built from the information of a LIDAR, (fused with the GPS and the IMU), which provides depth information within a range of 100 m in the case of outdoor driving sequences. On the other hand, the ARMS flow is estimated only from a single camera with much lower spatial resolution. This leads to errors in the flow when the car is both turning and moving forward as shown in the bottom panel of the outdoor driving sequence in Fig. 9. The ARMS flow seems to "see" only the dominant apparent motion of a left-to-right translation whereas the ground truth shows an expanding flow due to far structures in the scene. To further quantify the performance, as used in [13], we computed the average endpoint error (AEE) = $\sum ||(\hat{V} - V_{true})||_2$ where V_{true} is the ground truth derived from the dataset and $\hat{V}$ is the computed flow. The AEE was only computed over events rather than whole images. The performance was compared against the state of the art algorithms – EV-flownet [13] and Event based visual flow (EV-flow) [8]. The results of the quantification are presented in Table 2. The errors for the Indoor Flying conditions (In_Fly) were taken from [13]. We additionally report our error estimates for the outdoor driving conditions for which the error values were not provided with the dataset.

The proposed method shows remarkable improvements over the EV-flow and even though the algorithm is simple and works only on events, its performance matches that of the EV-flownet which requires elaborate learning network and is trained using both events and grayscale images. This means that to use it, one must have both event and image recordings for training of new scenes. Our method on the other hand only uses change events. We think that retraining the EV-flownet on binary images or only on events, or adding additional grayscale information into our algorithm would be a more suitable comparison and should close the performance gap between the two algorithms.

4.3 Event Based Prediction Using ARMS Flow

4.3.1 *Trivial Case*

The corrected direction estimates using ARMS flow can greatly improve the prediction of rigid object over traditional plane fitting methods. Fig. 7 shows the actual future events (green) and the predicted events for EDL (red) and ARMS (blue) flow using events that occurred 250 msec in the past. The figure shows that using ARMS flow, all the predicted events of the square form another square but if the directions are not the same as in case of the EDL, the predicted shape is not rigid anymore and does not form a square. To quantify the performance of the two flows, we compute how well the predicted events from local and corrected flow maintain the rigidness of the object. That is, we compute the affine transformation needed to map the predicted events to the actual events. To simplify, we assume zero rotation and perform

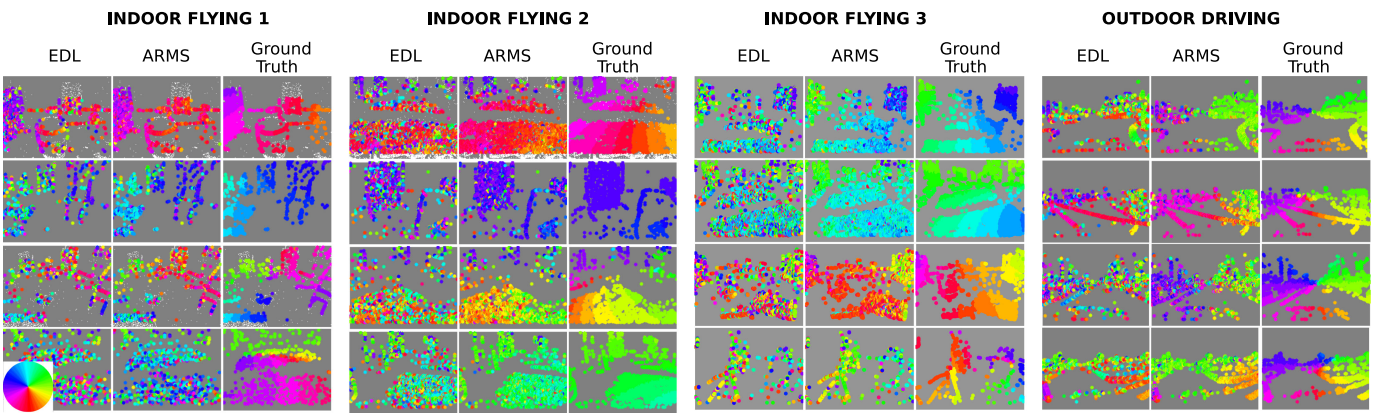


Fig. 9. Figure shows the flow results for events recorded from different conditions from the MVSEC benchmark data. The panels show flow directions from EDL and ARMS flow along with the ground truth directions. The ARMS flow vastly improves the EDL flow estimates and generally is close to the ground truth directions. We also show examples when the ARMS flow fails as shown in the bottom panels for Indoor flying 1, Indoor flying 3 and Outdoor Driving conditions.

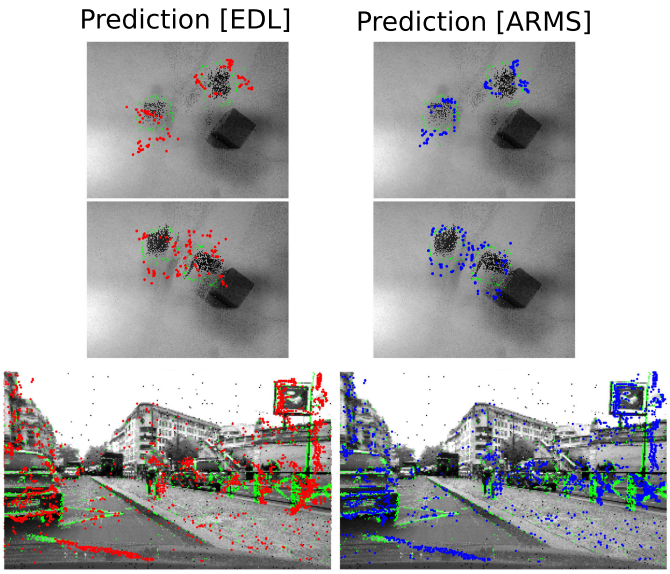


Fig. 10. Figure shows the performance of EDL and ARMS flow on prediction of events for the shapes and moving car scenario. The images show the actual events (green), the predicted events using EDL (red) and the predicted events using ARMS (blue). The figures show that the ARMS flow can greatly improve the prediction in both clean and cluttered and complicated environment invariant to the shapes or number of objects.

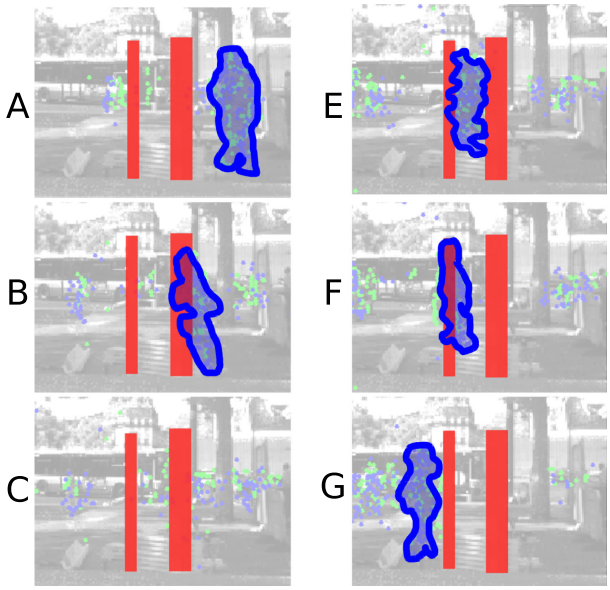


Fig. 11. Predictions of the motion of a person up to 200ms using ARMS flow in real world cluttered environment occluded by objects. The images show predictions when the pedestrian passes behind two poles (highlighted by red masks). The location of the pedestrian is masked (blue) by clustering the predicted event location and creating a single blob.

only translation and scaling. The graphs in Fig. 7 show the scaling and translation needed for the EDL and ARMS flow for a sequence of 360 msec broken into event clusters of 20 msec each. A perfect prediction would imply no scaling (i.e., scaling correction = 1) and no translation (translation correction = 0). The mean translation error for ARMS flow was 6.52 pixels per event versus 8.70 pixels per event for EDL flow. More importantly the scaling error in ARMS was only 0.085 compared to 0.141 in case of EDL. These results show that our proposed ARMS flow reduces the translation error and requires almost no scaling corrections showing that this flow can be used successfully to perform predictions on moving rigid objects.

4.3.2 Prediction of Multiple Objects and Moving Car

Next, we perform event by event predictions for the multiple moving objects and moving car scenarios mentioned in the previous section. Fig. 10 provides the prediction results from the non rectified and the rectified flow for these sequences. Contrary to the two previous sequences, as the scale from the scene is not easily extracted, we do not assess the impact of the prediction by measuring the affine deformation parameters and show only the prediction results. We managed thanks to the rectified flow, to predict events up to 250 msec. The ARMS flow can predict the event-by-event locations even in such a highly dynamic scene across all directions and again helps to maintain the affinity of different objects, such as the car, the person walking and the environment such as the dividers, poles and signs.

4.3.3 Prediction in Cluttered Scene With Occlusions

We also computed the flow and performed predictions at 200 ms in a more cluttered scenario where a pedestrian was tracked while passing behind stationary objects and across other pedestrians in a busy street scene. The results are presented in Fig. 11. The predictions very clearly match the actual events. This example is a typical showcase for the robustness of the prediction to occluding perturbations. We do not need to resort to more complex procedure such as some Kalman filtering to achieve accurate prediction.

4.3.4 Prediction of Moving Pedestrian on Street

To further measure the prediction capabilities using our flow method we placed the ATIS on a street corner and recorded pedestrians passing by. As in the trivial case example, for each incoming event, we make prediction on where the event will occur after 500 msec using the optical flow computed with both the local plane fitting and the new ARMS flow algorithm. We performed the transformation estimation for event clusters over 50 msec time windows. While the direction component of the flow for each event is used as it is, the speed component of the flow is normalized by the mean speed i.e., each event i in the event-cluster has speed M and direction θ_i, where M is the mean speed of all events and θ_i is individual flow directions. Using these predictions, a reconstruction of the motion is made based on local and corrected flow as shown in Fig. 12 by red and blue dots respectively. The figure shows that the ARMS flow can predict the position of the man up to next 500 msec very accurately. We used the transformation metric as used in the previous experiment to compare the performance of the two methods. The graphs show that the ARMS method outperforms the EDL through the sequence of recording for both scaling and translation corrections. The mean translation error for EDL was 7.1240 pixels while that for ARMS was 4.7558 pixels while the scaling error was 0.297 and 0.167 for EDL and ARMS respectively. Qualitatively, the

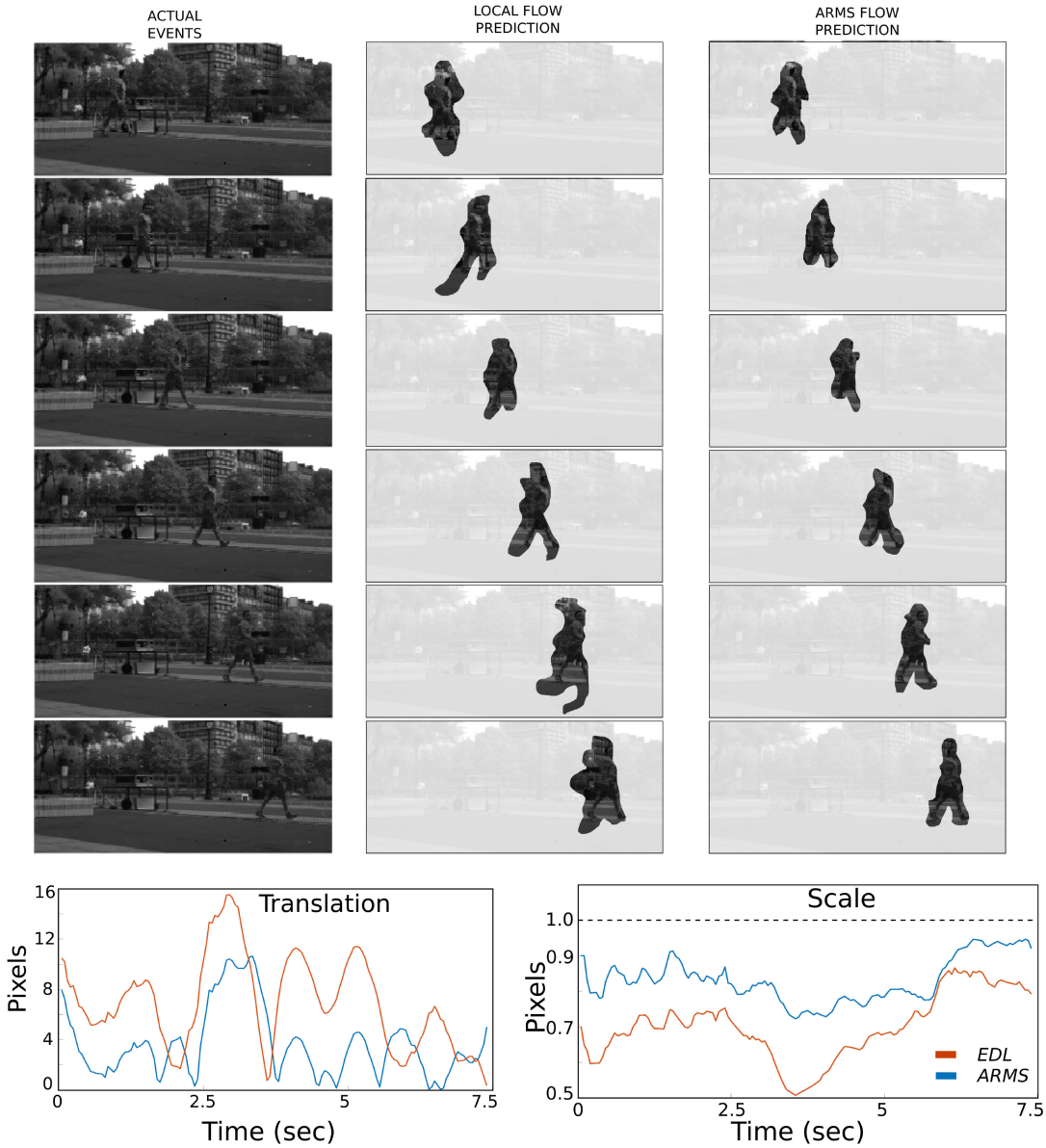


Fig. 12. Figure shows the possibility of performing predictions based on optical flow estimates for people passing by on a street at two different points in time. The images shows the actual events (green), predicted events using EDL (red) and predicted events using ARMS (blue) over grayscale images obtained from ATIS. The predictions from ARMS clearly show much better fit with the actual events while the predictions from EDL tend to create inflated shapes. This is quantified in the graphs showing the transformation required to fit the predicted events to the true events. The EDL required larger corrections in both scaling and translation compared to ARMS predictions.

cluster formed by the predictions based on the ARMS-Flow is less noisy and more compact and is much closer to the real events. This shows that our algorithm can maintain the shape on a rigid moving object even when the predictions are made on an event by event basis and therefore at very high temporal rates.

5 Conclusion

Event driven sensors provide an efficient sampling method to solve computer vision problems with scope for developing novel algorithms in temporal domain. Optical flow is an important feature for most vision based open problems and estimating fast yet robust flow is a crucial step. While some interesting algorithms have been developed to estimate visual flow using the event-driven sensors, they either fail to solve the aperture problem due to the emphasis on local spatio-temporal computation or are inefficient and do not really use the high event speeds of these sensors.

In this paper, we have presented a novel visual flow algorithm that not only solves the aperture problem but also performs on an event-by-event basis justifying the use of event-driven sensors. In fact, we exploit the intrinsic property of the event based optical flow algorithm, that allows for correcting the directions of erroneous local flow estimates. We have shown here that the algorithm works in real world scenarios, in case of both stationary and moving camera. The algorithm is invariant to the number of objects or their size and does not require additional processing steps such as object detection and tracking. This fast implementation allows us to perform truly event based prediction of moving objects from 250 to 500 msec in future without

TABLE 3
Benchmark Computation Times on an Intel E5-1603 Processor

Data	Num Events	Actual duration(sec)	Compute time(sec)	Rate (Evn/sec)
Shapes	111999	0.678	0.845	132510
BarSquare	1.25e6	5.8	8	156305
OutDay 1	5e6	13.41	25.9	132926
OutNight 1	5e6	17.2	23.3	114286
IndoorFly 1	5e6	29.87	28.2	192205
IndoorFly 2	5e6	19.58	28.16	127500
IndoorFly 3	5e6	24.21	26	172308

affecting the shape and size of the object. This is equivalent to making estimations of position of an object upto 10-25 frames in future when using a traditional frame-based camera. To the best of our knowledge, we could not find any methods using event-driven sensors that have attempted to perform such accurate predictions without any temporal binning of events. Further, these predictions are invariant to the size and number of independent objects in the scene. These predictions can allow higher order recognition and tracking layers to perform at the high temporal rates at which events are generated. Our future goals are to use this algorithm as part of an autonomous driving car sensor system to allow for fast collision detection and detect abnormal driver and pedestrian behavior. Further, the spatial scaling method works not only for plane fitting based local flow methods but any local flow methods that satisfy the condition that the flow magnitude is related to the contour of the edge or object in motion.

Since, our algorithm does not use higher order features, the true flow is only possible if an edge with true direction motion is present. In general, the flow improves depending on the presence of edges that are close to being orthogonal to the true flow direction. Also, since the flow is provided by the mean of local flows in a spatial window, the estimated flow is slightly moved away from the true flow. An improved prediction of the magnitude of flow could allow us to make predictions at even longer duration of up to a few seconds. A possible solution to improve the estimates could be by using the gray scale information provided by an ATIS like neuromorphic sensor. Analysing MVSEC and DAVIS data also show that ARMS flow can fail in certain scenarios especially in outdoor conditions where objects are far and the events on the camera plane themselves are not enough to compensate for the depth of the objects. This means that in cases when the car is making turns while moving forward the ARMS flow can only see the apparent motion of the whole scene moving whereas the depth information using a LIDAR would allow measure the expanding flow. In indoor flying cases, we found that while ARMS flow can remarkably improve the EDL flow output, errors occur if the drone makes large, sudden change in directions as this leads to discontinuity in the event plane and leads to large errors in local flow computation itself. The ARMS flow still performs close to ground truth for most of the duration of the flight sequences. Many new techniques for motion correction use contrast maximization methods [30] to segment events from object moving at different directions and speeds. This may allow for flow to be computed on events of individual objects. This could also improve the magnitude estimations but the events correction using this technique is still to some extent affected by the aperture problem [29].

In terms of the memory and CPU requirements, the algorithm was implemented in C++ running on single core Intel E5-1603 processor, achieving on average a computation rate of 120 kEvents/second [Table 3] and requires very small amount of memory that increases linearly with the pixels resolution of the sensor. While the traditional CPU is enough for real-time processing on a qVGA sensor, a parallel neuromorphic hardware implementation could make the algorithm independent of the sensor resolution and allow real time motion based visual processing for larger sensor arrays.

REFERENCES

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

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

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

[4] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.

[5] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[6] J. Y. A. Wang and E. H. Adelson, "Layered representation for motion analysis," in *Proc. IEEE Conf. Comput. VIs. Pattern Recognit.*, 1993, pp. 361–366.

[7] R. Benosman, S.-H. Ieng, C. Clerq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Netw.*, vol. 27, pp. 32–37, 2012.

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

[9] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Front. Neurosci.*, vol. 9, 2015, Art. no. 137.

[10] S. Seifozzakerini, "Analysis of object and its motion in event-based videos," to be published, doi: 10.32657/10220/46126.

[11] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nat. Neurosci.*, vol. 2, pp. 1019–1025, 1999.

[12] D. R. Valeiras, X. Clady, S. H. Ieng, and R. Benosman, "Event-based line fitting and segment detection using a neuromorphic visual sensor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1218–1230, Apr. 2019.

[13] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," in *Proc. Robot.: Sci. Syst.*, Jun. 2018, doi: 10.15607/RSS.2018.XIV.062.

[14] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 568–576.

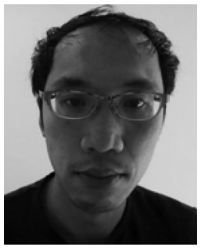
[15] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister, "Blind video temporal consistency," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 196.

[16] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3061–3070.

[17] M. Bai, W. Luo, K. Kundu, and R. Urtasun, "Exploiting semantic information and deep matching for optical flow," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 154–170.

[18] C. Bailer, K. Varanasi, and D. Stricker, "CNN-based patch matching for optical flow with thresholded hinge embedding loss," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2710–2719.
[19] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 41–48.
[20] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 25–36.
[21] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1647–1655.
[22] V. Vasco, A. Glover, Y. Tirupachuri, F. Solari, M. Chessa, and C. Bartolozzi, "Vergence control with a neuromorphic iCub," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 732–738.
[23] A. Glover and C. Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2203–2208.
[24] F. Rea, G. Metta, and C. Bartolozzi, "Event-driven visual attention for the humanoid robot iCub," *Front. Neurosci.*, vol. 7, 2013, Art. no. 234.
[25] H. Akolkar, D. R. Valeiras, R. Benosman, and C. Bartolozzi, "Visual-auditory saliency detection using event-driven visual sensors," in *Proc. Int. Conf. Event-Based Control Commun. Signal Process.*, 2015, pp. 1–6.
[26] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5419–5427.
[27] T. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.
[28] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, pp. 142–149, 2017.
[29] T. Stoffregen and L. Kleeman, "Event cameras, contrast maximization and reward functions: An analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12292–12300.
[30] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7243–7252.

Himanshu Akolkar received the MTech degree in EE from IIT, Kanpur, India, and the PhD degree in robotics from IIT, Genoa, Italy, after which he had a postdoctoral stint with the Université Piérre ét Marie Curie. He is currently a postdoctoral associate with the University of Pittsburgh. His primary research interest includes to understand the neural basis of sensory and motor control to develop an intelligent machine.



Sio-Hoi Ieng is currently an associate professor with Sorbonne Université, Paris, France, and a member of the Vision Institute, Paris. He was involved in the geometric modeling of catadioptric and non-central vision sensors and their link to the caustic surface. His current research interests include neuromorphic and event-based vision perception algorithms and computer vision, with a special reference to the understanding of general visual sensors, exploring cameras networks, and studying the connection between biologic and artificial vision.

Ryad Benosman received the MSc and PhD degrees in applied mathematics and robotics from the University Pierre and Marie Curie, Paris, France, in 1994 and 1999, respectively. He is currently an associate professor with the University Pierre and Marie Curie, Paris, leading the Natural Computation and Neuromorphic Vision Laboratory, Vision Institute, Paris. His work covers neuromorphic visual computation and sensing. He is currently involved in the French retina prosthetics project and in the development of retina implants and cofounder of Pixium Vision a french prosthetics company. He is an expert in complex perception systems, which embraces the conception, design, and use of different vision sensors covering omni-directional 360 degree wide-field of view cameras, variant scale sensors, and non-central sensors. He is among the pioneers of the domain of omni-directional vision and unusual cameras and still active in this domain. He has been involved in several national and European robotics projects, mainly in the design of artifcial visual loops and sensors. His current research interests include the understanding of the computation operated along the visual systems areas and establishing a link between computational and biological vision. He has authored more than 100 scientific publications and holds several patents in the area of vision, robotics, and image sensing. In 2013, he was awarded with the national best French scientific paper by the publication LaRecherche for his work on neuromorphic retinas.