Fast Event-based Harris Corner Detection Exploiting the Advantages of Event-driven Cameras

Valentina Vasco, *Student Member, IEEE*, Arren Glover, *Member, IEEE* and Chiara Bartolozzi, *Member, IEEE*

***Abstract*— The detection of consistent feature points in an image is fundamental for various kinds of computer vision techniques, such as stereo matching, object recognition, target tracking and optical flow computation. This paper presents an event-based approach to the detection of corner points, which benefits from the high temporal resolution, compressed visual information and low latency provided by an asynchronous neuromorphic event-based camera. The proposed method adapts the commonly used Harris corner detector to the event-based data, in which frames are replaced by a stream of asynchronous events produced in response to local light changes at μs temporal resolution. Responding only to changes in its field of view, an event-based camera naturally enhances edges in the scene, simplifying the detection of corner features. We characterised and tested the method on both a controlled pattern and a real scenario, using the dynamic vision sensor (DVS) on the neuromorphic iCub robot. The method detects corners with a typical error distribution within 2 pixels. The error is constant for different motion velocities and directions, indicating a consistent detection across the scene and over time. We achieve a detection rate proportional to speed, higher than frame-based technique for a significant amount of motion in the scene, while also reducing the computational cost.**

I. INTRODUCTION

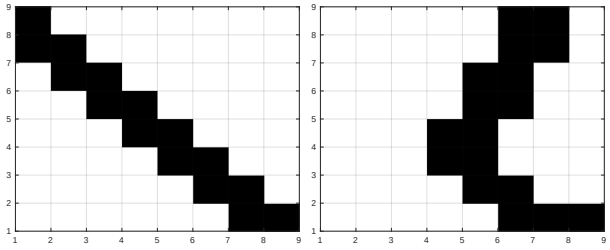
Using conventional frame-based cameras, several tasks such as stereo vision, motion estimation, object recognition and target tracking require the robust localisation of pixels of interest in images. For instance, in the field of visual motion estimation, the well known aperture problem makes the motion direction of an edge ambiguous, as only the motion component orthogonal to the primary axis of orientation of the object can be inferred based on the visual input. Corner detection and tracking between two or more frames, instead, can be used to produce the true flow. Therefore, corner features, defined as the region in the image where two edges intersect, must be detected consistently over time, and from different orientations and viewpoints. Typical processing in the frame-based paradigm occurs by looking at the appearance of patches of pixels in each frame, where measures derived from self-similarity [1], gradients [2], color [3] and curvature [4] have been proposed to characterize corner pixels.

Conventional cameras, while widely adopted, are not always optimal for robotics tasks as, during periods without motion, two or more images can contain the same redundant information, resulting in significant amount of computational resources wastage. Furthermore, the fixed frame rate at which images are acquired is not suitable for tracking fast moving targets: either tracking is completely lost above Nyquist rates, large displacements of features between frames introduce ambiguity when tracking multiple targets or targets can appear blurred in the direction of motion and therefore tracking is lost.

Event-based vision sensors (e.g. [5]) are biologically inspired sensors that produce an asynchronous stream of events in response to movements that occur in the sensor's field of view. In contrast to standard frame-based cameras, only the information from changing elements in the scene require processing, thereby removing data redundancy and reducing the processing requirements. In addition, events can be conveyed with a temporal resolution of 1 μs, allowing for precise computation of scene dynamics orders of magnitude faster than frame-based cameras. Responding only to changes in contrast in the field of view, event-driven sensors are natural contour detectors as events typically only occur on object edges. The corner detection problem can also be simplified as event-driven sensors inherently do not respond to uniform regions in the field of view. For instance, the well-known Canny edge detector was implemented in an event-driven way [6], exploiting this innate property of the sensor and achieving a more efficient and less resource demanding detection than the frame-based counterpart.

In this paper, we applied the frame-based Harris corner detection [2] to event-driven data, and showed that its performance benefits from the native properties of the event-driven sensors. While the innate edge enhancement facilitates the corner detection, the sensors' precise timing allows dense detection of fast moving corners, reducing the overall computational cost required by the frame-based method. Event-based corner detection was previously proposed [7], in which events that belong to the intersection of two or more motion planes are labelled as corners. The technique requires the inital calculation of optical flow, and is dependent on the flow direction being accurate. We propose that the variation around a corner can be detected directly by the pattern of active pixels in a small neighbourhood around the corner centre. While the data is motion driven (as motion inherently triggers events), we show that the resulting detection is independent on the speed and direction of the motion. The proposed algorithm is characterised and validated on the neuromorphic iCub humanoid robot [8], whose visual system is composed of two dynamic vision sensors (DVS) mounted as eyes in the head of the iCub, with total of 6 Degrees of Freedom (DoFs).

V.Vasco, A.Glover and C.Bartolozzi are with the iCub Facility, Istituto Italiano di Tecnologia, Italy. {valentina.vasco, arren.glover, chiara.bartolozzi}@iit.it. V.Vasco is also with Universitá di Genova, Italy.



(a) (b)

Fig. 1: A visualization of the asynchronous local contrast map when the local window includes an edge (a) and a corner (b). The black patterns correspond to "1" in the binary local map (i.e. where an event occurred). The local change in contrast is high in (b) along the two major axes, compared to (a) that is high along one direction.

II. THE ALGORITHM

The Harris corner detection [2] is one of the most widely used techniques to detect corner features in current frame-based vision processing, thanks to its reliability, low numerical complexity and invariance to image shift, rotation and lighting [9]. The method detects if a pixel is a corner by using a local window that is shifted by a small amount in different directions, to determine the average change in image intensity. The change is computed by calculating the spatial gradient of the intensity. A corner can be identified if a matrix containing the first derivatives of intensity has two large eigenvalues, meaning that the major intensity gradients centered around the pixel occur in two different directions. We do not disregard events, but augment the information carried by the single event when a corner is detected.

A. Event-based corner detection

The DVS sensor does not provide intensity measurements to which frame-based corner detection can be directly applied, but a stream of events characterized by the pixel position and the timestamp at which the event occurred. Since pixels only respond to local variations of contrast in the visual scene, for this application, we create an asynchronous local contrast map for each event, as shown in Fig. 1, and use the Harris score to compare the information generated from the current active pixel to that of surrounding pixels.

As a single event does not provide enough information alone, events need to be accumulated to create an informative representation of the world. However the data structure to use for processing the event stream is still an open question. A temporal window [10] is dependent on the velocity of the object in the environment and over- or under-estimating it results in a motion-blur effect or incomplete representation. As the visual stream of events is intrinsically related to the motion of the object, we chose a representation with a fixed number of events that is not affected by the speed, although is scene-dependent. We then need to integrate events over space to localize features in the scene.

Therefore, for each polarity, we consider a surface of events in the three-dimensional spatio-temporal domain, composed of the two dimensions of the sensor and an additional dimension representing time. For each incoming event, the surface maps the pixel position of the event to its timestamp, such that it will represent the latest timestamps evolving in time: $\Sigma_t : \mathbb{N}^2 \rightarrow \mathbb{R}, \mathbf{p} = (x, y)^T \rightarrow t = \Sigma_t(\mathbf{p})$ [11], and only the most recent N events are used to asynchronously update the surface $\Sigma_t(x, y)$.

For the current event $\mathbf{e_i} = (\mathbf{p_i}, t_i, pol_i)$, we define a local spatial window (W), L pixels wide, and associate to the pixels' positions inside the window 1 or 0 depending on whether an event is present on the surface at that pixel location, obtaining the *binarized* surface Σ_b:

$$\begin{aligned} \Sigma_b : \quad \mathbb{N}^2 \quad &\rightarrow \quad \mathbb{N} \\ if\ \exists\ \mathbf{e}\ at\ \mathbf{p} \quad &\rightarrow \quad \Sigma_b(\mathbf{p}) = 1 \\ otherwise \quad &\rightarrow \quad \Sigma_b(\mathbf{p}) = 0 \end{aligned} \tag{1}$$

We then compute the gradients of the obtained binary surface $\nabla\Sigma_b = [\frac{d\Sigma_b}{dx}, \frac{d\Sigma_b}{dy}]$ and use them to compute the symmetric matrix $M_{2\times2}$, as follows:

$$M(\mathbf{e_i}) = \sum_{\mathbf{e}\in W} g(\mathbf{e}) \begin{bmatrix} \frac{d\Sigma_b(\mathbf{e})}{dx}^2 & \frac{d\Sigma_b(\mathbf{e})}{dx}\frac{d\Sigma_b(\mathbf{e})}{dy} \\ \frac{d\Sigma_b(\mathbf{e})}{dy}\frac{d\Sigma_b(\mathbf{e})}{dx} & \frac{d\Sigma_b(\mathbf{e})}{dy}^2 \end{bmatrix} \tag{2}$$

where $g(\mathbf{e})$ is a Gaussian window function.

Similarly to the original Harris algorithm, we associate a score R to each event $\mathbf{e_i}$, computed as:

$$\begin{aligned} R(\mathbf{e_i}) = det(M) - k\ trace^2(M) = \\ \lambda_1\ \lambda_2 - k\ (\lambda_1 + \lambda_2)^2 \end{aligned} \tag{3}$$

where λ_1 and λ_2 are the eigenvalues of M.

Three cases will hold:

- if the patch includes few events due to the noise that occurs in the sensor, there is a negligible variation of contrast, leading both eigenvalues λ_1 and λ_2, and thus R, to be small;
- if the patch includes one edge (as in Fig. 1a), there is a high variation of contrast in the direction of the edge, leading one eigenvalue to be high and the other to be small, and thus R to become negative;
- if the patch includes two edges (as in Fig. 1b), there is a high contrast variation along the two major axes, leading both eigenvalues λ_1 and λ_2 to be high, and thus R to be positive: a corner event is probable. A special case is the line-ending, which can be seen as corner if the edge is thick enough to produce high contrast variation on both directions.

Based on these considerations, if $R(\mathbf{e_i})$ is greater than a set threshold S, the event $\mathbf{e_i}$ is labelled as corner event.

III. EXPERIMENTS

In the event-driven paradigm, the visual stream of events is dependent on the amount of motion in the camera's field of view (due to objects moving or due to the camera itself moving). Therefore the data generated by the camera is motion-dependent and the corner detection algorithm needs

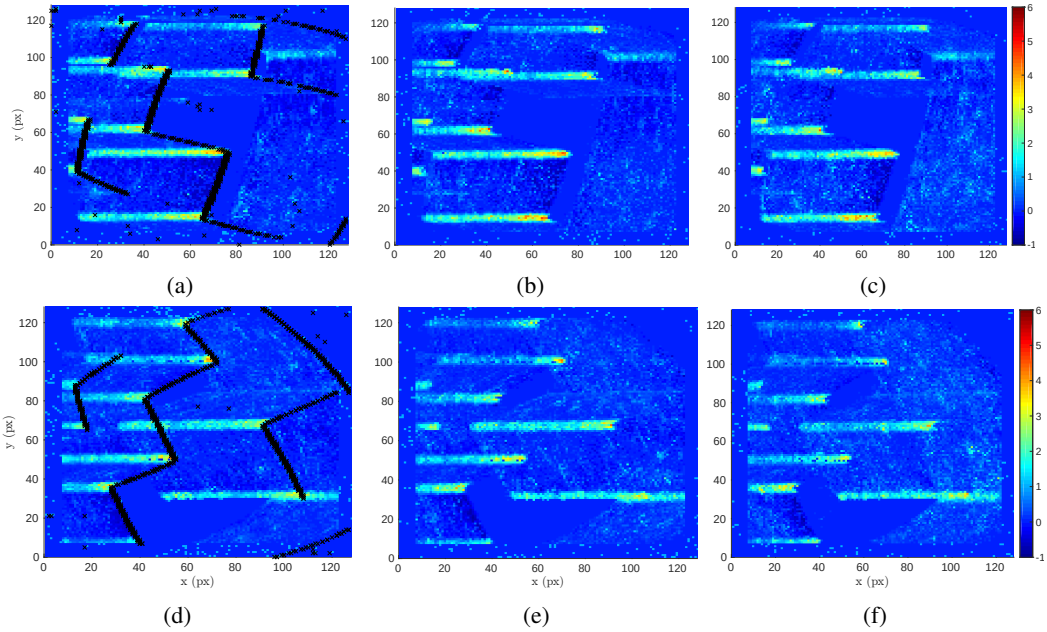


Fig. 2: Distribution of the score for different velocities magnitudes and corner orientations. Each row shows results for the dataset obtained with the same orientation and different magnitudes. The first row shows the score distribution for 15 degrees orientation and 10 deg/s (a), 40 deg/s (b) and 100 deg/s (c) magnitudes. The second row shows the score distribution for 60 degrees orientation and 10 deg/s (d), 40 deg/s (e) and 100 deg/s (f) magnitudes. Black dots represent the initial window of 1000 events. Only one polarity is shown for visualisation.

to be invariant to different motion types. For example, a corner oriented at 45 degrees to the direction of motion should produce similar edge velocities and strengths evenly around the corner centre. As the orientation angle changes, the relative strengths and velocities (due to camera aperture problems affecting optical flow) can also vary.

We characterised and tested the event-based corner detection on data collected from the dynamic vision sensor (DVS) mounted on the iCub robot [8], performing two different sets of experiments. The first set was performed by moving the iCub's eyes in front of a controlled pattern, in order to determine the detection sensitivity to various motion speeds and relative orientations of corners and velocity directions. In order to evaluate the detection response at different velocities, we moved the cameras at several speeds and also rotated the pattern to achieve different relative orientations. We used black and white "images", which are similar to the frame-based approach, with the difference that the contrast change is much sharper, without any gradient. Therefore the response due to change in angular magnitude and image rotation, as already shown in [2], should not significantly change in the event-driven method.

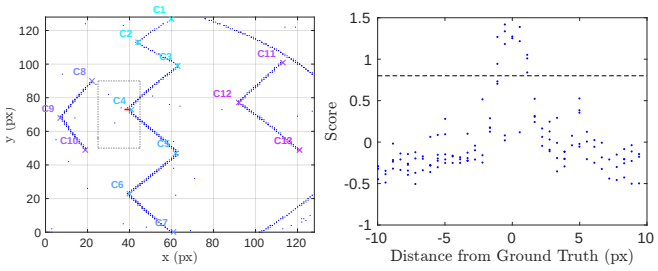
The second experiment was performed mimicking a typical scenario in which the iCub is used, with the robot looking around in the environment, where several objects are placed at different distances, to determine the robustness of the method in a real scene and the effectiveness and consistency of the detection over time.

A. *Characterization with a controlled input*

The stream of events from the iCub's camera was recorded while the robot was observing a static checkerboard, moving the eyes horizontally from left to right, at 7 different velocities, from 10 up to 100 deg/s, with a step of 15 deg/s. For each acquisition, the checkerboard was rotated at 7 different orientations, from 0 to 90 degrees, with a step of 15 degrees. For this experiment, we empirically selected $N = 1000$ events and a window of $L = 7$ pixels.

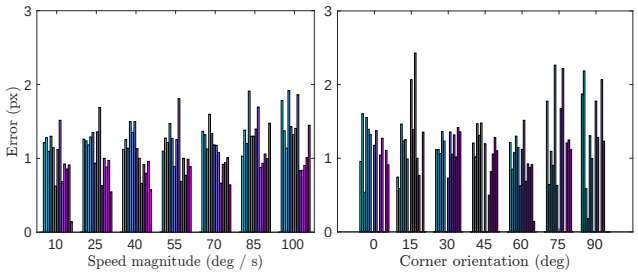
As shown in Fig. 2, events located on a corner exhibit a significantly higher score than events located on the edges. This indicates potential for speed and orientation invariant detection. Events on line endings produce a high score as well when the edge is thick (top line-endings in Fig. 2a and bottom line-endings in Fig. 2d). However, the line-ending is a feature point with a high information content, and is also considered a reasonable feature point for tracking. The score at the start is higher than the rest of the dataset. This seems related to the off polarity, as the score distribution is more uniform when we consider the on polarity. Importantly the scores calculated are consistent across different orientations and speeds qualitatively seen in Fig. 2.

To quantitatively assess the detection performance with different velocities and orientations, we compared the corner events detected by the proposed algorithm with a ground truth obtained by manually labelling the start and the end positions of each corner. As the motion profile was very simple (horizontal motion), we only cared about the corner position, disregarding any timing information and we con-



(a) (b)

Fig. 3: Slice of 1000 events and ground truth corners (a) and distribution of the score compared to the distance from the ground truth (b). The threshold S is marked by the dashed black line. Blue dots represent the score around the ground truth (red line) of each event inside the dotted black square within the slice of 1000 events.



(a) (b)

Fig. 4: Mean error at $S = 0.8$ for all the corners at fixed orientation (60 deg) and different speed magnitudes (a) and different orientations at fixed speed (10 deg/s) with respect to the corner's direction (b). Blue dots represent the initial window of 1000 events.

sidered that corner positions should not deviate from ground truth over time. In order to set a threshold, we considered the events located around a single corner within a slice of 1000 events (shown in Fig. 3a) and their score distribution with respect to the corresponding ground truth (shown in Fig. 3b). The distribution has a bell-shape, with a strong peak in the center (i.e. on the events located on the ground truth) that sharply decreases towards a steady state, as we move away from center (i.e. on the events located on the edges). If a threshold S is set equal to 0.8, we can easily isolate the high scores on corners from the lower edge scores with an estimated error distribution of 2 pixels.

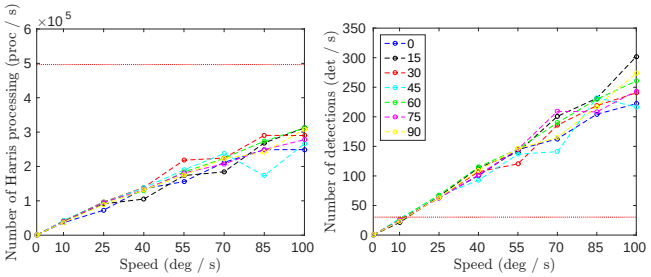
In order to evaluate how well this measure generalises across different corners, the threshold was applied to all the detected corners in the scene. The detection accuracy error was further characterised, by computing the distance between algorithmically detected corners and the ground truths, for every manually labelled corner/line-ending shown in Fig 3a, at different speeds and orientations. As shown in Fig. 4, the error is mostly consistent for all corners in all datasets with varying velocities and orientations. The typical detection error is still less than 2 pixels, reaching a mean value of 1.1 ± 0.3 px when varying the speed (Fig. 4a) and 1.2 ± 0.4 px when varying the orientation (Fig. 4b). The overall detection error is also comparable to the results obtained in [7] and thus we assert the event-based Harris detection has similar error distribution as the previously proposed intersection of motion constraints.

We compared the number of Harris processing required by frame and event-based methods, assuming the traditional camera to capture frames at 30 fps, with the same spatial resolution of 128×128 pixels of our event-based camera. While the traditional method computes the Harris score for each pixel, performing a global search of the maximum response over the entire frame, the event-based technique computes the score asynchronously and locally only when events occur. Therefore, the first requires a constant processing rate ($30 \times 128 \times 128 = 496480$ proc/s), and the second requires a variable processing rate depending on the number of processed events (which can vary with total number of edges in the scene and their velocities). As shown in Fig. 5a, the event-based corner detector drastically reduced the computational demand for all speeds when viewing the checker-board pattern. We also show the computational cost for a typical scene where several objects are placed in front of the robot, which moves its cameras at typical speeds (3, 5 and 20 deg/s), with the black crosses. The computation cost under typical conditions is therefore approximately $\sim 94\%$ lower than a frame-based camera, demonstrating the event-based advantage in terms of resource consumption.

Similar considerations apply for the number of detections per second, shown in Fig. 5b. For a fair comparison, we considered a single moving corner in the different datasets and computed the number of detections per second taking into account only the corner events that belonged to the associated ground truth line. While the frame-based detection rate is constant (~ 30 det/s assuming a single corner detection per frame), the event-based detection rate increases with velocity, achieving similar results for low speed (< 10 deg/s), but clearly outperforming the frame-based approach for higher speed (> 25 deg/s), reaching on average ~ 250 det/s for the largest velocity. We note that even the fast moving target produces spatially-dense observations, which has advantages for tracking as no information is lost (i.e. between frames), and has implications in removing tracking ambiguity between multiple targets. Also, while the frame-based approach suffers from 33 ms of latency, inherently due to the camera frame rate, the event-based detector processes event by event, by reducing the latency down to few μs.

B. Characterization for a real scene

To test the robustness of our method in a real scenario, we performed experiments in a typical iCub's environment, where several objects (a sponge, a car and a robot toy, a book, a tissue and a tea box) are placed on a table in front of the robot (as in Fig. 6) which in turn moves its eyes along several directions (shown in Fig. 7a) at a speed of 5 deg/s, while acquiring event-data from the scene. We empirically selected $N = 2000$ events and a window of



(a) (b)

Fig. 5: Comparison between the computational cost (a) and the detection rate (b) of the frame-based detector (dotted line) and the event-based detector, for different velocities of iCub's eyes and different orientations of the checkerboard. The cross markers indicate the number of processing for the complex scene (realistic velocities generally used on the robot have been tested). Both polarities are processed.

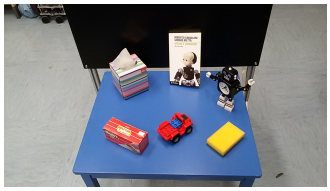
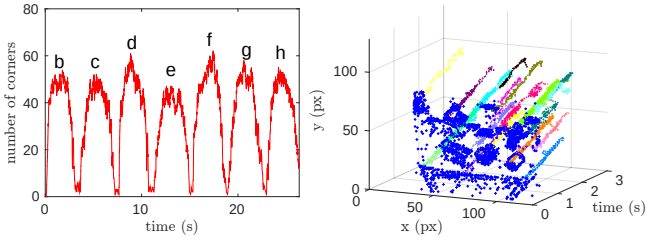


Fig. 6: Experimental setup for corner detection in typical iCub environments.

$L = 5$ pixels. The algorithm produced detection of many moving corners in the scene and examples of the detected corners for different motion directions are shown in Fig. 7. The sharpest corners are consistently detected over time and along different trajectories (e.g. the book and the sponge four corners). However, the low spatial resolution of the camera makes less-sharp corners more ambiguous in the binarised images, and the exact central position of corners (e.g. from the toy robot or toy car) are not as well defined.

Evaluating the detection consistency over time including changes in motion direction is important to produce a proper signal for tracking. Given a rough estimate of ~ 45 manually labelled corners in the visual scene, we grouped the algorithmically detected corner events that belonged to the same moving corner and computed the number of the grouped corner events over time. The total number of corners detected has a mean value of 45.7, remaining mostly consistent over time, indicating the corner detection algorithm responds consistently even in more cluttered datasets, as shown in Fig. 8a. During periods in which the eyes are changing direction, the low motion results in a low amount of events, and hence, corner detections. An example of traces in (x, y, t) space from the 'b' trajectory is shown in Fig. 8b. One corner was selected from each object in the scene and the (x, y) trajectory was manually tagged over the full dataset, shown in Fig. 9. The ground truth was computed for each corner by accumulating events over a period of 1s and manually labelling the central corner pixel. We visually identified a



(a) (b)

Fig. 8: Number of detected corners over time (a) and detected corner events over time in the (x, y, t) space (b) along the b trajectory (Fig. 7a). Blue dots represent the initial window of 2000 events corresponding to Fig. 7b and colors indicate corner events belonging to different moving corners.

radius of 5 pixels, marked by the green circles in Fig. 7b, within which corner events were considered as possible corner candidates. The limited testing range is necessary as many corners are spatially close making the analysis of actual corner accuracy difficult from a data association perspective.

As our data is motion-driven, during periods of no-motion we do not get detections, but importantly when motion occurs, consistent corner events occur in regions corresponding to the ground truth with an overall average error < 3 pixels (significantly lower than the 5 pixel limit). The error increases for objects with corners visually not well-defined: the robot toy and the car toy corners are detected with an error of 2.8 ± 1.4 px and 3.1 ± 1.3 px respectively. This also occurs as the spatially closest corners merge even within the selected radius of 5 pixels (e.g. it's difficult to disentangle corner events that belong to different car toy corners). However, the trajectories traced by such points on the image plane appear smooth and coherent to changes in time and in motion direction and we therefore propose that a suitable tracking method could be successfully applied.

IV. DISCUSSION AND FUTURE WORK

In this work, we have presented the event-driven Harris algorithm for corner detection using the event-driven DVS, which is able to detect moving corners in visual scenes with lower computation cost than a frame-based camera, and at a detection rate that is proportional to speed. Event-based Harris detection processes asynchronously each event whenever the corner moves by a pixel, thus requiring processing only during motion and achieving low latency. In comparison, the frame-based camera receives an observation at a set frame-rate regardless of any position change and with higher latency.

We used a data structure with fixed number of events, which was tuned according to the scene complexity, resulting in a higher value for the real scene dataset. This parameter however did not affect the score and the same threshold could be used for both datasets. Other representations are still under investigation.

We showed that, while motion is required to produce events, the algorithm was invariant to speed and relative

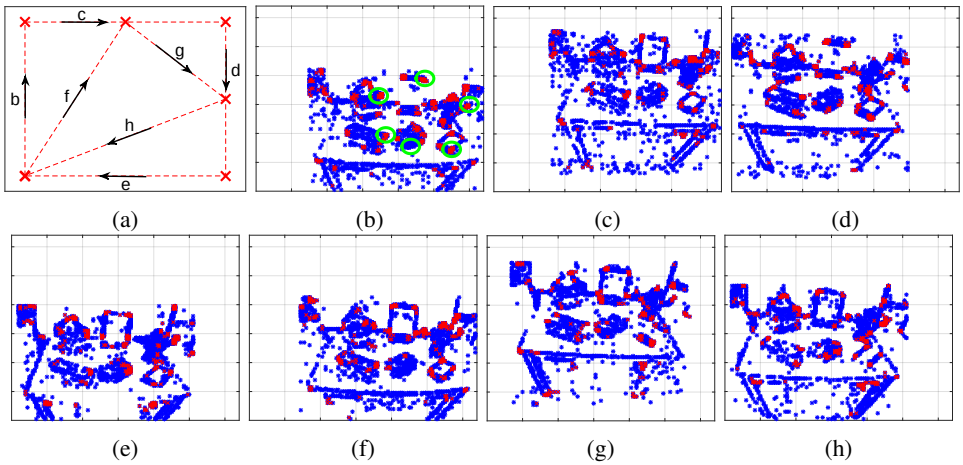


Fig. 7: Trajectories set on iCub's cameras and examples of corner events detected along different trajectories and at different time intervals: [0.3 - 0.4] s (b), [3.7 - 3.8] s (c), [8 - 8.2] s (d), [12.1 - 12.3] s (e), [15.8 - 16] s (f), [19.7 - 19.8] s (g) and [23.5 - 23.7] s (h). The green circles in (b) show the ground truth radius of 5 pixels. Both polarities are shown.

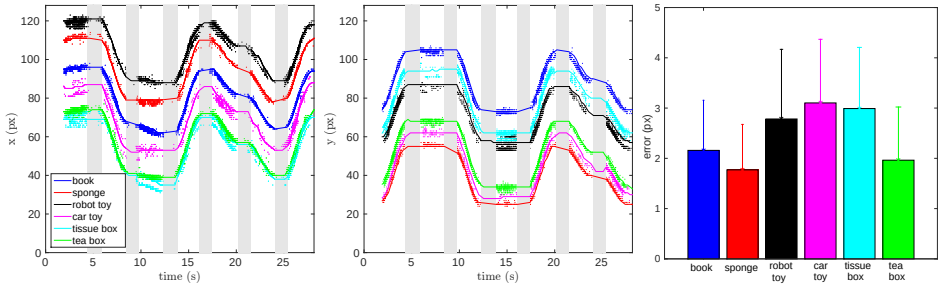


Fig. 9: Trajectories of the x (a) and y (b) coordinates of the detected corner events (dots) with corresponding ground truths (solid lines) and mean and standard deviation of the error for each object (c). Shaded areas indicate periods of non motion of the camera that do not produce detections.

corner orientation and the direction of the corner velocity. The latter being important if spatio-temporal gradients are used to compute optical flow as, in this situation, the aperture problem produces erroneous flow estimations. The algorithm was competitive with previous literature, achieving similar error distributions ($<$ 2 pixels), but without the need for the calculation of optical flow. In natural scenes the number of corners remained reasonable constant over time, and detection plots indicate a strong potential for robust tracking of corners on the event-driven iCub robot. We plan to use, in particular, these trackable features to generate a sparse optical flow that is unaffected by the aperture problem. Such flow can be used for learning entire scene flow statistics for characterising the ego-motion of the event-driven iCub.

References

[1] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover." *tech. report CMU-RI-TR-80-03*, p. 175, 1980.

[2] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Procedings of the Alvey Vision Conference 1988*, pp. 147–151, 1988.

[3] J. Chu, J. Miao, G. Zhang, and L. Wang, "Color-based corner detection by color invariance," *HAVE 2011 - IEEE International Symposium on Haptic Audio-Visual Environments and Games, Proceedings*, no. 1, pp. 19–23, 2011.

[4] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1376–1381, 1998.

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

[6] S.-H. Ieng, C. Posch, and R. Benosman, "Asynchronous Neuromorphic Event-Driven Image Filtering," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1485–1499, Oct. 2014.

[7] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015.

[8] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The iCub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[9] D. Parks and J. Gravel, "Corner Detection ," *International Journal of Computer Vision*, pp. 1–17, 2004.

[10] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural networks : the official journal of the International Neural Network Society*, vol. 27, pp. 32–7, mar 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22154354

[11] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, Feb. 2014.