Event-based Looming Objects Detection

Behnam Kamranian[1] and Howard Cheng[1]

Department of Mathematics and Computer Science
University of Lethbridge, Canada
b.kamranian@uleth.ca, howard.cheng@uleth.ca

Abstract. An event-based looming objects detection algorithm for asynchronous event-based cameras is presented. The algorithm is fast and accurate both in the detecting the correct number of objects as well as whether the objects are looming.

1 Introduction

In perception, looming is an optical phenomenon in which the size of a given object rapidly expands [18]. Looming often occurs when an object moves closer to the viewer. Fast and accurate looming detection is essential both in nature and robotics.

There are many looming object detection algorithms developed for conventional frame-based cameras (for example, [4,5,7,14,15,20]). However, conventional cameras are limited by their frame rates and also produce redundant data for parts of the scene that remain static. The Dynamic Vision Sensor (DVS) is designed based on the human retina [12]. The DVS is an event-based camera which asynchronously transmits events only when significant changes in the log-luminance of individual pixels are detected. There is no "frame" to collect events over a time interval, and no data is transmitted when there is no significant change. These cameras can react to events faster while their electrical and computational power requirement is lower.

The goal of this paper is to provide a real-time and automatic solution for the problem of detecting multiple looming objects. Optical flow is first computed using an event-based algorithm [17]. Clustering techniques are adapted for asynchronous optical flow events to identify potential objects, and the optical events for each objects are analyzed to determine if the object is looming. Our clustering algorithm does not require *a priori* assumptions on the shapes and number of objects and can adapt to changing number of moving objects in the scene. In addition, our algorithm can run on modest hardware without parallel processing.

2 Preliminaries

2.1 Event-based Cameras

The Dynamic Vision Sensor (DVS) is a neuromorphic camera which behaves similar to the human visual system by modeling the human retina [12]. Unlike

frame-based cameras which collect frames and transmit them synchronously at a fixed frame rate, the DVS asynchronously transmits events as soon as each event occurs. When there are no changes in the log-luminance in the scene, the DVS produces no output. When there is a significant change in the log-luminance of any pixel, the DVS asynchronously reports an event which is described by the coordinates,the timestamp, and the polarity $(+/-)$ of the change. The magnitude of the change is not reported. In the DVS, each pixel can adapt to its own intensity because they are independent from the other pixel sensors. As a result, the DVS has a very high dynamic range. A good survey on the event-based cameras and their applications can be found in [6].

2.2 Event-based Optical Flow

Ridwan and Cheng [17] presented an event-based optical flow algorithm that detect movements by identifying correlations among events. When objects move in a scene, the log-luminance changes occur mostly at the object boundaries. The pixels of a boundary edge will produce the same polarity along the direction of the motion over a period of time. Therefore, finding events of the same polarity in close proximity in time and space might be an indication of the motion.

The output of this algorithm is a stream of events containing the time, location and direction in eight compass directions ($\vec{v}_0$, ..., $\vec{v}_7$). For each DVS event that arrives, the algorithm searches the eight neighbours of the location for a matching recent event with the same polarity. Experimental results showed that each event can be processed in around two microseconds with very modest hardware. More details of the algorithm can be found in [17].

2.3 Event-based Object Clustering and Tracking

Barranco et al. presented a method [1] based on mean shift clustering and adapted this algorithm to process asynchronous events. To reduce the required computation time, this method processes events in parallel and in small packets of a few hundred events at a time. Using Kalman filters, this method can track multiple targets [11]. High temporal resolution results in accurate velocity measurements. Despite the advantages of using this method, it does require parallel processing to be feasible for real-time applications.

3 Single Object Looming Detection

In this section, we assume that there is only a single moving object in the scene. We describe an algorithm to detect if this object is looming, and present experimental results demonstrating its effectiveness. The input to our algorithm is the optical flow event stream from the event-based optical flow algorithm [17].

Object boundaries are identified by grouping similar optical flow events—if their angles differ by 45 degrees or less.. The boundary obtained consists mostly of the leading and trailing edge of the moving object. Object movement

is classified into three types: moving towards the viewer (looming); moving away from the viewer; or moving sideways. The boundary of a looming object moves away from the center of the object. When an object moves away from the viewer, its boundary moves towards the center. When an object moves sideways, the leading edge of the object moves away from the center while the trailing edge moves towards the center. The arithmetic mean of the locations of all boundary optical flow events is computed to obtain an interior point. Let $\vec{v}$ be the vector associated with the direction in the reported optical flow event, and $\vec{u}$ be the vector from the interior point to the event on the boundary. If $\vec{u} \cdot \vec{v} > 0$, we conclude that $\vec{v}$ is pointing away from the interior (Figure 1).

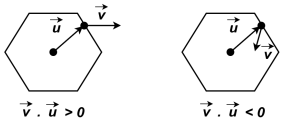

Fig. 1: Determining if an optical flow event is pointing away from the interior point or vise versa using dot product.

Optical flow events are collected into "pseudo-frames" of a certain length, and if the number of events pointing away from the interior is more than twice the number of events pointing towards the interior, our algorithm reports that a looming object is detected in the scene. To reduce the effect of noise, looming should only be reported if there is a significant number of vectors pointing away from the interior compared to the number of pixels in the scene. In our experiments, looming is reported only if the number of vectors pointing away from the interior is at least 0.5% of the total number of pixels in the scene.

The single object looming detection algorithm is shown in Algorithm 1. The algorithm produces an output event only if there is a looming object detected. Otherwise no output is produced. A queue $Q_{(x,y)}$ is used at each pixel to store recent optical flow events. The thresholds required are the length of the pseudo-frame L, and time thresholds T_{low} and T such that only those events between T_{low} and T seconds before the current event are considered recent. A set S is used to collect optical flow events into a pseudo-frame and a global variable *last* is used to record the timestamp of the last pseudo-frame. On average the complexity is constant for each event.

3.1 Experimental Results

We have performed experiments on our looming object detection algorithm on various scenarios. We have used simple objects with the DVS for these experiments. For visualization, an optical flow event is shown as a blue line moving towards a red dot. A data set was created to test the effectiveness of the algorithm by using thd DVS to capture motion in a scene. A brief of the description of the data set used is given below.

Round looming object: a round object is moving towards the viewer (Figure 2 (a)).

Algorithm 1 Looming detection algorithm.

procedure LOOMING$(x, y, t, \vec{v})$
  Remove all events $(x', y', t', \vec{v'})$ such that $t - t' > T$ from the front of $Q_{(x,y)}$.
  Add $(t, \vec{v})$ to the back of $Q_{(x,y)}$
  boundary $\leftarrow$ false
  for each (x', y') an 8-neighbour of (x, y) **do**
    Search in $Q_{(x',y')}$ for an event $e' = (x', y', t', \vec{v'})$ such that $T_{low} < t - t' \leq T$ AND $\vec{v}$ and $\vec{v'}$ are similar
    if NOT found **then**
      boundary $\leftarrow$ true
    end if
  end for
  if NOT boundary **then**
    return
  end if
  $S \leftarrow S \cup \{(x, y, \vec{v})\}$
  if $t - last > L$ **then**
    $(c_x, c_y) \leftarrow$ centroid of all events in S
    $pos, neg \leftarrow 0, 0$
    for each $(x, y, \vec{v}) \in S$ **do**
      $d \leftarrow ((x, y) - (c_x, c_y)) \cdot \vec{v}$
      if $d > 0$ **then**
        $pos \leftarrow pos + 1$
      else if $d < 0$ **then**
        $neg \leftarrow neg + 1$
      end if
    end for
    if $pos > 2 \times neg$ AND $pos > 0.005 \times$ total pixels **then**
      Report **LOOMING** at time t
    end if
    $S \leftarrow \{\}$
    $last \leftarrow t$
  end if
end procedure

Round object moving sideways: a round object is moving from right to left (Figure 2 (b)).

Square looming object: a square object is moving towards the viewer (Figure 2 (c)).

Square object moving sideways: a square object is moving from left to right (Figure 2 (d)).

For these experiments, the constants and the thresholds we have chosen experimentally are shown in Table 1. The results of our experiments are shown in Table 2.

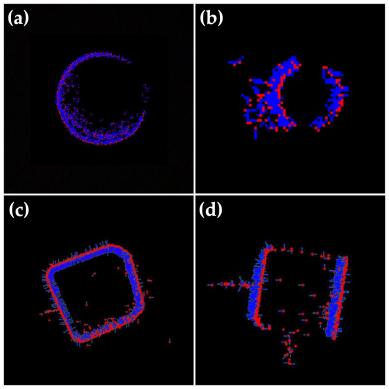

Fig. 2: (a) A round looming object. (b) A round object is moving sideways. (c) A square looming object. (d) A square object is moving sideways.

Table 1: Values of all thresholds and constants for all experiments.

Name	Value	Unit
COLS	180	pixels
ROWS	190	pixels
Timestamp Threshold (T)	25,000	μs
Low Timestamp Threshold (T_{low})	100	μs
Length of Pseudo-frame (L)	25,000	μs

4 Multiple Object Looming Detection

When there are multiple objects in the same scene, our looming detection algorithm (Section 3) fails to detect the looming objects. Our goal in this section is to separate the events in the scene into multiple objects using clustering, and then apply our single object looming detection algorithm to each segmented object.

Clustering algorithms generally require a set of data points to group them into different clusters. However, the optical flow event stream is asynchronous. New events can arrive at any time and old events also need to be removed. Some algorithms solve this problem by using pseudo-frames. Although there are some real-time clustering algorithms adopted for event-based data points [1,13], they required parallel processors or special FPGA hardware for real-time performance. Only the coordinates of the optical flow events are used for clustering. The label of each point along with the centroid of each cluster is reported by the clustering algorithm. We also use L as a parameter for the algorithm to adjust the length of pseudo-frame.

Table 2: Results of single looming object detection.

	Round looming object	Round object moving sideways	Square looming object	Square object moving sideways
Total number of events	668530	672201	850175	792649
Length of video (s)	3.92	2.81	5.10	10.07
Run Time per event (μs)	1.69	2.82	2.27	2.59
Decision	Looming	Not looming	Looming	Not looming

4.1 *K*-Means Event Clustering

The K-means algorithm [9] is a well-known clustering algorithm. As the number of clusters is not known in advance, the K-means algorithm is executed with different values of $K = 1, \ldots, M$, where M is the maximum number of clusters to consider. A "compactness" measure C is used to compare different outputs produced by clustering algorithms with different values of K:

$$C = \sum_{i=1}^{n} \|x_i - c_{l(x_i)}\|^2, \tag{1}$$

where $l(x_i)$ is the label assigned to event x_i, $\|x_i - c_{l(x_i)}\|$ is the distance between each data point x_i and each cluster's centroid $c_{l(x_i)}$. By plotting the compactness as a function of K, we can find the "elbow point" where the rate of reduction changes drastically [8,10,21]. The elbow point is a good candidate for the number of clusters. Figure 3 shows the elbow point and the change in the compactness

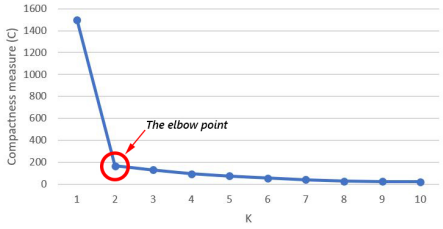

Fig. 3: The elbow point method.

measure as the number of clusters increases. There is no consensus on a mathematically rigorous definition of the elbow point [10]. The Kneedle algorithm [19] had been proposed for finding the elbow point, but experiments show that it was not well-suited for the data arising in our application.

4.2 The elbow method

Heuristically, the elbow point is the point at which the angle of the curve is the greatest (Figure 3). Only those points in which the decrease in compactness is greater than the average decrease over all values of K are considered, and the point with the largest angle is reported as the number of clusters (Algorithm 2). In the algorithm, M is the maximum number of clusters to consider, and C_i is the compactness measure when the K-means algorithm is used to cluster the events into i clusters.

4.3 Sequential *K*-means Clustering

Sequential K-means clustering is a variation of the standard K-means clustering algorithm that processes one data point at a time and update the clusters' centroids at each step [3]. centroids at a particular time. When a new data point

Algorithm 2 The elbow method

procedure ELBOW($M, C_1, \ldots, C_M$)
  $avg = \frac{C_1 - C_M}{M-1}$
  $\beta_{max} = 0$
  for $i = 1, \ldots, M-1$ **do**
    $\Delta_i = C_{i+1} - C_i$ ▷ Note: $\Delta_i < 0$
  end for
  for $i = 1, \ldots, M-2$ **do**
    if $\Delta_i \leq \Delta_{i+1}$ AND $-\Delta_i \geq avg$ **then**
      $\beta = \arccos\left(\frac{(1,\Delta_i)\cdot(1,\Delta_{i+1})}{\|(1,\Delta_i)\|\cdot\|(1,\Delta_{i+1})\|}\right)$
      if $\beta > \beta_{max}$ **then**
        $\beta_{max} = \beta$
        $K = i + 1$
      end if
    end if
  end for
  return K
end procedure

x is received, the algorithm chooses the centroid c_i closest to x and adds x to the corresponding cluster. The centroid c_i is updated by

$$c_{i+1} = c_i + \frac{1}{n}(x - c_i), \tag{2}$$

where n is the total number of data points assigned to that cluster, including x.

For each data point, the number of operations required is proportional to K because of the search for the nearest centroid. As a result, the update can be done very quickly for each point, and it is even feasible to perform K-means clustering for multiple values of K simultaneously. The compactness measure for each value of K can be used by the elbow method to determine the appropriate number of clusters.

4.4 Cluster Merging

When objects are too large, the clustering algorithms may fail to detect the correct number of clusters by dividing them into separate clusters. This is because these algorithms try to minimize the average squared distance between each data point and the centroid of the clusters. As a solution to this problem, we can merge these clusters to form a single cluster. Clusters that are connected as 8-neighbours are merged into one connected component as a new cluster.

5 Experiments and Results

Tje different proposed clustering algorithms described in Section 4 are evaluated. The algorithms are tested with event streams generated from both captured and simulated scenarios. The algorithms are tested with data sets shown in Table 3. The captured event streams were obtained with the DVS specified in Table 4.

Table 3: Captured data sets.

Data set	Description	Number of polarity events	Number of optical flow events	Number of pseudo-frames
1	A single ball is falling	14900	9074	18
2	Two round objects are moving sideways	113240	81214	291
3	A round object is looming	79850	52786	73
4	Two balls are rolling sideways	22026	14831	18
5	Four round objects are looming	40900	26302	35

Table 4: The DVS specifications used for experiments.

Name	Value
Model	DVS 240 B
I/O	USB2.0
Power consumption	Low/high activity: 30/60 mA @ 5 VDC
Number of columns ($COLS$)	180 pixels
Number of rows ($ROWS$)	190 pixels

All algorithms were implemented in C++ and the OpenCV library [2]. To evaluate the correctness of the cluster detection algorithms we reported the number of pseudo-frames in which the correct number of clusters was detected. To evaluate the quality of each detected cluster, we manually checked the labelling in each pseudo-frame. We manually labelled the looming results of each pseudo-frame and we compared them with the results generated by the algorithms. A looming object detected correctly is a true positive, while a true negative occurs when the algorithm produce no output when there are no-looming objects. The commonly used Recall and Precision measures [16] are computed.

The figures show the detected clusters in different colours and depict the centroid of each cluster by a dot surrounded by a circle with the same colour of its cluster. The detected looming clusters are shown as yellow circles on their centroids (Figure 4).

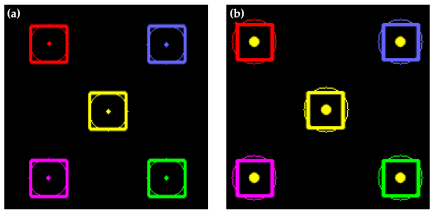

Fig. 4: (a) five detected clusters. (b) Five detected looming clusters

In Data Set 1, a ball is falling in front of the camera. The goal of this experiment is to determine whether the clustering algorithms is capable of detecting a single cluster. Figure 5 shows the optical flow events and output of clustering for one of the pseudo-frames of this data set. All of the results of experiments on captured data sets are reported in Table 5. Both sequential and OpenCV's K-means

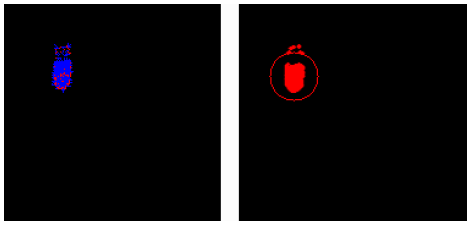

Fig. 5: A single ball is falling.

algorithms failed to detect the correct number of clusters in all pseudo-frames. Overall applying the merge algorithm enhanced the results drastically. Using the elbow method and sequential K-means algorithm, only five pseudo-frames have an the incorrect number of detected clusters. In this data set the sequential K-means is at least 54 times faster than OpenCV's K-means and 10 times faster than the mean shift algorithm without parallel processing (Section 2.3).

In Data Set 2, two round objects are moving sideways. The goal of this experiment is to compare the accuracy of algorithms when objects move straight versus when objects are rolling (Data set 6) in front of the camera. Figure 6 shows the optical flow events and clustering output for a pseudo-frame of this

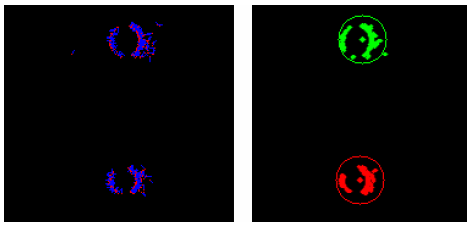

Fig. 6: Two round objects are moving sideways.

data set. All algorithms were able to detect the correct number of clusters in all pseudo-frames. The fastest algorithm is sequential K-means which on average took about 0.6 milliseconds to process each pseudo-frame.

In Data Set 3, a ball is approaching the camera. We moved the ball very close to the camera to see how the algorithms can perform when the dimensions of the objects are large or when they are close to the camera. None of the algorithm was able to detect the correct number of clusters. The reason is that when the object is so close to the camera, the algorithms separate it to multiple clusters to decrease the average squared distance from each event to the computed centroid. Figure 7 shows a pseudo-frame in this situation. Applying the merge algorithm enhanced the results. Figure 8 shows a pseudo-frame in which the merge algorithm was able to merge multiple clusters to a single cluster. However, due to both noises and lack of events in some parts of the object, the clustering algorithms were not able to detect a single cluster even by using the merge algorithm. The sequential K-means algorithm is again the fastest method and processed each pseudo-frame about 60 times faster than OpenCV's K-means algorithm.

For Data Set 4, two balls are rolling sideways. Figure 9 shows a pseudo-frame of the optical flow and clustering output of this data set. The algorithms were

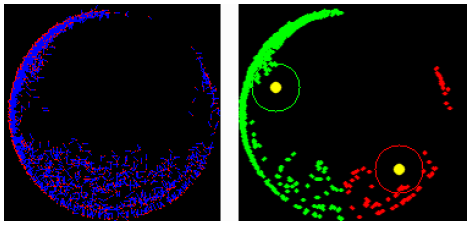

Fig. 7: A single looming ball which is incorrectly detected as two clusters. The merge algorithm was not applied.

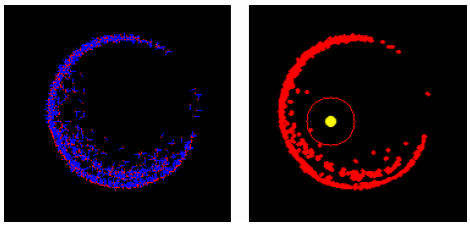

Fig. 8: Single ball is looming and detected as a single cluster by applying the merge algorithm.

able to detect the correct number of clusters in most pseudo-frames. Applying our merge algorithm enhances the results further.

For Data Set 5, the camera is moving toward four round objects in a solid white background. Figure 10 shows the optical flow events and clustering output for a single pseudo-frame of this case. Our algorithm was able to detect the correct number of clusters, though the recall rate is low because the camera is approaching the objects from an angle and it classifies the motion as sideways instead of looming.

The results of the experiments indicate the advantages of using sequential K-means algorithm compared to other methods. This is a real-time algorithm, and does not require any parameter adjustment. It can automatically adapt itself in all experiments cases to changing number of objects and movement types. Compared to other clustering algorithms, it is much faster and can process each pseudo-frame in less than 0.8 milliseconds depending on the size of the input. This is at least 30 times faster than OpenCV's K-means algorithm. In addition, the sequential K-means algorithm achieved the highest accuracy in cluster detection compared to other algorithms in most data sets.

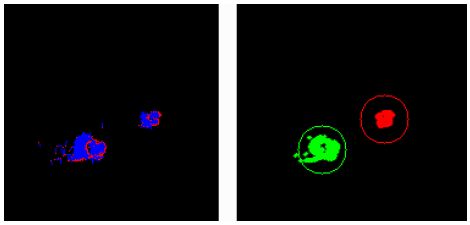

Fig. 9: Two balls are rolling sideways.

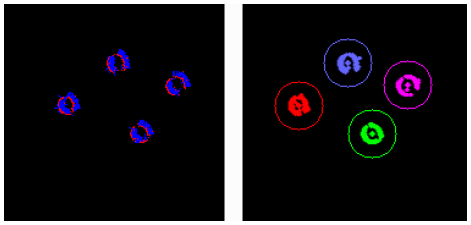

Fig. 10: Four round objects are looming.

Table 5: The results of experiments on captured data sets.

Dataset	Clustering Methods	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (ms)
			Precision	Recall	
1	K-means	88.88	72.22	100.00	28.125
	Seq. K-means	72.22	66.66	100.00	0.543
2	K-means	100.00	97.93	100.00	26.944
	Seq. K-means	100.00	96.90	100.00	0.693
3	K-means	56.16	95.71	91.17	41.596
	Seq. K-means	41.09	98.59	95.89	0.799
4	K-means	100.00	88.88	100.00	34.514
	Seq. K-means	100.00	80.55	100.00	0.632
5	K-means	25.71	100.00	7.14	36.391
	Seq. K-means	100.00	100.00	5.71	0.667

6 Conclusion

We presented a real-time looming object detection algorithm using event-based camera. It does not require any a priori knowledge of the number of objects in the scene and can adapt to changing number of objects. The proposed algorithm is significantly faster than the conventional K-means algorithm, and the accuracy for looming object detection is similar.

While our algorithm performs very well when objects are looming directly towards the camera, recall rate is low when the objects are looming towards the camera at an angle. Future works will address this limitation.

References

1. Barranco, F., Fermüller, C., Ros, E.: Real-time clustering and multi-target tracking using event-based sensors. CoRR **abs/1807.02851** (2018), `http://arxiv.org/abs/1807.02851`
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
3. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley-Interscience, 2nd edn. (2000)
4. Fülöp, T., Zarándy, A.: Bio-inspired looming object detector algorithm on the eye-ris focal plane-processor system. In: Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on. pp. 1–5. IEEE (2010)
5. Fülöp, T., Zarándy, A.: Bio-inspired looming direction detection method. In: Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on. pp. 1–6. IEEE (2012)

6. Gallego, G., Delbruck, T., Orchard, G.M., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2020). https://doi.org/10.1109/TPAMI.2020.3008413
7. Gil-Jiménez, P., Gómez-Moreno, H., López-Sastre, R., Bermejillo-Martín-Romo, A.: Estimating the focus of expansion in a video sequence using the trajectories of interest points. Image and Vision Computing **50**, 14–26 (2016)
8. Goutte, C., Toft, P., Rostrup, E., Nielsen, F., Hansen, L.: On clustering fmri time series. NeuroImage **9**(3), 298 – 310 (1999). https://doi.org/https://doi.org/10.1006/nimg.1998.0391, http://www.sciencedirect.com/science/article/pii/S1053811998903913
9. Hartigan, J., Wong, M.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)
10. Ketchen, D., Shook, C.: The application of cluster analysis in strategic management research: an analysis and critique. Strategic management journal **17**(6), 441–458 (1996)
11. Li, X., Wang, K., Wang, W., Li, Y.: A multiple object tracking method using kalman filter. In: The 2010 IEEE International Conference on Information and Automation. pp. 1862–1866 (June 2010). https://doi.org/10.1109/ICINFA.2010.5512258
12. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128 × 128 120 *db* 15 μs latency asynchronous temporal contrast vision sensor. IEEE Journal of Solid-State Circuits **43**(2), 566–576 (2008)
13. Linares-Barranco, A., Gómez-Rodríguez, F., Villanueva, V., Longinotti, L., Delbrück, T.: A usb3.0 fpga event-based filtering and tracking framework for dynamic vision sensors. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2417–2420 (May 2015). https://doi.org/10.1109/ISCAS.2015.7169172
14. Pantilie, C., Nedevschi, S.: Real-time obstacle detection in complex scenarios using dense stereo vision and optical flow. In: 13th International IEEE Conference on Intelligent Transportation Systems. pp. 439–444. IEEE (2010)
15. Park, S.S., Sowmya, A.: Autonomous robot navigation by active visual motion analysis and understanding. Proceedings of IAPR Workshop on Machine Vision Applications (1998)
16. Powers, D.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation (2011)
17. Ridwan, I., Cheng, H.: An event-based optical flow algorithm for dynamic vision sensors. In: International Conference Image Analysis and Recognition. pp. 182–189. Springer (2017)
18. Rind, F., Simmons, P.: Seeing what is coming: building collision-sensitive neurones. Trends in neurosciences **22**(5), 215–220 (1999)
19. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In: 2011 31st International Conference on Distributed Computing Systems Workshops. pp. 166–171 (June 2011). https://doi.org/10.1109/ICDCSW.2011.20
20. Subbarao, M.: Bounds on time-to-collision and rotational component from first-order derivatives of image flow. Computer Vision, Graphics, and Image Processing **50**(3), 329–341 (1990)
21. Thorndike, R.: Who belongs in the family? Psychometrika **18**(4), 267–276 (Dec 1953). https://doi.org/10.1007/BF02289263, https://doi.org/10.1007/BF02289263