Analysis of object tracking algorithms performance on event-based datasets

Alexandra-Claudia Olaru
Supervisor(s): Nergis Tömen, Ombretta Strafforello, Xin Liu
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

The event-based camera represents a revolutionary concept, having an asynchronous output. The pixels of dynamic vision sensors react to the brightness change, resulting in streams of events at very small intervals of time. This paper provides a model to track objects in neuromorphic datasets, using clustering. In addition, a non-linear filter is applied to correct the estimation of the object position. Both single and multi-object tracking algorithms are provided and their performance is analyzed using different metrics, including the clustering evaluation scores and the tracking accuracy. The accuracy is over 0.6 for multi-target tracking and more than 0.7 for single object tracking. Besides the proposed model[1], a comparison between different possible approaches for event-based data tracking is provided.

Keywords: Event-based data, single object tracking, multi-target tracking, Particle Filter, clustering

1 Introduction

Computer vision on dynamic scenes, described by high motion, is considered a challenge due to the limited clarity of the images. An innovative solution is an event-based camera, inspired by the human retina. Each pixel adjusts and reacts to temporal brightness changes, the output being an asynchronous event with the pixel address, characterized by an accurate timestamp. An event is identified by a tuple (x, y, t, p), representing the intensity change at the pixel with location (x, y) and time t, with polarity $p \in \{0, 1\}$, showing whether the intensity is increased or decreased. The advantages of using Dynamic Vision Sensors (DVS) are: very high temporal resolution and low latency, very high dynamic range and low power consumption [1]. Another important benefit is the fact that the size of the data set generated by the sensor is small, allowing for the usage of DVS in embedded real-time systems [2].

Being a relatively new concept, these asynchronous sensors are mainly used within the robotics community, and the research in other computer vision areas is restricted. Furthermore, the asynchronous output which depends on both the scene brightness change and the motion between the scene and camera raises concerns about the usage of classic algorithms [1]. Consequently, the available algorithms developed for this technology, as well as the datasets, are limited and the extent to which the event-based datasets can be used reliably to perform object detection and tracking is not well known.

Considering the restricted data and algorithms, this research aims to show that computer vision with event-based cameras can perform as well as the algorithms used with frame-based datasets. More specifically, this paper intends to answer the question "*How well can single and multiple object tracking perform using event-based data?*".

To answer the question proposed in this paper, the research was divided into multiple sub-tasks:

- What is the general distribution of events constructing the objects? How does the linearity of the system influence the algorithm?
- How can objects be detected and how can the objects be differentiated from the noise?
- What would the best characterization of an object be to track it?
- How can we estimate the trajectory and correct detection error?
- What is the difference between single and multi-object tracking for event-based data?

The main contributions of this research are:

- Propose a model to track objects in time, using a clustering algorithm to detect the moving objects and a non-linear filter to smooth their trajectory
- Analyse the model on the Neuromorphic-MNIST database, which was created using an Asynchronous Time-based Image Sensor (ATIS) [3]

The theoretical aspects of this model are provided, as well as the code and datasets used, aiming to contribute to the event-based vision research.

The outline of the paper is as follows: Section 2 presents the existing research related to event-based computer vision. In Section 3, several methodologies used to detect and track objects are presented, followed by the contribution and results in Section 4. Section 5 analyses the results and present the limitations of this research. Lastly, the ethical aspects of this research are described in Section 6, and Section 7 ends this paper.

2 Related work

The recent development of dynamic vision sensors influenced neuromorphic vision, and new methods are being proposed for object tracking [4]. In this section, prior work on event-based computer vision is described.

2.1 e-TLD (Tracking Learning-Detection): Event-based Framework for Dynamic Object Tracking

An object tracking model [5] uses a local search to track the object, and in case of failure, it does a global search to localize the object. Online learning, which is a technique to update the estimations when new data arrives, is required in this approach to consider the changes in object appearance. For the tracker, a binary classifier, which uses the online learning approach, is implemented and it is modified when

[1]The code for this paper is available on Github at https://github.com/aolaru11/Object-tracking-using-event-based-camera

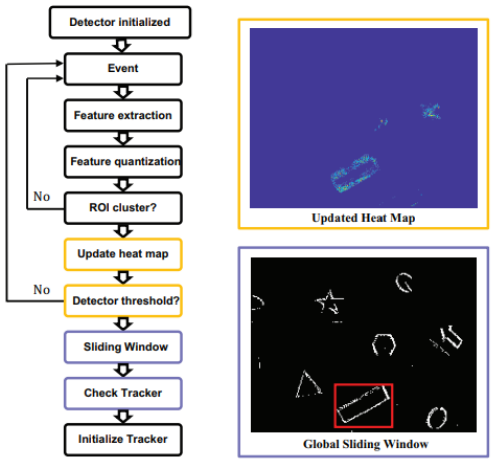


Figure 1: e-TLD model: Detector flowchart [5]

the region of interest is classified as an object. More specifically, when a region of interest is classified as an object, padding is created to extend the search area and the position of the object is updated with the region of interest having the highest score. If the tracker loses the object, a global sliding window search is performed to detect the object. Since the global sliding window search is more time-consuming than the local sliding window update of the tracker, it is used only when the tracking algorithm loses the object. Figure 1 shows the tracking model flowchart.

2.2 Real-time clustering and multi-target tracking using event-based sensors

Barranco et al. [6] proposed a model, which analyzes the event in real-time, using the Mean-Shift Clustering algorithm to determine the position of an object and the Kalman Filter to smooth the trajectory. The mean-shift method calculates the mean of the data distribution in the neighborhood of each point, shifting the computed mean until the processing converges. The events are processed in parallel, in batches, reducing the computational resource requirements. The trajectory of the center of mass is influenced by the number of events triggered, therefore a correction algorithm is required. The Kalman Filter is proposed, which uses a state vector (x,y) and velocity (v_x, v_y) for each center of mass, estimating the state vector from a measurement vector z_k, containing the position returned by the clustering algorithm. This model is highly influenced by the accuracy of the measured velocity, which depends on the type of Dynamic Vision Sensor. The clustering method has an F-score of over 0.9, reducing the computation cost by 88% compared to the frame-based method. The F-score is the harmonic mean between the precision 1 and the recall 2. Overall, the tracking error was 2.5 pixels.

$$Precision = \frac{True\_Positive}{True\_Positive + False\_Positive} \quad (1)$$

$$Recall = \frac{True\_Positive}{True\_Positive + False\_Negative} \quad (2)$$

Algorithm 1 Event-based particle filtering

```
for i = 1 → N do
    p_i ← p_start
    s_i ← 1
end for
R ← (t_0, p_start)
k_resample ← 0
for each event e_k happening at time t_k do
    for i = 1 → N do
        p_i ← p_i + M(p_i)
        s_i ← s_i + α f(e_k|p_i)
    end for
    if k ≥ k_resample + K then
        resample(s, p)
        k_resample ← k
    end if
    i_best ← argmin_{1≤i≤N} s_i
    append(R, (t_k, p_{i_best}))
end for
```

Figure 2: Algorithm proposed in [2]

2.3 Event-based Particle Filtering for Robot Self-Localization

The model proposed by Weikersdorfer et al. [2] uses a non-linear filter, namely a Particle Filter, to estimate the state of an observed system in time. For every step k, the algorithm, described in figure 2, maintains a list of particles, characterized by a score, which indicates how well the particles approximate the observation. The position of each particle is updated according to the motion model, which is a random diffusion with a normal distribution with zero mean and variance δ^2, depending on the features of the used sensor. Although this model can run in real-time, having minimal memory requirement, it does not consider all the features of events, such as the change of intensity.

Although multiple approaches are available for tracking objects in the event-based dataset, having good results, the models can be improved. Multiple objects can be tracked, computing clusters from the events which take place in a fixed time interval, similar to the approach presented in section 2.2. In addition, a non-linear filter, such as the Particle Filter can be applied to correct the trajectory of the centroids, which can be estimated with an error caused by the variation of the number of events available in time. The main advantage of using a non-linear filter is the fact that no real system is completely linear, a non-linear estimation has better results to approximate the observations in a real system.

Furthermore, the number of events generated for each object varies over time, a clustering method analyzing the events in real-time is a better approach, compared with the search within sliding windows [5] due to the robustness provided by the clustering method. More specifically, the clustering algorithm calculates the distance for each iteration without depending on the number of events generated in a given time frame, as the padding of a region-of-interest can do. A time frame is an event-based representation that encodes the events in a 2D map, using the following method: the coordinates (x, y) of an event are the indexes on the 2D map and the timestamp t represents the value.

3 Methodology

Object tracking has significant implications in computer vision, representing the approximation of the trajectory of

an object in the image plane over time [7]. An object is represented by multiple points, which are the events mapped to a 2D representation, with the event's coordinates (x, y) as the index on the 2D map and the timestamp t as the actual value. In order to perform the object tracking task, the center of mass of each object is computed and tracked over time. The center of mass, known as the centroid, is tracked instead of each point of the object because the number of events varies within time frames and a general description, which is guaranteed to be maintained in time, should be used.

Tracking a single object can be divided into two main sub-task, namely locating the object and determining its trajectory over time. However, a real system is usually represented by multiple objects and for this reason, multi-target tracking (MTT) is done. This task can be divided into three main parts: find the initial location, keep the identity of each object, and calculate the trajectory [8].

For this research both approaches were taken for object tracking, analyzing the Neuromorphic-MNIST dataset [3]. Firstly, the tracking algorithm was used for a single object in the image plane, described in Section 3.1, followed by a more general approach with multi-target tracking, presented in Section 3.2. A clustering algorithm was used to detect the moving objects and the event-based Particle Filter was applied to correct the position over time.

3.1 Single object tracking

In this section, the algorithm used to perform single object tracking will be described. A fixed time frame is used for the visualization of the object, multiple events recorded in the specified time frame representing the points of the object. Although the image plane contains one object, noise can be present and a clustering algorithm was used to detect the exact object, removing the noise.

A. Object detection

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) allows the detection of clusters with various shapes, being a non-parametric algorithm that assumes that a cluster is found in the high-density area [9]. The concept of this approach is the fact that a cluster is "a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples)" [10] [11], where a core sample represents a high-density area. The density linked to a point is computed by counting the points around that specific point, within a specified range. Clusters are represented by the points with a density above a given threshold [12]. One of the advantages of using this clustering algorithm is that it does not require a predefined number of clusters and it can discover clusters with arbitrary shapes.

In this paper, the scikit-learn Python library[2] was used for the implementation of DBSCAN. This implementation uses as a parameter a threshold value, epsilon, which represents the maximum distance between two samples to be considered neighbors. The parameter is approximated using the Nearest Neighbor algorithm. More specifically, the distances between each point and all the other points are computed, and the distances between each point and its closest neighbor are used to determine the optimal value of the epsilon parameter.

Initially, the events are divided into windows depending on a specified time interval. To detect the object in the image plane, the events, described by a location (x, y), a timestamp t, specified in microseconds, and a polarity p, which has the value 1 if there exists a change of brightness and 0 otherwise, are clustered. Each cluster represents an object and a predefined label is used to mark the noise. The aim of this clustering algorithm for single object detection is the fact that it can separate the noise from the actual object. More specifically, the event which is part of the object is assigned a cluster label, and the noise events are not considered in the cluster, having the predefined noise label. The centroid of a cluster, which is also an object, is computed using the formulas:

$$x_{centroid} = \frac{x_{min} + x_{max}}{2} \tag{3}$$

$$y_{centroid} = \frac{y_{min} + y_{max}}{2} \tag{4}$$

where (x_{min}, y_{min}), (x_{max}, y_{max}) represents the position of the events with the minimum, respective the maximum value of coordinates in a cluster and $(x_{centroid}, y_{centroid})$ is the position of the centroid for a given cluster. The centroid with coordinates $(x_{centroid}, y_{centroid})$ characterizes the object and it is tracked in time.

B. Tracking algorithm

The detection algorithm analyzes sets of events within a time frame and the number of events varies in time, leading to possible missing detection. For this reason, the position of the object, as well as its trajectory, should be corrected using an estimator. Most of the real systems are non-linear and a non-linear estimator, such as the Particle Filter, is preferred over a linear one, like the Kalman Filter, because it allows for a better approximation of non-linear systems. Furthermore, Particle Filter performs better for systems with non-Gaussian distribution and its estimation error can converge to zero, with the increase of the number of particles [13]. Also, it is important to notice that adaptation of the Kalman Filter exists to estimate non-linear systems, but their estimation errors do not converge to zero. The advantages of using a Particle Filter are important for event-based object tracking due to the representation of events in time frames, which can generate a

[2]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

different number of events, with various distributions, which can be represented by non-linear systems.

The Particle Filter, which implements a Bayesian estimator was applied in this research A Bayesian estimator is based on the Bayes' Rule, aiming to approximate the conditional probability density function of a state based on some measurements [13]. The filter algorithm maintains a set of particles, which are the hypothetical system states, to produce an estimation for the measurements. The hypothetical system states represent the points used to approximate some observations.

The general Particle Filter algorithm is computed by 5 steps:

- Randomly draw N particles p_i^k from a uniform distribution, with equal score s_i^k
- At each step k:
  - Predict the position of p_i^k using a predefined motion model
  - Update the score s_i^k, computing the probability $pr(s_i^{k1}|y_k)$, where y_k is the observation
  - Resample if necessary
  - Estimate the current position using the weighted average of particles coordinates

In the proposed model, a set of N particles p_i, representing the hypotheses over the current state of the system, is maintained. Each particle p_i is characterized by 2 parameters: location with coordinate (x, y) and a rotation value. Also, for each particle, a score is memorized and computed. The score shows how well the particles approximate the observation. Initially, the particles are randomly selected from a uniform distribution and they have an equal score. At each step k, their new position is estimated, considering a motion model, and the score is updated, using the distance between each particle and the observed point. The motion model follows the equation described in the paper **[?]** and it is a random diffusion using a normal distribution N(a^2) with zero mean and variance a^2. The average velocity of events is included in the motion model, to guarantee the movement of the particles.

$$p_i^{k+1} = p_i^k + M(p_i) + \alpha \tag{5}$$

$$M(p_i) = (N(\delta_r^2), N(\delta_r^2), N(\delta_\theta^2))^T \tag{6}$$

where δ_r represents the average moved distance for one event, depending on the sensor features, and δ_θ is the average rotation per event. α depends on the velocity of events on the x and y axes.

One important step in this algorithm is resampling because it ensures that particles with low scores are not used for the estimation of the observation. The normalized value of the score is used in the resampling and the result of resampling is a set with high score particles, which is a more accurate representation of the distribution. The duplication within particles is decreased by the usage of a normal distribution in the predict step, which can be seen as added noise. Sampling Importance Resampling was used, which involves a systematic resample algorithm. This method divides the samples into segments and it uses a single random offset to choose where to sample from all divisions, ensuring that the samples are equally apart. The weights are re-initialized after resampling, the particles having equal scores. The resampling is done when the effective particle count is smaller than a fixed proportion of the total number of particles to guarantee that the distribution has modified significantly.

The last step of this algorithm is to estimate the point which was analyzed from the measurements. The result of one iteration of this algorithm is the point calculated as the weighted average of the particles. Since the detection part of this model analyses events for each time frame, the Particle Filter considers as the observation the centroids provided by the clustering algorithm for each time frame. For this reason, the weighted average of particles' position represents the actual position of each object at different time frames.

3.2 Multi target tracking

In order to consider a better representation of a real system using event-based data, a multi-target tracking model is proposed. The moving objects are computed using a clustering algorithm, which detects the high-density areas in the image plane. Each cluster represents a moving object, described by the events within a time range. The centroid of each cluster will be tracked in time, instead of tracking each point of the object, because the events are not guaranteed to be constant in time, as the point of a frame-based object could be. Similar to the single object tracking from section 3.1, the clustering algorithm detects the object and a non-linear filter, namely the Particle Filter, is applied to smooth the trajectory of the center of mass.

A. Object detection

One of the challenges in multi-object detection is represented by the overlapping of objects and the order-preserving of the clusters. To minimize these possible effects, a Mean Shift Clustering Algorithm was used. The algorithm was selected over the DBSCAN one for multi-targets tracking because it does not require prior knowledge about the shape of the objects and it has a better estimation of the center of mass position, which represents the tracking parameter for each object. For this clustering algorithm, the implementation provided by Scikit-learn[3] was used. This centroid-based algorithm modifies the candidate of a centroid to be the mean of the points from a specified area. After the clusters are computed, the candidates are filtered to remove the near-duplicates. The size of the searched region is represented by a bandwidth parameter and the update for a new sample is done by obtaining the nearest centroid for a given sample.

[3] https://scikit-learn.org/stable/modules/clustering.html#mean-shift/

The mean shift vector computed for each centroid is the mean of the samples within its neighborhood, as specified in the formula:

$$m(x_i) = \frac{\sum_{x_j \in A(x_i)} Kernel(x_j - x_i) * x_j}{\sum_{x_j \in A(x_i)} Kernel(x_j - x_i)} \quad (7)$$

where $A(x_i)$ is the neighborhood of samples in an area of a given radius from x_i.

The events, characterized by the location (x, y), timestamp t, and polarity p, are divided into time windows, and the clusters, representing the moving objects, are computed from these events, ensuring that the order of clusters within consecutive time frames is maintained. Before applying the clustering algorithm, an anomalies detector, Isolation Forest [14], was used, and the noise is removed from the image plane. This method constructs a tree by recursively dividing the data and selecting a random attribute and a splitting value until the tree has reached a height limit or all data has the same value. The data points are sorted according to their path length, anomalies representing the points that are ranked at the top of the sorted list. A cleaned list containing the events, which are not considered anomalies, is then provided to the clustering algorithm.

B. Objects tracking

Similar to the algorithm described in Section 3.1, the Particle Filter was implemented to smooth the trajectory of the objects over time. Initially, a set of particles is generated for each cluster found in the first time window, and then for each step k, the new clusters are computed, maintaining the initial order of objects, and the Particle Filter algorithm is applied. The main steps of the Particle Filter implementation are:

- For each object o_i, initialize a set of particle SP_i
- For each object o_i:
  - Predict new position of the particles in SP_i
  - Update the score of particles using center of mass of o_i
  - Estimate the position of that object, using the weighted average in SP_i

The general tracking algorithm is provided in Algorithm 1.

Algorithm 1 Object tracking algorithm

Input: Set of events divided in B time frames
Output: Estimated position of objects in time

Compute clusters C_0 for the events within the first time frame
for each centroid c_i in cluster list C_0 **do**
  Generate a set of particles SP_i with particles p_j and score s_j
end for
for $k = 1 -> B$ **do**
  Compute new clusters C_k
  for each centroid c_i in cluster list C_k **do**
    1. Predict the position of each particle p_j in SP_i
    2. Update the score s_j of each particle p_j in SP_i
    3. Resample the set of particles SP_i
    4. Estimate new point using the weighted average of the particles in SP_i
  end for
end for

4 Results

This section presents the results of both single- and multi-object tracking, the following metrics are used: tracking visualization, the accuracy of the model using a manually annotated ground truth set, and different clustering evaluation metrics, including Silhouette score, Calinski&Harabasz Index, and Davies-Bouldin Index. To interpret the performance of the provided model, the Neuromorphic-MNIST dataset was used [1] and a new dataset containing multiple digits as a frame was generated for the evaluation of multi-target tracking, using the model described in [15].

Visualization of the tracking algorithm

The results of the proposed object tracking model can be visualized in Figure 3, which shows the evolution in time for both a single object and multiple targets. Furthermore, Figure 4 presents the impact of the Particle Filter on the position estimation, showing the difference between the computed centroid using the clustering algorithm and the approximated centroid using a non-linear Filter. It can be noticed that initially, the particles did not estimate the accurate centroid of the object, but with the modification of the particles' scores, the approximation is improved and the centroid of the object is correctly tracked.

Another significant aspect of the tracking visualization technique is represented by the representation of the particles, updated for each object, and their movement. In Figure 9, the motion of the particles, considering the average distance moved by an event, as well as its rotation. Only a sample of particles is shown in this figure, the arrows describing the movement and the position of the particles related to the objects.

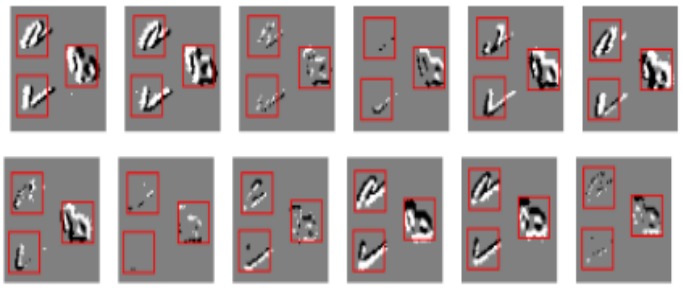


Figure 3: Visualization of the multi target tracking over time

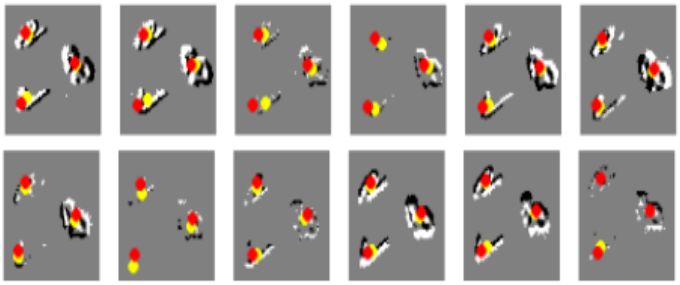


Figure 4: Difference between the centroid computed by the cluster algorithm (yellow dots) and the centroid estimated by the particle filter model (red dots).

Clustering accuracy score

Three different clustering performance metrics were used, namely the Silhouette coefficient, Calinski&Harabasz Index and Davies-Bouldin Index. As mentioned in Section 3.2, the DBSCAN algorithm was replaced by the Mean Shift Algorithm due to the advantages the second one presents, including a better computation of the centroids and the usage of kernel density estimation. To guarantee that the Mean Shift Algorithm performs better than the DBSCAN one, for multi-targets detection, a comparison between the results of each cluster approach for the same datasets was done, the results being presented in Figure 5. For the Silhouette score and Calinski&Harabasz Index, higher values show better results, while for the Davies-Bouldin Index lower values are representative of better clustering.

Clustering performance metric	Clustering Algorithm	Clustering average score
Silhouette score	Mean Shift	0.739
	DBSCAN	0.659
Calinski & Harabasz Index	Mean Shift	3263.1
	DBSCAN	1731.5
Davies-Bouldin Index	Mean Shift	0.343
	DBSCAN	0.455

Figure 5: Clustering performance evaluation for multiple objects in the image plan. Mean Shift algorithm has a higher accuracy.

Accuracy

The accuracy of this model was tested using the manually labeled ground truths for the N-MNIST dataset, bounding boxes being created around the objects. The Euclidean Distance between the center of the bounding box and the centroid estimated by the proposed object was calculated and the distance error for the tracking was 4.5. The accuracy of single object tracking is 0.7 and for multi-targets tracking, it is approximate 0.61.
The results of the Particle Filter depend on the number of particles selected to estimate the observed position. Figure 6 shows the link between the accuracy score and the number of

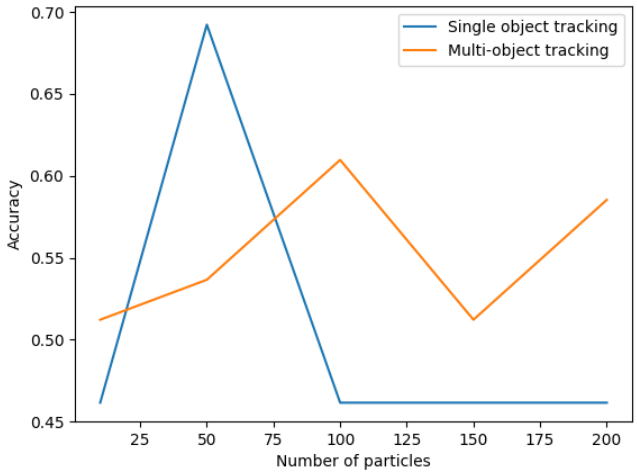


Figure 6: Variation of accuracy considering different number of particles using the N-MNIST dataset. The best result for multi-target tracking is around 0.6, generated using 100 particles for each object. The highest accuracy for single object tracking is 0.7, 50 particles being used to estimate each point.

particles used to estimate each centroid. The motion model used to predict the position of the particles operates with the average distance moved by an event, the average rotation, and the mean velocity of objects. In Figures 8a and 8b histograms with the velocity distribution can be observed. To calculate the velocity, for each object, the distances between the centroids in consecutive time frames on both x and y axes were computed, using the formula 8 and 9. The velocity used in the particle motion model was the mean of the histogram, on axis x, as well as y.

$$v_x = \frac{|x_i - x_{i+1}|}{time\_frame\_length} \tag{8}$$

$$v_y = \frac{|y_i - y_{i+1}|}{time\_frame\_length} \tag{9}$$

where i is the time frame index.

Another factor which influences the accuracy of the tracking is the time frame length. Increasing the interval of time, more events are considered and the features of objects are easier to be followed. The variation of accuracy relative to the time frame length is shown in figure 7.

5 Discussion

Analyzing the results obtained for the model provided, it can be seen that it is possible to perform object tracking on data generated using an event-based camera. Considering that the events generate objects with Non-Gaussian distribution, the results show that the Particle Filter Algorithm can be used to estimate the evolution of objects' movement in time, using the position of the centroids. One important aspect to notice is the fact that several available pieces of research apply the

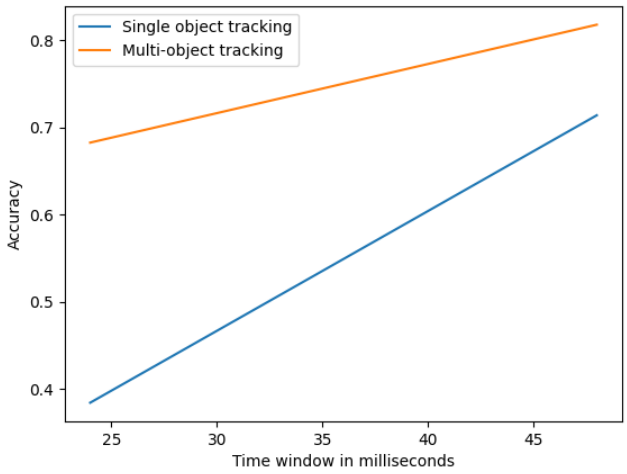


Figure 7: Variation of tracking accuracy relative to the different frame lengths. The highest accuracy for both single and multi object tracking is for the frame length of 48 milliseconds, with score 0.72 for single object and 0.82 for multi target tracking

Kalman Filter to smooth the trajectory of objects. However, for this research the Particle Filter was applied due to several advantages offered by this filter:

- It is a non-linear estimator, with better results on real networks, which are mainly nonlinear systems
- It can estimate non-Gaussian distribution
- It approximates the tracked centroid using multiple points, known as particles, resampling to ensure an accurate distribution over time.

Although the results of the algorithms show that the tracking algorithms can be used on an event-based dataset, with good accuracy, a non-linear filter is more complex than a linear one. It predicts the position using a motion model, which depends on the velocity of events, the average distance moved by an event, and its rotation. These variables are influenced by the parameters of the dynamic vision sensor, having a high impact on the accuracy of the results. For the N-MNIST dataset, these parameters are not clearly described, being a limitation for the implementation of these experiments. Also, the computation of an accurate velocity is not possible without having all the specifications of the camera, for this research the velocity of events is estimated using the movement of centroids. Furthermore, the results can be influenced by the initial sampling of the particles, which are randomly selected from a uniform distribution. For this research, the particles are selected from a uniform distribution within the range: $(x_{centroid} - \alpha, x_{centroid} + \alpha)$, respective $(y_{centroid} - \alpha, y_{centroid} + \alpha)$. Also, multiple resampling methods can be used for this estimator. The one used proposed in this paper, namely the systematic resampling, has good results compared with other tested methods.

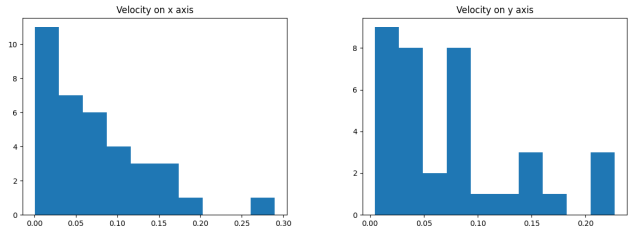
Another aspect that influences the overall results is the dataset which is not a complete representation of the real world, the analysis of the model is done on a toy dataset. However, the accuracy of the presented results was improved by creating a dataset with multiple randomly positioned digits in a frame. This dataset was built using the approach described in the paper [15]. This modified dataset shows the results for multitarget tracking, the initial model being updated to improve the outcome. For this research, initially, it was decided to implement Density-Based Spatial Clustering of Applications with Noise(DBSCAN) to detect the clusters, representing the moving objects. However, as was shown in Section 4, the Mean Shift clustering algorithm had better accuracy compared with the DBSCAN algorithm. Also, the Mean Shift algorithm is implemented to consider the center of mass when dividing the clusters, providing a better computation of the centroids, as well as a more precise computation of the clusters considering the variation of the number of events over time. To improve the model, in terms of run time efficiency, batches of events were analyzed together and divided into windows with a specific time interval.

The questions mentioned in section 1, have been analyzed to analyze the performance of the object tracking model proposed as the solution of this research. The algorithms are implemented to have good results on different data, generated from the events, including the non-linear systems, without a Gaussian distribution, which is usually the case of the event-based camera output. Moving targets can be detected using the clustering method and one of the best characterization of an object is the center of mass of the cluster, representing that object. The centroid is tracked over time because it can be computed from events within a time frame, irrespective of the number of events, while tracking each point of the object, namely each event, is not possible because of the asynchronous behavior of dynamic vision sensors. Analyzing the system distribution, a Particle Filter is used for trajectory correction because it is a robust estimator.

Overall, both single and multi-target tracking can be done on asynchronous event-based data and the results depend on the level of noise in the image plane, which influences the clustering accuracy. Another factor influencing the results is the motion model used to predict the movement of particles, influencing how well the trajectory of the objects is estimated by the Particle Filter. The datasets used to test the provided algorithms are composed of non-Gaussian distributions, simulating a real nonlinear system, but it does not entirely simulate the real world.

6 Responsible Research

The model provided in this research aims to track objects in event-based videos. The results of this model are presented in section 4 and they are discussed in section 5. The advantages of using the proposed model, as well as its limitation are



(a) Velocity x (b) Velocity y

Figure 8: Velocity variation on axes x and y for centroid for a sample of the dataset. Mean of histogram for axis x is 0.07, and for axis y is 0.08

presented and the reasons for selecting specific algorithms are described in detail. No ethical bias can be noticed in this model because it analysis each event and it does not use a pre-trained model, which can be biased. The experiments are designed to have as input objects represented from events, characterized by a location with coordinate (x,y), timestamp t, and polarity p and some features cannot be easily recognized by humans because the events might be sparse.

One of the main concerns when conducting research and implementing a model is related to the usage of existing materials. More specifically, the created model should not completely copy one of the available algorithms, but it should be a new solution, which can extend known methods. For this research, the model implemented in section 3 is built by two existing concepts, namely the clustering algorithm and a non-linear estimator, combining them and extending the Particle Filter Algorithms to use it on the event-based dataset. Consequently, no ethical issues can be derived from the implementation of this model.

Furthermore, the dataset used for testing the model is N-MNIST, which is released under the Creative Commons Attribution-ShareAlike 4.0 license. For this research, the dataset was not altered and it was mentioned as its source. Also, it is mentioned that the construction of this dataset was done in the absence of conflict of interests, showing that both from the perspective of this research and the initial construction of the database, no ethical issues are remarked. Also, the dataset contains handwritten digits and no personal information can be obtained from the data.

The results of this research can be reproduced following the methods described in 3 and the dataset [1] used for evaluating the performance of the proposed model is publicly available. All data points and features of the dataset were used and a pseudo-code was provided to ensure that the approach can be correctly replicated. Moreover, the implementation of the proposed model is available at the link and proper documentation was added to the code, to allow the reproduction of experiments.

7 Conclusions

An event-based model was proposed for object tracking, analyzing its performance on both single- and multi-target tracking. The algorithm was tested on two different datasets, both based on the Neuromorphic-MNIST dataset, simulating a nonlinear system, without a Gaussian distribution. The moving objects are detected using a clustering algorithm and an anomalies detector is applied, before computing the clusters to eliminate the noise as much as possible. Each cluster represents a moving object and its center of mass is computed and tracked over time. Lastly, a non-linear estimator, namely the Particle Filter is used to smooth the trajectory of objects and correct detection errors. After an in-depth examination of the event-based dataset, the conclusion was made that the Mean Shift clustering algorithms can provide better results for multi-object tracking.

Future work can be done to investigate the performance of an object tracking model in combination with a classification algorithm, focusing on tracing specific classes of objects. Also, the sampling of the estimator used for tracking the movement of the objects over time can be updated depending on the dynamic vision sensor used and more precise analysis of the velocity of events can be used to improve the accuracy of the model.

References

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[2] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 866–870, 2012.

[3] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, 2015.

[4] E. Piatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz, "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 35–40, 2012.

[5] B. Ramesh, S. Zhang, H. Yang, A. Ussa, M. Ong, G. Orchard, and C. Xiang, "e-tld: Event-based framework for dynamic object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3996–4006, 2020.

[6] F. Barranco, C. Fermuller, and E. Ros, "Real-time clustering and multi-target tracking using event-based sensors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5764–5769, IEEE, 2018.

[7] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, pp. 13–es, 2006.

[8] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021.

[9] T. N. Tran, K. Drab, and M. Daszykowski, "Revised dbscan algorithm to cluster data with dense adjacent clusters," *Chemometrics and Intelligent Laboratory Systems*, vol. 120, pp. 92–96, 2013.

[10] "2.3. clustering." https://scikit-learn.org/stable/modules/clustering.html#dbscan.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & knowledge engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[13] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[15] A.-D. Manolache, "Constructing complex event-based segmentation datasets," 2022.

A Appendix

A.1 Figures

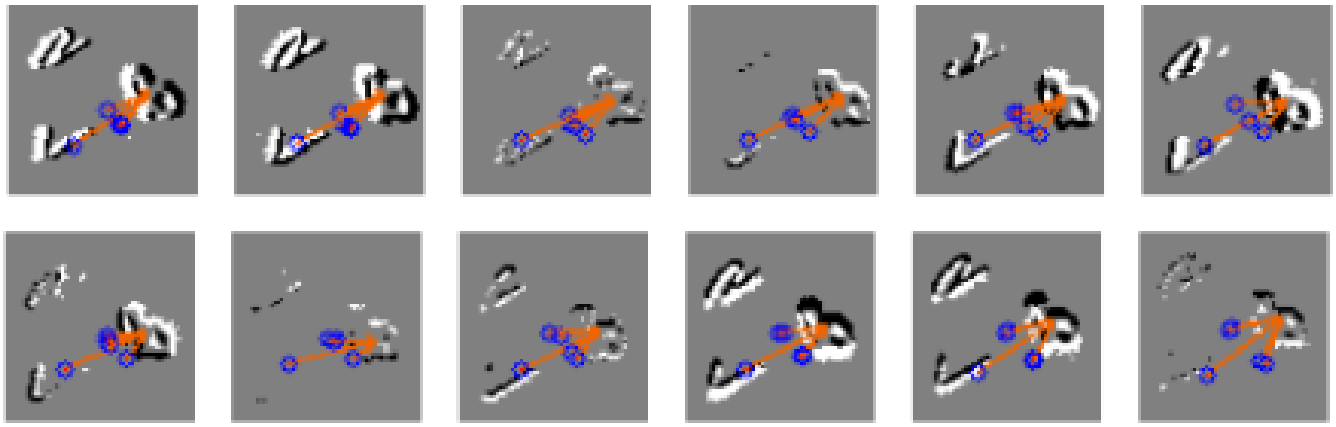


Figure 9: Sample particle movement for one of the objects in multi target tracking. In orange the trajectory to the center of mass of the object can be seen. The movement of particles can be seen in blue and the actual position of the particles in red.

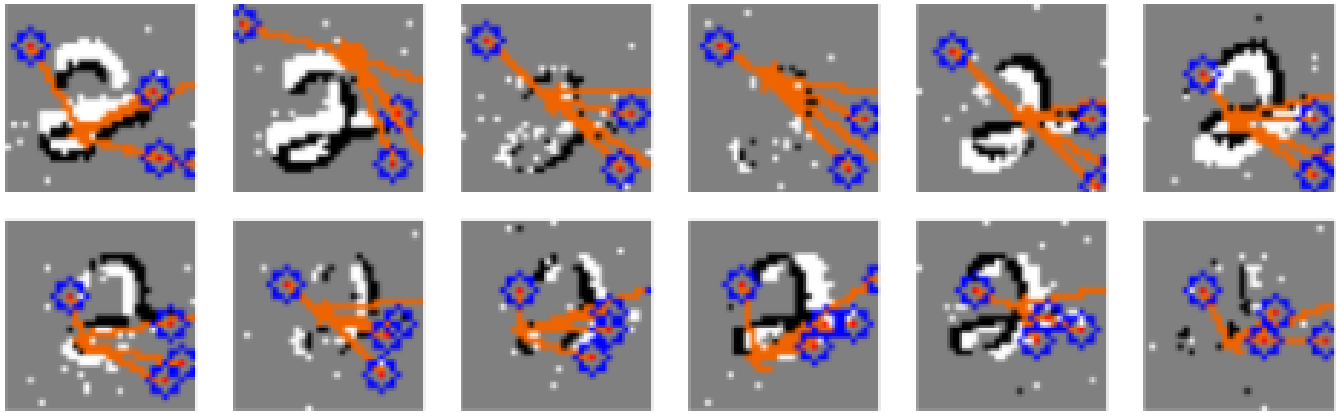


Figure 10: Sample particle movement for one object in the image plane. In orange the trajectory to the center of mass of the object can be seen. The movement of particles can be seen in blue and the actual position of the particles in red.