EVENT-BASED CLUSTERING AND LOOMING DETECTION

BEHNAM KAMRANIAN
Bachelor of Engineering, University of Najafabad, 2015

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

EVENT-BASED CLUSTERING AND LOOMING DETECTION

BEHNAM KAMRANIAN

Date of Defence: November 5, 2019

Dr. Howard Cheng Thesis Supervisor	Associate Professor	Ph.D.
Dr. John Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Matthew Tata Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Habiba Kadiri Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

To my parents and sister.

Abstract

Based on the sequential K-means algorithm, we present a real-time, accurate and automatic clustering method for asynchronous events generated by the optical flow algorithm of Ridwan and Cheng. The complexity of our algorithm does not increase with increasing number of events. We also designed an implementation of the elbow method capable of detecting the number of clusters without any a priori assumptions on objects. In addition we designed a merge algorithm capable of merging multiple touching clusters into one for enhancing the results of our clustering algorithm. The output of our clustering algorithm is then used with a single object looming detection algorithm to detect looming for multiple objects. We tested our algorithm on both simulated and captured data sets against two other well known algorithms. Our algorithm is fast and accurate both in cluster detection quality and looming detection quality.

Acknowledgments

First and foremost, I would like to extend my sincere gratitude to my supervisor, Professor Howard Cheng. Howard, I am truly honored and privileged to have conducted my research studies under your supervision. I can never thank you enough for taking a chance on me and teaching me what I know today. You have been both the most influential individual I have ever met and a real source of inspiration for me. You were not only a supervisor, but also an academic father who shared his broad knowledge with me that led to authentic research. I will consider the first day (14^{th} of September 2017) I met you as my second birthday. I will remember you as a symbol and example of a truly professional and at the same time kind and heart warming individual forever.

I would like to further extend my gratitude to my committee members, Dr. John Zhang and Dr. Matthew Tata, for their guidance and support throughout my research. Thank you Matthew for accessing me into your lab.

I also want to thank Dr. Hadi Kharaghani and Dr. Amir Akbary for their spiritual and technical supports to me all the time. I am truly honored of living in an environment which I could see you and talk to you for two years. Moreover, I would like to thank Sean Legge for his kindness and support all the time. I will really miss talking with you during coffee breaks.

The deepest thanks and appreciation go to my family. Words cannot express how grateful I am to them for all of the sacrifices that they have made on my behalf. Your prayer for me was what sustained me this far.

Many thanks to anyone else who supported me along the way, and to anyone else reading this thesis. ☺

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We use our eyes as part of our visual system to collect environmental information and transmit it to the processing part of the body, our brain, to interpret the surrounding world. Computer vision is a field that aims to provide computers with the same or even better capability. Computer vision is a multidisciplinary scientific field in which algorithms are designed to achieve an acceptable understanding through videos and digital images. These algorithms are developed to automatically extract useful information for automatic visual understanding. Video processing has many applications such as object detection and tracking, motion detection, video enhancement, navigation, etc. These applications can be used for different purposes such as robotics in which video cameras can be used as a sensor for the environment. There are many problems that need to be addressed and many algorithms that need to be developed to expand the area of video processing for use in computer vision and robotics. Detecting objects which are looming is one of these problems.

According to the study of perception, looming is an optical phenomenon in which the size of a given object rapidly expands [51]. When the object gets closer to the viewer, its image gets larger on the viewer's retina over time and a physiological response is triggered automatically to perceive that object as an approaching object. Looming detection is essential both in nature and robotics. Since fast looming movements are often dangerous, systems should be able to detect such situations. Therefore, looming detection does not focus on what is looming but the fact that something is approaching. Looming detection is also important in robotics and intelligent vehicles to give them the capability of avoiding

risky situations. Previously, conventional frame-based cameras have been used for processing in such situations. However, since their frame rate is relatively low, the reaction time of processes such as looming detection is limited. Also, because these cameras are not motion dependent, they detect and report all data regularly which wastes both electrical powers, computational resources and storage. Other techniques such as identifying the region of interest and ignoring the background is not optimal due to the extra computation, power and storage.

We also use our brain to distinguish different objects everyday. We cluster objects with similar features into groups in a way that the objects in the same group have similar features and the objects in different groups have different features [20]. There are many clustering algorithms developed for use within intelligent systems in different areas such as marketing, land use, city-planning, etc. The application of clustering in image processing, machine learning and robotics can lead to creating intelligent systems capable of distinguishing, tracking and avoiding objects and obstacles. Clustering is an unsupervised classification process and it can be used both as a stand-alone tool to get insight into data distribution or as a pre-processing step for other algorithms.

Neuromorphic engineering is a field of study in which biological nervous systems are studied to be simulated by Very Large Scale Integration (VLSI) systems [39]. The goal of this field is to replace transistor-based electronic circuits with neuron architecture inspired by human brain. The Dynamic Vision Sensor (DVS) is such a system simulating the human retina [36]. The DVS is an event-based camera which only transmits significant changes in log-luminance of any individual pixel. These changes are called events and may be detected and reported asynchronously by the camera as they happen. Therefore, in contrast with conventional frame-based cameras, these cameras can react to events faster while their electrical and computational power requirement is lower. However, the developed algorithms for frame-based cameras cannot be applied to theses cameras and new algorithms need to be developed to preserve the advantages of these cameras.

The goal of this thesis is to provide a real-time and automatic solution for the problem of detecting multiple looming objects. The proposed algorithm clusters the objects in the scene using clustering techniques adapted for asynchronous event-based input data transmitted by event-based cameras. The events in each cluster are analyzed to determine if the cluster contains a looming object.

1.1 Contribution

In this thesis, we proposed a real-time clustering algorithm based on the sequential K-means algorithm capable of clustering the optical flow-events generated by Ridwan and Cheng's optical flow algorithm [50]. Our algorithm is automatic and neither parameter adjustment nor a priori assumptions on the shape and number of objects in the scene are required. In addition, our algorithm does not require special hardware or parallel processors and can be run in accessible systems.

We used the output of our clustering method with Ridwan's single object looming detection algorithm [49] to solve the problem of looming detection for multiple objects. During our experiments on both simulated and captured data sets, we realized that Ridwan's algorithm cannot detect looming while the object is looming and moving sideways at the same time.

Beside our clustering algorithm, we implemented two other well-known clustering algorithms (K-means and mean shift algorithms) to evaluate our algorithm against them. We also designed a method capable of detecting the number of clusters without any a priori information about the objects. We called this method the elbow method. Last but not least, we designed an algorithm which merges multiple clusters to form a single one and improve the accuracy of the final clustering decision.

We show that a combination of using our proposed algorithms provides us the most accurate results in the shortest processing time. We also show that the complexity of our algorithm does not change if the number of events is increased.

1.2 Organization of Thesis

The background and motivation of this thesis is discussed in Chapter 2. In Chapter 3, our methods and the way we adopted different clustering algorithms for using with the event-based camera are described. In Chapter 4 we present the experimental results for both simulated and captured data sets. Finally in Chapter 5, the conclusion and possible future works in this area are discussed.

Chapter 2

Background

In this chapter, some background information and definitions are given. The conventional frame-based cameras are reviewed. Next, the Dynamic Vision Sensor is introduced, and the associated Address Event Representation is also described. Next, the problems of optical flow and looming object detection are introduced, including a brief overview of some of the previous algorithms for both frame-based cameras and the Dynamic Vision Sensor. Finally, a review is given for clustering algorithms that will be used in this thesis.

2.1 Conventional Frame-Based Cameras

Traditionally, a video is represented by a series of successive images called frames. Each of these frames contains an image taken by an array of imaging sensors. This is how conventional cameras record and represent videos. Although these cameras are commonly used and many algorithms have been developed for videos captured by these cameras, there are a number of drawbacks associated with them.

Conventional frame-based cameras have a frame rate, which is the number of frames they can capture per second. For example, typical cameras capture 24—30 frames per second (FPS). Depending on the type of camera, there is a frame captured every 30 to 40 milliseconds, even if the scene is unchanged or few changes are presented. This approach produces redundant data. Computational and electrical power are required to process them. Also, the amount of space required to store them is significant. Using compression techniques to deal with redundant data requires even more computational resources. Since these

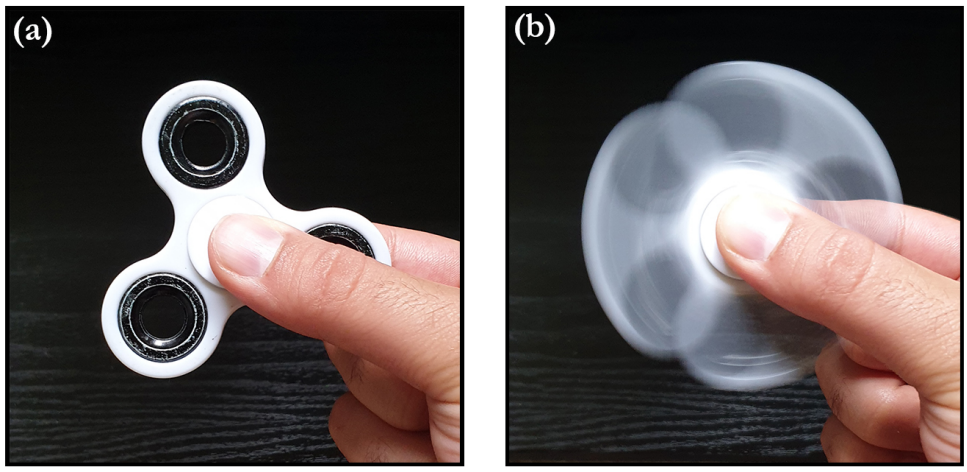


Figure 2.1: Motion blur in conventional cameras. (a) A fixed spinner. (b) A frame of a recorded video from a spinning spinner. Since the speed of the spin is higher than the frame rate of the camera, the motion is blurred.

cameras have to send captured frames synchronously at regular time intervals, there is a delay between the time the scene is changed and the time the data is captured and available for processing. This delay is called latency and it is important because it is the delay between what is happening in front of the camera and what is processed. When applications such as robotics or autonomous vehicles require a fast response to the environment, using a low-latency camera is important. On the other hand, higher latency can be tolerated for processing recorded videos. Since some changes may happen between two successive frames, motions at speed higher than the frame rate will be blurred and the algorithm processing the video cannot respond faster than its frame rate (Figure 2.1). This is known as aliasing and it happens when the frequency of changes is at least two times faster than the Nyquist rate [24]. When the range of intensity in the scene is high, it is highly probable that some details are lost using conventional frame-based cameras. For instance, most cameras cannot simultaneously capture both very bright and dark objects in acceptable details. Typically settings are available (e.g. aperture of the lens) to control the amount of light that reaches the sensors, and the setting is applied globally to all pixels. If there are both very bright or very dark objects in the same scene, the camera settings have to be tuned to allow either

Figure 2.2: An example of low dynamic range in conventional cameras.

the bright or dark object to be seen. Thus, we say that these cameras have low dynamic range. In Figure 2.2 there are two different images taken from the same scene with different settings in which only very bright or very dark objects can be seen in a single image. However, High Dynamic Range (HDR) is a technique that regular cameras use these days to solve this problem. In this technique, by taking many images with different exposure settings, a detailed image is obtainable over all brightness situations [13].

In summary, conventional frame-based camera have low dynamic range, produce large amounts of redundant data, use large amount of memory access and disk space to deal with these redundant data, and waste computational power, energy and time. Researchers have

Figure 2.3: The Dynamic Vision Sensor (DVS).

started to look for different imaging techniques capable of overcoming these drawbacks.

2.2 The Dynamic Vision Sensor

Neuromorphic engineering is a multidisciplinary field in which artificial neural components are built to mimic biological neurons [41]. The Dynamic Vision Sensor (DVS) (Figure 2.3) is a neuromorphic camera which behaves similar to the human visual system by modeling the human retina [36]. By considering the biological retina as a visual system capable of transmitting relevant information asynchronously [23], researchers found the solution of overcoming the drawbacks of conventional frame-based cameras.

There are key differences between conventional frame-based cameras and neuromorphic event-based cameras such as the DVS. Unlike frame-based cameras which collect frames and transmit them synchronously at a fixed frame rate, the DVS does not wait to collect frames and asynchronously transmits events as soon as each event occurs. An event occurs when the log luminance detected by a pixel changes by more than a predefined threshold. The output of the DVS is directly related to significant log luminance changes in a pixel. These changes can be caused by light changes as well as motion. When an object moves, significant changes happen usually only near its edges. In order to detect

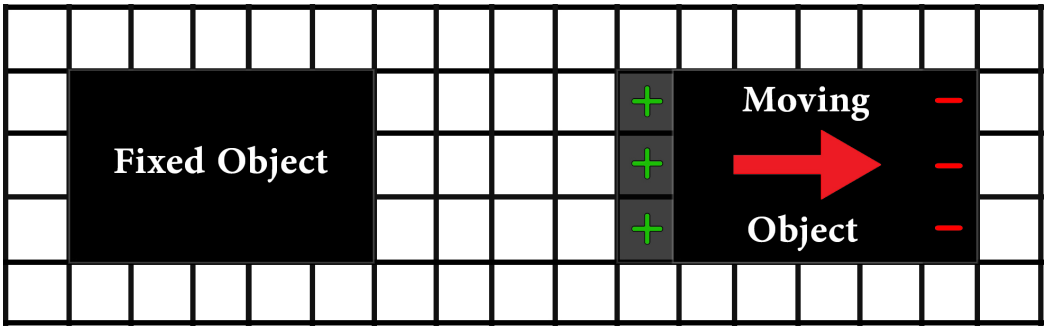


Figure 2.4: A simple horizontal move can cause events to be generated by the DVS.

movement, the log luminance of the object and the background need to have a considerable difference. By assuming a relatively constant background, as the object moves in the same direction for a period of time, its leading edge will cover the pixels of the background and its trailing edge will expose the background (Figure 2.4). Assuming the background is brighter than the object, the background has a higher log-luminance compared to the object. In this situation, the intensity changes at the leading edge are negative. While the changes at the trailing edge are positive. The situation is similar if we have a background with lower log-luminance than the object.

When there are no changes in the log-luminance of the scene the DVS is focused on, there is no output. As soon as a significant change in the log-luminance of any pixel occurs, the DVS reports an asynchronous event which is described by the coordinates of the pixel's location, the timestamp of the event and the polarity $(+/-)$ of the change. Although the event polarity indicates whether the change is from dark to bright or vice versa, the magnitude of the change is not available. Other information gathered from motion sensors provided in the DVS are also reported, but they are out of the scope of this thesis.

In the DVS, each pixel has an individual sensor. Each sensor only compares the current log-luminance to its previously triggered level. Therefore, each pixel can adapt to its own intensity because they are independent from the other pixel sensors. After detecting a significant difference between each pixel's intensity and its own previously triggered level, the sensor transmits an event. This is why the dynamic range of the DVS is very high—each

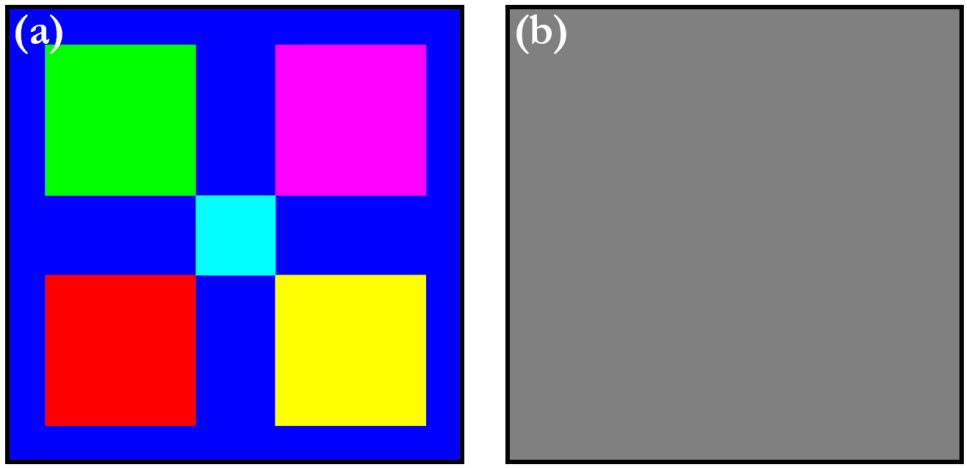
pixel adapts to the local luminance independently of other pixels. Even if one area is very bright and another area is very dark, significant log-luminance changes in both areas are still detected.

Reporting events asynchronously has many advantages over collecting frames. First of all, the DVS reports events as soon as they are detected by the pixel sensors. This makes the latency of the camera very low and makes it suitable for recording super high-speed motions with latency of microseconds. Also, unlike frame-based cameras which collect frames regardless of the amount of changes happening in the scene, the DVS only reports the significant changes in the log-luminance of the pixels and it does not produce any redundant data when no change occurs. This approach reduces power consumption as well as computing resources for processing the generated data and the amount of space needed to store them drastically. Additionally, unlike frame-based camera which have low dynamic range, having a high dynamic range makes the DVS less sensitive to special lighting conditions.

Working with the resulting event stream, however, can be challenging. Since the DVS reports events asynchronously, these events can come from any part of the image and they are in an unpredictable order. Therefore, algorithms need to process them in real-time. Also, instead of reporting the intensity, only the polarities of the changes are reported. Therefore, the only visible part of the object is its outlines, which if required, makes the visualization of the object difficult (Figure 2.5). Although some methods have been developed to reconstruct the original video by gathering the reported streams of the DVS [3, 30, 32], extra computation power and movement in the scene are required to obtain an approximation of the original object.

In addition, in situations in which the object and the background with different colours lead to the same luminance (Figure 2.6), no movement will be detected. This is because the DVS is not able to distinguish the background from the foreground object. Moreover, if the background and the object have similar colours, detecting motion in this situation is difficult even for human [38]. Since the DVS is modelled from the human retina, it cannot

Figure 2.5: Examples of the output of the DVS. The green dots indicate increases in the log luminance of those pixels (called ”positive changes”) and the red dots indicate decreases in log luminance (”negative changes”).



(c)

Color	(H,S,L)	(R,G,B)
	(0°,100%,50%)	(255,0,0)
	(120°,100%,50%)	(0,255,0)
	(240°,100%,50%)	(0,0,255)
	(60°,100%,50%)	(255,255,0)
	(180°,100%,50%)	(0,255,255)
	(300°,100%,50%)	(255,0,255)

Figure 2.6: Colours with the same luminance. (a) The original image. (b) The luminance of the image. (c) The luminance of the colours.

be used for detecting movements in such situations.

2.3 Address Event Representation (AER)

The Address Event Representation (AER) [6] is a communication protocol between neuromorphic systems. It is an asynchronous transmission protocol that transmits the generated outputs from the generator to the receiver using a narrow communication bus as well as a high-speed multiplexor [17, 40]. It takes advantage of the low frequency of events generated by the neuromorphic system and transports them through a high-speed bus. In computer systems, the bus is a channel through which two or more components or computers are connected and communicate with each other. When each event is generated by the DVS, the involved pixel's coordinates, as well as other information such as the polarity and the timestamp (for instance, x-coordinate, y-coordinate, timestamp, $+/-$), are sent as a code through the AER bus. This mechanism makes the neuromorphic chip act similarly to the human brain in a long-range communication point of view [35].

The Java Address-Event Representation (jAER) [29] and C Address-Event Representation (cAER) [8] libraries are two libraries available for accessing the DVS output in AER format. Although the Java-based jAER is a commonly used library, in this thesis we used the cAER library because it is designed to give direct low-level access to the DVS. Therefore, manipulating the DVS is easier and faster using cAER.

2.4 Optical Flow

In our daily lives analyzing, predicting and recognizing motion play a significant role. Detecting the direction of motion can help us in different situations. For instance, detecting the direction of the movement of cars can help us cross the street safely. Also, detecting and analyzing moving and stationary objects can help us build intelligent navigation systems with the ability to prevent collisions with surrounding objects. Motion detection requires finding a change in the position of an object relative to its surrounding environ-

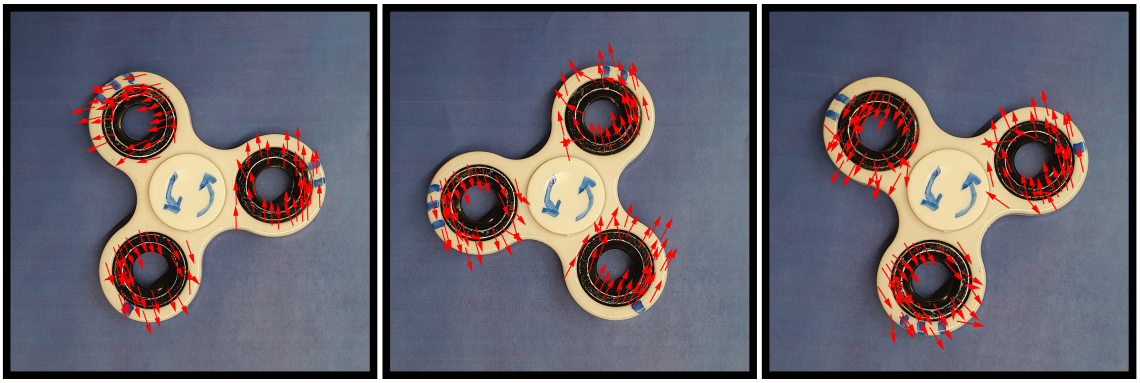


Figure 2.7: Optical flow patterns emerges by moving the object. In this figure (left to right) a spinner is spinning counterclockwise and the arrows indicate the direction of movement for different parts.

ment. Movement between a perceiver and objects in a scene generates patterns of visible motions of objects, surfaces and edges. These patterns are called optical flow. This term has various definitions in different areas of science. In computer vision, optical flow is the pattern of apparent motion indicated by a vector for each pixel in a scene. When objects in the scene move, a pattern of motion emerges (Figure 2.7). Navigation control, motion detection and classification are some of the applications of optical flow in computer vision and robotics [1, 57]. Optical flow is usually used to detect different types of movement and there are algorithms capable of classifying the movement of objects from the optical flow [28]. Many optical flow algorithms have been developed for conventional frame-based cameras [5, 14, 16, 28, 56].

A neurobiological algorithm is presented by Dramas et al. [14] which is able to calculate optical flow in real-time. In this algorithm, a direction of motion can be provided at each pixel in 8 or 16 directions. Also, it provides the distance of movement in pixels between two sequential frames. This algorithm uses thresholds to control minimum amount of edge energy, minimum luminance of two sequential frames and noise tolerance. As mentioned by the authors, the real-time calculation and the speed of this algorithm are its most important advantage. However, it is based on frame-based architecture and cannot be used with event-based sensors.

2.4.1 Event-Based Optical Flow Algorithms

Ridwan and Cheng [50] presented an approach that computes the optical flow from events reported by the DVS. In this algorithm, correlations among polarity events are identified to detect movements.

When objects move in a scene, the log-luminance changes occur mostly in boundary areas. By comparing the polarities of recent events and previous events in close proximity, the algorithm can detect motions. Having a series of pixels with the same polarity in a direction indicates that the object is moving toward that particular direction. When the object is moving, the pixels of the leading edge will produce the same polarity with the direction of the motion in a period of time. Therefore, finding events of the same polarity in close proximity and within an acceptable period of time might be an indication of the motion. The intensity of the moving object and the background must be different from each other, and the intensity of the background is assumed to be a constant value. Because there might be some noise reported by the DVS, this algorithm checks polarity changes close to recent events. A time threshold is also used to enhance the output of the algorithm in detecting motion and to tolerate noise. The algorithm produces output only when there are two matching events close to each other with the same polarity.

The input of this algorithm is an Address Event Representation (AER) stream and each of the events of this stream contains the timestamp, coordinates and polarity of the pixels. Two thresholds are used to control how recent the events are to be considered a match. The output of this algorithm is a stream containing the time, location and direction in eight compass directions as shown in Figure 2.8.

For each DVS event that arrives, the algorithm searches the eight neighbours of the location for a matching recent event with the same polarity. The match indicates which direction the object is moving. If multiple matches occur, the algorithm reports their average as the direction of movement. More details of the implementation can be found in [50].

In terms of computational complexity, the amount of processing per event is a small

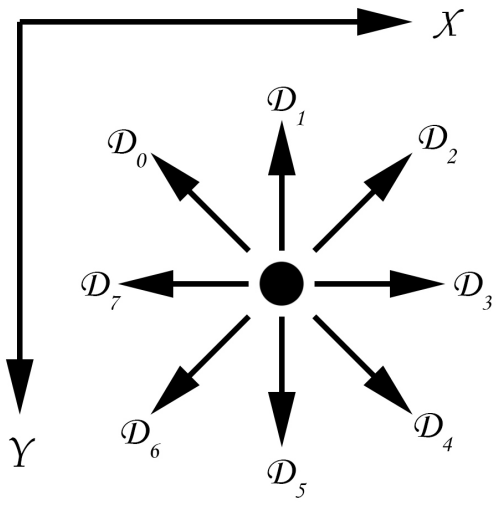


Figure 2.8: Eight compass directions.

constant. Experimental results show that using this algorithm, each event can be processed in around two microseconds.

The events generated from the DVS can be considered as derivatives, or changes, in log-luminance. It was shown by Ridwan [49] that the algorithm can be considered as an event-based version of the Reichardt Detector [48] which is a correlation-based motion detection method inspired by the flies' visual system.

2.5 Looming Object Detection

Looming is defined as a fast increase in the dimensions of an image of any given object [51]. This dimension expansion indicates an object approaching toward the perceiver or vice versa. Figure 2.9 shows an example of a looming object. When the size of the object image is increasing on the perceiver's retina, a physiological response leads to the perception of an approaching surface or object [21]. Since looming objects can be dangerous, this ability has been developed during many years of evolution in natural organisms [42, 43, 55]. Ragen and Beverley [47] proposed that looming objects are detected through data-processing channels which have been provided in the human visual pathway.

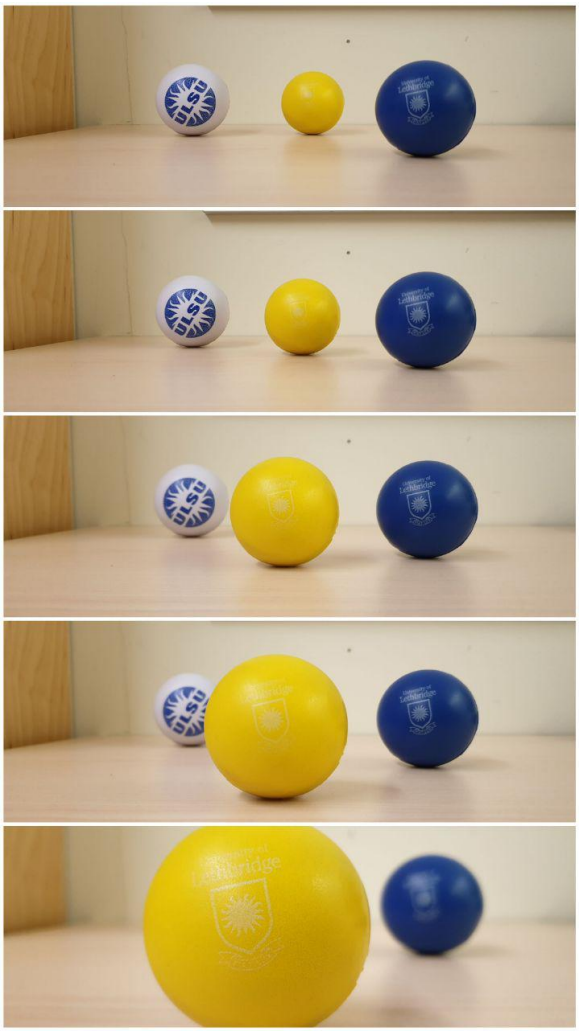


Figure 2.9: An example of a looming object—a ball is approaching toward the camera.

There are many looming object detection algorithms developed for conventional frame-based cameras (for example, [18, 19, 44]). We list a few of them as follows.

Subbaro [53] developed an algorithm that calculates the time-to-collision between a perceiver and an object by considering the first-order spatial derivatives of image motion. Park and Sowmyat [45] developed a real-time active visual motion understanding system capable of detecting obstacles around a moving robot. Their algorithm does not need any prior understanding of the environment. To detect the behaviour of looming objects, they first designed a real-time optical flow method. To segment pixels in 2-D space, they used a clustering method followed by calculating the looming features of the object. Then, they used a self-organizing feature map to classify the reported optical flow vectors. Finally, they computed the focus of expansion for the surfaces or objects to analyze the looming features. These looming features allow the robot to identify all surrounding obstacles and avoid them by finding a safer path.

Pedro et al. proposed a method [22] capable of computing the centre of expansion in a video sequence. In this method, their main goal is to compute the vanishing point of multi-frame interest point trajectories in continuous frames. Their method includes two main steps—simultaneously extracting the trajectories of points and projecting them towards their vanishing point [27]. After calculating the objects' motion flow and the intersection points, they use optical flow to link the continuous segments. The point trajectories are used to determine the centre of expansion.

2.5.1 Bio-inspired Looming Object Detection Algorithm

Fülöp and Zarándy [18, 19] developed looming object detection algorithms based on a neural circuit model of an object detection structure in the mouse retina. Their first algorithm is limited to detecting dark looming objects against a bright background [18], while the second algorithm removes this limitation [19].

In the second algorithm, a scene is divided into several overlapping areas called recep-

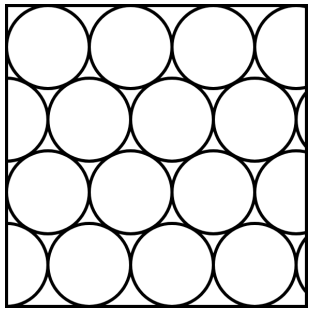


Figure 2.10: The scene is divided into receptive fields.

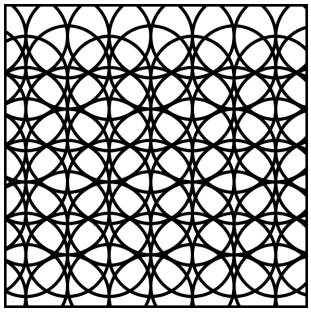


Figure 2.11: Overlapping receptive fields in the whole scene

tive fields (Figures 2.10 and 2.11). The algorithm attempts to determine the type of motion in each of these receptive fields. When the intensity of a pixel changes, we say that the pixel is active at that time. There are two channels in each pixel defined as a function in time—excitatory and inhibitory. The excitatory channel is triggered at a pixel if that pixel becomes active when it was inactive before. Similarly, the inhibitory channel is triggered if that pixel becomes inactive after having been active recently. A pixel that keeps changing will remain active and does not contribute to either channel.

For each frame, each receptive field computes the difference in the number of excitatory and inhibitory signals in that field. If that difference is larger than that in the previous frame, the algorithm concludes that there is a looming object in that receptive field as an expanding object leads to more excitatory pixels. If the difference is smaller, the algorithm concludes that there is a recessing object in that receptive field—that is, the object is moving away from the perceiver.

Although this method is capable of detecting looming or recessing motion in each particular receptive field, it has its limitations. When objects move into a receptive field, the algorithm reports looming motion instead of lateral motion. Similarly when they move out of a receptive field, the algorithm reports recessing motion. The objects that can be detected needs to be completely within a receptive field, and each receptive field is assumed to contain at most one object. Larger objects may trigger incorrect responses in multiple receptive fields.

2.5.2 Event-based looming object detection

Ridwan [49] presented an algorithm capable of detecting looming objects using DVS outputs. In this algorithm, she used the event-based optical flow algorithm [50] (Section 2.4.1) and the arithmetic mean of the locations of recent optical events at all boundary pixels to obtain an interior point. Then by calculating the dot product of the optical flow vector and the vector from the interior point to the event, the algorithm can detect if the boundary is moving away from or towards the interior point. A positive dot product indicates the event is moving away from the interior while a negative dot product indicates the event is moving towards the interior. If there are significantly more events moving away compared to events moving towards the interior, then a looming decision is made. Ridwan's experiments on her looming object detection algorithm included simple objects and round looming objects.

Although this algorithm accurately detected looming objects when there is a single object in the scene and can be used in real-time applications, it has a number of drawbacks. First, this algorithm does not work well for multiple moving objects in a scene. The reason is in the way the algorithm calculates the interior points of the objects. When multiple objects are in the same scene, the algorithm uses all of their boundary events to calculate the interior point. The calculated point is likely not correct and the algorithm fails to detect whether the objects are looming or not. Another issue with her algorithm is that the accuracy of the results is dependent on the shape of the objects. Given a concave object, the calculated interior point may lie out of the object. Therefore the algorithm's result will not be reliable. Also, using this algorithm to detect looming objects with an interior pattern is not possible. This is because when an object with interior patterns is in motion, its interior patterns will also be detected as boundary events. Thus, the algorithm detects them as separate objects and the problem with multiple objects in the same scene can occur here again. Figure 2.12 shows examples of these situation.

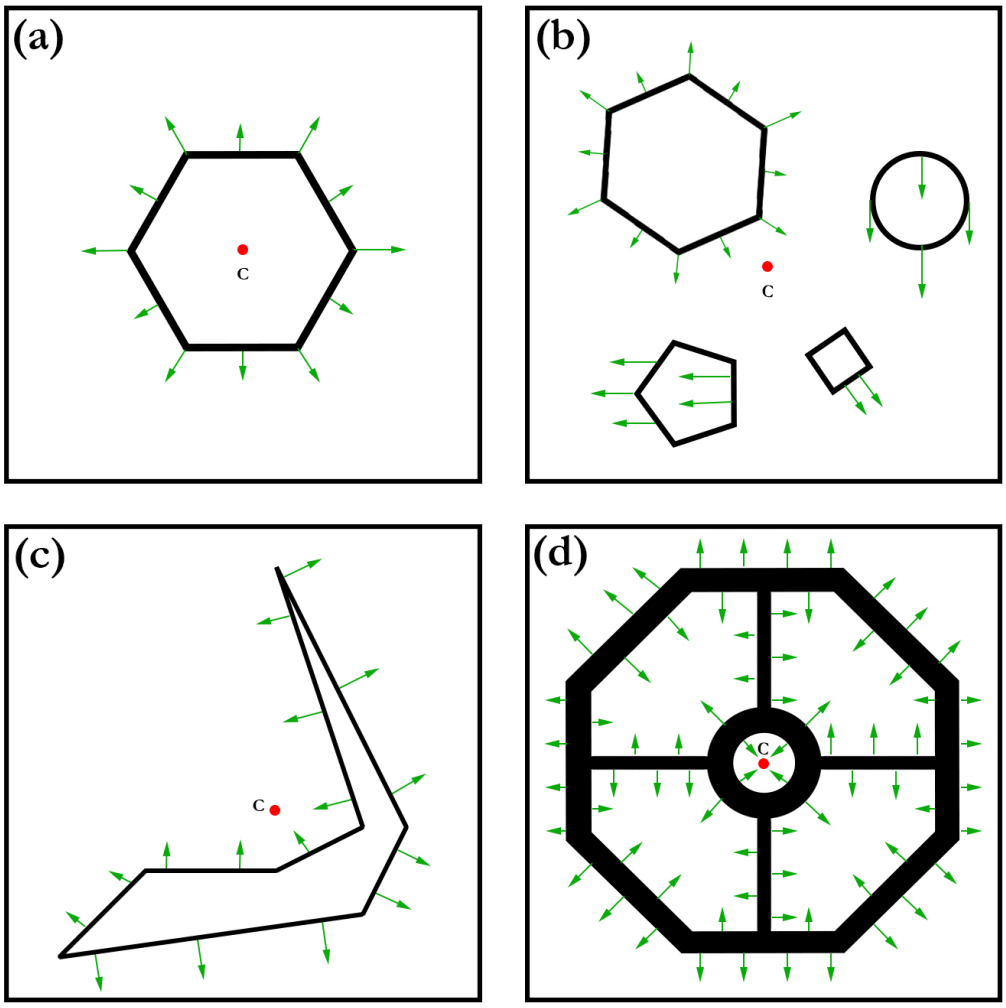


Figure 2.12: The limitation of Ridwan's algorithm. (a) single convex looming object that has its computed interior point inside it. (b) Multiple looming objects and their computed interior point which is in the middle of those objects. (c) A concave looming object that its computed interior point is outside it. (d) A looming object with internal patterns. Some parts moving towards the interior and some parts moving away from it.

2.6 Clustering

Given a set of points, clustering is the process of grouping these points into a small number of groups, so that the points within each group are similar to each other and points from different groups are not similar. Clustering has many applications in computer vision, including segmentation, object detection and recognition, and object tracking. For example, the points can encode various features (e.g. shape, size) of objects in a high-dimensional space, so that different objects of the same type are similar by some measure. Once clustering is done, a new object can be recognized as the same type as that represented by the cluster nearest to its encoded point. Clustering algorithms are unsupervised, as there are no known labels in the training set. The correctness of the labelling is often not easy to define, and may depend on how these labels are subsequently used.

Formally, let the set of training data points be $x_1, \ldots, x_n \in \mathbb{R}^m$ for some positive integer m. A clustering algorithm determines the appropriate number of clusters (K) in the data set, and assign each point a label $l(x) \in \{1, \ldots K\}$ such that $l(x) = l(y)$ if and only if the two points x and y are similar. The measure of similarity is typically a distance measure such as the Euclidean distance.

There are many different algorithms to perform clustering of data, with different requirements. We focus on two algorithms in this thesis. The K-means algorithm [26] is a well-known algorithm that requires the number of clusters to be known a priori. If this is not known there are methods to estimate the correct number of clusters [33]. On the other hand, the mean shift clustering algorithm does not require the number of clusters to be known. These two algorithms will be discussed in this section.

2.6.1 K-means Clustering

The K-means algorithm [26] repeatedly computes the cluster centroids and updates the clusters according to the updated centroids. The parameter K represents the number of clusters and must be specified as an input parameter. In this algorithm, the goal is to group

a set of n elements $(x_1, \ldots, x_n \in \mathbb{R}^m)$ into a set of $K(< n)$ clusters $S = \{s_1, \ldots, s_K\}$ based on the Euclidean distance between each point and the nearest cluster's centroid. To complete the clustering process there are three main steps:

1. initialize the clusters' centroids;
2. assign each element to a cluster;
3. update the clusters' centroids.

In the first step, initially K centroids $c_1, \ldots, c_K$ are chosen in some way, often randomly. In the next step, the element x_i is assigned to cluster s_j, according to the Euclidean distance. Given centroids $c_1, \ldots, c_K$, the cluster s_i is defined as:

$$s_i = \{x_\ell : 1 \leq \ell \leq n, and \, \|x_\ell - c_i\| \leq \left\|x_\ell - c_j\right\| \forall j, 1 \leq j \leq K\} \tag{2.1}$$

in which x_i is assigned to only one cluster even in the case that it could be assigned to more than one available clusters based on (2.1).

In the update step, the centroid of the clusters are recomputed according to the mean of the data points assigned to that cluster. The update formula is defined as:

$$c_i = \frac{1}{|s_i|} \sum_{x_j \in s_i} x_j \tag{2.2}$$

The cluster assignment and centroids update steps are repeated until the clusters' centroids do not change significantly.

Although the algorithm is guaranteed to produce a label for each data point, the results may not be the same by running this algorithm again for the same sets of data. The quality of the K-means results is sensitive to the initialization of centroids. Therefore, different initialization can lead to different results, some of which can be far from optimal. There are various methods to initialize or create clusters' centroids, with various advantages and disadvantages over each other [9].

Algorithm 1 K-means clustering

Input: $x_1, \ldots, x_n \in \mathbb{R}^m$, K
Output: Set of K clusters $S_i = s_1, \ldots, s_K$
initialize cluster centroids, $c_1, \ldots, c_k$
repeat
  for all $\ell = 1, \ldots, n$ **do**
    $L = 1$
    for $j = 1, \ldots, K$ **do**
      if $\left\|x_\ell - c_j\right\| < \|x_\ell - c_L\|$ **then**
        $L = j$
    $S_L = S_L \cup \{x_\ell\}$
  for $i = 1, \ldots, K$ **do**
    $c_i = \frac{1}{|s_i|} \sum_{x_j \in s_i} x_j$
until the centroids do not change significantly

In this thesis we have used the K-means algorithm provided by OpenCV's library [7]. This library has implemented the K-means algorithm with different initialization modes which can be chosen as an input parameter. As mentioned previously, one of these methods initializes the centroids randomly.

K-means++ is another initialization algorithm [2] for K-means provided by OpenCV [7]. In this algorithm, the first cluster's centroid is chosen uniformly at random. Then, it compares the distance of the rest of data points to the selected centroid to find the shortest distance. We define $D(x)$ as the shortest distance between each data point (x) and the nearest cluster's centroid computed so far. The following steps define the K-means++ algorithm:

1. Choose an initial centroid c_1 uniformly at random from the data set χ.

2. The data point x_i is selected as the next cluster's centroid with a probability of

$$p_i = \frac{D(x_i)^2}{\sum_{x \in \chi} D(x)^2} \tag{2.3}$$

3. Repeat the second step until a total of K centroids are selected

After computing all K centroids, the processing continues with the standard K-means algo-

rithm.

Although K-means is a commonly used algorithm, it has some limitations. The number of clusters (K) needs to be known a priori. Finding this number is actually one of the challenges to use this algorithm. The results of this algorithm can vary given the same set of data especially when the number of elements in the data set grows. This algorithm is limited to linearly separable clusters.

In contrast to supervised learning algorithms where ground truth data are available for evaluating the generated output, it is difficult to define an evaluation metric for evaluating the generated results of unsupervised algorithms. Also, since K-means needs to know the number of clusters a priori and does not learn it through the data set, we cannot evaluate it based on the number of clusters that we could have in different problems. For evaluating the K-means algorithm we can record the outcomes of the algorithm by running it for a range of K values and compare the outcomes together. There are different methods for determining the number of clusters from these data [33].

The Elbow method has been described by a number of survey articles [25, 31, 54], and is one of the methods for determining the number of clusters. A "compactness" measure is used to compare different outputs produced by clustering algorithms with different parameters. The sum of squared distance between each data point and its cluster's centroid is commonly used for the compactness measure. By increasing the number of clusters (K), the compactness is decreased to the extreme of reaching zero when the number of data points and K are equal. Intuitively, as K increases towards the correct number of clusters, the compactness decreases quickly. However, as K increases past the correct number of clusters, the compactness will continue to decrease but at a slower rate. Therefore by plotting the compactness as a function of K, we can find the elbow point where the rate of reduction changes drastically. Using this point we can determine the correct value for the number of clusters, such that increasing the number of clusters does not reduce the compactness considerably.

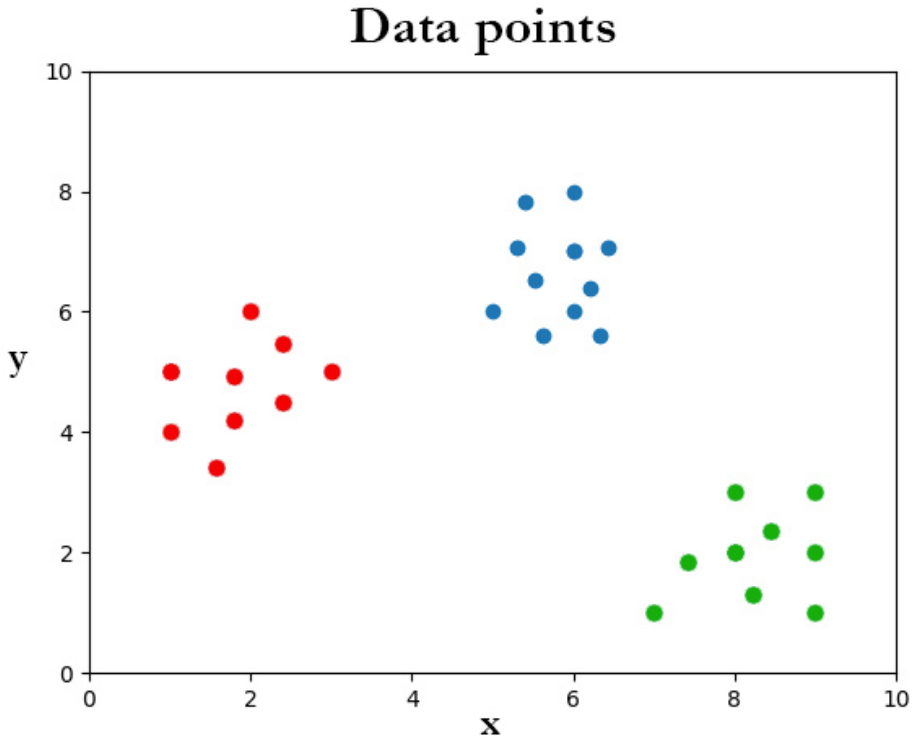


Figure 2.13: An example of three well separated clusters

Mathematically, the compactness is defined as

$$C = \sum_{i=1}^{n} \left\| x_i - c_{l(x_i)} \right\|^2 . \tag{2.4}$$

Figure 2.13 shows an example with three separated clusters. Figure 2.14 shows the elbow point and what happens to the compactness measure (C) as the number of clusters increases.

Although by looking at Figure 2.14 we can determine the K, this method is difficult to implement because there is no mathematically rigorous definition what the elbow point is [31]. In addition, the K-means algorithm needs to be executed multiple times for different values of K in order to use this method, which makes it less practical when time is limited.

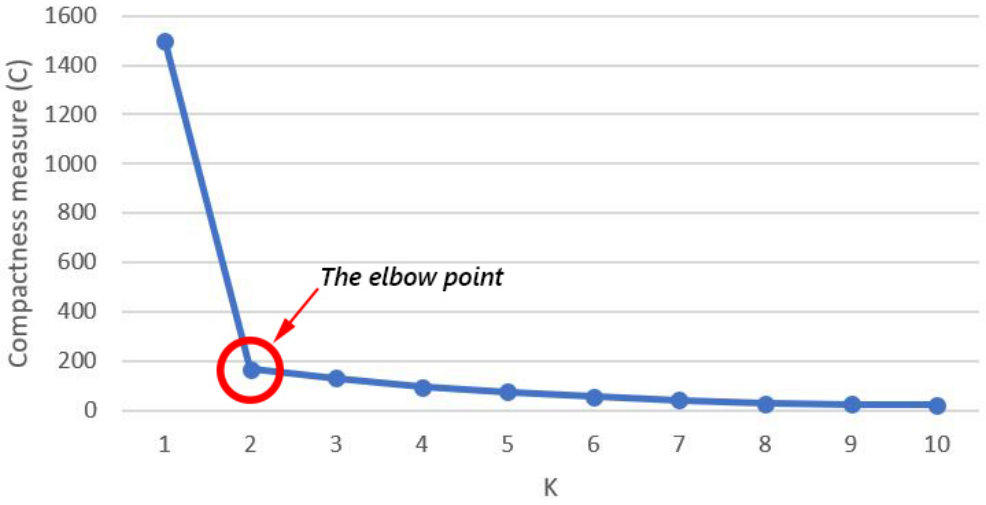


Figure 2.14: The elbow point method.

2.6.2 The Kneedle Algorithm

The Kneedle algorithm designed by Satopaa et al. [52] is a heuristic to detect the elbow point. It is a general algorithm which detects the elbow point for both online and offline applications. The elbow point in a discrete data set is defined according to the mathematical definition of curvature for continuous functions. Based on this definition, elbow points occur when a curve becomes more flat showing a decrease in curvature. This corresponds to a point that is furthest away from the diagonal connecting the start and end of the curve (Figure 2.15). In fact, it is easier to compute the vertical distances from each data point to the diagonal (Figure 2.16), as a point that has the largest vertical distance is also furthest away from the diagonal. In this algorithm, the curve is first normalized so that the diagonal connects $(0,1)$ to $(1,0)$, and a new distance curve is formed by the vertical distances of each point to the diagonal. To detect the elbow point in the normalized curve, the algorithm calculates the local maxima of the distance curve as candidates of the elbow point. For a local maximum to be considered the elbow point, the value of the distance curve must decrease below a certain threshold before the next local maximum. The value of the threshold is determined by a sensitivity parameter S to adjust how aggressive the algorithm detects

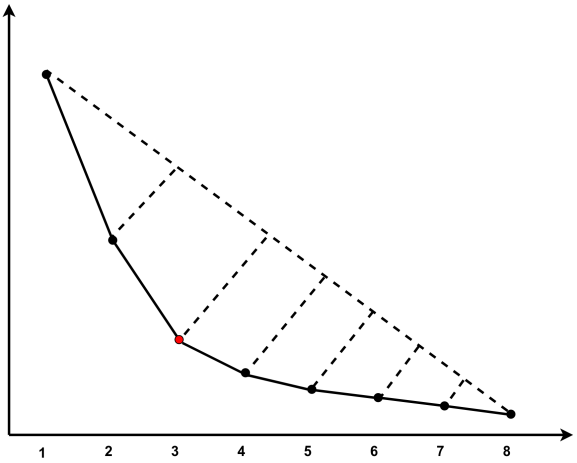


Figure 2.15: The elbow point using the Kneedle algorithm.

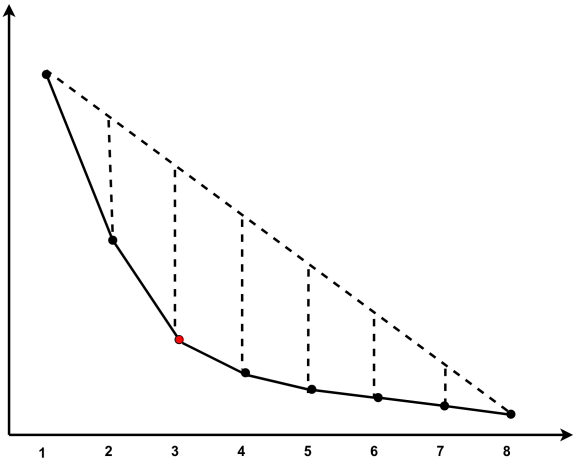


Figure 2.16: The vertical distances between the data points and the diagonal in the Kneedle algorithm.

elbows.

2.6.3 Mean Shift Clustering

Mean shift is a simple clustering algorithm that works well even if the number of clusters is not known a priori [10]. It is also nonparametric, so that there are no assumptions on the distribution of the underlying data.

Given the set of data points $x_1, \ldots, x_n \in \mathbb{R}^m$, we define the kernel density estimation function as

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i) \tag{2.5}$$

for $x \in \mathbb{R}^m$, where $K(x)$ is the kernel. The kernel is a decreasing function in $\|x\|$. For example, the Gaussian function

$$K_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|x\|^2}{2\sigma^2}} \tag{2.6}$$

is a commonly used kernel. Intuitively, if x is not close to any of the data points, $f(x)$ will be small. On the other hand, if x is a good representative for a group of some points, the value of $f(x)$ tends to be larger. The local maxima of the kernel density function are considered as the cluster representative, and the number of local maxima is the number of clusters. We can control the bandwidth of the Gaussian kernel by adjusting σ—a small value of σ means that points have to be very close to be considered to be similar, while a larger value of σ means that more variance is allowed in a cluster.

In order to find the local maxima of $f(x)$ in (2.5), an iterative hill climbing method is used. Starting at a point p, the mean shift is computed by taking the average of the data points each weighted by the kernel on the distance between p and that point:

$$p' = \frac{\sum_{i=1}^{n} K_\sigma(p - x_i) x_i}{\sum_{i=1}^{n} K_\sigma(p - x_i)}. \tag{2.7}$$

Thus, data points closer to p would have a higher influence on the mean shift. In the next

iteration, the mean shift is computed for the new point p', and so on. When the distance of the mean shift p' and p is less than a defined threshold (T), the algorithm stops and the final point is considered the representative of a cluster. This algorithm is described in Algorithm 2.

The process above is described for a single point. The mean shift algorithm performs this process starting at each of data points $x_1, \ldots, x_n$. The unique points from these n local maxima are the representative of the clusters, and the number of unique points is the number of clusters. The label is defined to be $l(x_i) = L$, where L is the cluster containing the local maxima obtained from starting the hill climbing method at point x_i.

As the hill climbing process starting at each point can be done independently from each other, it is possible to reduce the run time by performing them in parallel. This would, however, require hardware that supports parallel processing.

Algorithm 2 Single point mean shift for point x_i

Input: $x_1, \ldots, x_n \in \mathbb{R}^m,\ 1 \leq i \leq n,\ T, \sigma$
Output: p'
$p' = x_i$
repeat
  $p = p'$
  $shift = \vec{0}$
  $scale\_factor = 0$
  for all x_i **do**
    $weight = K_\sigma(p - x_i)$
    $shift = shift + weight \cdot x_i$
    $scale\_factor = scale\_factor + weight$
  $p' = \frac{shift}{scale\_factor}$
until $\|p - p'\| < T$
return p'

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

Barranco et al. presented a method [4] which is based on mean shift clustering and adapted this algorithm to process asynchronous events rather than conventional frames. This method does not require pixel intensities to be computed. Also, it can detect or track

Algorithm 3 Mean shift clustering

Input: $x_1, \ldots, x_n \in \mathbb{R}^m, T', T, \sigma$
Output: $l(x_i)$ for $i = 1, \ldots, n$
for all x_i **do**
  p_i = results from Algorithm 2 with input $x_1, \ldots, x_n$ and T
  while there are i and $j (i \neq j$ s.t. $p_i \neq p_j$ and $K_\sigma(p_i - p_j) \geq$ a defined threshold T' **do**
    $p_i = p_j$
  assign from labels $1, \ldots, n$ to unique values in $p_1, \ldots, p_n$
  $l(x_i) =$ label of p_i
return $l(x_i)$

clusters without knowing any a priori information about them such as the number or shapes of the clusters.

To reduce the required computational resources, this method processes events in parallel and in small packets of a few hundred events at a time. Also, using Kalman filters, this method is capable for visual tracking of multiple targets [34]. High temporal resolution makes the timing for measuring the velocity very accurate which is one of the advantages of this method. To evaluate this method, the authors used an existing dataset with different shapes, patterns, and speeds, as well as their own collected dataset. Also, they used the event-based sensor on a Baxter robot [12] to monitor real-world objects in manipulation tasks. This algorithm achieved a clustering accuracy with a reported F-measure of 95 percent and reduced the computational cost by 88 percent in comparison with frame-based methods. In addition, they reported the average error for tracking was 2.5 pixels and the clustering maintained a constant number of clusters during the time. Despite the advantages of using this method, it is not applicable in all situations. This is due to its computational requirements such as its dependency on parallel processing.

Chapter 3

Event-Based Clustering

When there are multiple objects in the same scene, Ridwan's approach [49] fails to detect the looming objects. The reason is that the algorithm calculates a single interior point of objects and compares it with the computed directions of the boundary events of those objects. The interior point in this situation might lie outside of each looming object, therefore the algorithm fails to detect whether the motion is looming or not. Since Ridwan's approach works only for detecting a single looming object, our goal is to separate the events in the scene into multiple objects and then apply the previous approach [49] to each segmented object. The procedure of our proposed approach is shown in Figure 3.1. In this chapter, the application of clustering algorithms and its adaptation to event-based cameras is described.

3.1 Event-Based Clustering

Clustering algorithms generally require a set of data points to group them into different clusters. However, event-based cameras only transmits information when it detects a significant log-luminance change at a specific pixel location. Since the output of event-based cameras is asynchronous, new events can arrive at any time and old events also need to be removed. Therefore since there is no collection of data points provided, it is difficult to use available clustering algorithms for segmenting asynchronous event-based data points. Some algorithms solve this problem by using pseudo-frames, which is the collection of all events received during some fixed interval of time. Collecting the events to make pseudo-frames and processing the events in a pseudo-frame can be slow depending on the length of

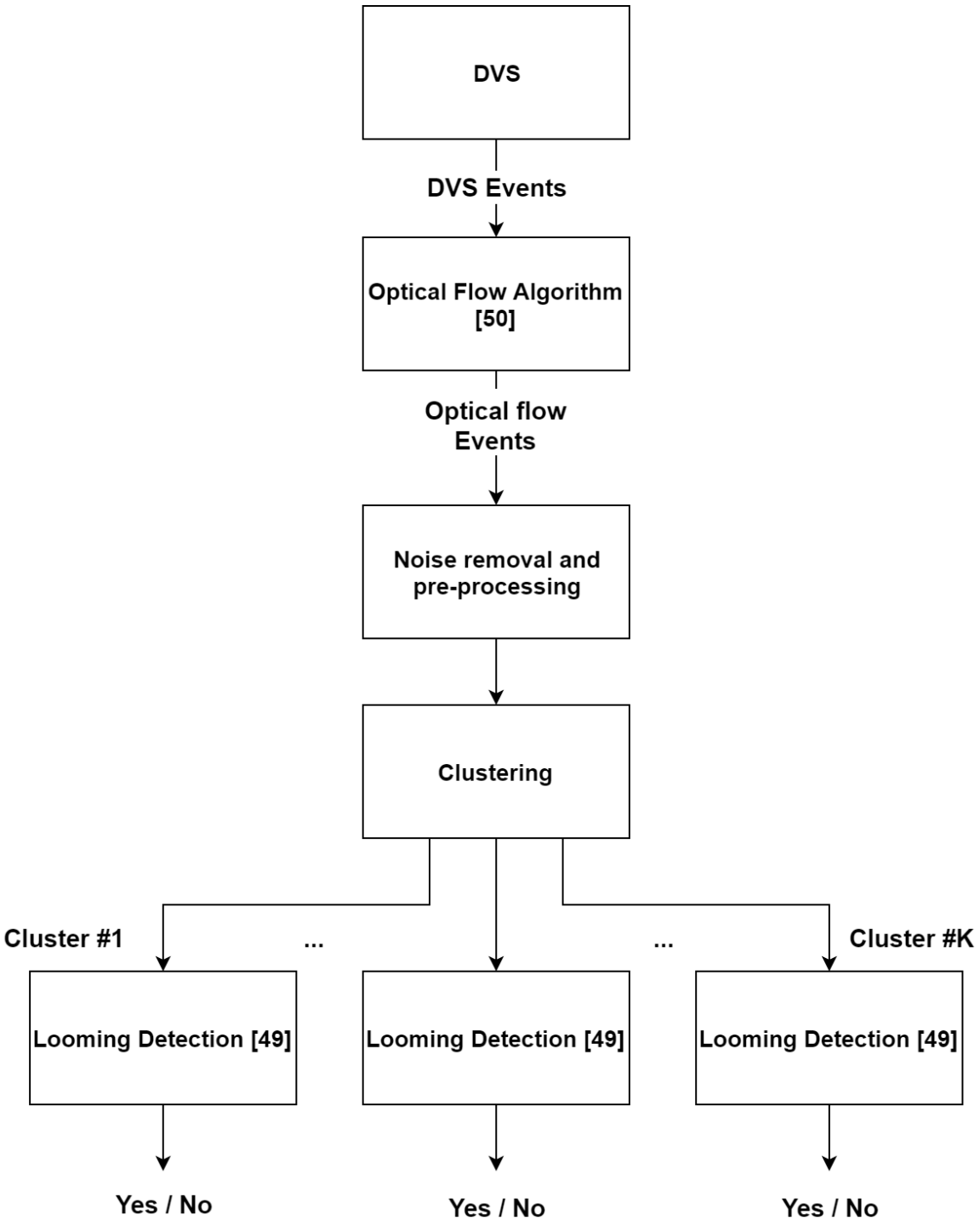


Figure 3.1: The procedure of detecting looming objects by clustering the events into objects using different algorithms.

the pseudo-frame. By increasing the length, the algorithm performs clustering less often but it reacts slower to changes in the scene. On the other hand, if we reduce the length, the program reacts faster but it needs to do clustering more often even when there are not enough data available for clustering. Therefore it is a trade-off between having a faster reaction or fewer calls to the clustering algorithm. One of the goals of this thesis is to determine what is acceptable in this regard.

This approach of course, can reduce one of the advantages of using event-based cameras which is its low latency. In addition, since making the pseudo-frames requires the algorithm to record events, process and report them, it cannot be considered as a real-time method. Although there are some real-time clustering algorithm adopted for event-based data points [4, 37], they required parallel processors or special FPGA hardware to process their data sets. Also, using mean shift algorithm requires adjusting two thresholds depending on the size of the objects in the scene and how long the algorithm takes to converge (T and σ in Algorithms 2 and 3) for each experiment case and real world situations. Another problem with the mean shift algorithm is that looming objects expand in size. Even if we determine the correct bandwidth for one pseudo-frame, the objects may expand beyond the set bandwidth. In this situation, the algorithm detects them as multiple clusters and reports inaccurate results. In this thesis our goal is to achieve a fast, sequential, simple and automatic software solution capable of executing in accessible hardware systems.

3.2 Noise removal pre-processing

Similar to Ridwan thesis [49], we first process the data to reduce possible noise. For each event generated by the optical flow algorithm, a search is conducted in its surrounding pixels (8-neighbourhood [24]) to find other events with similar directions. If a match is found, the event is collected to be processed by clustering algorithms. Otherwise, we consider that event as noise and ignore it. Isolated events are likely caused by noise, since edges in objects tend to generate similar events in close proximity.

The complexity of clustering algorithms is directly related to the number of data points they process. In this thesis, the directions of optical flow events are ignored, and we use their coordinates as the input to the clustering algorithms. 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.

3.3 K-Means Event Clustering

This section describes how the K-means algorithm is used to cluster optical flow events generated by Ridwan and Cheng's algorithm [50]. The K-means algorithm requires the number of clusters as an input, and we will also describe how to determine the number of clusters in our approach.

The output of Ridwan and Cheng's algorithm is an event stream containing the coordinates, timestamp and the direction of the optical flow events. The number of optical flow events is fewer than the number of polarity events from the camera. Since we are interested in the coordinates of the events, we collect the coordinates of each optical flow event and cluster them according to their locations. After collecting the coordinates every L microseconds, the K-means algorithm is used to cluster them. 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. The compactness measure (2.4) for each value of K computed by the K-means algorithm forms a curve. As K increases, the compactness measure decreases and forms a decreasing curve. To determine the number of clusters we designed a heuristic algorithm for detecting the elbow point of the generated curve.

3.3.1 The elbow method

Heuristically, the elbow point is the data point at which the angle of the curve at that point is the greatest (Figure 2.14). However, this is not sufficient as can be shown in Figures

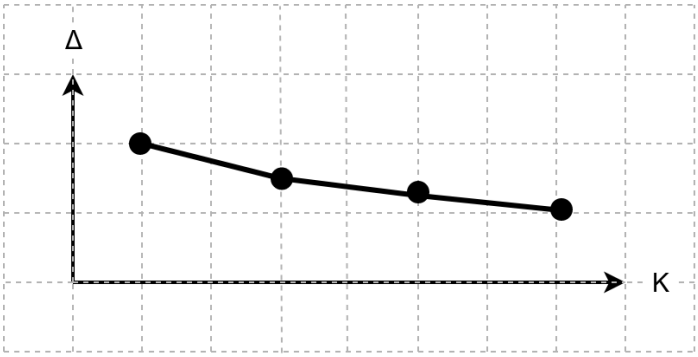


Figure 3.2: Three consecutive small decreases in compactness measure.

3.2 and 3.3. In Figure 3.2, the overall decrease in compactness measure is small, and there is no drastic decrease at any particular point to indicate that there is more than one cluster. In Figure 3.3, there is a large decrease followed by smaller decreases in compactness measure. However, the point with smaller decrease has larger angle compared to the point with larger decrease.

In order to avoid selecting elbow points which have a large angle but a small decrease in compactness measure, only those points in which the decrease in compactness is greater than the average decrease over all K are considered. In addition, a threshold ET is used to ensure that the largest angle is sufficiently large to indicate the presence of an elbow point. Otherwise, the algorithm will conclude that there is only one cluster.

The angle at a point on the curve can be computed by

$$\beta = \arccos\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|}\right) \tag{3.1}$$

where $\vec{u}$ and $\vec{v}$ are vectors represented by two consecutive segments in the curve (Figure 3.4). By extending one of the vectors and computing the angle between these two vectors according to (3.1), we can find the biggest angle among those segments where decreases are larger than the average and consider it as the elbow point. The algorithm reports this point as the number of clusters for the given data points. This process is described in Algorithm 4.

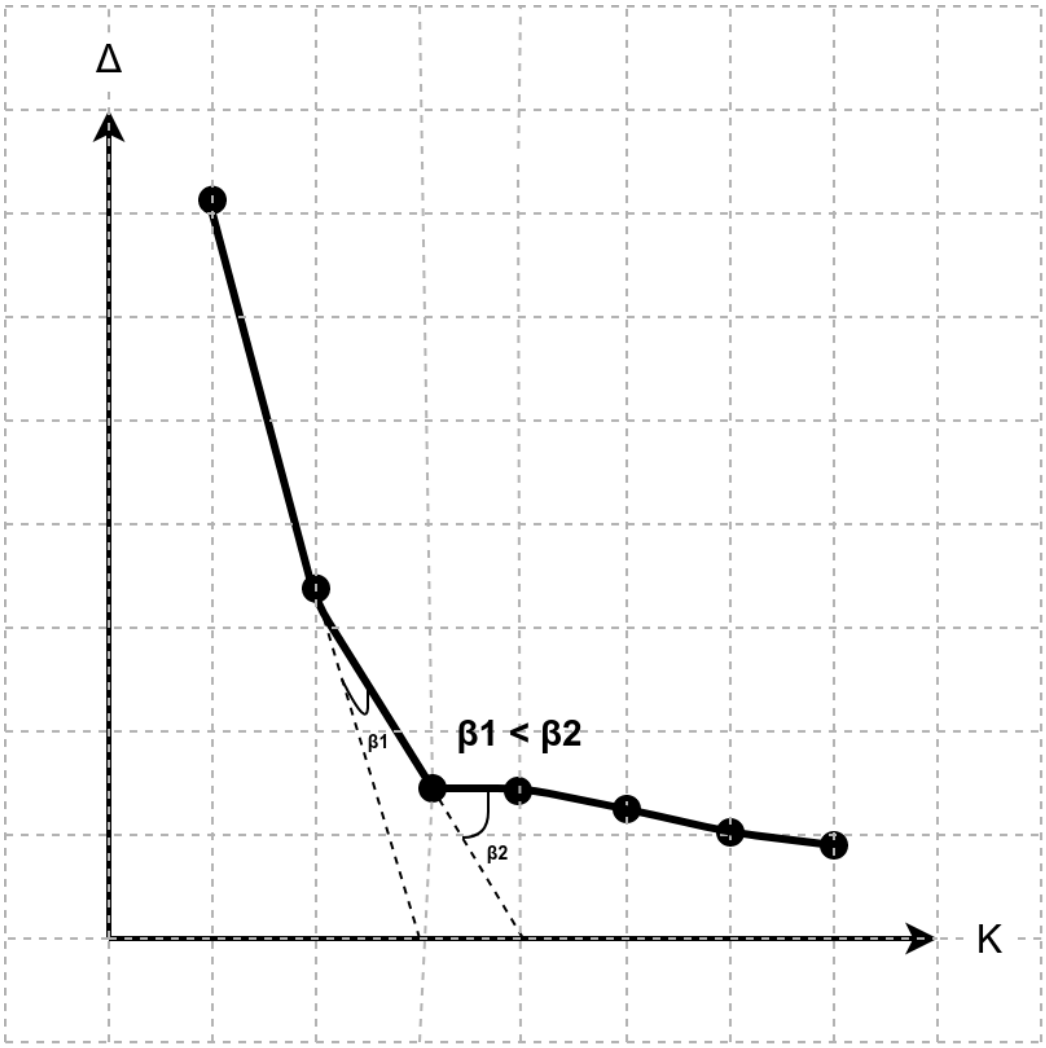


Figure 3.3: A large decrease followed by smaller decreases in compactness measure. However, the angle is larger at the point with a smaller decrease.

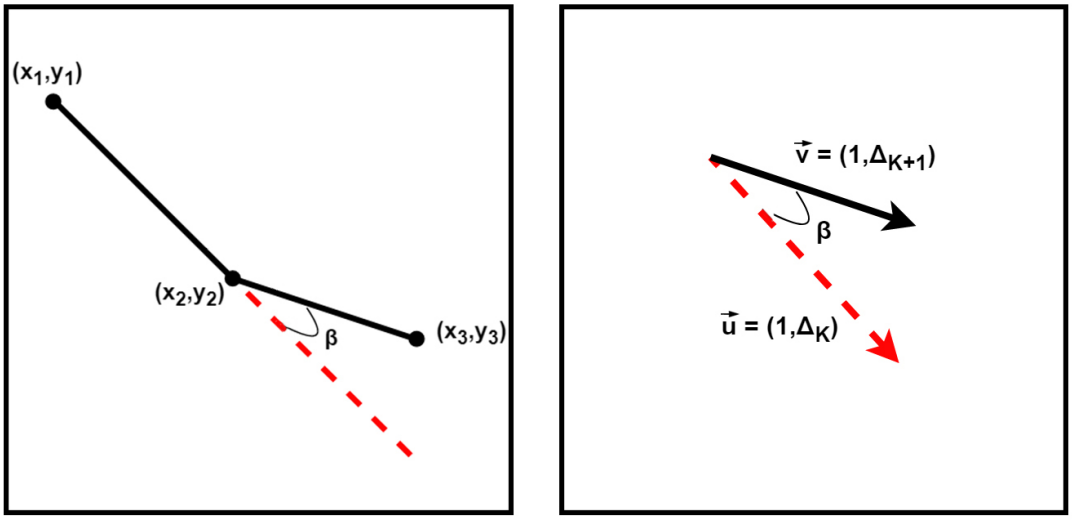


Figure 3.4: Two consecutive segments on the compactness measure curve. Δ_K is the difference between the compactness measure at point K and point $K-1$

Algorithm 4 The elbow method

Input:
M—the maximum number of clusters to consider
C_i $(1 \leq i \leq M)$—the compactness measure returned by K-means clustering on i clusters
ET—threshold to ensure angle is sufficiently large
Output:
K—Number of clusters

$avg = \frac{C_1 - C_M}{M-1}$
$\beta_{max} = 0$
for $i = 1, \ldots, M-1$ **do**
    $\Delta_i = C_{i+1} - C_i$ // $\Delta_i < 0$
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**
            $\beta_{max} = \beta$
            $K = i+1$
if $\beta_{max} < ET$ **then**
    $K = 1$
return K

3.4 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 [15]. The label of each point can be determined either at the time it is processed, or at the end after all data points are seen and the final centroids are computed. To avoid collecting optical flow events into pseudo-frames for processing, sequential K-means clustering can be used to process the events as they arrive. However, using Ridwan's algorithm [49] for the analysis of the movement of objects still requires us to collect the optical flow events for analyzing the directions.

The algorithm starts by assigning the first K data points to their individual clusters. Let $c_1, \ldots, c_K$ be the K cluster centroids at a particular time. When a new data point 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

$$c_{i+1} = c_i + \frac{1}{n}(x - c_i), \tag{3.2}$$

where n is the total number of data points assigned to that cluster, including x. Note that the results are not generally the same as the standard K-means clustering algorithm, since there is no reclassification of data points once the nearest cluster is found. In other words, it is similar to using the standard K-means clustering algorithm with only one iteration [15].

In the application of looming object detection, only the recent optical flow events are processed by Ridwan's algorithm. Therefore, the K-means clustering algorithm also needs to support the removal of events that have expired. While the sequential K-means clustering algorithm removes the need for collecting events into pseudo-frames for clustering, it does not normally supports the removal of data points. A simple modification can be made to handle the removal of data points. If x is a data point to be removed and it was added to

cluster i, then the cluster centroid c_i can be updated by

$$c_{i+1} = c_i - \frac{1}{n-1}(x - c_i), \tag{3.3}$$

where n is the total number of data points assigned to cluster i after x is removed.

While the method described above updates the centroids correctly as events arrive and expires, the looming object detection algorithm must remember the events and their labels in a queue so that the appropriate cluster can be updated as they expire. An alternate, simpler approach, is to modify (3.2) with a decay parameter $\alpha > 1$, so that

$$c_{i+1} = c_i + \frac{\alpha}{n}(x - c_i). \tag{3.4}$$

The centroid is then a weighted sum, so the most recent data point has the highest weight, while older data points are weighted with exponentially decreasing weights. As a result, older data points will be negligible in the calculations of the centroids without having to explicitly remove them. We can also maintain a weighted compactness measure similar to (2.4) to determine the quality of the clustering.

Furthermore, we can also simply restart the process at the beginning of each pseudo-frame. This is the approach we have adopted to form an automatic solution without adjusting the parameters.

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 (Figure 3.5) can be used by the elbow method (Section 3.1) or the Kneedle method (Section 2.6.2) to determine the appropriate number of clusters.

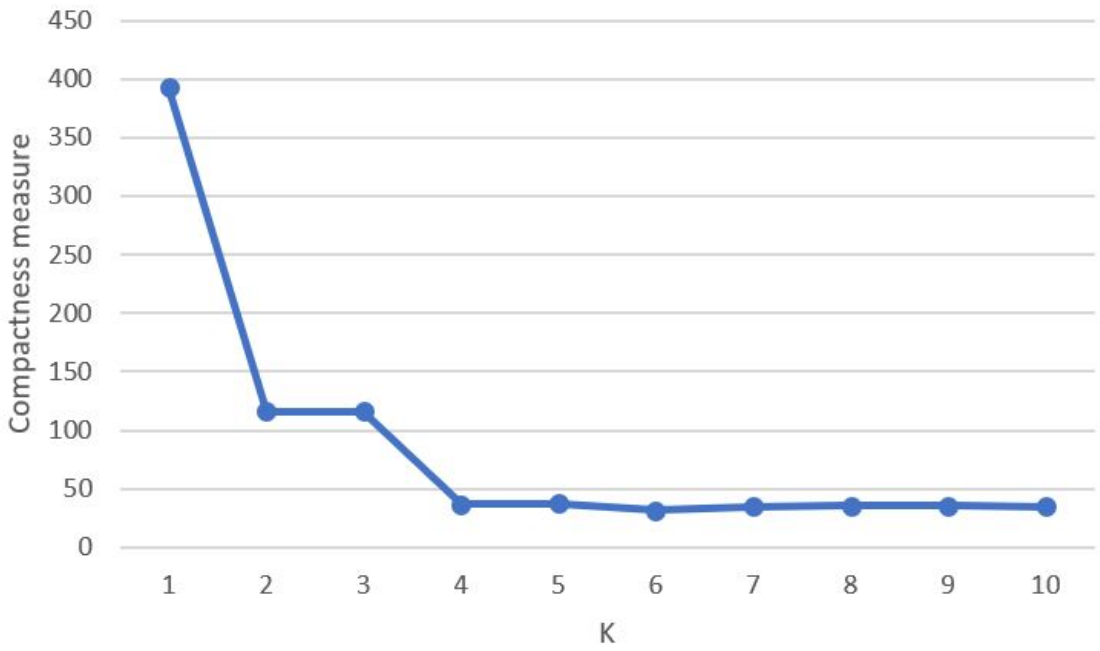


Figure 3.5: An example of compactness curve generated by sequential K-means algorithm.

3.5 Mean Shift Event Clustering

Similar to the previous sections, we use the coordinates of optical flow events as the input of the mean shift algorithm. As mentioned in Section 2.6.3, the mean shift algorithm does not require the number of clusters a priori. The mean shift algorithm reports the label of each point and the centroid of each cluster. Similar to the K-means algorithm, to collect enough data for the clustering process, we use a pseudo-frame with the length of L and collect the coordinates of the optical flow events into this pseudo-frame. Then the mean shift clustering algorithm is executed every L microseconds to cluster the optical flow events according to their coordinates.

To cluster the data points, Algorithm 2 is first executed on each optical flow event to shift them toward the centroid of each cluster. Then Algorithm 3 is applied on each optical flow event to assign them a label according to the cluster they belong to.

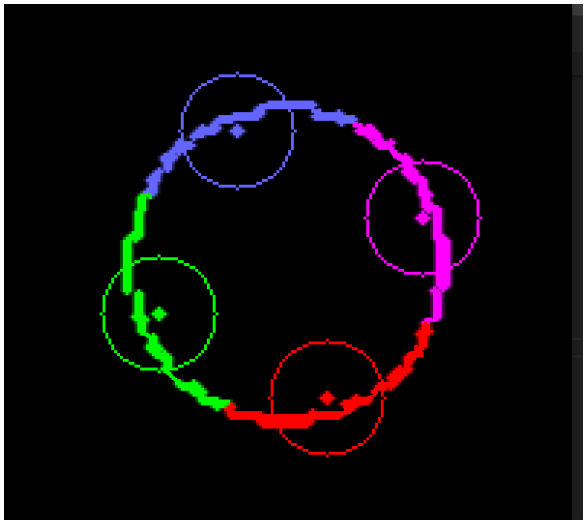


Figure 3.6: The algorithms incorrectly detected multiple clusters while there is only a single object in the scene.

3.6 Cluster Merging

By expansion the dimension of the objects, 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 or representative of the clusters. This can also happen when there is only a single object in the scene. Figure 3.6 shows an example of this situation.

As a solution to this problem, we designed an algorithm which merges these clusters and forms a single cluster. For each event, labels of adjacent events are merged using a union-find data structure. The union-find data structure allows the labels to be merged dynamically and a unique label is returned for all points that have been merged [11]. The process is described in Algorithm 5.

Figure 3.7 shows the output of merge method on the same data set. As can be seen, it successfully merged all clusters to one.

Algorithm 5 The merge algorithm

Input:
optical flow events, $\mathrm{l}(x,y)$
Output:
$\mathrm{l}(x,y)$

for all optical flow events(x,y) **do**
  for all optical flow events (x_2,y_2) in 8-neighbourhood of (x,y) **do**
    merge $\mathrm{l}(x,y)$ and $\mathrm{l}(x_2,y_2)$ // using Union-Find
for all optical flow events(x,y) **do**
  $\mathrm{l}(x,y) \leftarrow$ find$(\mathrm{l}(x,y))$ // using Union-Find

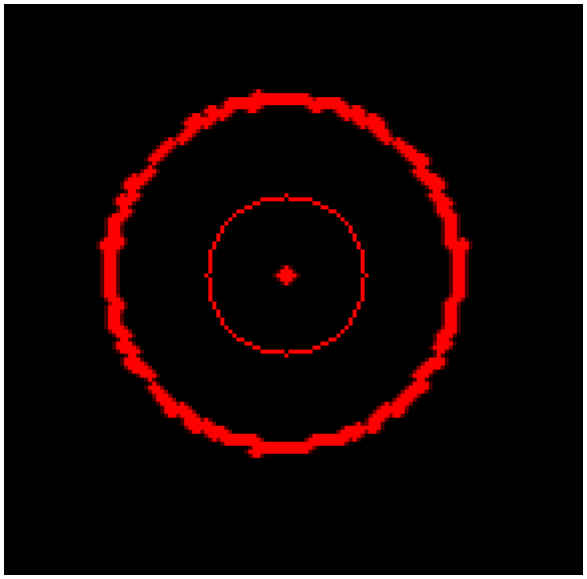


Figure 3.7: The merge algorithm successfully merged all clusters to one.

Chapter 4

Experiments and Results

In this chapter we evaluate the different proposed clustering algorithms described in Chapter 3. The algorithms will be tested with event streams generated from both captured and simulated scenarios. A comparison of the different clustering algorithms will be shown, and an analysis will be given.

4.1 Test Data Sets

To compare the effectiveness of different clustering algorithms we mentioned in Chapter 3, we test them with our data sets shown in Tables 4.1 and 4.2. Simulated data sets are shown in Table 4.1 and captured data sets are shown in Table 4.2. To generate the simulated scenarios, a program is used to generate objects with different shapes such as square, round etc. Also, we can choose the movement of objects to be looming, recessing or sideways as well as the direction of the movement. The output of the program contains the polarity events similar to those generated by the DVS. By using this output as the input of the optical flow algorithm we can obtain the optical flow events as the input of our clustering algorithms. We started with simulated data sets as the ideal test cases to get a baseline of the accuracies of the algorithms. Then, we compared the results with captured scenarios for further investigations. Regarding the captured event streams, we used the DVS with the specifications shown in Table 4.3. We captured different solid coloured objects on a solid coloured background not to reduce noise but because the Ridwan's looming detection algorithm [49] cannot handle internal patterns.

Table 4.1: Simulated data sets.

Data set	Description	Number of polarity events	Number of optical flow events	Number of pseudo-frames
1	A square object is looming while a round object is moving sideways	23744	17911	54
2	A round object and a square object are looming synchronously	18256	14906	37
3	A round object is looming continuously while a square object is looming during part of the event stream	16818	13087	46
4	Two non-covex objects moving sideways	9353	8628	57
5	A round object and a square object are moving toward and passing through each other	9744	9438	79
6	A round object is looming continuously while two square objects are moving sideways in parts of the event stream	15918	13083	56
7	Five square objects are looming synchronously	25710	18715	17
8	A square object is looming continuously	5142	3743	17
9	A square object is moving sideways.	1848	980	17
10	Two squares are passing by each other	21616	19880	178
	Two looming squares are approaching each other from the angle	16246	13346	33
	Two equal squares are approaching each other from the side	8344	6608	60

Table 4.2: 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 toward the camera	34270	19649	17
5	Two balls are rolling sideways	22026	14831	18
6	Four round objects are looming	40900	26302	35
7	Two humans are moving in front of the camera	37250	27073	84

Table 4.3: The DVS specifications used for experiments.

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

4.2 Experimental Setup

We implemented all of the mentioned algorithms in C++. As one of our approaches, we applied the conventional K-means algorithm on every pseudo-frames for different number of clusters. In this approach we use a version of the conventional K-means algorithm implemented in the OpenCV's library to cluster the optical flow events. We call this approach OpenCV's K-means. We also implemented the sequential K-means and mean shift algorithms and we adopted all algorithms for asynchronous optical flow events. 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.

By running the experiments, we manually labelled the results of each pseudo-frame and we compared them with the results generated by the algorithms. We defined the labels as follow:

- For looming movements:
  - **True Positive:** Detects looming while the object is looming
  - **True Negative:** Detects not looming while the object is not looming
  - **False Positive:** Detects looming while the object is not looming
  - **False Negative:** Detects not looming while the object is looming
- For sideways movements:
  - **True Positive:** Detects not looming while the object is moving sideways
  - **True Negative:** Detects looming while the object is not moving sideways
  - **False Positive:** Detects looming while the object is moving sideways
  - **False Negative:** Detects not looming while the object is not moving sideways

Table 4.4: The system specifications used for experiments.

Name	Value
CPU	Intel® Core™ M-5Y10c @ 0.80GHz
Number of cores	Dual-Core 4 MB cash
RAM	8GB 1600 DDR3 SDRAM
Operating System	Ubuntu 18.04.2 LTS 64-bit
Programming Language	C++
Libraries	libopencv 3.2.0, libstdc++, libGL, libcaer

To evaluate the quality of looming detection on each cluster, we compute Recall and Precision [46]. Recall is defined as the percentage of correct results that are correctly labelled by the algorithms. We can compute recall based on Equation (4.1)

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.1}$$

Precision is defined as the percentage of relevant results labelled by the algorithms. Precision is calculated by Equation (4.2)

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{4.2}$$

4.2.1 System Specification

The specification of the event-based camera and the computer system used for all our experiments are summarized in Tables 4.3 and 4.4, respectively.

For our experiments, various parameters are required for the algorithms we used. We present the values of the parameters for all data sets in this section and in Table 4.5. By choosing 1 for the elbow threshold (ET) we basically do not use this threshold for our experiments. For illustrating the direction of the movement, we used blue lines with red heads as shown in Figure 4.1. Our program shows the detected clusters in different colours and depicts the centroid of each cluster by a dot surrounded by a circle with the same colour of its cluster. Also, the program shows the detected looming clusters by yellow circles on

Table 4.5: The parameters value for all data sets.

Algorithm	Threshold	Symbol	Value
Optical Flow [50]	Timestamp (simulated)	(T)	2500 μs
	Low timestamp (simulated)	(T_{low})	100 μs
	Timestamp (captured)	(T)	6000 μs
	Low timestamp (captured)	(T_{low})	20 μs
OpenCV K-means	Number of attempts	-	3
	Type of initialization	-	K-means++
	Type of criteria	-	CV_TERMCRIT_EPS + CV_TERMCRIT_ITER
	Maximum iteration	max_iter	10
	Epsilon	-	1.0
Mean shift	Single mean shift threshold	T	1000
	Mean shift threshold	T'	9×10^{-10}
	bandwidth	σ	10
Elbow	Maximum number of clusters	M	10
	Elbow threshold	ET	1
Kneedle	Sensitivity parameter	S	0
-	Length of pseudo-frame	L	16667 μs

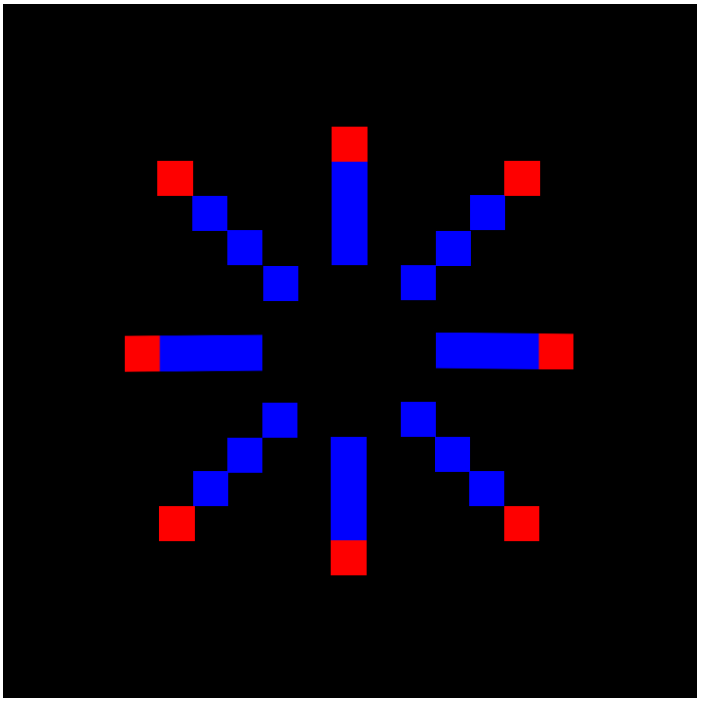


Figure 4.1: Direction of the movement for optical flow events.

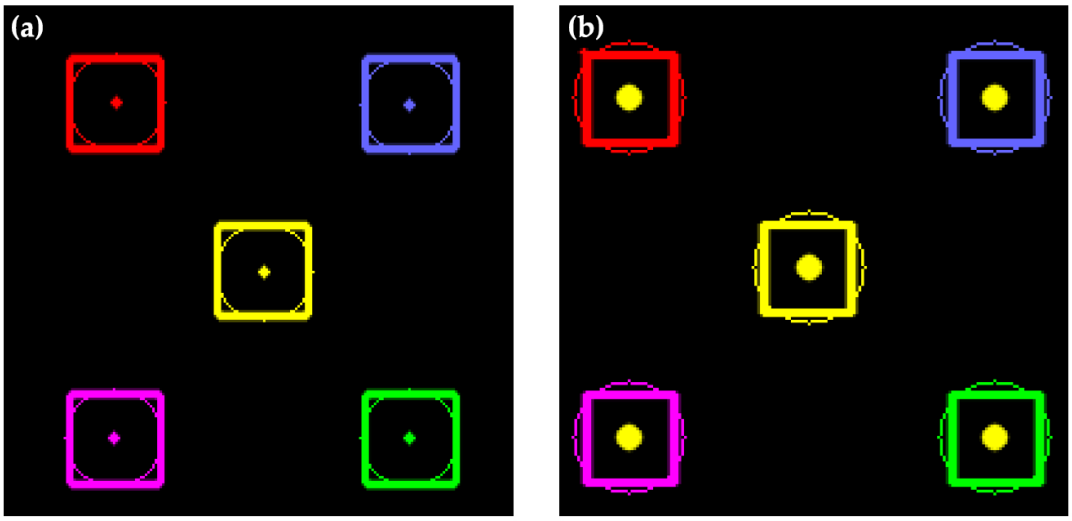


Figure 4.2: (a) five detected clusters. (b) Five detected looming clusters

their centroids (Figure 4.2).

4.3 Experiments on simulated data sets

In this section, we examine the effectiveness of the proposed clustering algorithms in different situations on simulated data sets. We compare these algorithms on the average time they take to process each pseudo-frame and their accuracy to detect the correct number of clusters. Also, for each detected clusters, we present the precision and recall of the looming detection algorithms.

4.3.1 Data Set 1

In this case, a square object is looming while a round object is moving sideways from down to up and vice versa. Figure 4.3 shows a pseudo-frame of the actual video and the output of the optical flow algorithm. Overall, two pseudo-frames did not have enough optical flow events required for our processing (Figure 4.4).

Table 4.6 shows the results of this experiment without using our merge algorithm. As can be seen, the elbow method has detected slightly more pseudo-frames with the correct

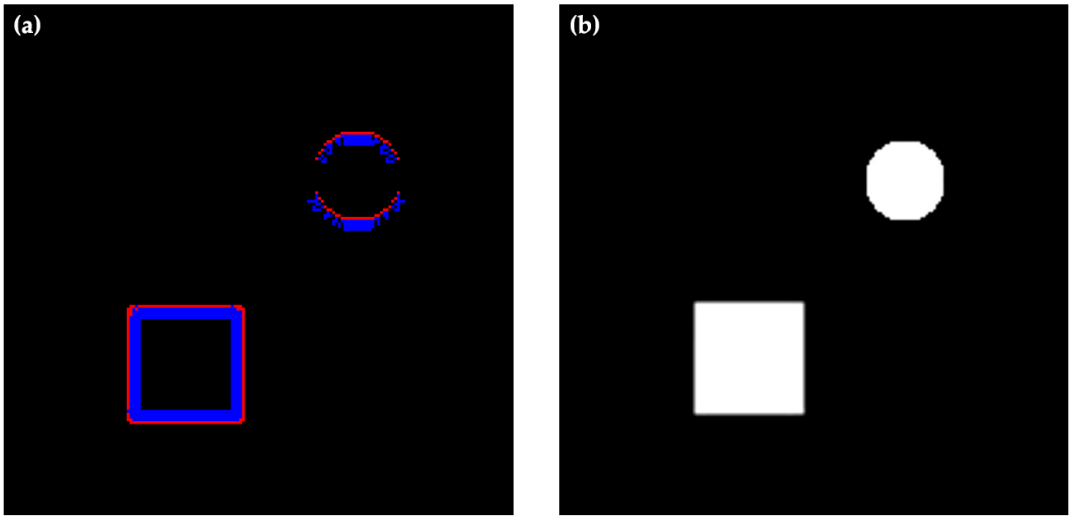


Figure 4.3: (a) The output of optical flow algorithm on data set 1. (b) A frame of actual video.

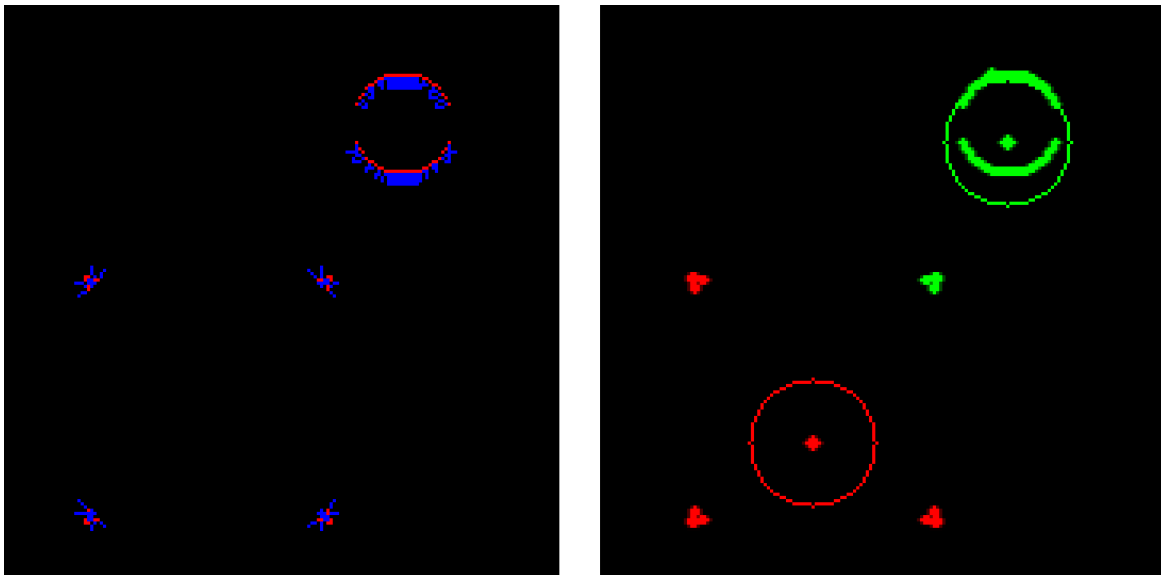


Figure 4.4: Insufficient optical flow events.

Table 4.6: The results of experiments on data set 1 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	31.48	100.00	98.78	0.677
	Kneedle	27.77	100.00	98.78	0.729
	Forced	100.00	93.82	98.70	0.668
OpenCV K-means	Elbow	92.59	100.00	98.78	33.251
	Kneedle	72.22	97.50	98.73	33.756
	Forced	100.00	100.00	98.78	33.185
Mean shift	-	48.14	96.15	96.15	4.781

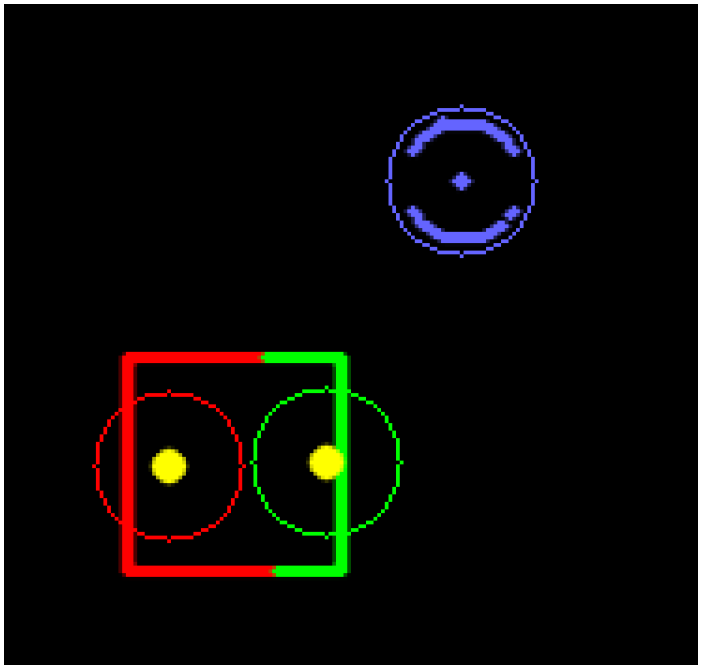


Figure 4.5: Incorrect number of clusters but correct looming detection.

number of clusters compared to the kneedle algorithm. Although the percentage of correct clustering is low, the quality of looming detection is high. This happens because applying the looming detection algorithm on the detected clusters was capable of detecting looming objects. This is highly dependent of the shape of labelled clusters. Figure 4.5 shows such a situation. In this case the square is detected as two clusters and each cluster is detected as looming correctly.

Additionally, to compare the effectiveness between the elbow method and the kneedle method, we also specified the correct number of clusters in the algorithm a priori and re-

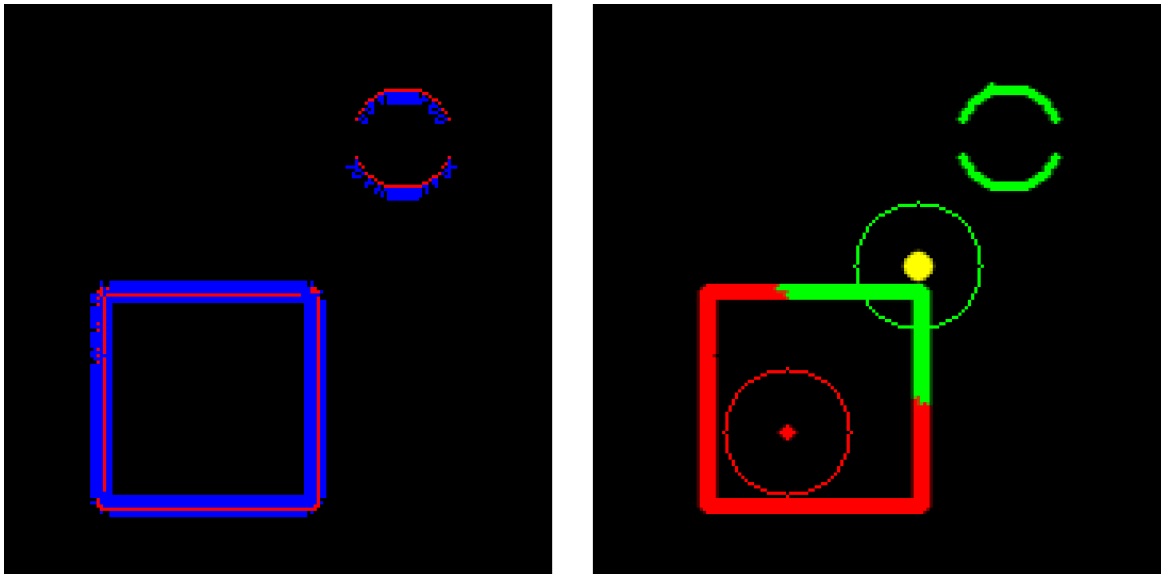


Figure 4.6: Incorrect looming detected due to incorrect detected cluster labelling.

Table 4.7: The results of experiments on data set 1 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (ms)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	98.78	0.710
	Kneedle	100.00	100.00	98.78	0.781
OpenCV K-means	Elbow	77.77	90.00	98.63	33.254
	Kneedle	77.77	90.00	98.63	33.822

ported the results as forced in Table 4.6. By forcing the correct number of clusters, the looming quality decreased. The reason is that in some pseudo-frames the shape of labelled clusters is different from the shape of objects. Figure 4.6 shows such a situation. As mentioned, since the looming detection algorithm is highly dependent of the shape of objects, in this case it failed to detect correctly due to incorrect labelling of the clusters in some pseudo-frames.

Table 4.7 shows the results of applying the merge algorithm on this data set. Using sequential K-means and our elbow method, the correct number of clusters was detected in all pseudo-frames. As can be seen, by merging the touching clusters, the merge algorithm improved the results of cluster detection drastically. Regarding the quality of looming detection, the algorithm worked very well on each cluster. It only failed to detect the loom-

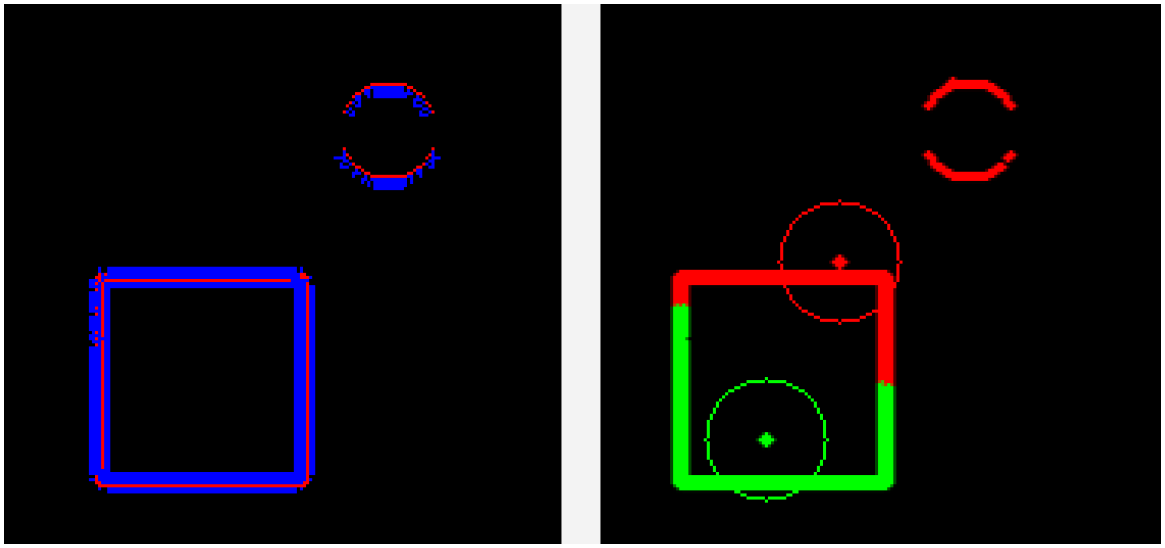


Figure 4.7: incorrect labelling of clusters.

ing object in one pseudo-frame. The results of using the sequential K-means and kneedle method were the same.

Using the K-means algorithm provided in OpenCV's library and elbow method without applying the merge algorithm, the correct number of clusters were reported in 50 out of 54 pseudo-frames. Although the number of pseudo-frames with correctly detected number of clusters is more than that for the sequential K-means algorithm, the quality of cluster detection is lower and in 19 out of 50 pseudo-frames incorrect cluster labelling were detected. Figure 4.7 shows a pseudo-frame in which the clusters are not labelled correctly. In 5 pseudo-frames the incorrect number of clusters were reported, but each detected cluster can be used with looming detection algorithm (Figure 4.8). Regarding the looming detection quality, the results were the same as the sequential K-means algorithm.

By using OpenCV's K-means algorithm and kneedle method, the correct number of clusters was detected in 39 pseudo-frames where among them 9 pseudo-frames detected the clusters with incorrect labels (Figure 4.7). In three pseudo-frames, the algorithm detected an incorrect number of clusters, but each cluster can be used with looming detection algorithm to arrive at the correct looming decision. However, two pseudo-frames had incorrect cluster labelling (Figure 4.6) which caused a lower precision in looming correctness compared to

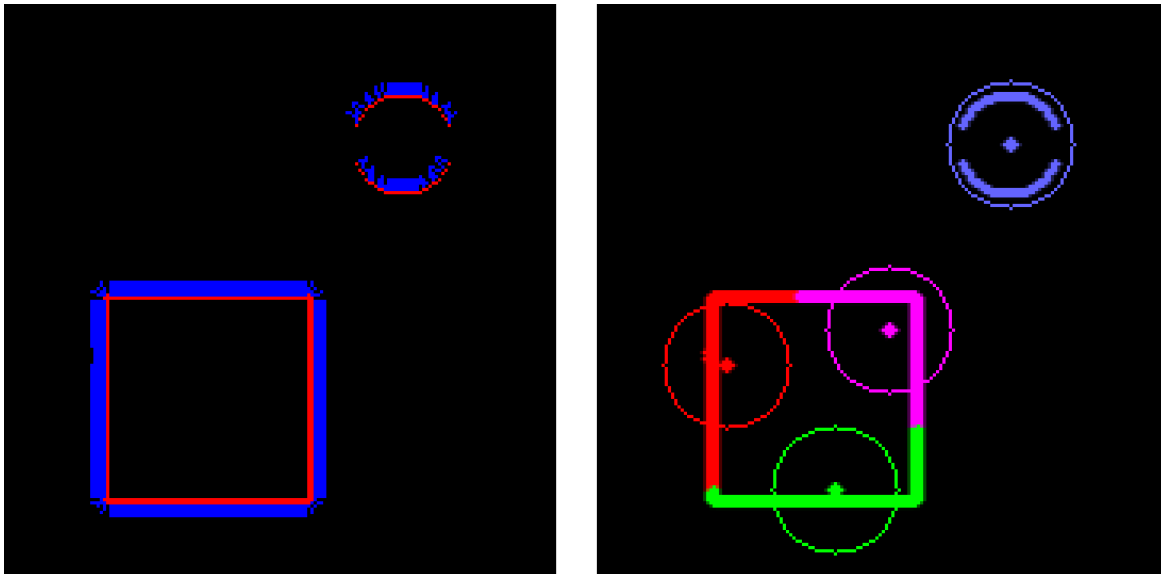


Figure 4.8: incorrect number of clusters, but can be used with looming detection algorithm.

the elbow method.

By forcing the OpenCV's K-mean algorithm to detect only two clusters, there were 15 pseudo-frames with incorrectly detected cluster labels. Among them, the incorrect labelling of clusters in one pseudo-frame caused the looming detection algorithm to fail detecting correctly.

By applying our merge algorithm on OpenCV's K-means algorithm, the elbow method reports 42 out of 54 pseudo-frames with the correct number of clusters. In the remaining 12 pseudo-frames, the merge algorithm merges two clusters to a single cluster. This problem occurs in pseudo-frames in which the clustering algorithm was not able to label clusters correctly (Figure 4.6, 4.7). This also affected on looming detection quality and caused a drop on the precision. The same results are acquired by using the OpenCV's K-means and kneedle algorithms.

Finally, by using the mean shift algorithm, the correct number of clusters was reported in 26 pseudo-frames. For the rest of the pseudo-frames the mean shift algorithm computed the cluster representatives to be as far as possible to each other, and labelled the events according to the computed cluster representatives (Figure 4.9). To solve this problem we

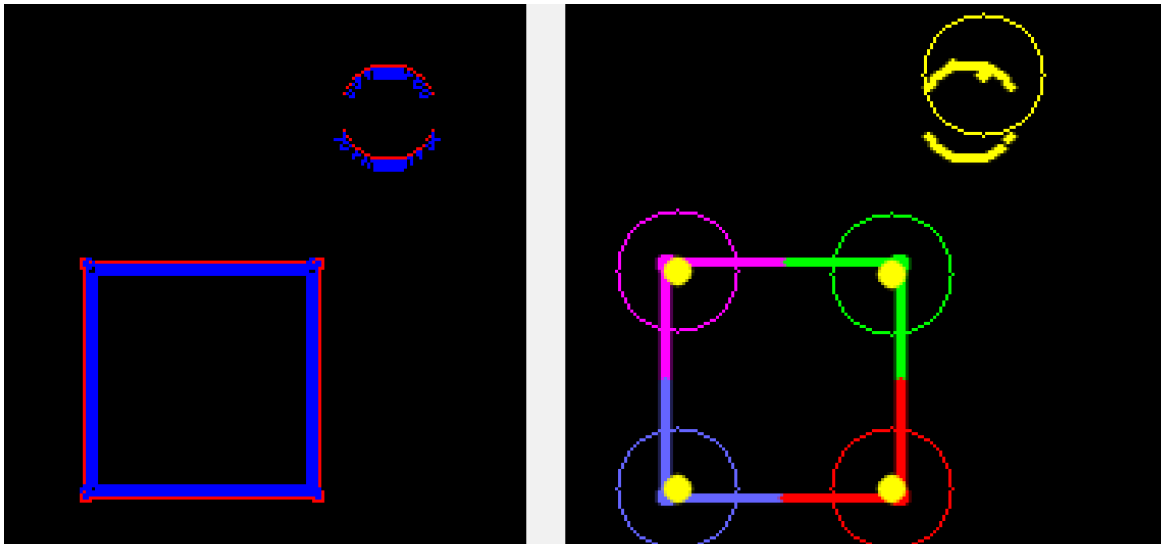


Figure 4.9: incorrectly computed cluster representative and clusters labelling by mean shift algorithm.

need to modify the bandwidth as a parameter to mean shift algorithm. Since the mean shift algorithm is not automatic and requires threshold adjustment according to each experiment case, we did not apply the merge algorithm on it.

As shown by these experiments, the sequential K-means algorithm achieved better results both in cluster quality and looming quality. Also, compared to the other algorithms it was much faster and the processing time per pseudo-frame is more than 46 times faster than OpenCV's K-means algorithm and 6 times faster than the mean shift algorithm. Our elbow method is also as fast and accurate as kneedle method. Applying the merge algorithm did not significantly increase the computational time.

The timing results shows that it takes the sequential K-means algorithm less than 0.8 milliseconds to process each pseudo-frame. The OpenCV K-means and mean shift algorithms require more than 33 and 4 milliseconds respectively to process each pseudo-frame. According to the length of pseudo-frames (L) that we selected for our experiments (about 16 milliseconds), the sequential K-means and mean shift algorithms have no difficulty with keeping up to process each pseudo-frame. On the other hand, OpenCV K-means is too slow and its processing time goes beyond the length of the pseudo-frame.

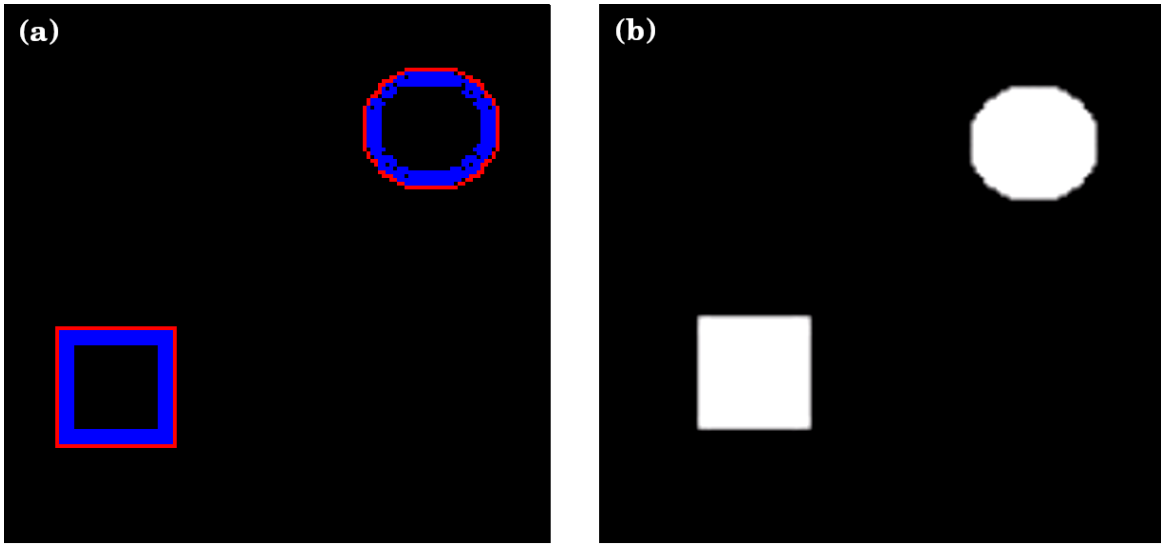


Figure 4.10: (a) The output of the optical flow algorithm on data set 2. (b) A frame of actual video.

Table 4.8: The results of experiments on data set 2 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (ms)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	100.00	0.547
	Kneedle	100.00	100.00	100.00	0.644
	Forced	100.00	100.00	100.00	0.536
OpenCV K-means	Elbow	100.00	100.00	100.00	36.071
	Kneedle	100.00	100.00	100.00	37.611
	Forced	100.00	100.00	100.00	35.995
Mean shift	-	70.27	100.00	100.00	7.194

From now on we will be brief and only show the results of the experiments for each data set unless there are something more to say.

4.3.2 Data set 2

In this experiment (Figure 4.10) two simulated objects (square and round) are looming simultaneously. As can be seen in this figure, all the optical flow arrows are pointing toward the outside which indicates the objects are looming. The results of using different clustering algorithms without applying the merge algorithm is shown in Table 4.8 and the results of applying the merge algorithm is shown in Table 4.9. Overall, this data set has 37 pseudo-frames. As can be seen in the results, both sequential K-means and OpenCV's K-means

Table 4.9: The results of experiments on data set 2 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	100.00	0.562
	Kneedle	100.00	100.00	100.00	0.673
OpenCV K-means	Elbow	100.00	100.00	100.00	36.151
	Kneedle	100.00	100.00	100.00	36.520

algorithms reported perfect results. However the mean shift algorithm was not able to detect clusters correctly and the correct number of clusters were reported in 26 out of 37 pseudo-frames.

Again in this experiment the computation time shows the advantage of using sequential *K*-means over other methods. With the same accuracy, it is more than 64 times faster than OpenCV's *K*-means algorithm and the computation time per pseudo-frame takes only 0.562 milliseconds.

4.3.3 Data set 3

This experiment is the same as the second experiment regarding the number, shape and type of movements of the objects. However, in this experiment the round object is looming continuously while the square object is looming during only part of the event stream. This means that the number of clusters are not constant during the whole event stream. The goal of this experiment is to detect whether the algorithms are able to adapt the detected number of clusters as soon as a new object starts moving or a moving object stops moving in the scene. Figure 4.11 shows a frame of the actual video and the output of the optical flow algorithm while the square object does not move. As can be seen in this figure since the square object has stopped moving, no events are reported by the optical flow algorithm, therefore the number of clusters will change during parts of the event stream. Table 4.10 shows the results of using clustering algorithm without applying our merge algorithm. In this experiment, using both sequential and OpenCV's *K*-means and both elbow and kneedle methods, 17 out of 46 pseudo-frames were reported with correct number of clusters. Since

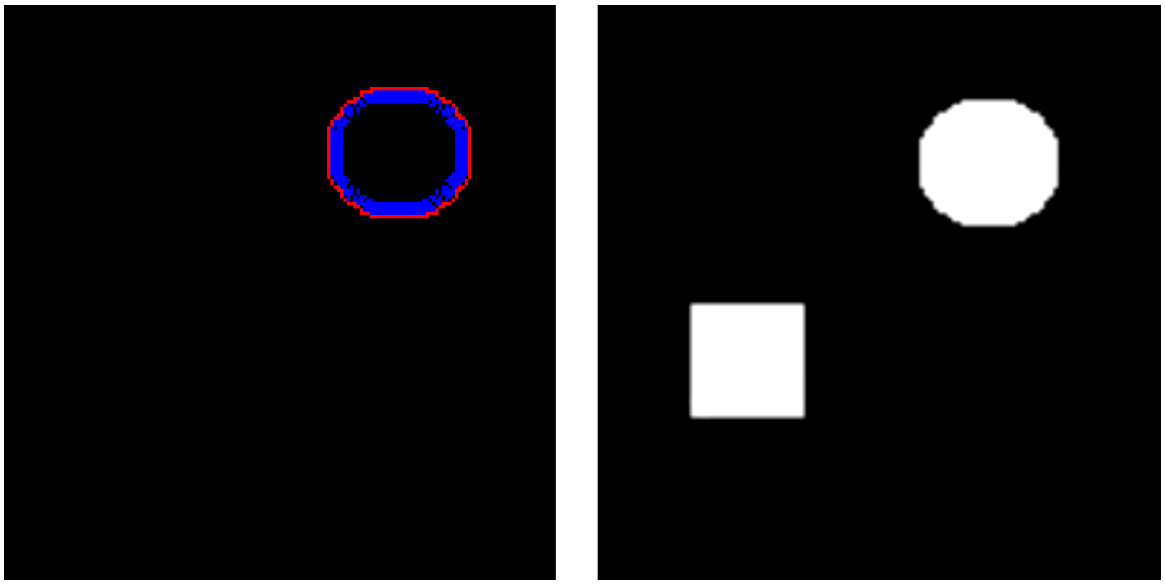


Figure 4.11: The square object has stopped moving while the round object keeps looming.

Table 4.10: The results of experiments on data set 3 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	36.95	100.00	97.05	0.686
	Kneedle	36.95	100.00	97.05	0.712
	Forced	100.00	100.00	97.05	0.667
OpenCV K-means	Elbow	34.78	100.00	97.05	30.898
	Kneedle	34.78	100.00	97.05	31.027
	Forced	100.00	100.00	97.05	30.031
Mean shift	-	78.26	100.00	97.05	4.033

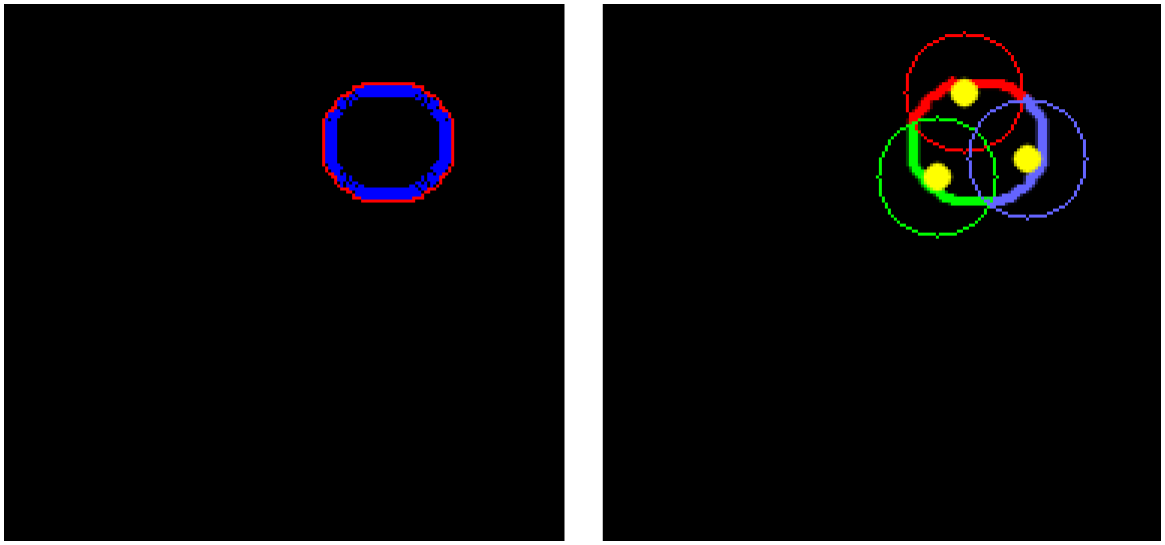


Figure 4.12: Sequential K-means and OpenCV's K-means algorithm failed to detect the correct number of clusters.

Table 4.11: The results of experiments on data set 3 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	97.82	100.00	97.05	0.695
	Kneedle	97.82	100.00	97.05	0.728
OpenCV K-means	Elbow	95.65	100.00	97.05	31.500
	Kneedle	95.65	100.00	97.05	32.147

the square stops looming after a while, the method detects the round object as multi clusters in 29 pseudo-frames (Figure 4.12). Although this is an incorrect labelling, each of these clusters are able to be used with looming detection algorithm to obtain the correct looming decision. Therefore the quality of looming detection is the same with forcing the number of clusters to be correct. The results reported by the mean shift algorithm shows that this algorithm was more successful than the other methods in detecting more pseudo-frames with the correct number of clusters. However this is highly dependent in the bandwidth chosen and could be different in different experiments.

Table 4.11 shows the results of applying our merge algorithm on this experiment. As can be seen, the sequential K-means algorithm was able to detect more pseudo-frames with correct number of clusters. Overall, there were four pseudo-frames in which the optical

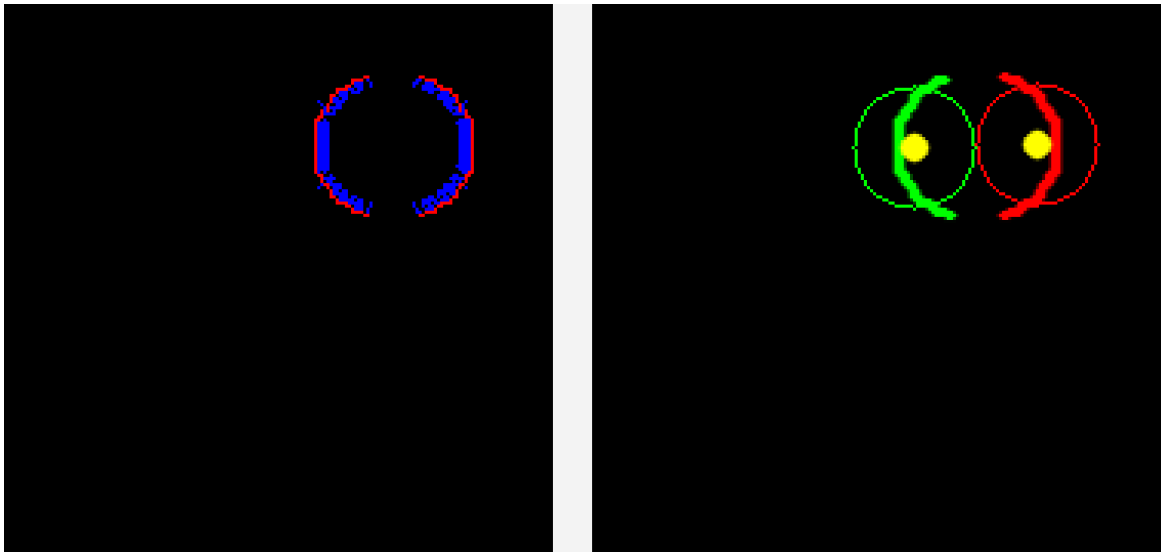


Figure 4.13: The round object was reported partly by optical flow algorithm.

flow algorithm did not report the whole round object (Figure 4.13). As can be seen in this figure the merge algorithm enhanced the results drastically. The sequential K-means algorithm reported one of these pseudo-frames as separate clusters and detected the rest as a single cluster. The OpenCV's K-means algorithm however detected two of them as separate clusters and two of them as single cluster. We should note that by considering only the optical flow outputs and without any prior assumption of the shape of objects, even human may cluster them as separate clusters.

Again in this experiment, there is a drastic difference between the computational time of the sequential K-means algorithm and OpenCV's K-means algorithm. Using sequential K-means algorithm we obtained more accurate results than the results reported by OpenCV's K-means and at least 46 times faster.

4.3.4 Data set 4

In this experiment two non-convex objects are moving sideways back and forth. We know that the looming detection algorithm may fail to detect looming for non-convex objects (Secession 2.5.2). Therefore the goal of this experiments is to determine whether the clustering algorithms are able to cluster this data set. A frame of the actual video and the

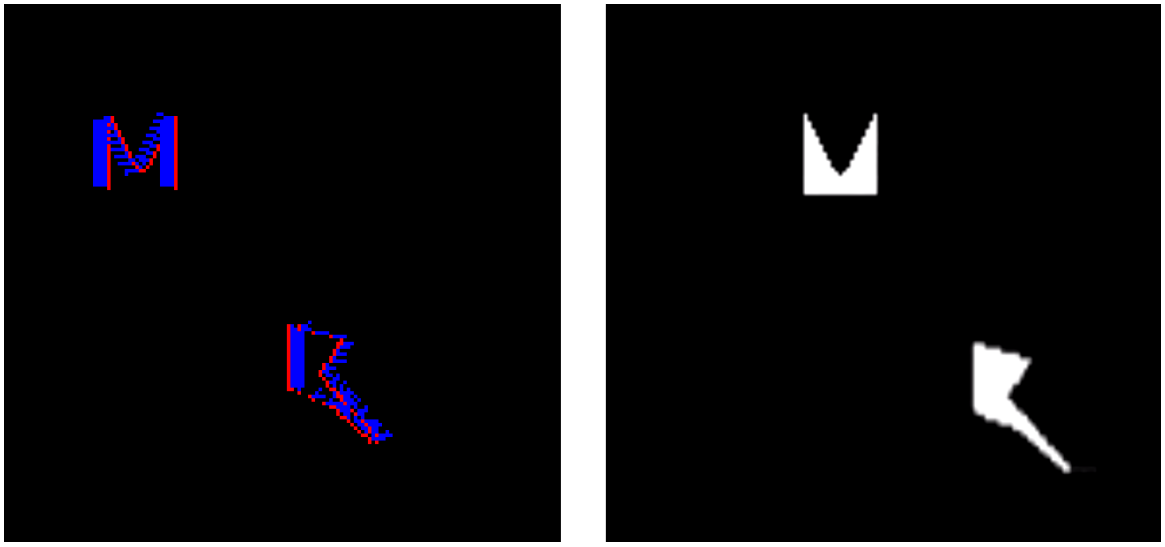


Figure 4.14: Two non-convex objects are moving sideways.

Table 4.12: The results of experiments on data set 4 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Processing time / Pseudo-frame (ms)
Sequential K-means	Elbow	100.00	0.690
	Kneedle	100.00	0.719
	Forced	100.00	0.686
OpenCV K-means	Elbow	100.00	23.082
	Kneedle	100.00	23.143
	Forced	100.00	23.046
Mean shift	-	100.00	1.067

optical flow events are shown in Figure 4.14. We showed the clustering results without applying the merge algorithm in Table 4.12 and the results of applying our merge algorithm are shown in Table 4.13. We did not check looming detection quality in this case. This data set has 57 pseudo-frames in which all of them clustered correctly using all algorithms. However using the sequential K-means algorithm, the computational time is at least 33 time faster compare to OpenCV's K-means algorithm and five times faster compere to the mean shift algorithm.

Table 4.13: The results of experiments on data set 4 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters $(\%)$	Processing time / Pseudo-frame (ms)
Sequential K-means	Elbow	100.00	0.697
	Kneedle	100.00	0.727
OpenCV K-means	Elbow	100.00	23.330
	Kneedle	100.00	23.360

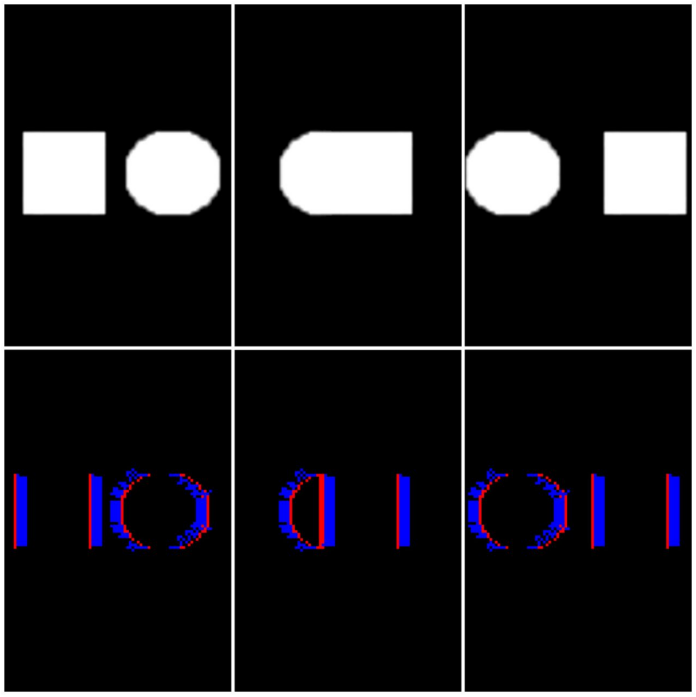


Figure 4.15: Two objects are moving toward and passing through each other.

4.3.5 Data set 5

In this experiment, a round object and a square object are moving toward and passing through each other. The goal of this experiment is to check the ability of the clustering algorithms to see to what extent they can cluster objects in merging situation. Figure 4.15 shows three pseudo-frames of before, at the moment and after merging of the objects. These pseudo-frames are taken form the actual video and the optical flow events. Overall this data set has 79 pseudo-frames in which the objects are merging in 35 pseudo-frames and in the rest of them the objects are separated. Table 4.14 shows the results of applying different clustering algorithms without using the merge algorithm and Table 4.15 shows the results of

Table 4.14: The results of experiments on data set 5 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	31.64	64.55	100.00	0.650
	Kneedle	31.64	69.62	100.00	0.668
	Forced	100.00	79.74	100.00	0.645
OpenCV K-means	Elbow	34.17	69.62	100.00	19,300
	Kneedle	34.17	68.35	100.00	19.403
	Forced	100.00	74.68	100.00	19.302
Mean shift	-	39.24	64.55	100.00	0.921

Table 4.15: The results of experiments on data set 5 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	31.64	64.55	100.00	0.662
	Kneedle	31.64	69.62	100.00	0.673
OpenCV K-means	Elbow	34.17	69.62	100.00	19.546
	Kneedle	34.17	68.35	100.00	20.413

applying the merge algorithm. As can be seen in these tables, based on the detected clusters, the looming detection algorithm detected looming incorrectly in many pseudo-frames. This detection decreased the precision of this algorithm in detecting looming objects. Since the objects are moving through each other, the optical flow events are pointing away from the centroid of the detected clusters, which cause the looming detection algorithm to detect looming (Figure 4.16). Since the DVS only generates events for leading and trailing edges of the square, in this case cluster detection would be difficult even with human supervision. Therefore, even applying the merge algorithm cannot enhance the results. However, despite of having the same accuracy, still the sequential *K*-means is faster than the other methods.

4.3.6 Data set 6

In this case, a round object is looming continuously while two square objects are moving sideways in parts of the event stream. Similar to data set 3, the number of clusters in this data set is also changing. Figure 4.17 shows a frame of the actual video and events generated by optical flow algorithm. As can be seen, since one of the squares is not moving, no events

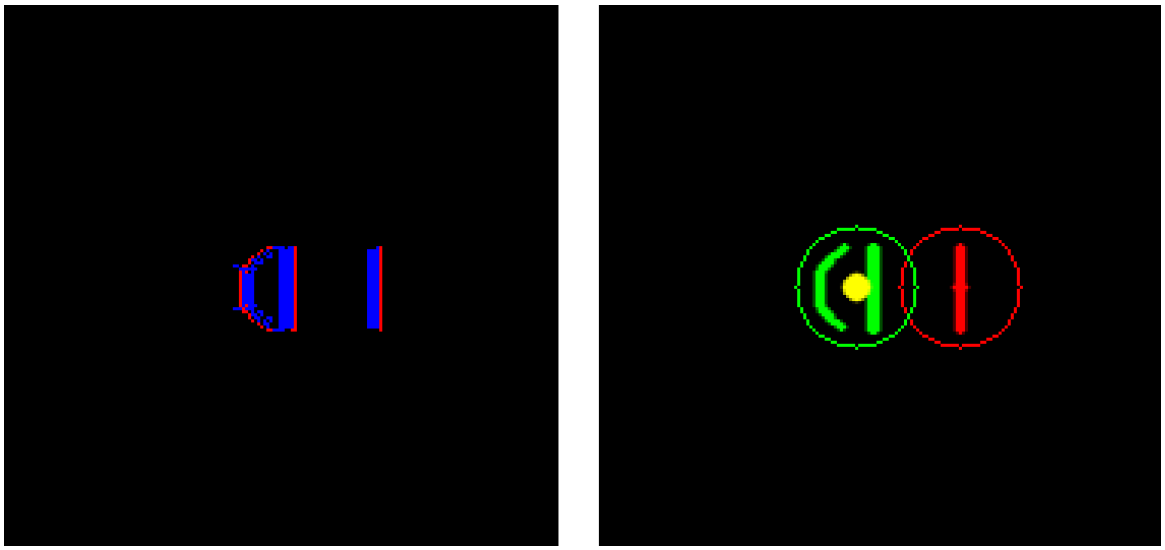


Figure 4.16: Incorrect looming detection.

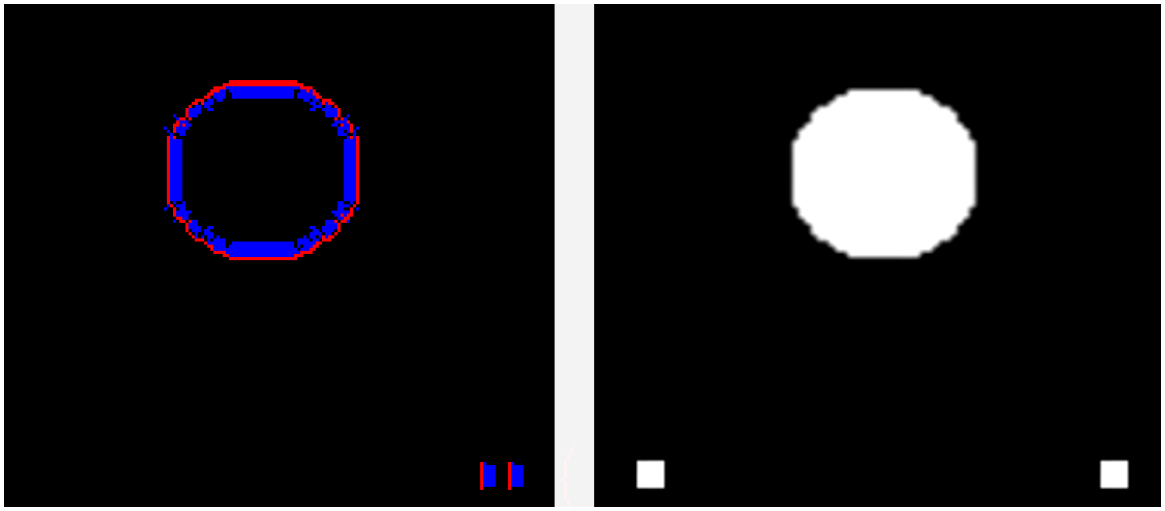


Figure 4.17: A continues looming round object and two squares which moving sideways in parts of the event stream.

Table 4.16: The results of experiments on data set 6 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	19.64	100.00	100.00	0.605
	Kneedle	19.64	100.00	100.00	0.820
	Forced	100.00	100.00	100.00	0.570
OpenCV K-means	Elbow	30.35	98.79	98.79	31.006
	Kneedle	23.21	100.00	98.19	31.149
	Forced	100.00	98.79	98.79	29.317
Mean shift	-	66.07	100.00	100.00	2.365

Table 4.17: The results of experiments on data set 6 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	92.85	99.39	97.60	0.651
	Kneedle	87.50	97.59	97.60	0.826
OpenCV K-means	Elbow	75.00	96.38	97.60	28.204
	Kneedle	76.78	97.57	98.17	29.968

were reported for it by optical flow algorithm. The results of using different clustering algorithm without applying the merge algorithm is shown in Table 4.16. The algorithms did not detected the correct number of clusters most of the time. However, by applying our merge algorithm we got the results showen in Table 4.17. Similar to Figure 4.12 as the round object gets closer to the camera, the clustering algorithms detect it as multiple clusters. In this situation by applying the merge algorithm we can correct this problem and report the correct number of clusters more often. The reported results by the sequential K-means algorithm are the most accurate and the fastest compared to other algorithms.

4.3.7 Data set 7

In this experiment five squares are looming synchronously. Figure 4.18 shows a frame of the actual video and the optical flow events. The results of using different clustering algorithms without applying the merge algorithm are shown in Table 4.18 and the results after applying the merge algorithm are summarized in Table 4.19. Since the resolution of the DVS is limited, adding more objects in the scene reduces the accuracy of the clustering

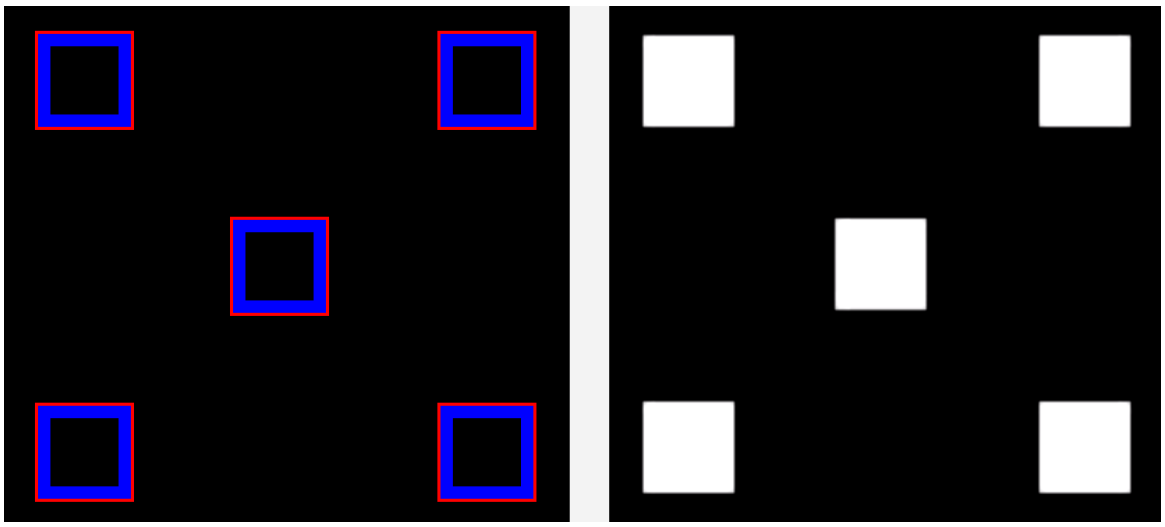


Figure 4.18: Five equal squares are looming.

Table 4.18: The results of experiments on data set 7 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	64.70	100.00	66.66	0.832
	Kneedle	64.70	100.00	66.66	0.928
	Forced	100.00	100.00	69.24	0.819
OpenCV K-means	Elbow	5.80	100.00	41.81	52.121
	Kneedle	0.00	100.00	41.81	55.318
	Forced	100.00	100.00	100.00	54.088
Mean shift	-	47.05	100.00	100.00	58.708

Table 4.19: The results of experiments on data set 7 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	64.70	100.00	65.95	0.961
	Kneedle	64.70	100.00	55.55	1.002
OpenCV K-means	Elbow	5.80	100.00	15.38	57.987
	Kneedle	0.00	100.00	0.00	58.031

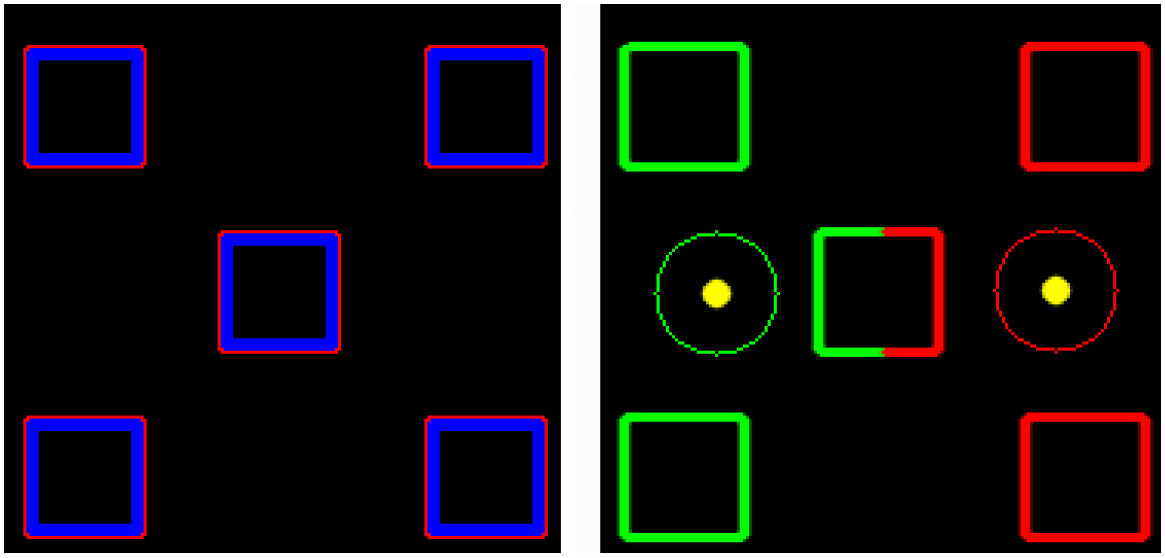


Figure 4.19: incorrect cluster labelling by OpenCV's K-means algorithm.

algorithm. In this situation, using the sequential K-means algorithm with the elbow method gives us the highest accuracy and fastest detection per pseudo-frames. For the OpenCV's K-means algorithm and elbow method, only one pseudo-frame with the correct number of clusters was reported and by using this algorithm and kneedle, no pseudo-frame with correct number of clusters was detected. This algorithm labelled the clusters incorrectly and divided them into two clusters. Figure 4.19 shows a pseudo-frame in which the OpenCV's K-means algorithm failed to detect the correct number of clusters. Although the mean shift algorithm detected the correct number of clusters in 47 percent of the pseudo-frames, it is still less accurate than the sequential K-means algorithm. In this experiment, it took the mean shift algorithm about 59 milliseconds to process a single pseudo-frame which is the highest processing time among all clustering algorithms. The sequential K-means however is at least 52 times faster than other algorithms. This happens because there are significantly more events in each pseudo-frame in this data set than in other data sets. The complexity of the OpenCV's K-means algorithm and the mean shift algorithm is highly affected by the number of events in a pseudo-frame. However, the run time of the sequential K-means algorithm for each event is not affected.

Table 4.20: The results of experiments on data set 8 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	0.00	100.00	100.00	0.641
	Kneedle	0.00	100.00	100.00	0.712
	Forced	100.00	100.00	100.00	0.635
OpenCV K-means	Elbow	0.00	100.00	100.00	24.838
	Kneedle	0.00	100.00	100.00	26.451
	Forced	100.00	100.00	100.00	24.284
Mean shift	-	100.00	100.00	100.00	2.585

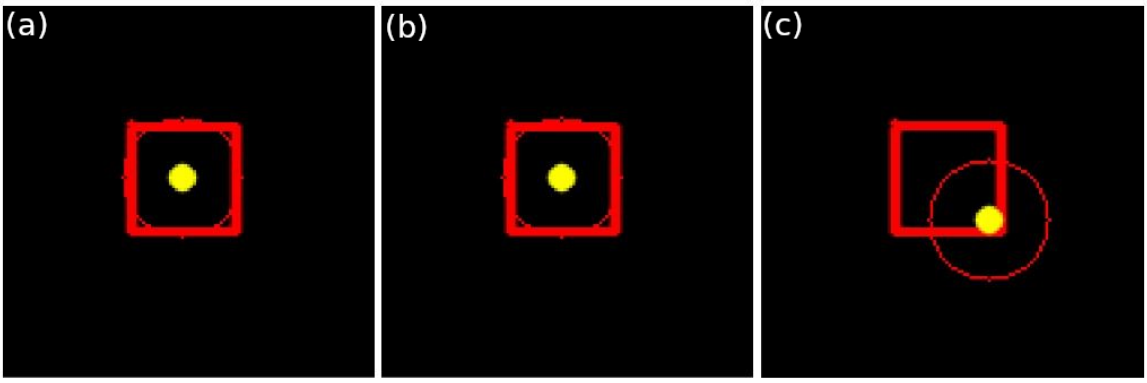


Figure 4.20: (a) The computed centroid by sequential K-means algorithm while the merge algorithm is applied. (b) The computed centroid by OpenCV's K-means algorithm while the merge algorithm is applied. (c) The computed cluster representative by mean shift algorithm.

4.3.8 Data set 8

In this experiment, a square is looming continuously. The goal of this experiment is to check the ability of the clustering algorithms to detect a single cluster. We gathered the results of this experiment without using the merge algorithm in Table 4.20. As can be seen in this table, the only algorithm that was able to detect correct number of clusters in all pseudo-frames is mean shift. However this is highly dependent to the bandwidth we set as a parameter for this method. For example, by adjusting the bandwidth to five, the mean shift algorithm could not detect any pseudo-frame with correct number of clusters. In addition, compared to other methods, using the mean shift algorithm, the computed cluster representative is not in the centre of the object (Figure 4.20). By applying our merge algorithm, both sequential and OpenCV's K-means algorithms were able to detect

Table 4.21: The results of experiments on data set 8 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	100.00	0.635
	Kneedle	100.00	100.00	100.00	0.794
OpenCV K-means	Elbow	100.00	100.00	100.00	25.008
	Kneedle	100.00	100.00	100.00	27.230

Table 4.22: The results of experiments on data set 9 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	0.00	100.00	100.00	0.425
	Kneedle	0.00	100.00	100.00	0.708
	Forced	100.00	100.00	100.00	0.421
OpenCV K-means	Elbow	0.00	100.00	100.00	12.395
	Kneedle	0.00	100.00	100.00	14.291
	Forced	100.00	100.00	100.00	12.212
Mean shift	-	100.00	100.00	100.00	0.923

correct number of clusters in all pseudo-frames. We gathered the results of applying the merge algorithm on this experiment in Table 4.21.

4.3.9 Data set 9

This experiment is similar to data set 8. However, in this case a single square is moving sideways. Since the DVS cannot detect changes at the top and down edges of the square, we experimented on this case to examine the clustering algorithms in such situations. Tables 4.22 and 4.23 show the results of this experiment without and with the merge algorithm respectively. As can be seen in this Tables, even by applying the merge algorithm, the sequential *K*-means and OpenCV's *K*'means algorithms were not able to detect any pseudo-

Table 4.23: The results of experiments on data set 9 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	0.00	100.00	100.00	0.446
	Kneedle	0.00	100.00	100.00	0.669
OpenCV K-means	Elbow	0.00	100.00	100.00	13.424
	Kneedle	0.00	100.00	100.00	14.140

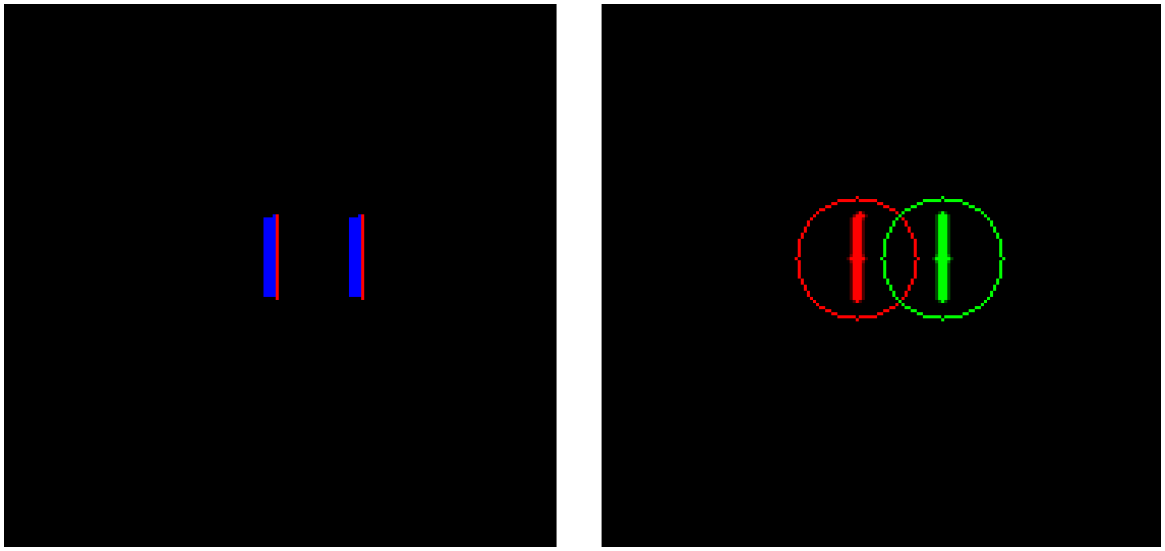


Figure 4.21: A square is moving sideways.

frames with correct number of clusters. Although the results seem incorrect, the logic of the clustering algorithm is correct. This is because of the way the DVS generates events only for the leading and trailing edges. Therefore, the clustering algorithms observe the square as two columns and detect them as separated clusters (Figure 4.21). Since these two columns do not touch each other, the merge algorithm cannot merge them into a single cluster. In this situation, detecting the correct number of cluster would be challenging even for human without any prior assumptions on the shape of the object and number of clusters.

The mean shift algorithm on the other hand was able to detect single cluster in all pseudo-frames. However, as mentioned before this correct detection is highly dependent to the dimensions of the object matching the bandwidth parameter.

4.3.10 Data set 10

In this experiment we attempt to determine to what extent the objects can get close to each other and still be clustered as different objects by the clustering algorithms. Since in previous experiments we observed that the merge algorithm enhances the final results, we apply this method to report the results of this experiment. These results are summarized in Table 4.24. For this purpose we considered different scenarios. As the first scenario which

Table 4.24: The results of experiments on data set 10. The numbers are the minimum distance between the objects such that the clustering algorithms report two distinct clusters instead of one.

		Distance (pixel)		
Clustering Methods	Elbow detecting method	Passing	Looming	Approaching
Sequential K-means	Elbow	1	1	7
	Kneedle	1	1	10
	Forced	1	1	5
OpenCV K-means	Elbow	16	1	7
	Kneedle	14	1	7
	Forced	7	1	5
Mean shift	-	31	5	4

we showed its results in the table under the passing column, two equal squares are passing by each other. The second experiment is on the situation which two looming squares are approaching each other from one of their corners. The results of this case are shown under the looming column. As for the last case, two equal squares are approaching each other. The results of this case are shown under the approaching column. Figure 4.22 shows a frame of the actual video of these case and their optical flow events. The numbers shown in the table are the minimum distance between objects such that the clustering algorithms report two distinct objects instead of one. As can be seen, the sequential K-means algorithm acts better in terms of clustering the close objects compared to other methods.

4.4 Experiments on captured data sets

In this section, the effectiveness of the clustering algorithms are experimented on captured data sets. As in the previous section, we compare the algorithms on their ability to cluster different objects, by reporting the percentage of the pseudo-frames in which the correct number of clusters are reported. The quality of the cluster labelling are also checked in each pseudo-frames. Then we report the precision and recall of the looming detection algorithm for the detected clusters.

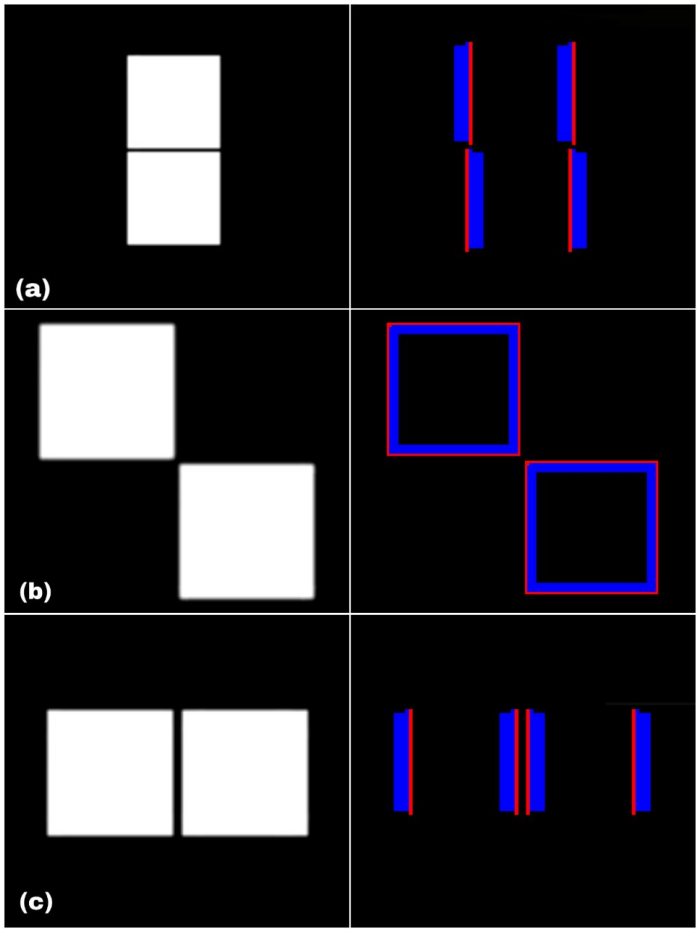


Figure 4.22: (a) Two squares are passing by each other. (b) Two squares are looming and one of their corners are approaching to each other. (c) Two squares are approaching to each other.

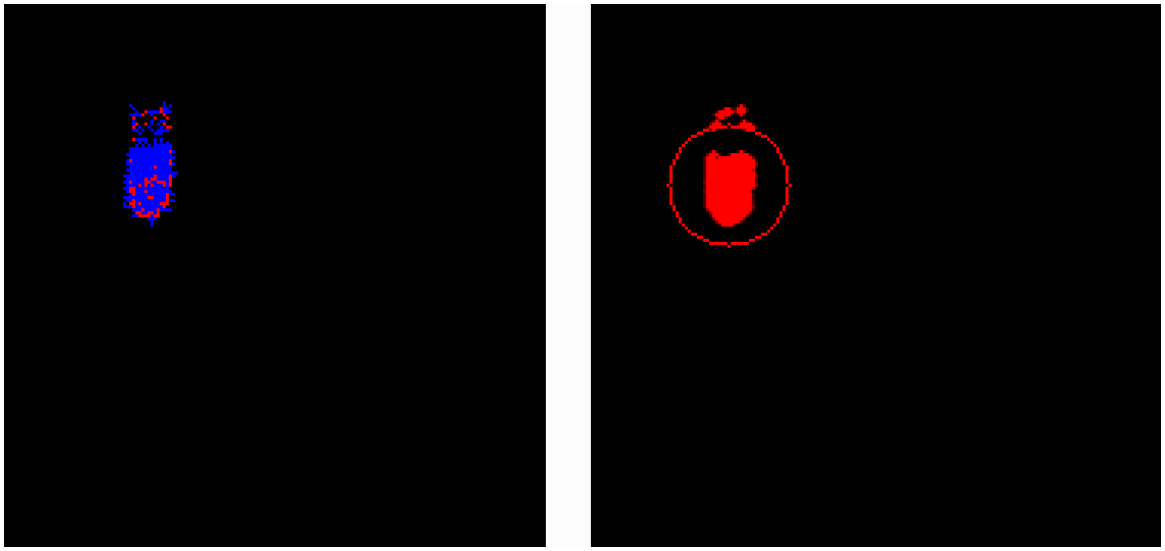


Figure 4.23: Single ball is falling (sideways movement). This pseudo-frame is recorded when the merge algorithm has been applied.

Table 4.25: The results of experiments on captured data set 1 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	0.00	22.22	100.00	0.570
	Kneedle	0.00	22.22	100.00	0.608
	Forced	100.00	88.88	100.00	0.544
OpenCV K-means	Elbow	0.00	0.00	0.00	28.730
	Kneedle	0.00	0.00	0.00	29.269
	Forced	100.00	83.33	100.00	27.266
Mean shift	-	100.00	83.33	100.00	5.811

4.4.1 Data set 1

In this data set, a ball is falling in front of the camera. The goal of this experiment is to determine whether the clustering algorithms are capable of detecting a single cluster while we working on captured data sets. Figure 4.23 shows the optical flow events and output of clustering for one of the pseudo-frames of this data set. Without using the merge algorithm, the results are reported in Table 4.25. As can be seen in this table, both sequential and OpenCV's *K*-means algorithms failed to detect correct number of clusters in all pseudo-frames. Only the mean shift algorithm was able to detect the correct number of clusters in all pseudo-frames as the bandwidth matches size of the object. However this is a coincident

Table 4.26: The results of experiments on captured data set 1 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	72.22	66.66	100.00	0.543
	Kneedle	77.77	66.66	100.00	0.754
OpenCV K-means	Elbow	88.88	72.22	100.00	28.125
	Kneedle	88.88	72.22	100.00	29.474

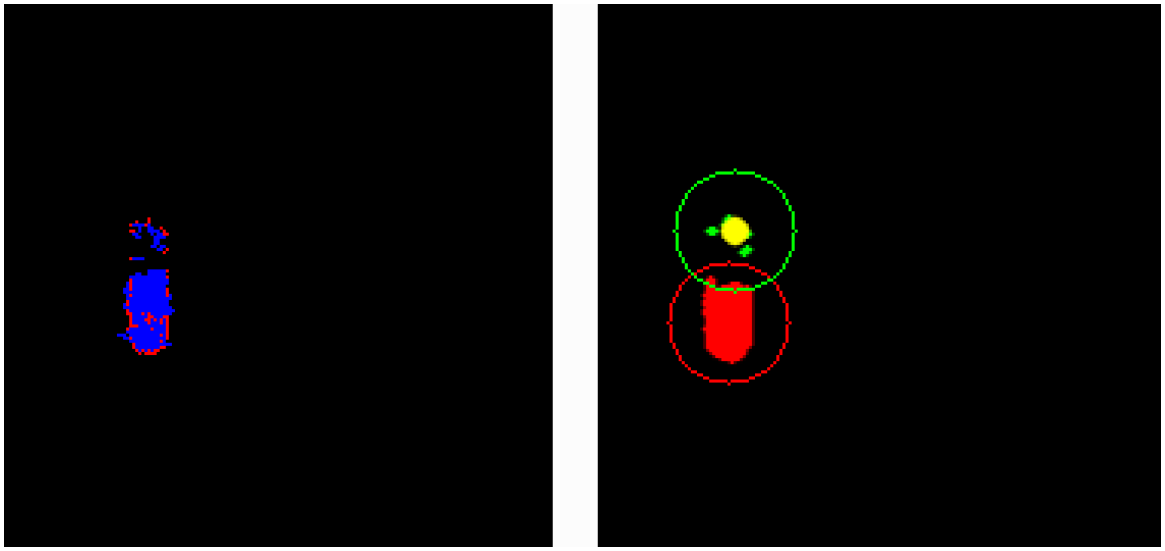


Figure 4.24: Incorrect number of clusters due to the noisy conditions.

and changing the bandwidth leads to reporting totally different results.

By applying our merge algorithm, the algorithms reported the results summarized in Table 4.26. Overall applying the merge algorithm enhanced the results drastically. Using the elbow method and sequential K-means algorithm, only five pseudo-frames with the incorrect number of detected clusters were reported. The reason is because these pseudo-frames are very noisy, and even noise removal pre-processing was not able to remove all noises (Figure 4.24). Using the kneedle method and sequential K-means algorithm the incorrect number of clusters were reported in four pseudo-frames and using OpenCV's K-means and both elbow and kneedle methods, two pseudo-frames reported incorrect number of clusters. This is because the sequential K-means produces tighter clusters compared to the OpenCV's K-means algorithm. By using the merge algorithm, the labelled clusters by

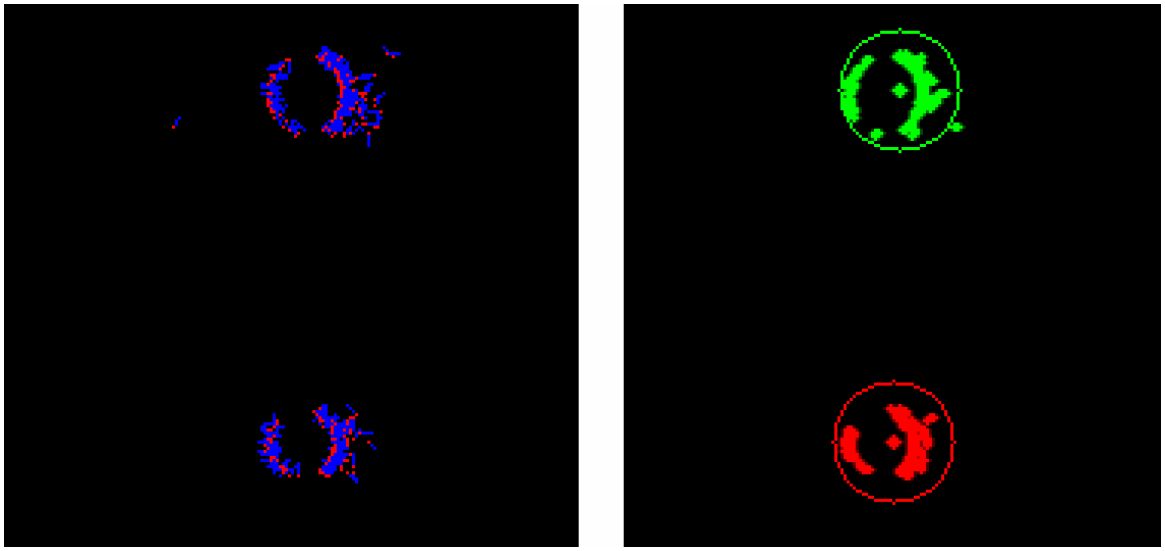


Figure 4.25: Two round objects are moving sideways.

the sequential K-means remain separate while they merge to one clusters for reported clusters by the OpenCV's K-means algorithm. This is why the OpenCV's K-means algorithm is reporting better results. However, qualitatively, even human may say they are separate clusters. In addition, 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.

4.4.2 Data set 2

In this data set two round objects are moving sideways. To capture this data set, we used two black round magnets and moved them using two other magnets under a white surface. 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 4.25 shows the optical flow events and clustering output for a pseudo-frame of this data set. The results of applying different clustering algorithms without using the merge algorithm are summarized in Table 4.27. As can be seen, all algorithms were able to detect the correct number of clusters in all pseudo-frames. However, the precision of looming detection for the mean shift algorithm is the lowest. This is because of the way the mean shift algorithm calculates the representative of the clusters. By applying the merge algorithm, same results

Table 4.27: The results of experiments on captured data set 2 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	96.90	100.00	0.652
	Kneedle	100.00	96.90	100.00	0.716
	Forced	100.00	96.90	100.00	0.648
OpenCV K-means	Elbow	100.00	97.93	100.00	26.151
	Kneedle	100.00	97.93	100.00	26.245
	Forced	100.00	97.93	100.00	26.021
Mean shift	-	100.00	95.53	100.00	2.576

Table 4.28: The results of experiments on captured data set 2 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	96.90	100.00	0.693
	Kneedle	100.00	96.90	100.00	0.731
OpenCV K-means	Elbow	100.00	97.93	100.00	26.944
	Kneedle	100.00	97.93	100.00	27.456

have reported. These results are shown in Table 4.28. Again in this case the fastest algorithm is sequential *K*-means which on average took about 0.6 milliseconds to process each pseudo-frame.

4.4.3 Data set 3

In this experiment, 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. Table 4.29 shows the results of this experiment without using the merge algorithm. As can be seen by these results, 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 multi clusters to decrease the average squared distance form each event to the computed centroid. Figure 4.26 shows a pseudo-frame in such situation. By applying the merge algorithm we get the results shown in Table 4.30. Overall, applying the merge algorithm enhanced the results. Figure 4.27 shows

Table 4.29: The results of experiments on captured data set 3 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	0.00	98.63	97.29	0.731
	Kneedle	0.00	97.26	97.26	0.787
	Forced	100.00	100.00	90.41	0.674
OpenCV K-means	Elbow	0.00	95.83	94.52	41.653
	Kneedle	0.00	95.83	94.52	41.997
	Forced	100.00	100.00	87.67	41.495
Mean shift	-	0.00	93.24	94.52	19.465

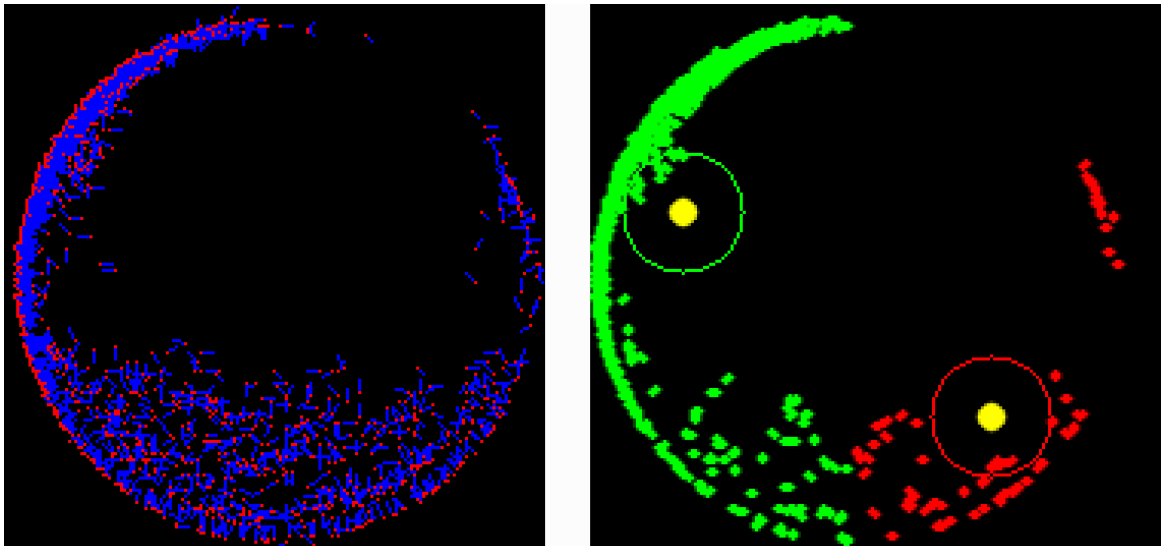


Figure 4.26: A single looming ball which incorrectly detected as more than a single cluster. In this pseudo-frame the merge algorithm was not applied.

Table 4.30: The results of experiments on captured data set 3 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	41.09	98.59	95.89	0.799
	Kneedle	45.20	98.55	93.15	0.879
OpenCV K-means	Elbow	56.16	95.71	91.17	41.596
	Kneedle	57.73	95.71	91.17	42.323

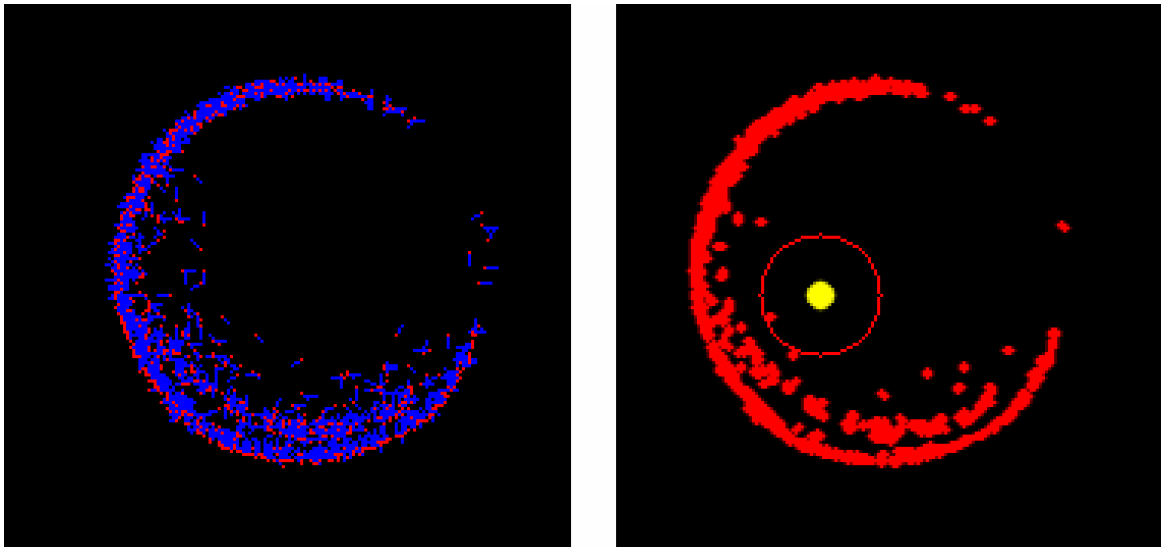


Figure 4.27: Single ball is looming and detected as a single cluster by applying the merge algorithm.

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. In this experiment, the size of object did not match with the bandwidth parameter and caused the mean shift algorithm to fail detecting the correct number of clusters in all pseudo-frames.

Regarding the computational time, the sequential K-means algorithm is the fastest method and processed each pseudo-frame about 60 times faster than OpenCV's K-means algorithm. Compared to the mean shift algorithm, the sequential K-means algorithm is about 26 times faster.

4.4.4 Data set 4

In this data set, two balls are rolling toward the camera. Figure 4.28 shows the output of optical flow events and clustering in one of the pseudo-frames of this case. The results of this experiment without applying the merge algorithm is shown in Table 4.31 and the results by applying the merge algorithm is shown in Table 4.32 As can be seen in both cases, all

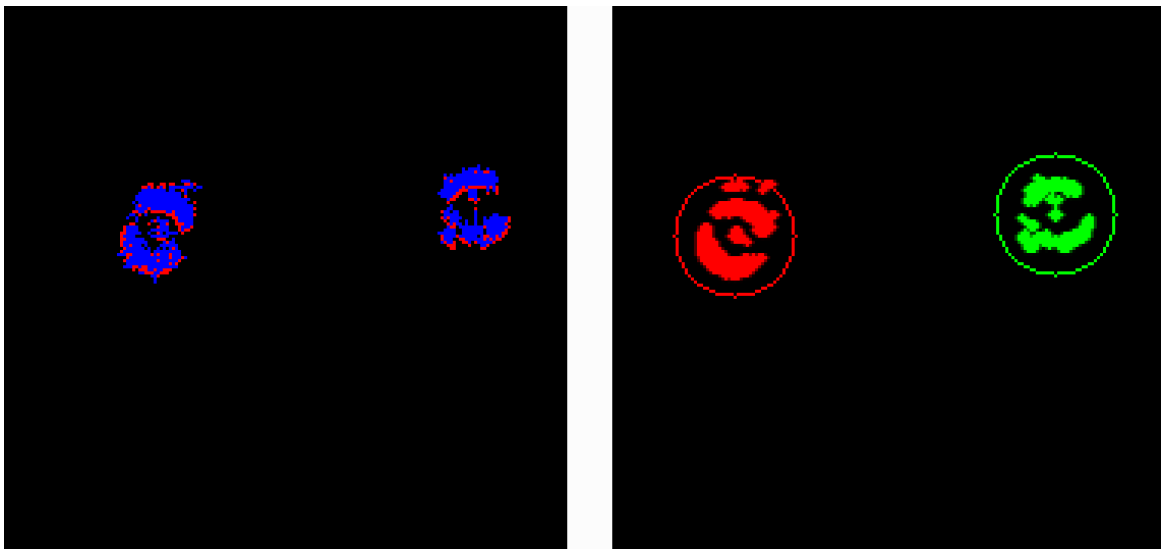


Figure 4.28: Two balls are rolling toward the camera (looming).

Table 4.31: The results of experiments on captured data set 4 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	32.35	0.757
	Kneedle	100.00	100.00	32.35	0.783
	Forced	100.00	100.00	32.35	0.644
OpenCV K-means	Elbow	100.00	100.00	32.35	38.953
	Kneedle	100.00	100.00	32.35	41.769
	Forced	100.00	100.00	32.35	38.918
Mean shift	-	100.00	100.00	32.35	36.593

Table 4.32: The results of experiments on captured data set 4 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	32.35	0.787
	Kneedle	100.00	100.00	32.35	0.921
OpenCV K-means	Elbow	100.00	100.00	32.35	39.948
	Kneedle	100.00	100.00	32.35	41.199

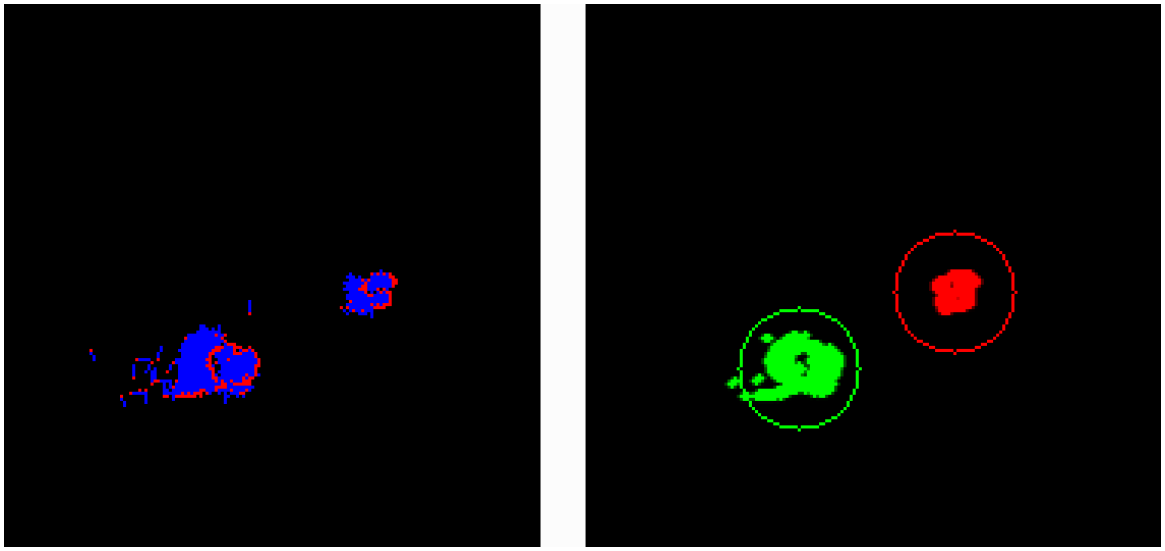


Figure 4.29: Two balls are rolling sideways.

methods were able to detect the correct number of clusters in all pseudo-frames. However, the recall of the looming detection is very low. This is because as a real world example, these balls are not moving toward the centre of the camera's lens and their movement is a combination of looming and sideways movement. We realized that Ridwan's looming detection algorithm [49] cannot detect whether the object is looming correctly in such situations and the objects need to be just moving toward the camera's lens to be detected as a looming object by Ridwan's algorithm.

4.4.5 Data set 5

In this case, two balls are rolling sideways. Figure 4.29 shows a pseudo-frame of the optical flow and clustering output of this data set. The results of experiments on this data set without applying the merge algorithm is shown in Table 4.33. As can be seen the algorithms were able to detect the correct number of clusters in most pseudo-frames. However, by applying our merge algorithm, we can enhance these results to 100 percent. Table 4.34 shows the results after applying the merge algorithm.

Table 4.33: The results of experiments on captured data set 5 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	83.33	80.55	100.00	0.616
	Kneedle	83.33	80.55	100.00	0.691
	Forced	100.00	80.55	100.00	0.594
OpenCV K-means	Elbow	94.44	88.88	100.00	34.349
	Kneedle	94.44	88.88	100.00	34.615
	Forced	100.00	88.88	100.00	34.289
Mean shift	-	33.33	88.88	100.00	13.996

Table 4.34: The results of experiments on captured data set 5 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	80.55	100.00	0.632
	Kneedle	100.00	80.55	100.00	0.731
OpenCV K-means	Elbow	100.00	88.88	100.00	34.514
	Kneedle	100.00	88.88	100.00	34.880

4.4.6 Data set 6

In this case, the camera is moving toward four round objects in a solid white background. Figure 4.30 shows the optical flow events and clustering output for a single pseudo-frame of this case. The results running different clustering algorithms without applying the merge algorithm are shown in Table 4.35 and the results after applying the merge algorithm are shown in Table 4.36. The sequential *K*-means algorithm is the most accurate and the

Table 4.35: The results of experiments on captured data set 6 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	5.71	0.570
	Kneedle	100.00	100.00	5.71	0.687
	Forced	100.00	100.00	5.71	0.562
OpenCV K-means	Elbow	25.71	100.00	7.14	36.284
	Kneedle	25.71	100.00	6.42	36.302
	Forced	100.00	100.00	6.42	36.211
Mean shift	-	25.71	100.00	7.14	13.459

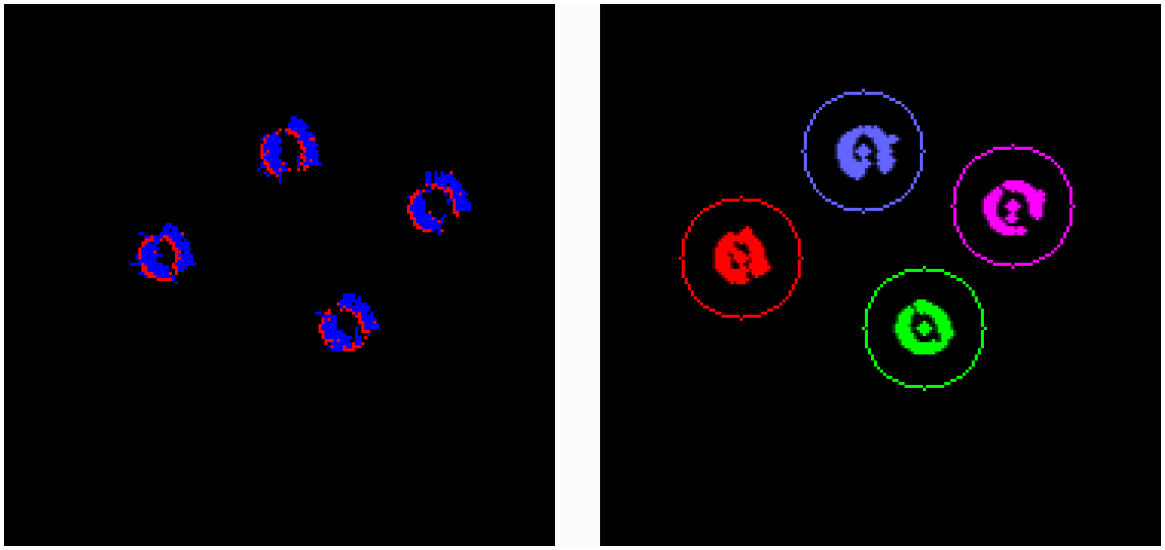


Figure 4.30: Four round objects are looming.

Table 4.36: The results of experiments on captured data set 6 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters (%)	Looming Correctness (%)		Processing time / Pseudo-frame (*ms*)
			Precision	Recall	
Sequential K-means	Elbow	100.00	100.00	5.71	0.667
	Kneedle	100.00	100.00	5.71	0.703
OpenCV K-means	Elbow	25.71	100.00	7.14	36.391
	Kneedle	25.71	100.00	6.42	36.820

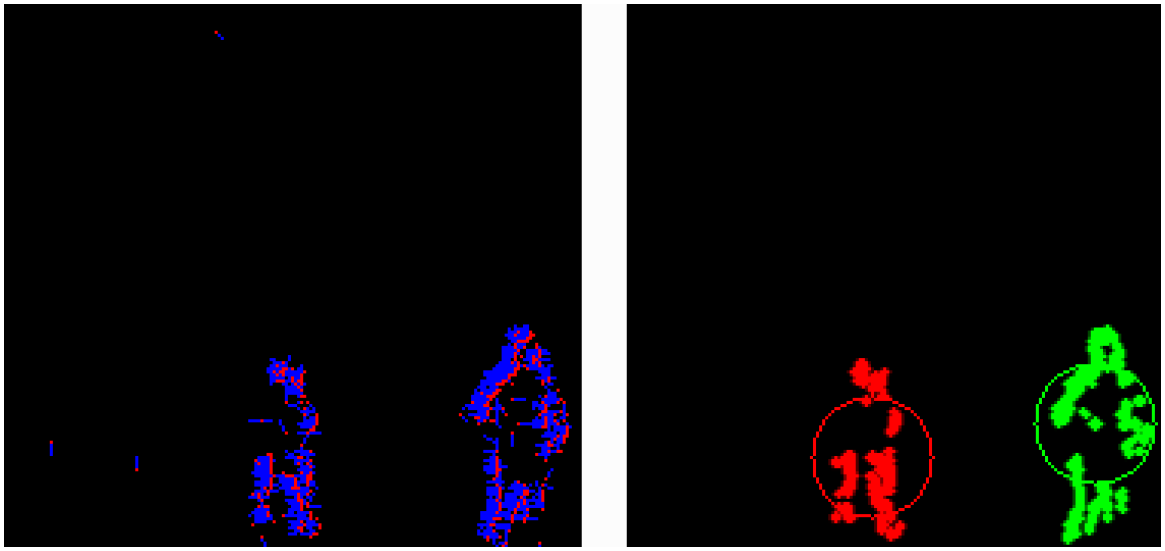


Figure 4.31: Two humans are moving in front of the DVS.

fastest method compared to other methods. It took this algorithm about 0.6 milliseconds to process each pseudo-frame and it was able to detect correct number of clusters in all pseudo-frames.

4.4.7 Data set 7

In this case, two humans are moving in front of the camera. The goal of this experiment is to examine the ability of different clustering algorithms in clustering other types of objects than geometrical objects. Figure 4.31 shows the output of the optical flow algorithm and clustering algorithms in a single pseudo-frame of this data set. The results of this experiment are shown in Tables 4.37 and 4.38 with and without applying the merge algorithm respectively. Since we are aware that the looming detection algorithm only works for convex objects and the way the optical flow algorithm generates events when a human is moving may not be a convex shape, we did not test the looming quality for this particular test. As can be seen, regarding the cluster detection the sequential and OpenCV's K-means algorithms worked perfect and detected correct number of clusters in all pseudo-frames. The mean shift however detected correct number of clusters in 86 percent of pseudo-frames. Again in this case, the sequential K-means algorithm was able to process each pseudo-

Table 4.37: The results of experiments on captured data set 7 without applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters $(\%)$	Processing time / Pseudo-frame (ms)
Sequential K-means	Elbow	98.80	0.685
	Kneedle	98.80	0.745
	Forced	100.00	0.634
OpenCV K-means	Elbow	100.00	27.325
	Kneedle	100.00	27.390
	Forced	100.00	27.222
Mean shift	-	86.90	3.152

Table 4.38: The results of experiments on captured data set 7 by applying the merge algorithm.

Clustering Methods	Number of clusters detecting method	Correct number of clusters $(\%)$	Processing time / Pseudo-frame (ms)
Sequential K-means	Elbow	100.00	0.713
	Kneedle	100.00	0.788
OpenCV K-means	Elbow	100.00	27.393
	Kneedle	100.00	27.407

frame about 38 times faster than OpenCV's K-means algorithm and about 5 times faster than the mean shift algorithm.

4.5 Discussion

The results of the experiments indicate the advantages of using sequential K-means algorithm compared to other methods. First of all, this is a real-time algorithm. Also, this algorithm does not require any parameter adjustment and it can automatically adapt itself in all experiments cases. 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 and more than about four times faster than the mean shift algorithm. In addition of being automatic and fast, the sequential K-means algorithm achieved the highest accuracy in cluster detection compared to other algorithms in most experiment cases.

Due to the way that the sequential K-means algorithm works, it only searches for the closest clusters and then does a simple update. The complexity remains the same even if there are many events in the pseudo-frame. Therefore the complexity of processing each event is still the same. On the other hand, with the conventional K-means (Algorithm 1) and mean shift (Algorithms 2 and 3) algorithms, their complexity depends on the number of data point in each pseudo-frame. Therefore if the number of data points in pseudo-frames is large, the sequential K-means algorithm is much faster. This behavior is confirmed in simulated data set 7 (Table 4.18).

For our experiments with looming detection algorithm, we chose the length of the pseudo-frames (L) to be about 16 milliseconds. By comparing the timing results of the clustering algorithms to this number, we realize that only the sequential K-means algorithm is able to process each pseudo-frame in less than 0.8 milliseconds which is significantly less than this time. The OpenCV's K-means algorithm is too slow and its processing time is longer than the length of each pseudo-frame. As for the mean-shift algorithm, depending

on the number of events in each pseudo-frame, its processing time can takes from about 5 milliseconds to more than the length of the pseudo-frame. Therefore, the K-means algorithm and mean shift algorithm will not be able to process a pseudo-frame before the next one is available.

Our elbow method is a fast and accurate algorithm capable of detecting the number of clusters without any prior assumption on the shape or number of the clusters. As can be seen by results, it is as accurate and fast as kneedle method (even more accurate and faster in some cases).

Finally, our merge algorithm enhanced the results of cluster detection in most of the experiments. Regarding the processing time, it is also a very fast method. Overall, the combination of using sequential K-means, elbow and merge algorithms provides us the most accurate results with the lowest processing time. It is automatic and neither prior assumptions on the shape and number of clusters nor parameter adjustment is required. In addition, for processing the events, our algorithms do not require special hardware or parallel processors and they can be run on accessible systems such as laptops efficiently.

Regarding the quality of looming detection algorithm, using the output of the sequential K-means algorithm and looming detection algorithm provided us the most accurate results in most of our experiments. As for the remaining experiments, its results were as good as using other clustering algorithms with looming detection algorithm. During our experiments we realized that the Ridwan's looming detection algorithm [49] fails to detect the looming object while the object is both looming and moving sideways at the same time.

However, our algorithms have some limitations which are due to the way the DVS generates the events. First of all, due to the limited resolution of the DVS, extreme expansion of the objects or having many objects in the scene may affect the accuracy of clustering algorithms. Also, since the DVS generates events only for the leading and trailing edges, the edges and other parts of the object may be missing. In this situation, since our merge algorithm looks for different labelled events in the neighbourhood of each event, it may fail

to merge the clusters. In addition, the lack of events causes the algorithms to detect an incorrect number of clusters especially when the objects are moving sideways (Figure 4.21). This situation is difficult even for human to detect without any prior assumptions on the shape and number of clusters. Last but not least, although we pre-process data to remove noise, depending on the amount of noise, the results might still be affected by noise.

Chapter 5

Conclusion

In this thesis we have proposed a real-time, automatic and accurate algorithm capable of clustering events generated by event-based cameras. This algorithm is based on the sequential K-means algorithm which we adopted that for using with asynchronous transmitted event based input data. Our algorithm is real-time and can assign events to the clusters as they arrive. Also, we heuristically proposed an algorithm called the elbow method (Algorithm 4) which can detect the number of clusters without any prior assumptions on the number or shape of the objects. In addition, we designed an algorithm called the merge algorithm (Algorithm 5) which is capable of merging incorrect detected multiple clusters to form a unique one. We used the clustering output with Ridwans looming detection algorithm [49] on each cluster to solve the problem of multiple object looming detection.

We compared our method with two other well-known clustering algorithms (the mean shift and the K-means algorithms) in their accuracy, processing time and looming detection quality on various test scenarios for both simulated and captured data sets. Through these experiments we showed that the combination of using our algorithms (the sequential K-means, elbow method and merge algorithm) provides us the most accurate results in lowest processing time. We also showed, in contrast with the conventional K-means and mean shift algorithms which collect data to compute the clusters' centroid or representative, our sequential K-means algorithm looks for the closest cluster as each event arrives and does a simple update recompute the centroids of the clusters. Therefore, its complexity to process each event is still the same no matter the number of events in pseudo-frames is small or

large.

The looming detection algorithm requires collecting data for the processing of the event directions. We collect events in pseudo-frames with the length of about 16 milliseconds. We showed that the only clustering algorithm which is able to process cluster detection within this time in all test cases is the sequential K-means algorithm. The processing time of this algorithm is less than 0.8 milliseconds. The conventional K-means algorithm is too slow and could not process cluster detection within the length of pseudo-frame in any experiment cases. As for the mean shift algorithm, the processing time was from about 5 milliseconds to about double of the length of pseudo-frame depending on the number of events in each pseudo-frame. Also, during our experiments, we realized that the Ridwan's looming detection algorithm [49] fails to detect the looming object while the object is both looming and moving sideways in the scene.

Last but not least, our algorithm does not require the adjustment of any parameters and it does not require any special hardware or parallel processors to process data efficiently.

5.1 Limitation

The limited resolution of the DVS affects the accuracy of our algorithm when there are many objects in the scene. This can also happen due to the extreme expansion of the object. In addition to limited resolution, due to the way the DVS generates events only for moving edges, our algorithm detects the leading and trailing edge as different clusters. This however may be detected the same way as our algorithm by human.

As our merge algorithm looks for touching clusters, lack of events causes it to fail merging multi clusters to a single one. Also, depending the amount of noise, our algorithm results may be affected in extreme noisy conditions.

5.2 Future Work

Although our algorithms have delivered promising results, despite of having some limitations due to the way the DVS generates the events, there are many directions for future studies. We list some of them below.

Training the algorithm: By training our algorithm to be able to detect different objects in various situation using machine learning and deep learning algorithms we may be able to obtain better results. We may be able to do this by providing various data sets including captured event streams from different objects in different types of movement. By modifying the training algorithms to receive event streams per pseudo-frames as input, we may be able to train the algorithm for detecting different objects and types of movement. In this situation, in addition of clustering the objects, the algorithm may be able to detect their shape or type of the movement. Therefore, even if some events are missing, our algorithm may still consider it as single object rather than multiple objects.

Applying different algorithms: Since our proposed clustering algorithm is real-time and accurate, we can apply other algorithms to each detected clusters and pursue different purposes such as algorithms with ability to track the movement of each detected clusters, algorithms to detect the shape of the objects or even more accurate looming detection algorithms.

Computing velocity: Computing the velocity of each event may help us cluster the objects even more accurately by using the velocity of events as another input parameter to our clustering algorithm.

Clustering polarity events: We can modify our algorithm to cluster the polarity events instead of optical flow events for using it with different algorithms. This can be used in different applications such as tracking the movement of each object using the polarity events instead of the optical flow events.

Distinguishing between stationary and moving objects: By placing the camera on a moving platform, it may be desired to distinguish between moving and stationary objects.

The DVS also generates other information gathered from other sensors such as accelerometer. Using these information may be used to cluster objects according to their position compared to camera. In this case, by adopting our algorithm we may be able to cluster stationary and moving objects in different groups based on whether the camera is static or moving.

Bibliography

[1] K. Aires, A. Santana, and A. Medeiros. Optical flow using color information: preliminary results. In *23º Annual ACM Symposium on Applied Computing*, 2008.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] P. Bardow, A. Davison, and S. Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016.

[4] F. Barranco, C. Fermüller, and E. Ros. Real-time clustering and multi-target tracking using event-based sensors. *CoRR*, abs/1807.02851, 2018.

[5] S. Beauchemin and J. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[6] K. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):416–434, 2000.

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] User guide: caer. `https://inivation.com/support/software/caer/`. Accessed: 2019-04-25.

[9] M. Celebi, H. Kingravi, and P. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013.

[10] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[11] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[12] S. Cremer, L. Mastromoro, and D. O. Popa. On the performance of the baxter research robot. In *2016 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, pages 106–111, Aug 2016.

[13] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, page 31. ACM, 2008.

[14] F. Dramas, S. Thorpe, and C. Jouffrais. Artificial vision for the blind: a bio-inspired algorithm for objects and obstacles detection. *International Journal of Image and Graphics*, 10(04):531–544, 2010.

[15] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

[16] W. Enkelmann. Obstacle detection by evaluation of optical flow fields from image sequences. *Image and Vision Computing*, 9(3):160–168, 1991.

[17] J. Franco, J. del Valle Padilla, and S. Cisneros. Event-based image processing using a neuromorphic vision sensor. In *2013 IEEE International Autumn Meeting on Power Electronics and Computing (ROPEC)*, pages 1–6. IEEE, 2013.

[18] T. Fülöp and A. Zarándy. 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*, pages 1–5. IEEE, 2010.

[19] T. Fülöp and A. Zarándy. Bio-inspired looming direction detection method. In *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, pages 1–6. IEEE, 2012.

[20] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*, volume 20. Siam, 2007.

[21] J. Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[22] P. Gil-Jiménez, H. Gómez-Moreno, R. López-Sastre, and A. Bermejillo-Martín-Romo. Estimating the focus of expansion in a video sequence using the trajectories of interest points. *Image and Vision Computing*, 50:14–26, 2016.

[23] T. Gollisch and M. Meister. Eye smarter than scientists believed: neural computations in circuits of the retina. *Neuron*, 65(2):150–164, 2010.

[24] R. Gonzalez and R. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[25] C. Goutte, P. Toft, E. Rostrup, F. Nielsen, and L. Hansen. On clustering fmri time series. *NeuroImage*, 9(3):298 – 310, 1999.

[26] J. Hartigan and M. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[27] L. He, T. Chia, and C. Yang. A geometric invariant approach to human face verification. *Journal of information science and engineering*, 22(3):511, 2006.

[28] B. Horn and B. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[29] User guide: jaer. `https://inivation.com/support/software/jaer/`. Accessed: 2019-04-25.

[30] S. Jayasuriya, O. Gallo, J. Gu, T. Aila, and J. Kautz. Reconstructing intensity images from binary spatial gradient cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–26, 2017.

[31] D. Ketchen and C. Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[32] H. Kim, S. Leutenegger, and A. Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[33] T. Kodinariya and P. Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[34] X. Li, K. Wang, W. Wang, and Y. Li. A multiple object tracking method using kalman filter. In *The 2010 IEEE International Conference on Information and Automation*, pages 1862–1866, June 2010.

[35] User guide: libcaer. `https://inivation.com/support/software/libcaer/`. Accessed: 2019-04-25.

[36] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120 *db* 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[37] A. Linares-Barranco, F. Gmez-Rodrguez, V. Villanueva, L. Longinotti, and T. Delbrck. A usb3.0 fpga event-based filtering and tracking framework for dynamic vision sensors. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2417–2420, May 2015.

[38] Z. Lu, L. Lesmes, and G. Sperling. The mechanism of isoluminant chromatic motion perception. *Proceedings of the National Academy of Sciences*, 96(14):8289–8294, 1999.

[39] A. K. Maan, D. A. Jayadevi, and A. P. James. A survey of memristive threshold logic circuits. *IEEE Transactions on Neural Networks and Learning Systems*, 28(8):1734–1746, Aug 2017.

[40] M. Mahowald. *VLSI analogs of neuronal visual processing: a synthesis of form and function*. PhD thesis, California Institute of Technology, 1992.

[41] C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[42] T. Münch, R. Da Silveira, S. Siegert, T. Viney, G. Awatramani, and B. Roska. Approach sensitivity in the retina processed by a multifunctional neural circuit. *Nature neuroscience*, 12(10):1308, 2009.

[43] M. O'shea, C. Rowell, and J. Williams. The anatomy of a locust visual interneurone; the descending contralateral movement detector. *Journal of Experimental Biology*, 60(1):1–12, 1974.

[44] C. Pantilie and S. Nedevschi. Real-time obstacle detection in complex scenarios using dense stereo vision and optical flow. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 439–444. IEEE, 2010.

[45] S. S. Park and A. Sowmya. Autonomous robot navigation by active visual motion analysis and understanding. *Proceedings of IAPR Workshop on Machine Vision Applications*, 1998.

[46] D. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[47] D Regan and K. Beverley. Looming detectors in the human visual pathway. *Vision research*, 18(4):415–421, 1978.

[48] W. Reichardt and M. Egelhaaf. Properties of individual movement detectors as derived from behavioural experiments on the visual system of the fly. *Biological Cybernetics*, 58(5):287–294, 1988.

[49] I. Ridwan. Looming object detection with event-based cameras. Master's thesis, Lethbridge, Alta.: University of Lethbridge, Dept. of Mathematics and Computer Science, 2017.

[50] I. Ridwan and H. Cheng. An event-based optical flow algorithm for dynamic vision sensors. In *International Conference Image Analysis and Recognition*, pages 182–189. Springer, 2017.

[51] F. Rind and P. Simmons. Seeing what is coming: building collision-sensitive neurones. *Trends in neurosciences*, 22(5):215–220, 1999.

[52] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, June 2011.

[53] M. Subbarao. 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.

[54] R. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, Dec 1953.

[55] Y. Wang and B. Frost. Time to collision is signalled by neurons in the nucleus rotundus of pigeons. *Nature*, 356(6366):236, 1992.

[56] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *European conference on computer vision*, pages 211–224. Springer, 2006.

[57] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.