Temporal Pattern-Based Active Marker Identification and Tracking Using a Dynamic Vision Sensor

Matthias Hofstetter

Master Thesis
January 2012

Prof. Dr. Markus Gross
Prof. Dr. Tobias Delbruck
Dr. Christof Bühler

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Graphics Laboratory ETH Zurich

Abstract

Marker tracking is widely used in applications like motion capture and pose estimation. This thesis describes the development of an active marker tracking algorithm using a Dynamic Vision Sensor (DVS) as recording device. The DVS's event driven behaviour opens novel opportunities to develop a tracking algorithm with a high temporal resolution at low costs. The tracking algorithm combines both the spatial and temporal information that is induced by the DVS in response to changes in illumination.

To ensure a continuous change in the illumination, blinking light emitting diodes (LED) are used as active markers. This changing illumination causes a continuous stream of events induced by the blinking of the LED. The thesis formulates a mathematical model to track these LEDs using the DVS's key features: Its high temporal resolution and temporal accuracy. Especially accounting for the temporal information the developed method can handle external influences like camera movements, vibrations, background activity and noise. The thesis also explores the relation between the LED's blinking pattern and the LED's velocity to. It shows that a high LED velocity requires a high LED blinking frequency to accurately measure the LED's trajectory.

The thesis describes distinct features to uniquely identify a blinking LED and compares their advantages and disadvantages. Using the LED's features the tracking algorithm is able to handle temporary object disappearance, occlusion and trajectory crossing. Aiming at demonstrating the basic capability the developed algorithm is able to identify up to 5 LEDs with blinking patterns between 100Hz and 1kHz.

Zusammenfassung

Die Verfolgung von Markern ist weit verbreitet in Applikationen zur Erfassung von Bewegungsabläufen und der Abschätzung von Posen. Diese Arbeit beschreibt die Entwicklung eines Algorithmus zur Verfolgung von aktiven Markern. Als Aufnahmegerät wird ein Dynamic Vision Sensor (DVS) verwendet. Der DVS ist ein ereignisbasierendes Aufnahmegerät, welcher neue Möglichkeiten für die Entwicklung eines Algorithmus mit hoher zeitlicher Auflösung und geringem rechnerischem Aufwand eröffnet. Der Algorithmus zur Verfolgung der Marker kombiniert sowohl die räumliche, als auch die zeitliche Information, welche von der DVS bereitgestellt wird, als Antwort auf eine Veränderung der Beleuchtung.

Um eine kontinuierliche Veränderung der Beleuchtung zu gewährleisten, werden blinkende Leuchtdioden (LED) als aktive Marker verwendet. Das Blinken der LED bewirkt eine kontinuierliche Generierung von Ereignissen durch den DVS. Die Arbeit beschreibt ein mathematisches Modell, um die blinkenden LEDs auf Grund dieser Ereignisse zu Verfolgen. Dabei werden die herausragenden Eigenschaften des DVS verwendet: Dessen hohe zeitliche Auflösung und zeitliche Präzision. Unter Berücksichtigung der zeitlichen Information ist der entwickelte Algorithmus in der Lage externe Einflüsse zu behandeln. Darunter fallen Bewegungen der Kamera, Vibrationen, Aktivitäten im Hintergrund und Rauschen. Die Arbeit untersucht weiter die Beziehung zwischen dem zeitlichen Muster, mit welchem die LED blinkt, und deren Geschwindigkeit. Es wird aufgezeigt, dass die Verfolgung einer LED mit hoher Geschwindigkeit möglich ist, sofern die Frequenz, mit welcher die LED blinkt, angepasst wird.

Die Arbeit beschreibt Eigenschaften um eine blinkende LED zu identifizieren und vergleicht diese auf Grund ihrer Vor- und Nachteile. Mit Hilfe dieser Eigenschaften ist der Algorithmus in der Lage ein temporäres Verschwinden der LED zu behandeln, sowie die räumliche Überlappung zweier LEDs. Die Arbeit umfasst die Entwicklung einer Demonstration um die Fähigkeit des Algorithmus aufzuzeigen, bis zu 5 blinkende LEDs zu erfassen. Das Blinken der LED wird dabei mit Frequenzen zwischen 100Hz und 1kHz betrieben.

Contents

List of Figures

List of Tables

1

Introduction

Conventional image processing methods rely on operating on the entire image in each frame, touching each pixel many times. As a result, this approach leads to high costs of computation and memory communication bandwidth, especially for high frame rate applications. In addition, the frame rate limits both, response latency and temporal resolution even for high-speed cameras and therefore greatly complicates registration and tracking of fast moving objects.

The Dynamic Vision Sensor (DVS) is designed to overcome this problem. The output of the DVS consists of asynchronous address-events that signal scene reflectance changes at the times they occur. This means essentially that an event occurs whenever the illumination of a particular pixel has changed. Every event consists of the coordinate of the pixel, the sign of change in illumination and the timestamp of the event. Instead of transmitting the frame, the DVS transmits only the events of changing pixels. This asynchronous response property means that the events have a very short latency and the timing precision of the pixel response is not quantized to the traditional frame rate, but by the pixels response dead time.

Today high-speed cameras like the Phantom FLEX developed by Vision Research achieve 2.570 frames per second with a resolution of 1.920 x 1.080 pixels [Sci]. Moreover the CamRecord CV allows image recording up to 10’000 frames per second and offers almost single photon detection capability with a resolution of 1280 x 1024 pixels [Opt]. A third example is the i-Speed FS camera by Olympus. This camera has a frame rate of up to 1’000’000 with a resolution of 1280 x 1024 [Oly]. Clearly the event-driven sensors have the potential to out-perform high speed cameras not only in terms of speed but also in terms of information redundancy. The advantages of the DVS compared with other high-speed cameras are the following:

1. Reduced costs
2. Smaller data volumes

3. New paradigms for image processing algorithms

While high-speed cameras like the Phantom FLEX cost between \$50'000 and \$150'000 engineering samples of the DVS has costs around \$2500. A key advantage of the DVS is the small data volume generated. In contrast to classic cameras transmitting the information of each pixel, the DVS only transmits the events of the changed pixels. This allows the development of new algorithms to analyse the information provided by DVS and use of standard USB interfaces. Clearly, event-driven sensors open new opportunities to gain both, novel image analysis approaches and performance improvement.

1.1. Dynamic Vision Sensor

Conventional vision sensors see the world as a series of frames. Successive frames contain enormous amounts of redundant information, wasting memory access, RAM, disk space, energy, computational power and time. In addition, each frame imposes the same exposure time on every pixel, making it difficult to deal with scenes containing very dark and very bright regions [Del12a].

The Dynamic Vision Sensor (DVS) solves these problems by using patented technology that works like your own retina. Instead of wastefully sending entire images at fixed frame rates, only the local pixel-level changes caused by movement in a scene are transmitted - at the time they occur. The result is a stream of events at microsecond time resolution, equivalent to or better than conventional high-speed vision sensors running at thousands of frames per second. Power, data storage and computational requirements are also drastically reduced, and sensor dynamic range is increased by orders of magnitude due to the local processing. The sensor is shown in figure 1.1.

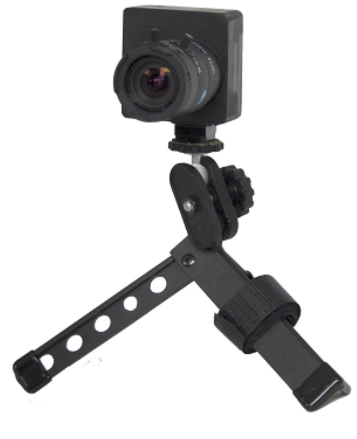


Figure 1.1.: *The image shows the DVS128.*

The DVS functionality is achieved by having pixels that respond with precisely-timed events to temporal contrast. Movement of the scene or of an object with constant reflectance and illumination causes relative intensity change; thus the pixels are intrinsically invariant to scene illumination and directly encode scene reflectance change. This is illustrated in figure 1.2

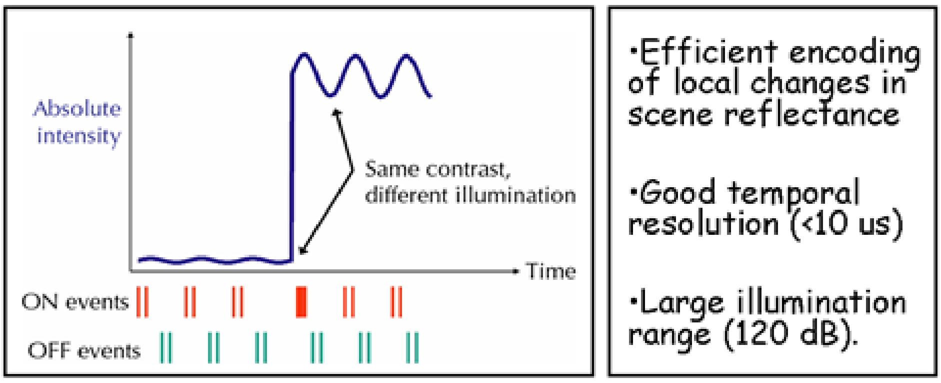


Figure 1.2.: *The figure shows the generation of on- and off-events depending on the relative intensity changes.*

The pixel uses a continuous-time front end photoreceptor,(inspired from the adaptive photoreceptor), followed by a precision self-timed switched-capacitor differentiator (inspired by the column amplifier used in the pulsed bipolar imager). The most novel aspects of this pixel are the idea of self-timing the switch-cap differentiation and self-biasing the photoreceptor. This pixel does a data-driven AD conversion (like biology, but very different than the usual ADC architecture). Local capacitor ratio matching gives the differencing circuit a precisely defined gain for changes in log intensity, thus reducing the effective imprecision of the comparators that detect positive and negative changes in log intensity. An overview of the DVS's architecture is shown in figure 1.3.

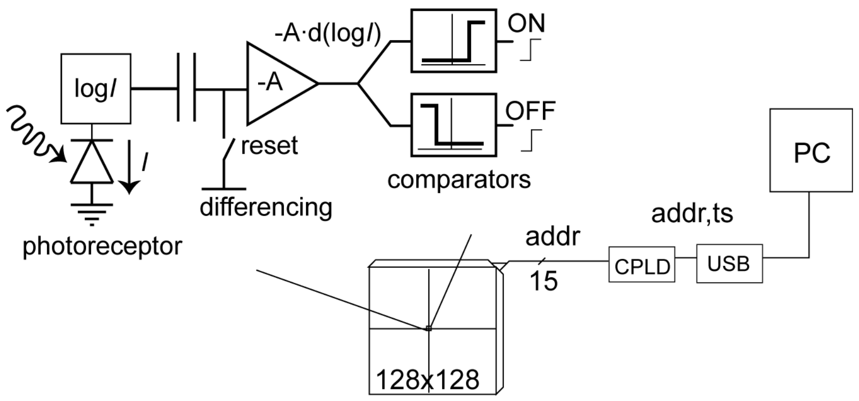


Figure 1.3.: *The figures shows an overview of the DVS's architecture.*

1.2. Motivation

The goal of this project is to distinguish and track specific spatio-temporal patterns produced by blinking light-emitting diodes (LED). These patterns could be different in terms of both frequencies and binary sequence similar to Morse codes. Since the frequencies of these patterns could be easily higher than the frame rate of a frame-based camera this is an ideal application for the DVS. To distinguish and track these patterns, the algorithm has to fulfil distinct criterions listed in section 1.3. Furthermore, the goal of the project is to develop an algorithm which combines the spatial and temporal information of events generated by the DVS.

Possible applications for the DVS are given in situations with fast moving objects like in traffic control, speed estimation and particle tracking. Combined with the principle of tracking blinking LEDs described above, a possible application could be in medicine. For example, to analyse the tremor of a patient, which is usually between 4 and 15Hz, requires a camera with a high temporal resolution to capture the motion in all its details. Using the above described LEDs as markers on the patient and the DVS to capture the scene with a high temporal resolution the algorithm should be able to capture the motion very precisely at low costs. The developed marker tracking algorithm is therefore suitable in problem situations which require a high temporal resolution of the tracked marker trajectories.

In order to test the performance, correctness and robustness of the developed algorithms the project includes the creation of a demonstration application. This application supports the user in easily creating and modifying test cases for an arbitrary number of blinking LEDs. With the help of these test cases the developed algorithms are evaluated regarding their performance, accuracy and robustness.

1.3. Goals for the Tracking and Classification Algorithm

The LED's blinking pattern has to be identified and classified by the identification algorithm. Identification involves the extraction of features representing unique properties of the blinking LEDs. The classification task on the other hand has to recognize whether a LED was already seen before based on the features obtained by the identification.

1. The algorithm has to cope with multiple LEDs (e.g. more than 3).
2. The identification has to be performed in real time.
3. Additionally the algorithm has to learn unknown patterns on its own.

The tracking algorithm has to track the identified blinking LEDs to record their path.

1. The tracking algorithm has to track multiple LEDs in space (e.g. more than 3).
2. The tracking has to be performed in real time.
3. The tracking algorithm has to be robust in a way that it can cope with temporary object disappearance, object trajectory crossing, camera movements / vibrations as well as different external light influences.

2 Related Work

This chapter provides an overview at the current state of research in marker tracking. Marker tracking has been exploited for many decades. Current marker tracking algorithms are well established and are able to cope with special situations like temporary object disappearance. There are plenty of specialized marker tracking implementations for various applications and environments.

Marker tracking with blinking pattern is not yet done. This chapter presents distinct papers containing interesting aspects and ideas considered in this master thesis. It also includes papers using the dynamic vision sensor (DVS) as recording device to solve a specific problem. Recall that blinking LEDs are especially suited for both the event-driven DVS camera and identification of multiple objects.

[DxYx10] The paper aims at the problem of tracking moving object under complex background. A new method is proposed, which is based on machine vision. This method takes advantage of image analysis technology to detect the profile of specified moving object and compute its center. According to the center of moving object in the several previous frame image, the proposed method predicts the possible center in the next frame. By taking this predicted center as the input the method computes the real center in the next frame.

[FP98] The system proposed in this project consists of three stages. In the first stage, all moving objects are detected using a temporal differencing algorithm. These are described as motion regions. The tracking process involves correlation matching between a template and the current motion regions obtained by temporal differencing. The motion region with the best correlation is tracked and is used to update the template for subsequent tracking.

[HX10] The paper presents a two-stage object tracking method by combining a region-based method and a contour-based method. First, a kernel-based method is adopted to locate the object region. A Kalman-Filter is used to predict the objects location while a mean-shift algorithm will find the location that maximizes a similarity measurement based between the target and candidate model. After the target localization with the kernel-based method, the authors adopt the diffusion snake to evolve the object contour in the object feature space, in order to improve the tracking precision.

[HH11] This paper presents a set of real-time tracking system. The tracking system uses a combination of Camshift and Kalman filter algorithm. If the target appears as occluded, the system tracks the object by using the speed detection and prediction of a Kalman filter. Otherwise, the system tracks the object by using Camshift.

[LS08] Marker tracking is widely used in Augmented Reality. Markers containing digital barcode patterns can not only be used for pose tracking, but also to uniquely distinguish thousands of objects. In the paper the authors describe three new marker designs that occupy significantly less space and therefore reduce the amount of visual pollution in the augmented area. The paper describes techniques to handle cases where the marker is partially occluded or out of sight.

[ST94] The paper discusses the definition of a good feature to track. A feature with a high texture content can still be a bad feature to track. It further describes how the feature has to be adapted over time to track the features dissimilarity over time. Because of the potentially large number of frames through which a given feature can be tracked, the dissimilarity would increase by using a too simple motion model to adapt the feature over time.

[Del08] Conventional image sensors produce massive amounts of redundant data and are limited in temporal resolution by the frame rate. This paper reviews our recent breakthrough in the development of a high-performance spike-event based DVS that discards the frame concept entirely, and then describes novel digital methods for efficient low-level filtering and feature extraction and high-level object tracking that are based on the DVS spike events.

[MY11] This paper proposes a method to recognize bare hand gestures using a DVS camera. The paper attempts to classify three different hand gestures made by a player during rock-paper-scissors game.

[CD09] Balancing small objects such as a normal pencil on its tip requires rapid feedback control with latencies on the order of milliseconds. The paper describes how a pair of DVS's is used to provide fast visual feedback for controlling an actuated table to balance an ordinary pencil on its tip. The estimate of the pencil is maintained as a Gaussian in the Hough space of possible lines.

[DL07] Fast sensory-motor processing is challenging when using traditional frame-based cameras and computers. The system consists of a 128x128 retina that asynchronously reports scene reflectance changes, a laptop PC, and a servo motor controller. The setup can be used to implement a fast sensory-motor reactive controller to track and block balls shot at a goal. Moving objects are tracked by an event-driven cluster tracker algorithm that detects the ball as the nearest object that is approaching the goal.

3

Temporal Pattern-Based Active Marker Identification and Tracking

This chapter describes the central work of the master thesis. The basic concepts and the key methods for temporal pattern-based active marker identification and tracking are discussed.

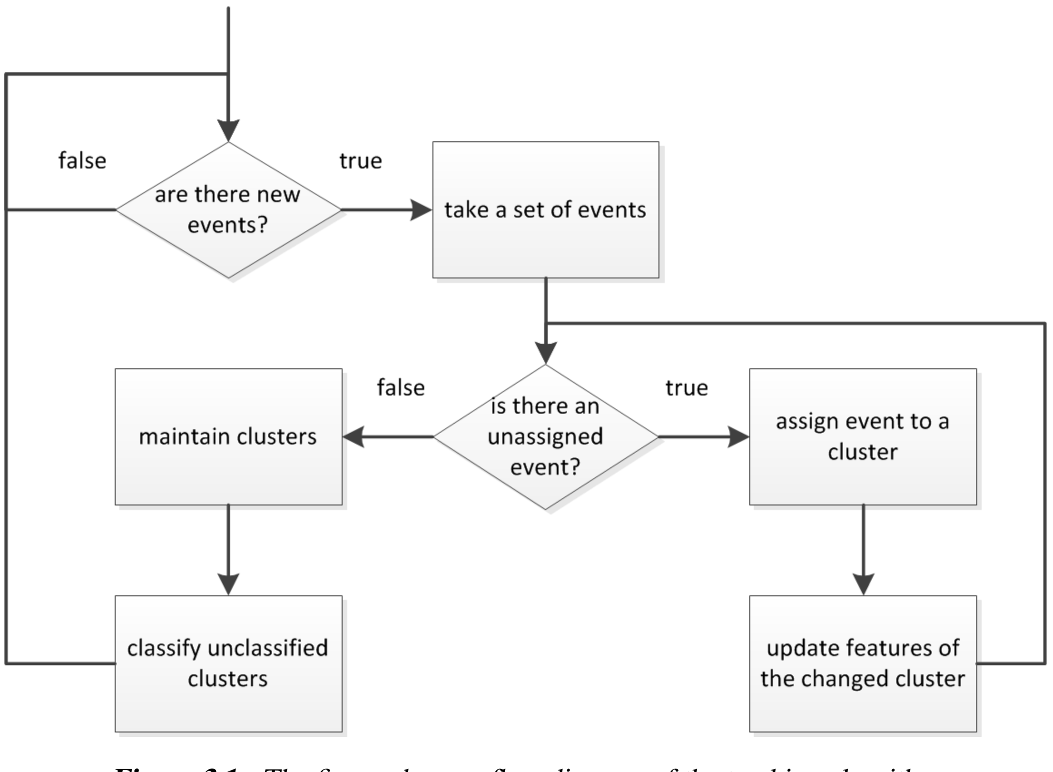

false
are there new events?
true
take a set of events
false
is there an unassigned event?
true
maintain clusters
assign event to a cluster
classify unclassified clusters
update features of the changed cluster

Figure 3.1.: *The figure shows a flow diagram of the tracking algorithm.*

The tracking algorithms general idea is shown in figure 3.1. Because of the DVS's event driven behaviour the algorithm only has an input if there are new events. The algorithm will process a whole set of events if there is one available. First the algorithm clusters events generated by the same blinking LED. For each event the algorithm assigns the event to the cluster representing the events blinking LED. Whenever the algorithm adds an event to a cluster the cluster will immediately recompute its features like its position or its size.

After all available events are processed, the tracking algorithm maintains the set of clusters. It removes and deletes clusters that are no longer used. The remaining clusters are classified according to the cluster's computed features by the algorithm. For each unclassified cluster the algorithm searches for the best possible classification. If there is now satisfying classification the algorithm will register the clusters blinking pattern as a new one.

The work is divided into a set of work packages. The ordering of these packages illustrates the project's development process. The list of packages and their description is shown in the following enumeration.

1. **Behaviour Analysis of the DVS:** This section gives a view on how the DVS perceives the blinking pattern of the LED. The results of different experiments are summarized to show the limits of the DVS but also its potentials.
2. **Event Filtering:** The filter reduces the number of events that have to be processed by the tracking algorithm. The section describes a straightforward and efficient filter to filter out events not generated by a blinking LED.
3. **Event Clustering:** Describes the approach used to assign an event to a cluster. The goal is to group events produced by the same blinking LED.
4. **Basic Features of a Cluster:** Using the set of events assigned to a cluster the algorithm is able to compute several features like the clusters spatial position and its extension. These features can then be used to serve as a criterion for the above described event clustering.
5. **Features Used to Identify a Blinking LED:** In this section the methods used to extract the LED's blinking pattern are presented. It shows different approaches and different metrics describing the pattern and discusses their advantages and disadvantages.
6. **Classification of a Blinking LED:** Once the blinking pattern of a LED has been identified the cluster has to decide whether the LED is already known. The classification algorithm has a list of known blinking pattern. The goal of this section is to find the best matching blinking pattern for an observed LED. If there is no assignment possible, the LED's blinking pattern has to be registered as a new one.
7. **Experiments for the Classification of a LED:** Describes the different experiments to analyse the quality and performance of the algorithm to identify the LED's blinking pattern. The results of the experiments are presented and discussed in this section as well.
8. **Tracker Combining Spatio-Temporal Information:** The section presents the final algorithm used to track blinking LEDs. It uses the different approaches presented in the thesis to combine the events spatial and temporal information.

Parameter	*Quantity*
Camera	iniLabs ETH DVS128
Objective	Computar Vari Focal H3Z4512CS-IR
LED Frequency [Hz]	100-1000
Blinking Pattern	Square wave
Duty cycle [%]	50

Table 3.1.: *General setup for all experiments.*

3.1. Experimental Setups

The purpose of this section is to present the different experimental setups used for the experiments in the thesis. There are 3 different setups which are described with their default values. In the concrete experiments we will refer to one of the 3 experiments and explain the changed parameters in detail.

The descriptions of the DVS camera and its devices used in all experiments are listed in table 3.1. The table also shows the default blinking pattern used in the experiments and its additional parameters.

3.1.1. Stationary Blinking LED

This experimental setup uses a stationary LED as illustrated in figure 3.2. The blinking pattern of the LED is controlled by a function generator which is able to generate arbitrary blinking patterns. The DVS is placed in front of the LED at various distances and connected with a computer where the algorithm is running.

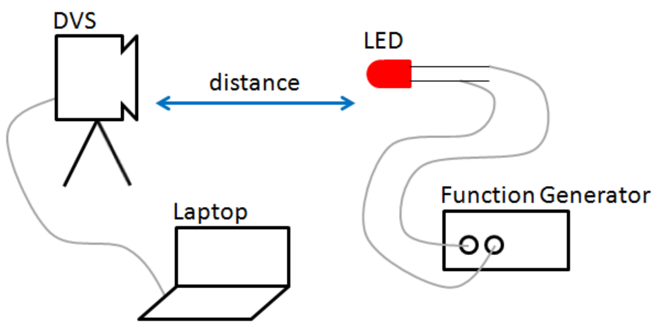


Figure 3.2.: *Setup for an experiment with a stationary blinking LED.*

The default parameters for the experiments are illustrated in table 3.2.

Parameter	*Quantity*
Distance [cm]	50-300
Velocity [pixel / sec]	0
LED	Standard red capsule LED width 5mm diameter
Frequency Generator	Hawlett Packard 33120A
Power [V]	off=0 / on=3

Table 3.2.: *Default values for the experiment with a stationary blinking LED.*

Parameter	*Quantity*
Velocity [pixel / sec]	0 - 300
Servo Motor	Hitec HSR-5995TG

Table 3.3.: *Default values for the experiment with a moving blinking LED mounted on a robot-arm.*

3.1.2. Back and Forth Moving LED

This setup is very similar to the one described in section 3.1.1. The difference is that the LED is mounted on a robot-arm. With the help of a servo motor the arm moves continuously from right to the left and vice-versa. The servo motor is controlled by the computer while the LED is controlled by a function generator. The setup is shown in figure 3.3.

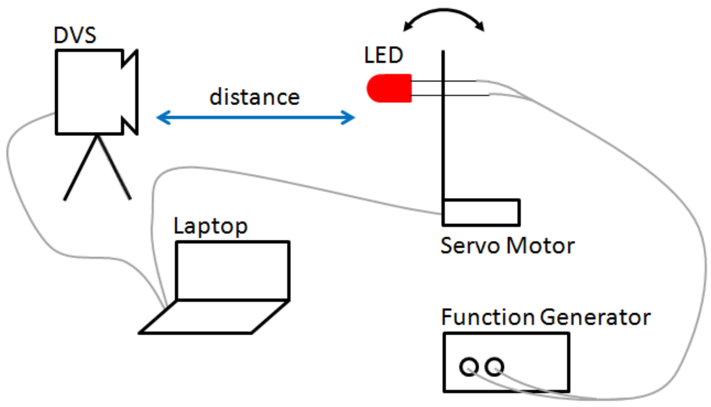


Figure 3.3.: *Setup for an experiment with a moving blinking LED mounted on a robot-arm.*

Most of the experiments default values are the same as shown in table 3.2. The additional and changed parameters for this experiment are listed in table 3.3.

Parameter	*Quantity*
Velocity [pixel / sec]	0 - 300
Servo Motor	Maxon DC Motor M96908
Power Supply	Bench Power Supply - Farnell Tops 3D
LED	LED_RGB_THT 754-1492-ND
Micro Controller	ATMEL AVR32 AT32UC3B
Power [V]	off=0 / on=3

Table 3.4.: *Default values for the experiment with a moving blinking LED mounted on a cylinder.*

3.1.3. Rotating LED

This experimental setup allows to track the blinking LED with a controlled velocity on a circular path as represented in figure 3.4. The LED is controlled by a micro controller which is able to generate any kind of blinking pattern. A battery holder is used as power supply for the LED and the micro controller. The unit consisting of the LED, the micro controller and the battery holder is mounted on a cylinder which rotates around its axis driven by a servo motor. The servo motors velocity is controlled by an external power supply.

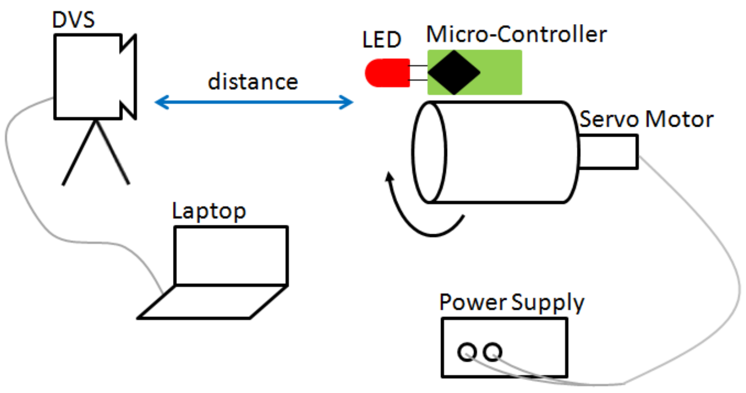
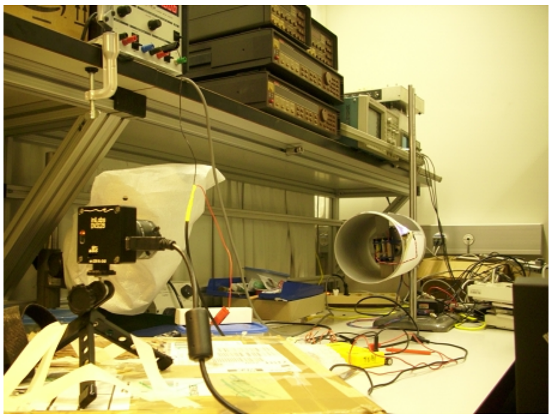


Figure 3.4.: *Setup for an experiment with a moving blinking LED mounted on a cylinder.*

The changed parameters as well as the new ones are shown in table 4.1 while the rest of the parameters are adopted from the tables 3.2 and 3.3.

The LED is part of a 5x1cm PCB used in the Electronics for Physicists: Digital course. It has an Amtel AVR32 micro controller. The Atmel USB chip is a high-performance Atmel 32-bit CPU optimized for small code size. It is supported by the GCC compiler, so one can use the entire open-source GNU toolchain with Eclipse IDE to program it [Del11]. To create blinking patterns for the LED a C program was adapted for the micro controller in the thesis.

This experimental setup in practice is shown in figure 3.5. The figure 3.5(a) shows the rotating cylinder with the mounted LED and micro controller on the right side. On the left side the DVS sensor is visible in front of the cylinder. Figure 3.5(b) illustrates the setup during the experiment. Here the cylinder is rotating around its axis. The cylinder's rotation causes a blurring of the blinking LED because of its high velocity.


(a) The figure shows the cylinder with the attached AVR32 micro controller and the LED. The DVS is visible in front of the cylinder.

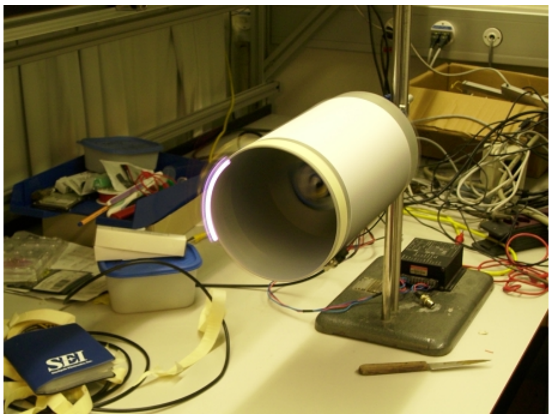

(b) Shows the setup with a blinking LED attached on the rotating cylinder.

Figure 3.5.: *The figures are showing the experimental setup in practice.*

Parameter	*Quantity*
Distance [cm]	50
Frequency [Hz]	100

Table 3.5.: *Setup for the experiment to analyse the density of events produced by a stationary blinking LED.*

3.2. Behaviour Analysis of the DVS

In order to develop an algorithm to identify and track blinking LEDs using a DVS camera, we first analyse the DVS's fundamental characteristics by recording and analysing the data stream from a single blinking LED. This gives views on how the DVS perceives the temporal patterns produced by blinking LEDs. The derived results and insights serve as a basis to develop the mathematical model to identify and classify the LEDs according to their pattern.

3.2.1. Spatial Distribution of Events

To identify temporal patterns by means of a conventional frame-based camera, the pattern recognition algorithm has to find the region where changes occur. In contrast to that the DVS intrinsically generates events containing the coordinates of changing pixels. Using this information an algorithm is able to directly identify the changed regions. This direct identification of changing regions is one of the key advantages of the event-driven behaviour of the DVS.

In the experiments described below the blinking LED is operated with a frequency between 10Hz to 1kHz. Therefore, the DVS will produce around 10 to 1000 events per second for each pixel observing the blinking LED. The first question to answer would therefore be whether it is possible to estimate the position of a blinking LED based on the number of generated events of a particular pixel over time. This definition corresponds to a density of events where the density is computed pixel wise as the number of events per time unit.

Stationary LED

A first analysis of the density of events over time is presented in this section where a stationary blinking LED is observed by the DVS. The setup of the experiment corresponds to the one presented in section 3.1.1. The changed parameters for the experiment are shown in table 3.5.

To visualize the number of events generated by each pixel x, y a density $density(x, y)$ was computed as shown in formula 3.1. Here the variable $events$ corresponds to the set of all created events where each event e has a location $e.x, e.y$. The symbol $\mathbb{I}$ represents the indicator function and $\triangle t$ is the duration of the observation.

$$density(x, y) = \frac{\sum_{e \in events} \mathbb{I}_{e.x=x \, \wedge \, e.y=y}}{\triangle t} \tag{3.1}$$

Metric	*Results*
Signal Average Density [events / sec]	588.69 ± 13.51
Background Average Density [events / sec]	0.5 ± 0.07
Signal Size [pixels]	101

Table 3.6.: *Shows the quantitative results of a blinking stationary LED.*

For the experiment we expect that the DVS produces significant more events for those pixels observing the LED's blinking pattern. These pixels have therefore a high density of events and can easily be identified.

Figure 3.6 shows the density of events per pixel. It is evident that some of the pixels have a striking higher density than others. These pixels are spatially close together.

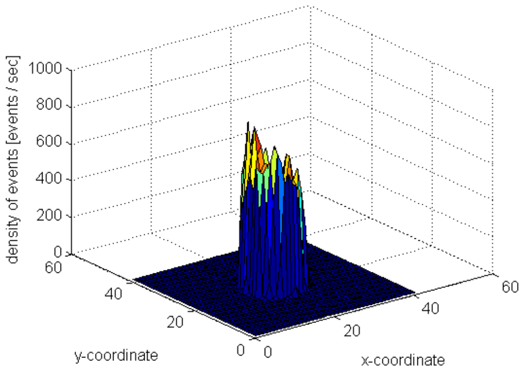


Figure 3.6.: *Shows the density of events per pixel. The x and y axis corresponds to the pixel locations while the z axis represents the number of events per time unit.*

Table 3.6 shows the quantitative results of the experiment. To determine the LED's size on the DVS in pixels we take the half of the highest density found as a threshold. The pixels having a density higher than this threshold are considered as pixels observing the blinking LED. This is shown in formula 3.2 where an indicator function $\mathbb{I}$ is used to determine the LEDs size.

$$size = \sum_{x} \sum_{y} \mathbb{I}_{\{density(x,y) > \max_{dx,dy}(density(dx,dy))/2\}} \tag{3.2}$$

The quantitative results in table 3.6 are showing the same behaviour as the figure. The density of events is significant higher for those pixels observing the blinking LED than for the background pixels.

As expected there are pixels producing significantly more events than others. According to the distribution of these pixels it is evident that these pixels have observed the blinking LED. The position of the LED can be clearly identified in the density plot - irrespective of the noise-induced spikes. Furthermore the plot allows the identification of each pixel observing the blinking LED since the LED-induced density of events is higher than for the background pixels.

Parameter	*Quantity*
Distance [cm]	50
Velocity [pixel / sec]	10
Frequency [Hz]	100, 300

Table 3.7.: *Setup for the experiment to analyse the density of events produced by a moving blinking LED.*

In conclusion the detection of a stationary blinking LED is straightforward provided that the LED-induced event rate exceeds that of the background noise of the DVS.

Moving LED at Fixed Velocity

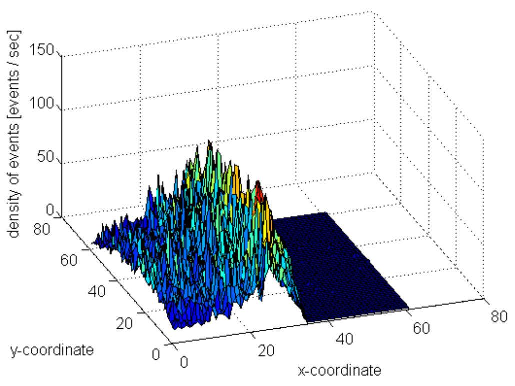
The next question to answer is whether the events produced by the DVS allow estimating the position of a moving blinking LED. By accounting for specific constraints on the timestamps of the events a tracking algorithm should directly be able to estimate the position of the LED at every point in time. In this experiment we focus on the path of a blinking LED moving at a constant velocity, and the characteristics of the events belonging to this path. The setup used for the experiment is described in section 3.1.2. Table 3.7 shows the changed parameters used in this experiment.

For the experiment we expect that the pixels observing the blinking LED have the same characteristics as described in section 3.2.1 for a stationary one. The pixels observing the LED during its movement should have a significantly higher density of events because of the LED's blinking. If we increase the frequency of the LED's blinking the DVS will generate more events and the identification of the pixels observing the LED will be more reliable.

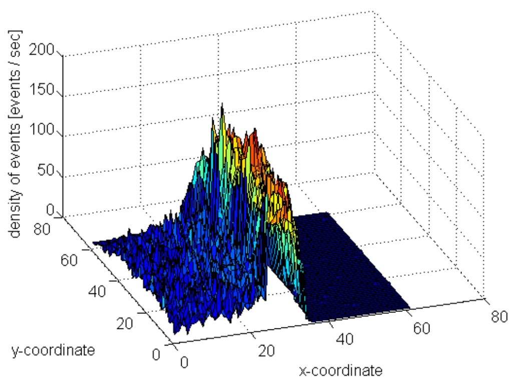
The density map of a moving blinking LED is shown in 3.7. The path of the moving blinking LED is clearly visible with its dense event generation. The spikes on the left side of the figure are induced by the movement of the robots arm. The increased frequency in the experiment leads to a better separation of the pixels observing the blinking LED and the ones observing the robots arm.

The quantitative results are shown in table 3.8. The results clearly show that the density of events in the region of the robots arm and the one in the region of the background are almost constant. The density for the LED induced events increases with an increasing frequency. The three densities are computed by taking manually for each region a set of pixels and computing their densities.

In contrast to a stationary LED (see figure 3.6), the noise of the events not induced by the LED is now significant higher. Consequently, object tracking algorithms have to cope with a reduced signal to background ratio in order to estimate the position of a moving LED. The reduced signal to background ratio is obvious since the moving LED is only for a certain amount of time observed by the same pixel of the DVS. Therefore the number of events per pixel generated by the blinking LED is significantly smaller than in the case of a stationary LED. As the experiment



(a) Moving blinking LED with a frequency of 100Hz.



(b) Moving blinking LED with a frequency of 300Hz.

Figure 3.7.: *The two figures illustrate the normalized number of events generated by a moving blinking LED. The x and y axis corresponds to the pixel locations while the z axis illustrates the density of events per pixel.*

Metric	*Results at 100Hz*	*Results at 300Hz*
Signal Average Density [events / sec]	57.15 ± 1.42	150.57 ± 2.96
Arms Average Density [events / sec]	34.76 ± 0.77	32.41 ± 0.66
Background Average Density [events / sec]	0.06 ± 0.03	0.04 ± 0.02

Table 3.8.: *Setup for the experiment to analyse the density of events produced by a moving blinking LED.*

shows the background activity is not the significant criterion. The most important factor is the activity of the arms movement because its density is clearly higher than the one of the background.

As visible in table 3.7 the frequency was tripled to 300Hz in this experiment. The increased frequency allows a much better distinction and helps to find the path and the current position of the moving blinking LED in a more stable and reliable way. This effect is shown in figure 3.9(b).

Discussion

Using the density of events to get a first rough estimation of the position of a blinking LED seems to be a very simple and reliable approach. It should be used as a prior to decide which regions of the image are important for a further analysis by some more sophisticated methods to compute the position of the blinking LED and its classification.

Parameter	*Quantity*
Distance [cm]	50
Frequency [Hz]	100, 200, 300
Velocity [pixels / sec]	10

Table 3.9.: *Setup for the experiment to analyse the impact of an increasing velocity of the blinking LED.*

3.2.2. Impact of Velocity to the Spatial Distribution

In order to gain further information on the effect of the LED velocity on the signal characteristics an additional experiment was made. Here the DVS observes a blinking LED at increasing velocities. The LED frequency is kept constant. Based on the equations described below, a modified signal to background ratio (SBR) is calculated for the different LED velocities. The setup of the experiment corresponds to the one presented in section 3.1.2 using the robot-arm. The changed parameters of the experiment are listed in table 3.9.

To evaluate the SBR of the events generated by the blinking LED a mask encompassing the path of the blinking LED is used. The mask of the path is shown in figure 3.8. The mask is manually adjusted to contain approximately all signal-carrying pixels.

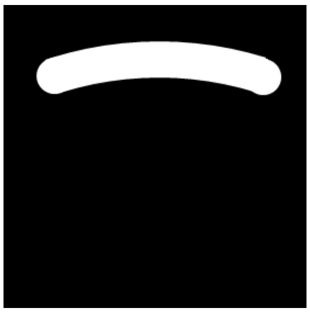


Figure 3.8.: *The binary map represents the path of the blinking LED. The pixel represented by white are those pixels collecting events generated by the moving blinking LED.*

Using this precomputed mask the number of events belonging to the signal and the number of events belonging to the background can be computed. This two numbers are used to calculate a ratio between the signal's events and background events. This corresponds to a SBR adapted to our problem situation. The needed equations are shown in 3.3 and 3.4. Formula 3.3 represents the density of events produced by the blinking LED. The function map_{Signal} is equal to one if the corresponding x, y coordinate has a white value and zero otherwise. The sum of events is then divided by the number of white pixels in map_{Signal} and the time span of the observation $\triangle t$.

$$density_{Signal} = \frac{\frac{\sum_{e \in events} map_{signal}(e.x, e.y)}{\sum_x \sum_y map_{signal}(x, y)}}{\triangle t} \tag{3.3}$$

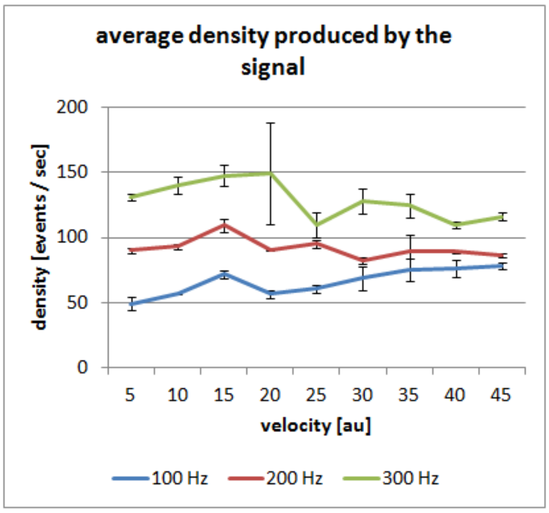
Using the formula shown in 3.3 we are able to compute the ratio by dividing the signal's density of events $density_{Signal}$ by the background density $density_{Background}$ which was computed according to the formula 3.3 using a second map $map_{Background}$. The map $map_{Background}$ is defined similar to map_{Signal} and contains a set of pixels observing only background activity. Formula 3.4 shows the mathematical formulation for the SBR.

$$\text{signal to background} = \frac{density_{Signal}}{density_{Background}} \tag{3.4}$$

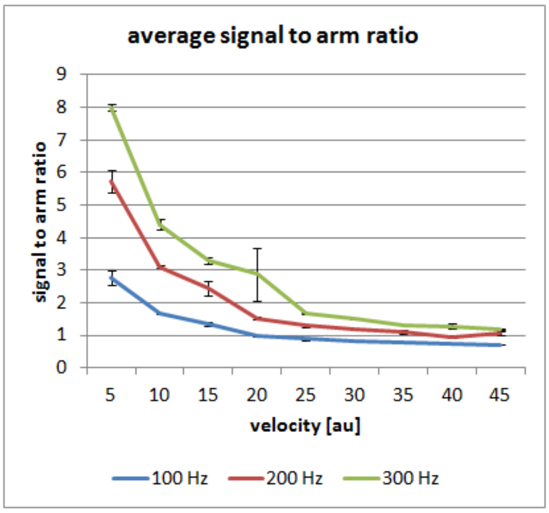
As observed in section 3.2.1 the background activity is low and therefore the SBR in all cases large. A more interesting ratio would therefore be a signal to arm ratio (SAR). With a third map map_{arm} we are also able to compute the density of events in the region of the robots arm. The SAR is then defined by an adapted versions of the formulas 3.3 and 3.4.

In the experiment the velocity of the blinking LED was continuously increased to investigate the effect of an increasing velocity to the above defined SBR and SAR. The observation time corresponds here to the time the LED needs to move from the left to the right side on the arm. According to the observation in section 3.2.1 we expect that the two ratios decrease with an increasing velocity of the LED.

Figure 3.9 shows the results for the experiment with an increasing LED velocity and a constant LED frequency. The two figures show the average density of events produced by the signal and the corresponding SAR. The computed ratio clearly decreases with a higher velocity of the blinking LED while the signals average density $density_{Signal}$ seems to be almost constant.



(a) Shows the density of events generated by a moving blinking LED with an increasing velocity.



(b) The figure shows the ratio of the density of events generated by the signal divided by the density of events generated by the arm movement.

Figure 3.9.: *The two figures are showing the amount and distribution of events generated by a blinking LED mounted on a moving back and forth robot arm.*

As expected, the SAR decreases at increasing LED speed, thus object identification and tracking will be more challenging at higher velocities. On the other hand, the decreasing SAR can be compensated by increasing the LED frequency, thus simply generating more signal-induced events per pixel. According to the results derived from the experiments in section 3.2.1 the SBR behaves similar to the SAR but is always significantly larger than the SAR.

3.2.3. Representation of the LED's Blinking Pattern

A fundamental question this thesis has to answer is the following: Which representation can be used to recognize the LED's blinking pattern? The blinking pattern is here defined as the pattern of the LED's blinking. It describes how many times the LED is turned on an off relative to the period. Formula 3.5 shows the definition of the blinking pattern P where each transition t_i defines at which time $t_i.t$ the LED is turned on $t_i.s = on$ or off $t_i.s = off$.

$$P = \{t_1, ..., t_N\} \tag{3.5}$$

The blinking pattern defines a periodic signal and is therefore repeated after a certain amount of time T. This time T is called the blinking patterns period. Of course the period has to be larger than the largest timestamp in P. This is shown in formula 3.6.

$$T > \max_{t_i \in P} t_i.t = t_N.t \tag{3.6}$$

Histogram

This representation stores the measured time spans between an on- to an off-event and vice-versa in two histograms. The histogram helps to analyse the distribution of the time spans. The mathematical notation of the measured intervals $\triangle t$ is shown in formula 3.7 where e represents a new event and $e.t$ the timestamp of its creation.

$$\triangle t = e.t - \max_{c \in C} c.t \tag{3.7}$$

We search now for the latest event in a set of candidates $C(e)$ to compute the time span between the two events. This helps to overcome the situation when two or more on- or off-events are created instead of only one which occurs if there is a large change in the measured illumination. The definition of the set of candidates is shown in equation 3.8. The candidates $c \in C(e)$ are required to have the same spatial position as the event e and a different type. Since there are only the types on and off, the definition using the inequality is sufficient.

$$C(e) = c : c \in events \wedge c.x = e.x \wedge c.y = e.y \wedge c.type \neq e.type \tag{3.8}$$

The histogram is normalized to compare histograms generated over different observation periods. The normalization is done by dividing each time bin of the histogram $h(i)$ by the sum of all bins as shown in formula 3.9.

Parameter	*Quantity*
Distance [cm]	50
Frequency [Hz]	100

Table 3.10.: *Setup for the experiment to analyse the time distribution between on- to off-events and vice-versa.*

$$h_{normalized}(i) = \begin{cases} 0, & \text{if } h(i) = 0 \\ \dfrac{h(i)}{\sum_j h(j)}, & \text{otherwise} \end{cases} \tag{3.9}$$

The following experiment has to answer the question whether the histogram can be used to represent the LED's blinking pattern. Section 3.1.1 describes the setup of a stationary LED used in this experiment while the changed parameters are shown in table 3.10.

Because of the fact that the DVS generates an event whenever the illumination changes it will generate an event whenever the DVS observes a transition of the blinking LED's state: it produces an event whenever the LED is either turned off or on. Next we will discuss the calculation of the interval between two events depending on the frequency of the signal. This idea is illustrated in formula 3.10 where the interval corresponds to the time span between an on- to the next off-event or vice-versa. In the case of a squared pattern with 50% duty cycle the two intervals are the same.

$$\text{interval} = \frac{1\text{sec}}{\text{frequency}}/2 \tag{3.10}$$

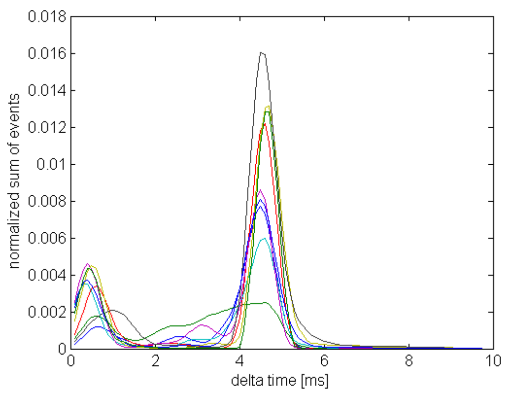
We expect for this experiment in each histogram a peak around 5000 microseconds. This corresponds to the expected interval between two events derived from formula 3.10.

The histograms of the measured intervals between two events using a squared signal at a frequency of 100Hz are shown in figure 3.10. The histograms clearly reveal two major peaks of events. The experiment also shows that there is a significant amount of noise in the high frequency range. This is especially true for the histogram depicting on-off events.

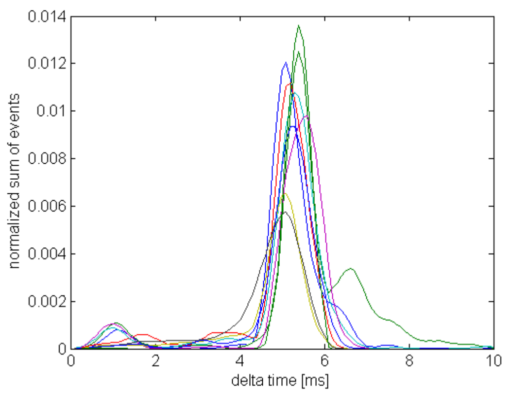
The quantitative results of the experiment are listed in table 3.11.The two histograms peaks are slightly different although the signal duty cycle is 50%.

The experiment has shown that the DVS is able to detect the pattern of a squared signal at a frequency of 100Hz. The noise present in the histograms leads to the conclusion that an algorithm identifying the pattern of a blinking LED has to be able to distinguish between the high frequency noise and the signal itself.

The temporal position of the main peaks does not correspond to the expected temporal position of 5000 microseconds. The respective main peaks for the on-off and off-on events are at 4578 microseconds and 5266 microseconds. There seems to be a certain delay which could be related to the high frequency noise visible in the histogram. A final explanation for the peaks temporal



(a) Time distribution between an on- to the next off-event.



(b) Time distribution between an off- to the next on-event.

Figure 3.10.: *The two histograms represent the time distribution between two events at a frequency of 100Hz. Each curve represents one pixel within the LED-illuminated area.*

Metric	*Results*
Average Temporal Position of on-off Peak [μsec]	4577.78 μsec $\pm$ 27.78
Average Temporal Position of off-on Peak [μsec]	5266.67 μsec $\pm$ 57.74
LED Size [# pixels]	97

Table 3.11.: *Setup for the experiment to analyse the time distribution between on- to off-events and vice-versa.*

position is not yet possible, but it could be explained by a different on-off and off-on latency.

Transition History

This representation indicates at which time the LED is turned on or off. The change of the LED's state is called transition in the following. A transition defines the state of the LED after the transition's timestamp. The transition history th is then an ordered set of transitions describing the LED's blinking pattern over time. This is shown in formula 3.11 where the transition t_i is defined as shown in formula 3.5.

$$th = \{t_1, ..., t_N\} \tag{3.11}$$

The transition history represents the state of a particular pixel over time. The state of a pixel is equal to one, if the last event was an on-event, otherwise the state is equal to zero. The mathematical notation is shown in formula 3.12 where the function $f(x, y, t)$ defines the state at the pixel x, y for a given timestamp t. The variable l in the formula corresponds to the last event observed at this particular pixel.

Parameter	*Quantity*
Distance [cm]	50
Frequency [Hz]	100

Table 3.12.: *Setup for the experiment to visualize the transition history for different frequencies.*

$$f(x,y,t) = \begin{cases} 1, & \text{if } l = \arg_{l \in \{e \in events: e.x = x \wedge e.y = y \wedge e.t < t\}} \max l.t \wedge l.type = \text{ on} \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

A simple experiment was performed to analyse the representation of a LEDs blinking pattern using a transition history. The setup for the experiment with a stationary LED is described in section 3.1.1 while the changed parameters are listed in table 3.12.

For this experiment we expect that the LED's state changes uniformly from an off- to an on-state and vice-versa. Especially the change of the LED's state should be visible in any pixel observing the LED.

Figure 3.11 visualizes the transition history at a frequency of 100Hz. It is clearly visible that the transition histories of the different pixels do not look the same.

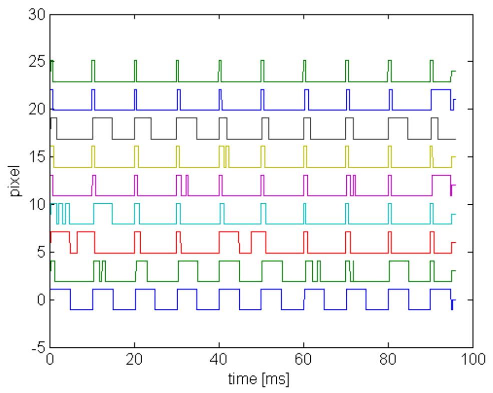


Figure 3.11.: *The figure shows the transition history at a frequency of 100Hz. Each line indicates the states of a particular pixel observing the blinking LED. A high state means that the last event was an on-event while a low state indicates an off-event.*

The experiment has shown that a classification algorithm using the LED's blinking pattern as a criterion has to combine the transition histories of all pixels to figure out the LED's blinking pattern. Although the transition timing is highly synchronized, there is a significant amount of noise present in each transition history. The changes from an on- to an off-state and vice versa do not look the same. The algorithm has therefore to handle the two changes of the LED's state separately to figure out the correct transition.

Discussion

The experiments have shown that an algorithm to identify and classify the temporal pattern of a blinking LED could use the two metrics tested, histogram and transition history, as features of the blinking pattern. The histogram can be used because its peaks allow an identification of the underlying frequencies. A problem occurs if the algorithm has to distinguish between two signals having the same interval histograms. This problem is illustrated in figure 3.12 where both blinking patterns will look the same if we just consider the histogram. Since the time span between different transitions is equally distributed in the two blinking patterns the corresponding histograms are identical.

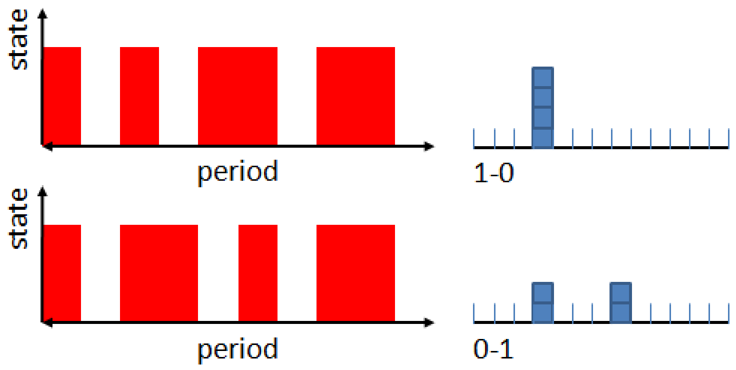


Figure 3.12.: *The two temporal patterns on the figure's left side are different in their behaviour. At the same frequency both temporal patterns will produce the same histograms as shown on the right side of the figure.*

Although histograms do not preserve the absolute time within the signal they provide a stable metric to classify a huge amount of temporal signals in a stable way.

The transition history for a set of pixels is a second metric that can be used to identify and classify the temporal pattern. To use these transitions one has to reduce the noise within the signal in order to have a reliable classification which has to be possible since there are multiple pixels observing the same blinking pattern at least for large markers. Nevertheless it seems that this approach leads to better results since there is no confusion of similar temporal patterns.

3.2.4. Blinking Pattern Recognition and its Limitations

In this section the output of the DVS is analysed to gain insights on its behaviour for different signal characteristics. It analyses the DVS ability to recognize the LED's blinking pattern at different frequencies. The histogram provides different metrics to estimate the DVS's performance in terms of accuracy, sharpness and separation. Since the transition history is just another representation for the same signal it is sufficient to perform the experiment using only the histograms.

Parameter	*Quantity*
LED	red / infra-red
Distance [cm]	50
Frequency [Hz]	50-950

Table 3.13.: *Setup for the experiment to analyse the DVS's temporal accuracy using different frequencies for the blinking pattern of the LED.*

Accuracy

To understand the behaviour of the camera and its ability to extract a high frequency signal the above computed histograms are further investigated to derive statistical relevant views. To extract the temporal pattern of a blinking LED the DVS has to deliver accurate data of the observed LED. In terms of histograms this would mean that the position of the peak in the histogram corresponds to the true temporal position of an event. This section has to answer the question whether the temporal position of a peak is correct or not at different frequencies. In order to find the position p of the peak in the histogram we have to search the bin which has the highest value. The formula is shown in 3.13 where the threshold t_{noise} has to ensure that the peak does not represent high frequency noise.

$$p = \max_{p} h(p),\ p > t_{noise} \tag{3.13}$$

Section 3.1.1 describes the setup of a stationary LED used in the experiment. Table 3.13 summarizes the changed values. To analyse the impact of different types of LEDs the experiment was made using a red LED as well as an infra-red one.

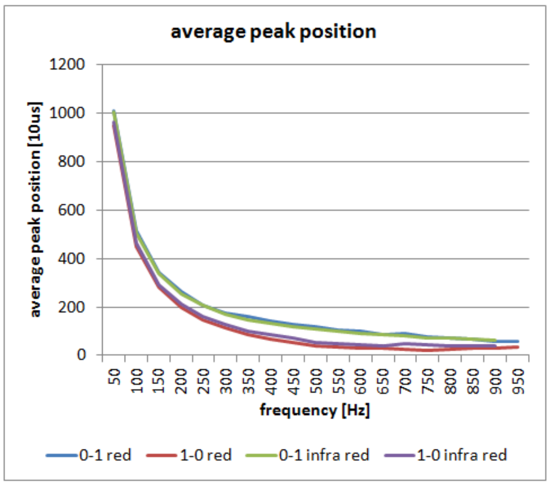
For the experiment we assume that the peaks temporal position decreases with an increasing frequency. This follows directly from formula 3.10 defining the interval between two events given the frequency.

Figure 3.13 shows the statistic of the measured positions of the peak representing the signal. The peaks temporal position clearly decreases with an increasing frequency. The corresponding standard error decreases as well but is for all frequencies low.

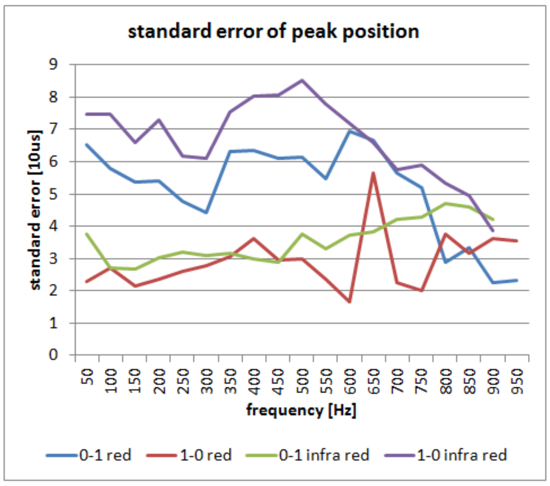
As expected the peaks temporal position decreases with an increasing frequency. The low standard error leads to the conclusion that an algorithm using the DVS is able to compute the interval between two events reliably.

To estimate the quality of the extracted peaks the positions of the peaks are compared with their true positions. The error is shown in formula 3.14 where p represents the measured temporal position and $p_{expected}$ the expected temporal position computed according the formula shown in 3.10.

$$error = |p - p_{expected}| \tag{3.14}$$



(a) The figure represents the average interval between two events.

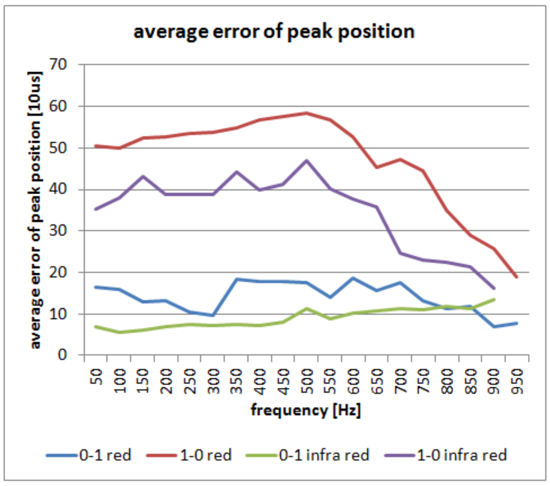


(b) Illustrates the corresponding standard error of the interval between two events.

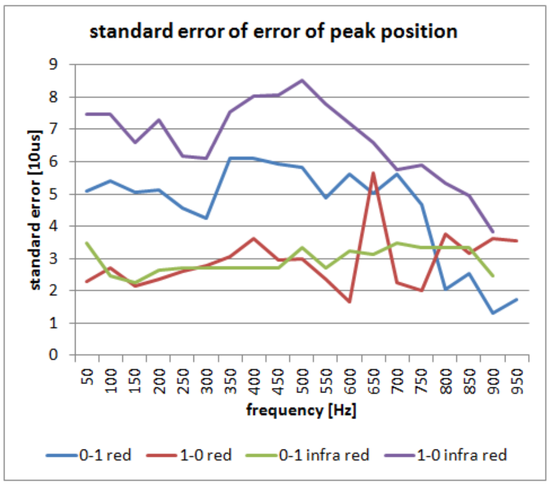
Figure 3.13.: *The figures show the measured time span between on- to off-events (1-0) and vice-versa (0-1). The measurements of a classical red and an infra-red LED are compared.*

As observed in section 3.2.3 there is an error of the extracted peaks temporal position. For the experiment we expect that the error is constant.

The error is represented in figure 3.14. It is clearly visible that the error is much smaller for the intervals between an off- to an on-event while the error between an on- to an off-event seems to be higher in general. Especially the error between an on- to an off-event can be improved by using an infra-red LED. This behaviour is expected from the pixel circuit; since pixel bandwidth is proportional to intensity, off-on transitions will be more precise.



(a) The figure represents the average error two events.



(b) Illustrates the corresponding standard error of the interval between two events.

Figure 3.14.: *The figures show the temporal error of the measured time between an on- to off-event (1-0) compared with the expected time and vice-versa (0-1).*

As expected the peaks temporal error is constant up to a certain frequency. The peaks wrong temporal position is due to the camera latency for incoming on events. This latency is not the same for on and off events and has therefore to be considered by the algorithm. The decreasing error occurs only at high frequencies and is likely to be caused by a quantization error.

Sharpness

Another criterion for the quality of the interval is the sharpness of the peak. To measure the peaks sharpness the full width at half maximum was computed. It is an expression of the extent of a function, given the difference between two extreme values at which the function is equal to half of its maximum. The smaller the width, the sharper is the corresponding peak and the better the quality of the histogram. The mathematical notation of the two extreme values is show in formula 3.15 where h represents the histogram, x_1 and x_2 the extreme points and x_{max} the position of the maximum in the histogram.

$$h(x_1) = h(x_2) = \frac{1}{2} * h(x_{max}) \tag{3.15}$$

The full width at half maximum is then defined as the difference between x_1 and x_2 as shown in formula 3.16.

$$d = |x_1 - x_2| \tag{3.16}$$

The experimental setup corresponds to the one used for the experiment to determine the DVS's accuracy in section 3.2.4. For the experiment we expect that the sharpness is not affected by an increasing frequency.

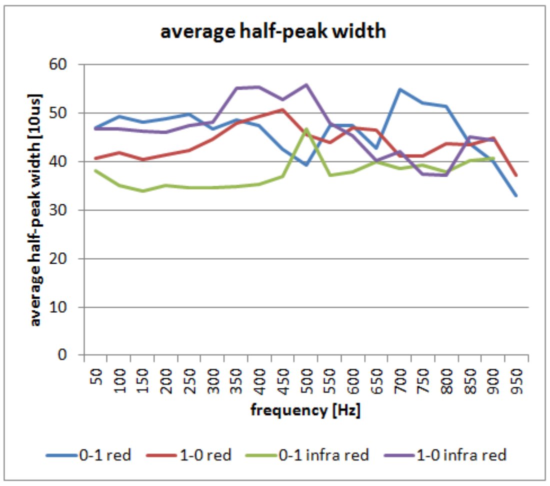
Figure 3.15 represents the results of the experiment with an increasing frequency. The width at half the maximum of the peaks is almost constant for all frequencies and also the corresponding standard error is very small.

To conclude we can say that an increasing frequency does not have a significant impact on the sharpness of the peaks.

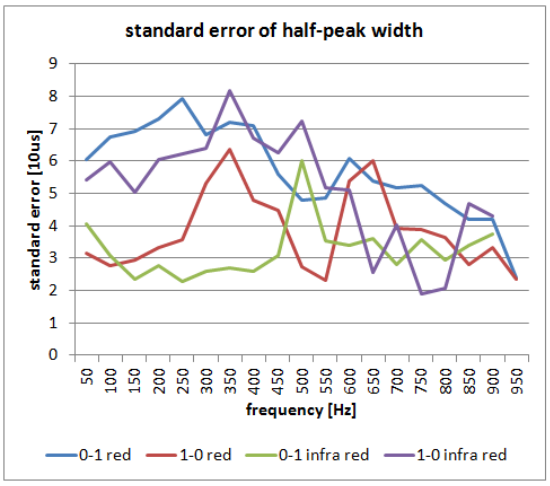
Support

The support is here defined as the fraction of events belonging to a particular peak. The higher the fraction, the more dominant is the corresponding peak. Therefore a peak generated by a blinking LED should have a support as large as possible. The definition of the support is shown in formula 3.17 where l corresponds to the left extreme of the peaks half maximum and r to the right. The definition of the two extremes is derived from formula 3.15. This peaks sum of histogram bins has to be normalized by the total sum of the histogram.

$$s = \frac{\sum_{i=l}^{r} h(i)}{\sum_{i=1}^{n} h(i)} \tag{3.17}$$



(a) *The figure represents the average width at half the maximum of the histogram representing the time distributions*



(b) *Represents the corresponding standard error of the width at half the maximum.*

Figure 3.15.: *The figures show the temporal sharpness of the measured time between an on- to off-event (1-0) compared with the expected time and vice-versa (0-1). The sharpness is defined as the width at half the maximum at each peak.*

The setup for the experiment is the same as the one presented in section 3.2.4. For the experiment we expect that an increasing frequency improves the support of the signal. The reason for this expectation is that an increasing frequency increases the signal to background ratio defined in formula 3.4.

The results of the experiment are represented in figure 3.16. The support clearly increases with an increasing frequency.

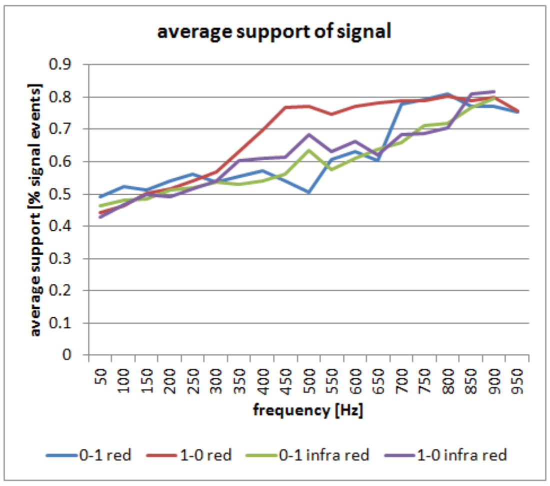
As expected the peaks support increases with an increasing frequency. The higher the frequency of the blinking LED, the better is the corresponding signal to background ratio.

Separation

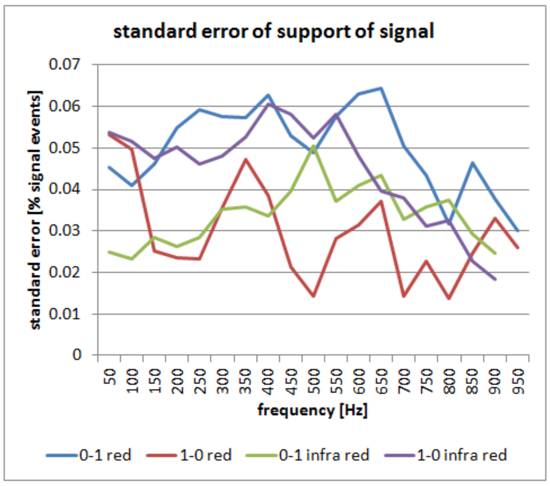
The quality of the histograms depends also on the ability to separate events caused by noise from those generated by the blinking LED itself. If there is more than one peak within the histogram, one generated by noise and one by the blinking LED, the two peaks have to be separable. To decide whether the two peaks are separable or not one can compare their distance at their half maximum height. The width between the two peaks is based on the peaks half maximum as shown in formula 3.15. The computation of the distance is shown in formula 3.18 where r_{noise} represents the right extreme of the peak representing the noise and l_{signal} the signal peaks left extreme.

$$\text{distance} = \max(0, r_{noise} - l_{signal}) \tag{3.18}$$

In the experiment the frequency was continuously increased to estimate the frequencies influ-



(a) The figure represents the average support of the peak considering half the maximum.



(b) Illustrates the corresponding standard error of the support at half the maximum.

Figure 3.16.: *The figures show the support of the peak created by the blinking LED. The support is defined as the fraction of events falling within a peak. To compute the extension of the peak the width at half the maximum was used. The histograms of the measured time spans between on- to off-events (1-0) and vice-versa (0-1) are shown.*

ence on the distance between the peaks. A detailed description of the experimental setup is shown in section 3.2.4.

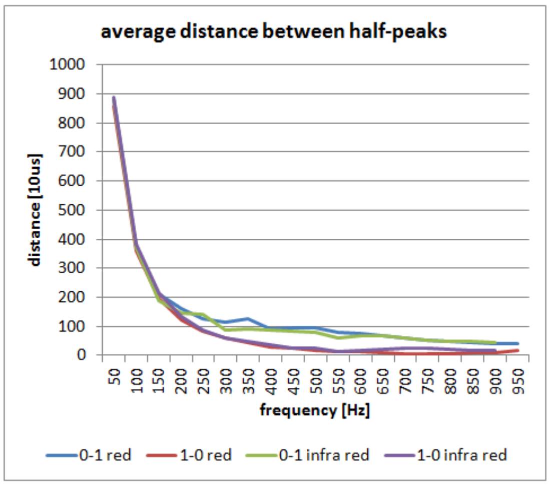
For the experiment we expect that the distance between the two peaks at their half maximum decreases with an increasing frequency. The temporal position of the peak caused by the high frequency noise should be constant with an increasing frequency while the position of the peak caused by the LED induced events decreases.

Figure 3.17 shows the statistics of the measured results of the experiment. The distance between the two peaks at their half maximum decreases with an increasing frequency. E.g. at 500Hz, the infra-red LED produces peaks separated by 833 microseconds for the histogram between off- to on-events and 250 microseconds vice-versa.

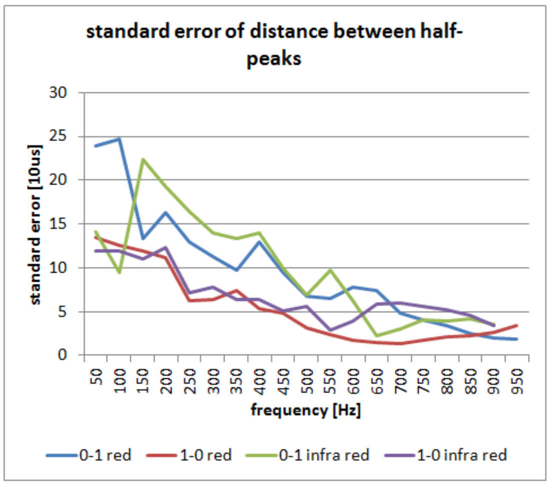
The experiment shows that the separation between off- to on-events from the noise seems to be much easier than the one for on- to off-events. This correlates with the observation already discussed in section 3.2.4 where we measured the accuracy of the peaks. Furthermore the infra-red LED provides a better separation from noise than the visible LED. This characteristic limits the maximal frequency usable since a sufficiently high frequency would make the distinction from noise impossible.

Discussion

The experiment has shown that it is not possible to increase the frequency arbitrarily. There is a maximum frequency at which it is no longer possible to distinct the signals peak from the noise. In the experiment it seems that this point is somewhere between 500Hz and 600Hz. By combining the information from all pixels observing the blinking LED it would perhaps be



(a) *The figure illustrates the measured average distance at half the maximum height between the peaks of the histogram.*



(b) *Represents the corresponding standard error of the measured distances between the peaks at half the maximum of the histogram.*

Figure 3.17.: *The figures represent the average of the distance between half the maximum of the two peaks in the histogram as well as the corresponding standard error. The histograms of the measured time spans between on- to off-events (1-0) and vice-versa (0-1) are shown.*

possible to observe a frequency beyond this limit. Nevertheless there will be an upper limit for the frequency.

3.3. Event Filtering

This section describes a filter to filter out events not produced by a blinking LED. Using the results of section 3.2 we are able to define distinct characteristics for events generated by the same blinking LED. These characteristics are described in the following enumeration:

1. **Spatial Closeness:** The LED is a compact object. An event produced by a blinking LED has therefore neighbouring events at almost the same spatial location.

2. **Temporal Closeness:** The temporal closeness describes the characteristics that a blinking LED produces a set of events having almost the same timestamp.

The filter has therefore to find events having a spatial and temporal closeness to each other. The filter's goal is to group neighbouring events together having almost the same timestamp. It is inspired by the labeling algorithm known in computer vision [BB82]. The filter process is illustrated in figure 3.18.

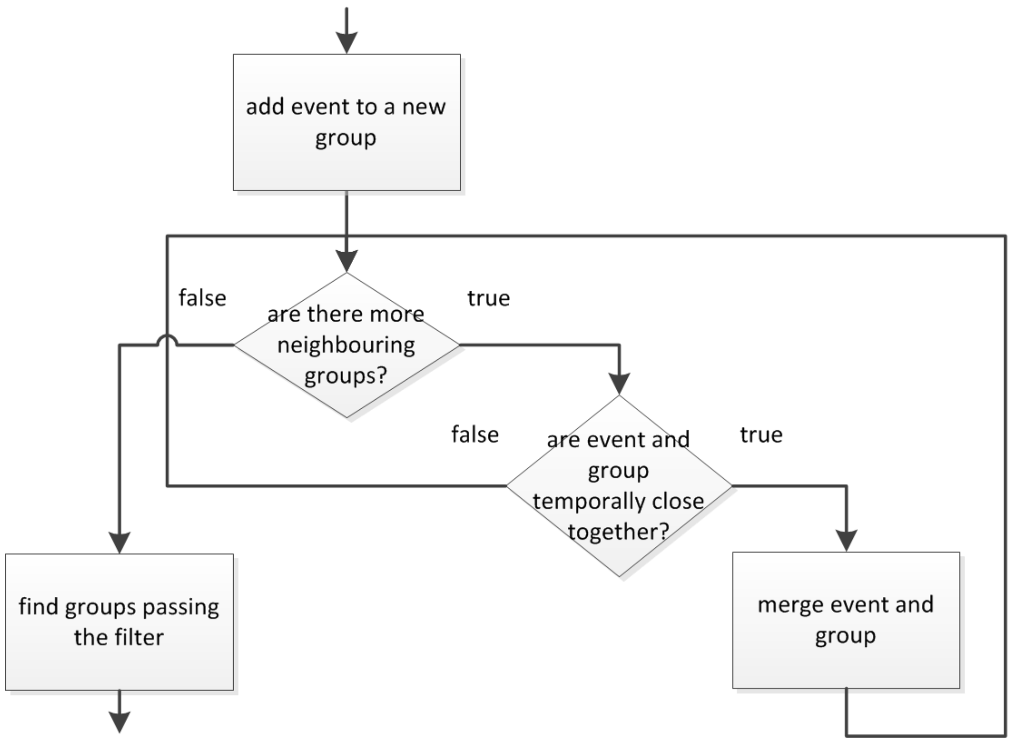

add event to a new group
false
are there more neighbouring groups?
true
false
are event and group temporally close together?
true
find groups passing the filter
merge event and group

Figure 3.18.: *The figure illustrates the process filter out events not generated by a blinking LED.*

Instead of using a gray scaled or binary image like in classical connected component labeling the filter uses a map m. The map is a three dimensional array having in the first dimension the blinking LED's state and in the second and third dimension the x and y coordinate of the pixel locations. The map is used to store references to groups having almost the same timestamp. A detailed description of the filters tasks is shown in the following enumeration:

1. Add the event e to a new group $S_{current} = S_{N+1} = \{e\}$ where N represents the number of groups used by the filter $S = \{S_1, ..., S_N\}$.

2. Store a reference in the map at the events location as shown in formula 3.19.

$$m(e.type, e.x, e.y) = S_{current} \tag{3.19}$$

3. Check all neighbouring locations in the map m considering the events type $e.type$. The connectivity used by the filter is 4-connected.

   a) Compare the event's timestamp $e.t$ with the average timestamp of the neighbouring groups. Formula 3.20 shows the calculation of the average. Here m stores the reference to the groups of events S_i and dx, dy the offset based on the chosen connectivity.

   $$\mu_i = \frac{1}{|S_i|} \sum_{e \in S_i} e.t,\ S_i = m(e.type, e.x + dx, e.y + dy) \tag{3.20}$$

   b) If the events timestamp $e.t$ is close to the groups average timestamp μ_i the algorithm merges the events group $S_{current}$ with the group S_i. The condition is shown in equation 3.21 where $t_{temporal}$ is the threshold for the temporal closeness.

   $$|e.t - \mu_i| \leq t_{temporal} \tag{3.21}$$

   According to the experiments in section 3.2.4 the parameter $t_{temporal}$ is set to 50 microseconds.

   c) Formula 3.22 shows the mathematical representation of merging the two clusters $S_{current}$ and S_i.

   $$S_{current} = S_{current} \cup S_i \tag{3.22}$$

   d) After merging the reference map m is updated. The references pointing to the group S_i are replaced by pointers to $S_{current}$. This is illustrated in formula 3.23.

   $$m(e_c.type, e_c.x, e_c.y) = S_{current},\ \forall e_c \in S_i \tag{3.23}$$

4. At the end the filter has to find groups containing events produced by a blinking LED. If a group has more than 2 events it is possible that the two events are generated by a blinking LED because of their spatial and temporal closeness. The resulting set of groups passing the filter S_{pass} is shown in formula 3.24.

$$S_{pass} = \{S_i \in S : |S_i| \geq 2\} \tag{3.24}$$

This adapted version of a labelling algorithm allows an efficient filtering for events generated by a blinking LED. It reduces the amount of events that have to be processed by the tracking algorithm. The groups S_i are only used here to filter out events not produced by a blinking LED and are different to the clusters in the following chapters.

3.4. Event Clustering

This section describes how the algorithm groups events induced the same blinking LED together. Such a group of events is called cluster in the following. The problem is that an algorithm has to distinguish between an event generated by a blinking LED and noise. An algorithm needs therefore methods based on different metrics to distinguish between the two. Since the algorithm has to be purely event driven it has to assign every incoming event to one of the existing clusters. If there is no assignment possible a new cluster has to be generated.

The goal of the project is to evaluate different approaches to identify blinking LEDs. In order to have a flexible framework to evaluate these different approaches the algorithm has to make decisions based on cost functions. By changing the cost function the algorithm should be able to make new decisions and therefore new assignments.

A cost function has to represent the cost of assigning an object o to a cluster i. Therefore the cost function gives an ordering of the solutions according to their costs. The optimal cluster i for a given object o has to minimize this cost function. This is the only requirement for the definition of a possible cost function. The classification process is illustrated in figure 3.19. The algorithm searches for each incoming event the best matching cluster. It computes for each cluster i a cost to assign an event e to cluster i. By comparing the computed costs the algorithm has to decide whether there is a satisfying solution or not. If there exists such a solution the event is added to the best matching cluster i^*, otherwise a new candidate cluster is created and the event added to the new cluster.

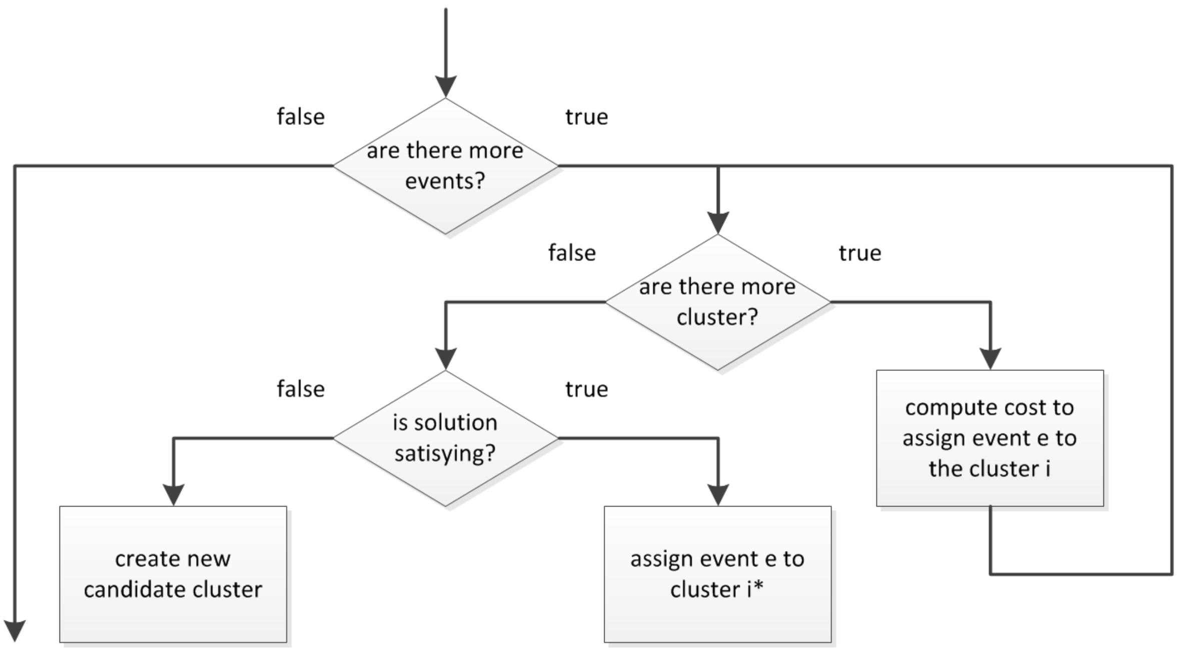

false
true
are there more events?
false
true
are there more cluster?
false
true
is solution satisying?
compute cost to assign event e to the cluster i
create new candidate cluster
assign event e to cluster i*

Figure 3.19.: *The figure illustrates the process to find the best assignment for an event e to a cluster i using a cost functions.*

3.4.1. Different Methods to Assign an Event to a Cluster

This section describes different methods for the assignment of events to a cluster. It describes how the algorithm finds the best cluster i^* for a given event e and how the found solution is evaluated to decide whether there is a valid assignment possible or not.

In this particular example of assigning events to a cluster the event e is represented by the object o. Since the assignment of objects is a very general concept it is explained by objects rather than events to enhance that it can be used in various problem settings.

Threshold Based

To assign a given object o to a cluster i the algorithm can simply search for the best solution. To do this the algorithm evaluates all possible assignments i of the object o and chooses the one minimizing the cost function R. This is shown in formula 3.25.

$$i^* = \min_i R(i|o) \tag{3.25}$$

Since there is no guarantee that the minimizing assignment i^* is a good solution the algorithm has to compare the cost for the assignment with a threshold. If the costs are below the threshold t_{cost} the algorithm assigns the object o to the cluster i^*. Formula 3.26 illustrates the rejection function for the assignment.

$$\min_i R(i|o) = R(i^*|o) < t_{cost} \tag{3.26}$$

Probability Based

Another method to assign an object o to a cluster i based on cost functions is given by probabilities. Here the algorithm uses a Boltzmann definition of weights to compute probabilities out of the costs [Buh11]. The definition of weights is shown in formula 3.27.

$$w(i|o) = e^{-\beta(R(i|o))} \tag{3.27}$$

To compute probabilities out of the weights the algorithm has to normalize the weights. This is represented in formula 3.28. The parameter β represents the sharpness of the weights.

$$p(i|o) = \frac{w(i|o)}{\sum_\alpha w(\alpha|o)} = \frac{e^{-\beta(R(i|o))}}{\sum_\alpha e^{-\beta(R(\alpha|o))}} \tag{3.28}$$

The effect of the sharpness β is illustrated in figure 3.20. A small β leads to a uniform distribution while a large β assigns the object to the cluster with lowest cost with very high probability.

To assign an object o to a cluster i the algorithm has to determine the assignment maximizing the probability i^* as shown in formula 3.29.

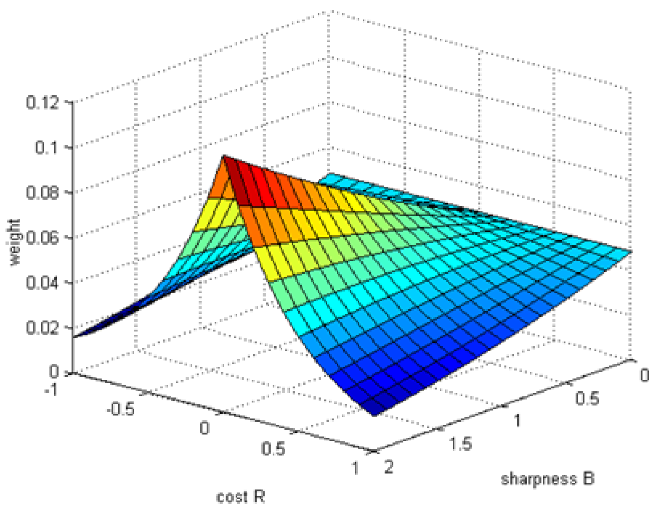


Figure 3.20.: *The figure shows the resulting Boltzmann weights depending on the cost R and the sharpness β.*

$$i^* = \max_i p(i|o) \tag{3.29}$$

Again the algorithm has to use a rejection function to decide whether the best possible assignment is a valid one. Formula 3.30 illustrates the rejection function based on probabilities. If the probability is above the threshold $t_{probability}$ the algorithm assigns the object o to the cluster i^*. The rejection function depends on the chosen β. A small β only assigns an object o to a cluster i^* if the corresponding cost $R(i^*|o)$ is significant smaller than the others costs $R(i|o)$. A larger β on the other hand assigns an object to a cluster also if there is only a small difference in the computed costs $R(i|o)$.

$$\max_i p(i|o) = p(i^*|o) > t_{probability} \tag{3.30}$$

Fuzzy

This approach uses again the definition of probabilities presented in section 3.4.1. Here the algorithm does not assign an object to a single cluster. Instead of that it assigns the object o according to the probabilities $p(i|o)$ to multiple clusters i. The weight of the object in the cluster i is given by the probability $p(i|o)$. An object belongs therefore to multiple clusters with different weights. The definition of weights is shown in formula 3.31. The sum of all assignment weights has to be equal to one.

$$w_{io} = p(i|o), \quad \sum_\alpha w_{\alpha o} = \sum_\alpha p(\alpha|o) = 1 \tag{3.31}$$

3.4.2. Cost Functions to Evaluate Assignments of an Event

To assign an event e to a cluster i the algorithm has to compute the quality of the possible assignments corresponding to the R metric of section 3.4.1. There are many possible metrics to compute this cost. Two obvious possibilities are discussed below.

1. **Spatial proximity** One metric would be the spatial distance between the event e and a candidate cluster i. The smaller the distance the lower would be the corresponding cost.
2. **Size** The size of a cluster i can influence the decision. If there are two clusters having the same spatial distance from the event e one would add the event to the bigger one.

Using cost functions to assign an event e to a cluster i allows testing different features for the assignment while the same framework for the assignment can be used. The only criterion for a cost function would be that it minimizes the cost for the optimal cluster i^*.

Spatial Closeness

As a first indication to which cluster i an event e belongs gives the spatial position $c(i).x, c(i).y$ of the cluster i and the position $e.x, e.y$ of the event e. If the distance between the two is small or even zero, the probability that the event e belongs to cluster i is much higher than if the distance between the two is very large. Thus the cost function has to consist of a term depending on the spatial distance between the event and the cluster. Formula 3.32 shows this cost function where i represents a cluster and e a particular event. The function D represents a dissimilarity and its value is high if the distance is large and low otherwise. The argument is therefore the distance between cluster i and the event's location.

$$R_{distance}(i|e) = D(\sqrt{(c(i).x - e.x)^2 + (c(i).y - e.y)^2}) \tag{3.32}$$

The used function for the dissimilarity D is the quadratic distance function where the argument d represents the distance between two positions. This is shown in formula 3.33.

$$D(d) = d^2 = (c(i).x - e.x)^2 + (c(i).y - e.y)^2 \tag{3.33}$$

Spatial Closeness and Boundary

If the boundary of a cluster i is known we can also use this information in the cost function. Instead of using the distance between the event and the cluster's center the cost function considers the distance to the cluster's boundary. In the case of circular clusters, the cost is shown in formula 3.34 where $c(i).r$ represents the cluster's radius.

$$R_{distance}(i|e) = D(\max(0, \sqrt{(c(i).x - e.x)^2 + (c(i).y - e.y)^2} - c(i).r)) \tag{3.34}$$

Parameter	*Default Value*
β	0.7
$t_{probability}$	0.6

Table 3.14.: *Shows the parameters used for the event assignment.*

3.4.3. Discussion

In the final algorithm the probability based assignment shown in section 3.4.1 was used to assign an event to one of the clusters. The two major reasons for the decision are:

1. In contrast to the cost based assignment the probability based allows the comparison of the best solution with all others. The probability of an assignment contains implicit information about all other assignments through the probability normalization.
2. A Fuzzy assignment of events is not appropriate since an event belongs to exactly one LED. The additional noise induced by false assignments of events leads to the conclusion that a rejection of uncertain assignments leads to a more stable tracking algorithm than one using Fuzzy assignment. Also, we did not have time to consider the computational and algorithms complexity of multiple assignments.

Table 3.14 shows the default values of the parameters used in the tracking algorithm.

The cost function used in the first implementation of the algorithm was the spatial cost function considering also the assumed circular boundary of the object. It is a straightforward approach and corresponds best to the physical properties of the LED marker.

3.5. Basic Features of a Cluster

This section describes some basic features of a cluster. The features are measurements and statistics computed out of the events assigned to a particular cluster. Such features are for example the cluster's spatial position, its lifetime or its activity.

Some of the features presented in this section are used by the cost functions presented in section 3.4.2 for the assignment of events. These features are also used by the tracking algorithm to track the LED's movement while others are used to maintain the clusters by the tracking algorithm.

3.5.1. Position

The position of the cluster is perhaps the most elementary property of the cluster. Out of the events assigned to the cluster the algorithm has to compute its spatial position.

There are plenty of different methods to compute an objects center. Using the profile of an object one can average a set of edge points to compute a center of the region [DxYx10]. Other methods are using a segmentation to compute the objects centroid.

A simple and efficient way to compute the clusters spatial position is by using the central moments known from image analysis. The definition of the ij image moments is shown in formula 3.35 where I represents the intensity values of a grayscaled image. Simple properties of the image which are found via image moments are the objects area (or total intensity), its centroid, and information about its orientation [Kil01].

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \tag{3.35}$$

This formula has to be adapted to our problem situation because there are no intensity values I. Instead of intensities the formula uses the set of events $c(i).e(t)$ assigned the cluster i and falling into a certain time window. The maintenance of the events can be done efficiently by using a circular list as data structure. The list has a fixed size. It is large enough to hold events back in time in the needed time interval. The interval corresponds to the maximum LED pattern period. The definition of this set of events $e(t)$ above a time threshold t is described in formula 3.36 where $c(i).e$ represents the set of all events assigned to the cluster i. The threshold t is the oldest allowed event time $e.t$ for the events in the circular list.

$$c(i).e(t) = \{e : e \in c(i).e \wedge e.t > t\} \tag{3.36}$$

The adapted formula for the computation of moments is shown in 3.37.

$$M_{ij} = \sum_{e \in c(i).e(t)} e.x^i e.y^j \tag{3.37}$$

The centroid of the events can be computed by using directly the raw moments. This is shown in equation 3.38.

$$\{c_x, c_y\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\} \tag{3.38}$$

The advantage of the central moments is that they can be used to compute the distribution of the assigned events and their orientation. But the quantization error introduced in discrete signals presents problems, especially when dealing with small-size images. Since we do not use the moments as feature descriptors the error has not significant impact on the algorithm. There exists a method to calculate moments from a continuous signal and is able to cope with the quantization error [LK07].

3.5.2. Contour

The contour of a cluster represents the LED's silhouette. There are various approaches for the representation of a contour. A possible representation can be an union of a number of representative lines [SN09]. Other approaches reduce the computation time by using a pixel representation considering also the pixels neighbourhood [KR06].

To represent the contour of a blinking LED we use a straightforward representation. The blinking LED is not considered as a compact object represented by a binary image as shown in the approaches [SN09] and [KR06]. In contrast to that we consider the events assigned to a blinking LED as a distribution of spatial positions. Without compromising the key goals of this thesis we simplify the derivation of the contour of the object: In fact, we approximate the LED's shape by an ellipse with center c_x, c_y and half axes $2\sigma_x, 2\sigma_y$.

Using the definition of moments shown in equation 3.37 the algorithm is able to compute the variance in x and y direction. Formula 3.39 shows the mathematical definition.

$$\sigma_x^2 = M_{20} - c_x M_{10}, \quad \sigma_y^2 = M_{02} - c_y M_{01} \tag{3.39}$$

If we assume now that the distribution of events follows a normal distribution then it is proven that 95% of the events fall within 2 times the standard deviations of the mean. Thus, the actual contour of an LED is defined by formula 3.40.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c_x + 2\sigma_x \cos t \\ c_y + 2\sigma_y \sin t \end{pmatrix}, \; 0 \leq t \leq 2\pi \tag{3.40}$$

To justify the usage of a normal distribution with mean μ and standard deviation σ we used the Maximum Entropy Method. The resulting distribution derived by the entropy maximization with the above defined constraints is then a normal distribution [Buh11].

The doubled standard deviations $2\sigma_x$ and $2\sigma_y$ can therefore be used to determine the clusters boundary in x and y direction. Additionally (though not used here) the algorithm is able to directly compute the clusters orientation by using the image moments as presented in formula 3.41.

$$\phi = \frac{1}{2}\arctan(\frac{2M_{11}/M_{00} - c_x c_y}{(M_{20}/M_{00} - c_x^2) - (M_{02}/M_{00} - c_y^2)}) \tag{3.41}$$

3.5.3. Velocity

The velocity of the object measures the clusters speed and the movement direction. This is shown in formula 3.42 where $c(i).x$ represents the clusters current position and $c(i).x_{last}$ the last measured position. The time span $\triangle t$ represents the time span between the two measured positions.

$$c(i).v = \frac{c(i).x - c(i).x_{last}}{\triangle t} \tag{3.42}$$

3.5.4. Lifetime

The lifetime of a cluster is the difference in microseconds from the clusters birth and the current timestamp of the algorithm. The timestamp of the clusters birth is given by the first event assigned to it. This is shown in formula 3.43 where t represents the algorithm's current timestamp and $c(i).l(t)$ the function of the clusters lifetime.

$$c(i).l(t) = t - \min_{e \in c(i).e} e.t \tag{3.43}$$

3.5.5. Activity

The activity of a cluster measures the number of assigned events per time unit. It corresponds to a density of events and is a good criterion to decide whether the cluster observes a blinking LED or not. This is one of the key observations presented in section 3.2.

To observe a certain time window the algorithm uses again a circular list to maintain the events. Formula 3.44 shows the definition of the activity $c(i).a$ where $c(i).e(t)$ represents the observed time window and $c(i).A$ the clusters area in pixels. The area can be computed out of the clusters boundary information presented in section 3.5.2.

$$c(i).a = \frac{c(i).e(t - t_{threshold})}{c(i).A} \tag{3.44}$$

3.6. Features Used to Identify a Blinking LED

To classify and identify the pattern of a blinking LED the algorithm has to extract various features from each cluster. Using the information of these features the algorithm computes the LED's blinking pattern. This section describes the features the algorithm computes to find the blinking pattern of an LED.

3.6.1. Cluster Transition History

The transition history of a cluster (which is different than pixel transition of section 3.2.3) represents how the cluster changes its state over time. In other words it represents at which time the LED is turned on and off. The transition history has to use the events assigned to a cluster to compute the state of the LED at each point in time. Using all information available the algorithm has to reduce the influence of noise. This section describes different approaches to compute the transition history out of these events.

For every change of the state a transition t has to be created which contains the information in which state the cluster is after the transition and at which time the change occurs. The two properties of a transition t are described in the following enumeration.

1. $t.s$: Represents the clusters state after the transition t.
2. $t.t$: Defines the timestamp of the transition. It represents the time at which the transition occurs.

The clusters transition history $c(i).th$ is an ordered set of transitions t_i. The definition of the set is shown in formula 3.45.

$$c(i).th = \{t_1, ..., t_n\},\ t_i.t < t_{i+1}.t \tag{3.45}$$

Election of Cluster Transitions from Pixel Transitions

This approach to determining cluster transitions uses an election to find the transitions of the states of a cluster. The general idea is to observe a time interval and to compute the sum of events for each type. The majority of events determine the current state of the cluster. The formula is shown in 3.46 where $c(i)$ represents the cluster with index i and $c(i).e(t-w, t+w)$ the clusters set of events falling into the interval $[t-w, t+w]$. The parameter w represents the size of the observed interval.

$$v(t) = \sum_{e \in c(i).e(t-w,t+w)} \begin{cases} 1, & \text{if } e.type == \text{ on} \\ -1, & \text{otherwise} \end{cases} \tag{3.46}$$

The mathematical notation of the set $c(i).e(t-w, t+w)$ is shown in formula 3.47 where $c(i).e$ corresponds to the set of all assigned events to the cluster $c(i)$.

$$c(i).e(t-w,t+w) = \{e : e \in c(i).e \wedge e.t > t-w \wedge e.t < t+w\} \tag{3.47}$$

If the state of the cluster $c(i).s$ is off and most of the events vote for an on state the algorithm has to change the clusters state and to create a new transition at time t. Formula 3.48 describes this condition where $v_{threshold}$ is a predefined threshold that has to be exceeded by $v(t)$ to allow a change of the state. It is sufficient to define $v_{threshold}$ equal to one.

$$c(i).s == \text{off} \wedge |v(t)| > v_{threshold} \tag{3.48}$$

Of course the same approach can be used to detect a transition from an on to an off state. Figure 3.21 illustrates the transition history. The dotted red line represents the final position of transition in time computed on the incoming events visible in the figure. In this case it accurately computes the squared wave with 50% modulation.

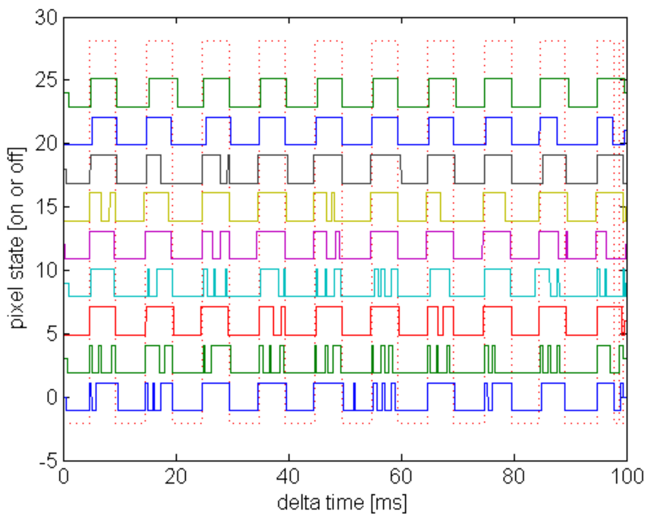


Figure 3.21.: *The figure shows the resulting transition history based on the election. The red dotted line indicates the final position of the transition.*

Kernel-Based Cluster Transition Estimation

However, this voting approach does not allow constraining the clusters transition history by a known or estimated temporal marker pattern. Therefore we developed the following Kernel-based approach.

This approach uses a kernel to convolve the clusters stream of incoming events with it. Convolution has to do with the multiplication of functions [Kre05]. The incoming events are represented by a function g. This function returns for each timestamp t the density of events observed at a particular time t. The mathematical notation is shown in formula 3.49 where i corresponds to the index of the cluster we are interested in and $c(i).e$ the cluster's set of assigned events.

$$g_i(t) = |\{e : e \in c(i).e \wedge e.t = t\}| \tag{3.49}$$

By convolving this function g with a kernel we are able to compute different kinds of density of events by using different kernels f. Formula 3.50 represents the mathematical equation for the convolution where t represents a particular timestamp and w the size of the kernel.

$$(f * g)(t) = \int_{-w}^{w} f(\tau)g(t+\tau)d\tau \tag{3.50}$$

If the kernel f is chosen as a rectangular kernel the result of the convolution corresponds to the sum of events in the interval defined by the kernel's size. It also represents a box-car low pass filtering of the event density over time. The definition of a rectangular kernel is given in formula 3.51 where w corresponds to the size of the kernel.

$$f(\tau) = \begin{cases} 1, & \text{if } \tau > -w \wedge \tau < w \\ 0, & \text{otherwise} \end{cases} \tag{3.51}$$

Using the result of the convolution the algorithm has then to search for possible transitions. A transition at time t has to satisfy two criterions:

1. On the one hand the transition has to be a local maxima of the convolution $(f * g)(t)$. This observation is obvious if we look at the convolution as some kind of density of events in a certain time interval. Then the correct position of the transition has to maximize the density of events.
2. On the other hand the result of the convolution has to be above a certain threshold in order to reduce the influence of noise. To find such a threshold the algorithm has to search for a global maximum in an interval that contains at least one valid transition. The size of this interval has to be chosen appropriate to ensure this condition. The global maximum represents the number of events of the most dominant cluster transition in the interval.

To find a local maximum the algorithm has to search in a particular interval for the maximum. The size of the interval is directly related to the size of the kernel f. Since the size of f is represented by w the size of the interval for the search of a local maximum is w as well. The equation is shown in 3.52 where w defines the size of the interval $[t - w, t + w]$ within the algorithm searches for the local maxima at time t. The parameter w is set to 200 microseconds based on the experiments performed in section 3.2.4.

$$m_{local}(t) = \max_{t-w<\tau<t+w}(f(\tau)) \tag{3.52}$$

The same principle is used to find a global maximum. Instead of using a interval represented by the size w a size $s > w$ is used. The minimal size for s has to be chosen in a way that there is at least one valid transition within the interval. The mathematical notation for the global maxima is shown in 3.53. Here s is set to 10000 microseconds to ensure the presence of a transition.

$$m_{global}(t) = \max_{t-s<\tau<t+s}(f(\tau)) \tag{3.53}$$

The global maxima m_{global} can then be used as a threshold to find a valid transition of the state.

Criterion	*Election*	*Kernel-based*
Distance	constant	adaptive
Accuracy	continuous	continuous
Noise	constant	adaptive

Table 3.15.: *Lists the characteristics for each proposed solution to generate the transition history.*

The necessary condition for all transitions is illustrated in formula 3.54. The first condition ensures that the transition at time t is a local maxima in the interval with size w. The second condition defines the maximal deviation $f_{deviation}$ from the global maxima m_{global}. Since a transition is described by a bunch of events and the chosen window size s for the global maxima is large enough to ensure at least one valid transition, the global maxima m_{global} contains the number of events of the most dominant transition. The deviation $f_{deviation}$ then serves as a relaxation for this best found transition.

$$\{t : \ (f * g)(t.t) \geq m_{local}(t.t) \wedge (f * g)(t.t) \geq m_{global}(t.t) * f_{deviation}\} \tag{3.54}$$

Discussion

To evaluate the different approaches to generate the cluster's transition history a decision matrix was used to find the best solution. The decision criteria are described below.

1. **Distance** Evaluates the approaches ability to handle different distances of the blinking LED. Since a LED very close to the DVS will produce much more events because of its size than a LED far away, the algorithm has to adapt itself to compute the transitions.
2. **Accuracy** This criterion is used to describe whether a solution is able to find the correct time of the transition or if there is a certain delay or a low resolution.
3. **Noise** Describes the algorithm's ability to handle noise.

The evaluation is shown in table 3.15. According to this evaluation the Kernel-based method is the best approach. This approach is more robust to noise than the election.

3.6.2. Transition History Sampling

Some method like the fast Fourier transformation requires a discrete version of the signal. Therefore the cluster's transition history has to be sampled. Since the transition history provides a stable and noise reduced version of the signal it is used as input for the sampling process.

The clusters transition history $c(i).th(t)$ returns for a given timestamp t the cluster's state at this particular time. The sampling s is defined as shown in formula 3.55. Here f_s represents the sampling frequency. The sampling is therefore a discretization of the cluster's state in time.

$$s = \left\{ c(i).th\left(\frac{1\text{sec}}{f_s} * j\right)\right\},\ j \in \mathbb{N} \tag{3.55}$$

If the transition history is sampled we have to consider the Nyquist-Shannon sampling theorem. A sufficient number of samples of the signal should be taken so that the original signal is represented in its samples completely. Also it should be possible to recover or reconstruct the signal completely from its samples. The number of samples to be taken depends on the maximum signal frequency present. The sampling theorem states that a continuous time signal can be completely represented in its samples and recovered back if the equation 3.56 is satisfied. Here f_s is the sampling frequency and W is the maximum frequency present in the signal [Chi03].

$$f_s \geq 2W \tag{3.56}$$

3.6.3. Marker Blinking Pattern Period Estimation

A fundamental property of the signal is its period. The period is the smallest value of T satisfying equation 3.57 for all t.

$$g(t+T) = g(t),\ \forall t \in \mathbb{R} \tag{3.57}$$

In essence, the period T is the smallest amount of time it takes for the function to repeat itself. Once an algorithm was able to determine the period of the signal it is able to extract the absolute signal by superimposing the observed pattern using the found period. This section describes different methods to find the period of the signal or its frequency.

Transition-Based Efficient Autocorrelation and Period Estimation

The autocorrelation function is used to measure the match between the signal and its time delayed version [Chi03]. Informally, it is the similarity between observations as a function of the time separation between them. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal which has been buried under noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. It is often used in signal processing for analysing functions or series of values, such as signals in the time domain [Agr09]. The autocorrelation function is shown in formula 3.58 where τ represents the lag of the signal f.

$$R_{ff}(\tau) = \int_{-\infty}^{\infty} f(t)f(t+\tau)dt \tag{3.58}$$

The equation 3.58 has to be adapted to our problem situation where we have to perform the autocorrelation on the clusters observed transition history $c(i).th$ defined in section 3.6.1. This adaptation is shown in formula 3.59.

$$R_{c(i).th,c(i).th}(\tau) = \int_{start}^{end} c(i).th(t)c(i).th(t+\tau)dt \tag{3.59}$$

Such a correlation is an expensive operation. It would be a good idea to discretize the integral in equation 3.59 and to replace it by a sum.

We assume now that the mathematical function representing the LED's transition history is equal to one at a particular time, if the LED is turned on and minus one, if it is turned off. This is shown in formula 3.60 where t corresponds to a particular timestamp.

$$c(i).th(t) = \begin{cases} 1, & \text{if LED is on at } t \\ -1, & \text{otherwise} \end{cases} \tag{3.60}$$

In order to simply the following calculations we define a second function $c(i).th_\tau(t) = c(i).th(t+\tau)$. Figure 3.22 illustrates the result for the integral in formula 3.59. Whenever the value of $c(i).th(t)$ is the same as the one of $c(i).th_\tau(t)$ the result is 1, otherwise -1. The result is indicated by the green area of the figure. The sign changes only if there is a change of the state in $c(i).th(t)$ or $c(i).th_\tau(t)$.

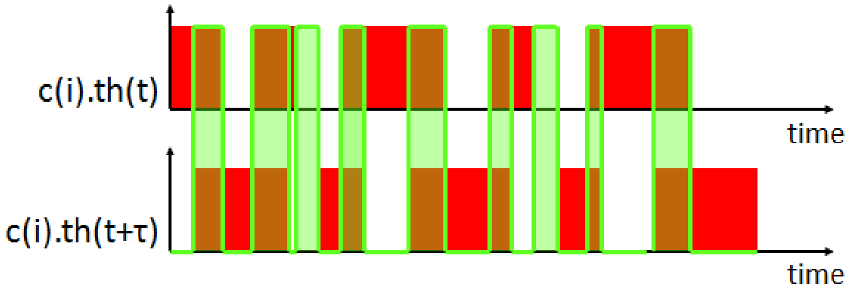


Figure 3.22.: The green area shows the result of the correlation function. Here $c(i).th$ corresponds to the LED's observed transition history and $c(i).th_\tau = c(i).th(t+\tau)$.

Since these changes of the LED's state occur only at a discrete number of positions we can accelerate the computation of the integral. We define an ordered set of transitions $T = \{t_1, ..., t_n\}$ where T is the union of all transitions in $c(i).th(t)$ and $c(i).th_\tau(t)$. The ordering of the set is based on the timestamp of each transition which is illustrated in formula 3.61. Here $t_i.t$ represents the timestamp of the i-th transition t_i.

$$T = \{t_i : t_i, t_{i+1} \in c(i).th(t) \cup c(i)_\tau.th(t) \wedge t_i.t < t_{i+1}.t\} \tag{3.61}$$

This ordered set of transitions can be used to reformulate the integral in equation 3.59. The number of transitions in T is represented by $n = |T| - 1$. The resulting summation is represented in formula 3.62. The first part of the equation where the states of the two functions are compared looks still the same. The second part computes the amount of time $t_{i+1}.t - t_i.t$ until the next change of the state happens. By multiplying this time span we are able to compute again the same area.

$$C_{c(i).th,c(i).th_\tau}(\tau) = \sum_{i=1}^{n-1} c(i).th(t_i.t)c(i).th_\tau(t_i.t) * (t_{i+1}.t - t_i.t) \tag{3.62}$$

To find the signal's period an algorithm has to search for local maxima of the autocorrelation function. The offset τ maximizing the autocorrelation function is then a possible period of the signal. This is represented in formula 3.63.

$$period = \max_\tau C_{c(i).th,c(i).th_\tau}(\tau) \tag{3.63}$$

The algorithm performs the autocorrelation function for different time intervals $[t_{start}, t_{end}]$ and searches in each interval for the local maxima. The interval's size $t_{end} - t_{start}$ has to be larger than the largest LED period. Once the algorithm was able to determine the local maxima a K-Means clustering algorithm can be used to search the most reliable period out of the different maxima. In statistics and data mining, K-Means clustering is a method of cluster analysis which aims to partition n observations into k groups in which each observation belongs to the group with the nearest mean [Seg07]. The clusters used in K-Means clustering are called groups in the following to prevent confusion with the tracked clusters representing a marker.

Given is a set of observations $(x_1, ..., x_n)$ where each observation is a maxima of the autocorrelation function. K-Means clustering has to partition the n observations into k groups where $k \leq n$. This groups $S = (S_1, ..., S_k)$ have to minimize the within cluster sum of squares. This is shown in formula 3.64 where x_j is an element of group S_i and μ_i the mean of all points in S_i.

$$cost_k = \arg_S \min \sum_{i=1}^{k} \sum_{x_j \in S_j} \|x_j - \mu_i\|^2 \tag{3.64}$$

Since the number of groups is unknown we need an additional condition to have a criterion whether the current number of groups is a good choice or not. This is shown in formula 3.65 where the within-cluster sum of squares is compared with a threshold. If the cost is below the threshold the number of groups is sufficient otherwise we have to increase this number. The threshold t_{cost} is defined as the number of candidate periods n multiplied with the allowed error of 100 microseconds.

$$cost_k \leq t_{cost} = n * 100\mu sec \tag{3.65}$$

The process to find the period of the signal using K-Means includes the following steps:

1. Perform K-Means clustering with a single group $k_0 = 1$.
2. Compute the cost as the within-cluster sum of squares as shown in equation 3.64.
3. Check whether the condition in equation 3.65 is satisfied.
4. If the condition is satisfied the solution is valid and the algorithm will give the result back. Otherwise the number of groups is increased by one $k_n = k_{n-1} + 1$ and the process starts again with step 1.

By analysing the resulting set of groups S_i the algorithm computes the period. The algorithm searches for the groups having the largest number of elements (candidate periods) as shown in 3.66.

$$S_{max} = \arg_{S_i} \max |S_i| \tag{3.66}$$

The period is then computed by taking the average of the assigned candidate periods x_i. The mathematical notation is shown in formula 3.67.

$$\text{period} = \frac{1}{|S_{max}|} \sum_{x_i \in S_{max}} x_i \tag{3.67}$$

Fourier Transformation

Another possibility to find the period of the signal is given by the Fourier transformation. In mathematics, the discrete Fourier transform (DFT) is a specific kind of discrete transform, used in Fourier analysis. It transforms one function into another, which is called the frequency domain representation, or simply the DFT, of the original function (which is often a function in the time domain). But the DFT requires an input function that is discrete and whose non-zero values have a limited (finite) duration. Such inputs are often created by sampling a continuous function [Smo10].

Because of the properties of the Fourier Transformation the algorithm has just to determine the first local maxima which correspond to the frequency of the signal. Out of this frequency the period of the signal can directly be determined by using the equation 3.68.

$$\text{period} = \frac{1 \text{ sec}}{\text{frequency}} \tag{3.68}$$

To reduce the influence of noise the Fourier Transformation is performed over different sampling intervals. A Bartlett periodogram is used to maintain the results. The Bartlett periodogram is based on the notation of creating a pseudo-sample sequence by dividing a long sequence of samples into P non-overlapping segments of D samples each [Ror10]. If the original sequence is $x[k]$, the p-th segment can be expressed as shown in formula 3.69.

$$x_p[n] = x[pD + n] \tag{3.69}$$

Next the discrete-frequency sample spectrum is computed for each of the P segments.

$$S_p[m] = \frac{D}{T} \left| \sum_{n=0}^{D-1} x_p[n] e^{\left(\frac{-j2\pi mn}{D}\right)} \right|^2 \tag{3.70}$$

Criterion	*Autocorrelation*	*Fourier Transformation*
Effort	moderate	moderate
Time	depends on signal	constant
Accuracy	high	high
Range	dynamic	limited

Table 3.16.: *Lists the advantages and drawbacks for each proposed solution to compute the period of the signal.*

The individual sample spectra for these segments are then averaged to form the periodogram. The arithmetic average of the P different sample spectra at each frequency is shown in formula 3.71.

$$S_B[m] = \frac{1}{P}\sum_{p=0}^{P-1} S_p[m],\ m = 0, 1, ..., D-1 \tag{3.71}$$

The calculation of the blinking LED's frequency is straightforward. First of the algorithm has to search for the first local maxima m_{local} in the periodogram defined in formula 3.71. Using the formula 3.72 the frequency can directly be calculated where D is the number of samples and f_s the sampling frequency [Sun01].

$$\text{frequency} = \frac{f_s * m_{local}}{D} \tag{3.72}$$

Discussion

This section evaluates the different approaches used to find the period. The advantages and drawbacks are listed in table 3.17. The criteria are described blow.

1. **Effort:** Evaluates the expected computational effort of the chosen method.
2. **Time:** The expected amount of time the method uses to find the blinking LED's period.
3. **Accuracy:** Describes the methods potential accuracy to compute the period.
4. **Range:** The range indicates whether the method is able to find an arbitrary large period or whether the method finds only periods in a limited interval.

The evaluation does not lead to a clear decision which approach has to be used by the algorithm. The two approaches have to be compared by performing an experiment to evaluate their performance in practice.

3.6.4. Extracting the Temporal Pattern

This section presents different approaches to extract the pattern of the blinking LED. The computation of the temporal pattern is based on precomputed features like the LED's transition history and its period. We assume that the searched blinking pattern of a marker is unknown.

The temporal pattern has to describe the state of the blinking LED at every point in time. Since we search for a periodic signal the temporal pattern describes the behaviour of the blinking LED completely because the temporal pattern will be repeated as long as the DVS is able to observe the blinking LED. The LED's temporal pattern can therefore be used as a feature to uniquely identify a blinking LED if there are no two LED's having the same pattern. The classification of an LED based on its temporal pattern is discussed in section 3.7.

The blinking pattern's mathematical definition is shown in formula 3.73. The pattern is an ordered set of transitions t_i. Each transition has a timestamp $t_i.t$ of the transition's occurrence and a state $t_i.s$. Every two transitions t_i and t_{i+1} defines the cluster's state in the interval $[t_i.t, t_{i+1}.t]$. The cluster's state in the interval corresponds to the state $t_i.s$ Because of the pattern's periodicity the largest timestamp $t_N.t$ has to be equal to the corresponding period.

$$P = \{t_1, ..., t_N\}\,,\ t_N.t = period \tag{3.73}$$

Majority Election of Temporal Pattern

This approach divides the temporal pattern uniformly into a finite number of time slots where each slot represents a certain amount of time $\triangle t$. This corresponds to a discretization of the time domain of the signal. To do the discretization a frequency resolution r is used. The relation between the temporal width of a slot and its resolution is presented in equation 3.74.

$$\triangle t = \frac{1 \text{ sec}}{r} \tag{3.74}$$

The formula to find the index of the time slot for a given transition t is shown in 3.75 where the function f is used to compute the index. The function takes as an argument a transition t and the cluster $c(i)$ to which the transition is assigned. The LED's period is represented by $c(i).p$.

$$j = f(t, c(i)) = \lfloor \frac{t.t \bmod c(i).p)}{r} \rfloor \tag{3.75}$$

To simplify the mathematical notation we introduce an ordering on the set of transitions in the clusters transition history $c(i).th$. The ordering of transitions t_i is based on their timestamp $t_i.t$ as shown in formula 3.76.

$$t_i.t < t_{i+1}.t,\ t_i, t_{i+1} \in c(i).th \tag{3.76}$$

For each of the slots in the LED's blinking pattern the algorithm votes for the state. This is done by superimposing the extracted transition history using the modulo of the signal period.

The mathematical notation of the election is shown in formula 3.77 where $t_i.s$ represents the state of the transition t_i. A transition t_i votes for its state $t_i.s$ in the interval $[t_i.t, t_{i+1}.t]$. This corresponds to the time span from transition t_i to the next transition t_{i+1}.

$$v(j) = \sum_{i=1}^{n} \begin{cases} 1, & \text{if } f(t_i, c(i)) \leq j \wedge f(t_{i+1}, c(i)) > j \wedge t_i.s = \text{on} \\ -1, & \text{if } f(t_i, c(i)) \leq j \wedge f(t_{i+1}, c(i)) > j \wedge t_i.s = \text{off} \\ 0, & \text{otherwise} \end{cases} \tag{3.77}$$

To determine the state of a particular slot the algorithm has just to look at the sign of the votes at the particular slot as illustrated in formula 3.78. Here $s(j)$ represents the state of the signal at slot j.

$$s(j) = \begin{cases} 1, & \text{if } v(j) > 0 \\ 0, & \text{if } v(j) < 0 \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{3.78}$$

Since we know the state for every time slot the temporal pattern of the blinking LED is fully known. The formula to compute the state at an arbitrary point in time t for a cluster $c(i)$ is shown in 3.79. The function f is again used to compute the index of the time slot given a timestamp t and the cluster $c(i)$.

$$c(i).s(t) = s(f(t, c(i))) \tag{3.79}$$

K-Means Clustering of Temporal Pattern

Using the LED's period, this approach finds transitions having nearly the same relative temporal position within the LED's blinking pattern. This means the timestamp of the transition is nearly the same if we consider the blinking pattern's periodicity. Formula 3.80 illustrates the relative timestamp t_{rel} of a transition with time $t.t$ of a blinking pattern with period $c(i).p$.

$$t_{rel} = (t.t \bmod c(i).p) \tag{3.80}$$

The goal is to find accumulations of transitions having almost the same relative timestamp t_{rel}. Such accumulations indicate the existence of a cluster transition at a particular temporal position.

To find these accumulations based on the relative timestamps a K-Means clustering is used. K-Means clustering will produce a set of groups S where each group S_i has a list of relative timestamps t_{ij} assigned to it. Each group represents an accumulation of cluster transitions at a particular relative timestamp. These timestamps within a group are very close to each other since K-Means clustering minimizes the within-cluster sum of squares. The within-cluster sum of squares uses the relative timestamps t_j and the mean of the group μ_i. This is shown in formula 3.81. The n in $cost_n$ indicates that the cost was computed by considering n transitions.

$$cost_n = \arg_S \min \sum_{i=1}^{k} \sum_{t_j \in S_i} \|t_j - \mu_i\| \tag{3.81}$$

K-Means clustering is performed the same way as presented in section 3.6.3 using the iterative process to iteratively increment the number of groups k. The criterion for incrementing the number of groups k is the within-cluster sum of squares shown in formula 3.81. The sum has to be below a threshold c_{cost} illustrated in equation 3.82. The threshold t_{cost} is defined as the number of transitions n multiplied with the allowed error of 100 microseconds.

$$cost_n < t_{cost} = n * 100 \tag{3.82}$$

K-Means clustering has to be performed separately for the on and off states. The mathematical definition of the groups produced by K-Means clustering for the on states is shown in 3.83.

$$S_{on} = \{S_{on,1}, ..., S_{on,k}\}, \quad S_{on,i} = \{t_{i1}, ..., t_{im}, t_{ij}.s = on\} \tag{3.83}$$

By analysing these groups S_i the algorithm extracts those groups where the number of assigned relative timestamps exceeds a certain threshold $t_{confidence}$. The threshold $t_{confidence}$ is determined heuristically.

$$S_{on,confidence} = \{S_{on,i} : |S_{on,i}| > t_{confidence}\} \tag{3.84}$$

The resulting set of groups $S_{on,confidence}$ is used to reconstruct the original signal observed by the DVS. To compute the temporal position of the valid transitions we use simply the average of the timestamp of all transitions within a group $S_{on,i}$. The state of the transition corresponds to the state of the group. The resulting set of transitions is shown in formula 3.85.

$$T_{on} = \left\{ t_i : t_i.t = \frac{1}{|S_{on,i}|} \sum_{t \in S_{on,i}} t.t \wedge t_i.s = on \right\}, \ S_{on,i} \in S_{on,confidence} \tag{3.85}$$

The total set of transitions in the signal corresponds to the union of all transition having an on state T_{on} and off all transition having an off state T_{off}. This is shown in formula 3.86.

$$T_{signal} = T_{on} \cup T_{off} \tag{3.86}$$

Since the set of transitions T defines at which time the state of the blinking LED is changed the set T describes the behaviour of the LED at each point in time t. Formula 3.87 shows the formula to find the state of the cluster $c(i)$ at time t. We search for the closest transition t before the searched time $t_{searched}$ minimizing the temporal distance.

$$c(i).s(t_{searched}) = t_{min}.s, \ t_{min} = \arg\min_{t \in T} t_{searched} - t.t, \ t_{searched} - t.t \geq 0 \tag{3.87}$$

Grouping of Temporal Pattern

The above definition of K-Means clustering in section 3.6.4 is an expensive method. The problem with K-Means clustering is that it takes a set of cluster transitions $T = \{t_1, ..., t_n\}$ and groups all transitions within this set T. If a new transition t_{n+1} is available from the clusters transition history $c(i).th$ the clustering has to be repeated. In order to improve the efficiency of the algorithm we have to look for another solution which does not require performing K-Means clustering for every new transition. This method finds accumulations of transitions using a grouping approach.

A better approach would be if we assume that the new transition belongs to a new group S_{k+1} which contains only the new transition as an element. This would not increase the cost defined as the within-cluster sum of squares since a new group with one element has the element's value as its mean $\mu_{k+1} = t_{n+1}$. This is shown in equation 3.88

$$cost_{n+1} = \arg_S \min \sum_{i=1}^{k} \sum_{t_j \in S_i} \|t_j - \mu_i\| + \|t_{n+1} - \mu_{k+1}\| = \arg_S \min \sum_{i=1}^{k} \sum_{t_j \in S_i} \|t_j - \mu_i\| = cost_n \tag{3.88}$$

Since we have assumed that the solution based on n transitions satisfies the condition shown in 3.82 the solution defined in 3.88 satisfies the condition as well. According to the principle of K-Means clustering we search for the smallest number of groups satisfying the condition of the maximal allowed cost. The algorithm has therefore to check whether there is a solution based on fewer groups. The process is described in the following enumeration:

1. Mark the new group S_{k+1} as the changed group $S_{changed}$.
2. Find for the changed group $S_{changed}$ the nearest group S^* to it. This is shown in formula 3.89 where S represents the set of all groups and μ_i the mean of group S_i.

$$S^* = \arg_{S_i \in S} \min \|\mu_i - \mu_{changed}\| \tag{3.89}$$

3. Merge the two groups.
4. Compute the cost of the solution using the within-cluster sum of squares as shown in formula 3.88.
5. If the cost still satisfies the condition in equation 3.82 the algorithm marks the merged group as the changed group and goes back to step 2. Otherwise the searched number of groups k has been found.

Using the resulting set of groups the algorithm is able to compute the blinking pattern's transitions. This is done by using the mean μ_i of groups S_i as the temporal position. The set of transitions is computed using the approach presented in 3.6.4 for K-Means clustering.

Criterion	*Election*	*K-Means*	*Grouping*
Effort	low	high	low
Accuracy	discrete	continuous	continuous

Table 3.17.: *The table shows the advantages and drawbacks for each proposed solution to compute the LED's blinking pattern.*

Discussion

This section lists the advantages and drawbacks for each approach to find the best approach to extract the LED's blinking pattern. The list in shown in table 3.17. The criteria are described blow.

1. **Effort:** The expected computational effort of the method.
2. **Accuracy:** Describes the methods potential accuracy to compute the LED's blinking pattern.

K-Means clustering is of course the worst approach in terms of effort. The approaches Grouping and Election differ in their accuracy because the first one uses a discretization of the time domain whereas the other one works in a continuous time domain. To estimate the impact of the discretization on the resulting identified temporal pattern a further experiment is required to observe the impact in practice.

3.6.5. Extracting the Temporal Pattern's Histogram

The goal of this section is to derive a representation of the marker's blinking pattern based on histograms as discussed in section 3.2.3. The advantage of this representation is that it does not require the calculation of the blinking pattern's period. The histograms can be used as features to uniquely identify a blinking LED. The identification of the LED's blinking pattern using histograms has significant drawbacks as discussed in section 3.2.3. The classification of an LED based on its histograms is discussed in section 3.7.

The histogram represents the distribution of time spans between an on to an off event and vice-versa. It serves as a criterion for the identification of the LED's blinking pattern with the disadvantages discussed in section 3.2.3. Formula 3.90 shows the mathematical definition of the histogram $h_{on,off}$ where $\triangle t$ represents the measured time between an on to an off event. The histogram $h_{off,on}$ represents therefore the time distribution between off to on events.

$$h_{on,off}(\triangle t) = h(\triangle t) + 1 \tag{3.90}$$

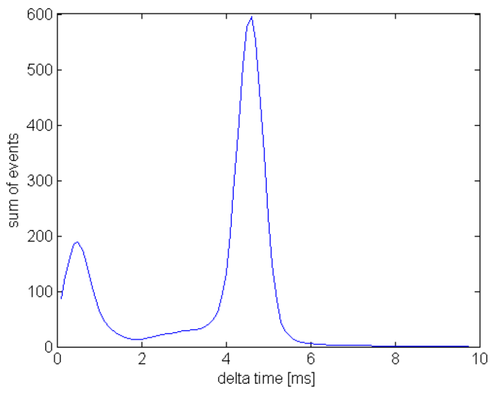
To reduce the fluctuation in the histogram caused by noise a Gaussian mixture model can be used. A Gaussian Mixture Model is a parametric probability density function represented as a weighted sum of Gaussian component densities. Gaussian Mixture Models are commonly used as a parametric model of the probability distribution of continuous measurements or features

[Rey08]. The Gaussian mixture model is based on a model where each sample contributes to the model. Thus it is more like a noise reduction method.

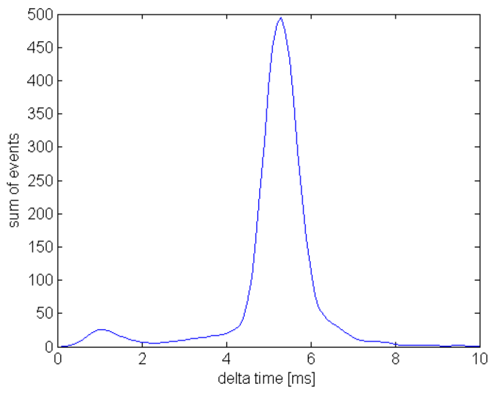
For our problem situation it is sufficient to reduce the impact of noise by using a Gaussian distribution for the definition of the histogram. Formula 3.91 shows the adapted definition of the histogram where h corresponds to the histogram and $h(t)$ to a particular bin of the histogram. The value of the bin is increased depending on the time between the two events μ and the desired standard deviation σ. The standard deviation σ depends on the resolution of the histogram and the noise within the observed time distribution.

$$h_{on,off}(t) = h(t) + \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(t-\mu)^2}{\sigma^2}\right)} \tag{3.91}$$

Figure 3.23 represents the histograms of the time spans between an off to an on-event and vice-versa. Using a Gaussian distribution to assign weights to the bins of the histogram should enhance the robustness against noise.



(a) The figure illustrates the histogram of the measured time distributions between on to off events.



(b) The histogram of the time distributions between off to on events.

Figure 3.23.: *The figures illustrate the generated histograms using a Gaussian distribution to assign weights to the bins.*

Cluster-Centered Histogram

In this approach a map is used to determine the time distributions between events. This map is positioned relative to the spatial center of the cluster and stores the state of the last event and the corresponding timestamp. The relative positioning helps to reduce the noise caused by a moving blinking LED. Two maps are used for the computation of the histogram. The first map s stores the state for each relative position. This is shown in formula 3.92 where $c(i).x$ corresponds to the clusters position and $e.x - c(i).x$ to the position of an event e in the map relative to the clusters position. The map s therefore stores the last observed state at each relative position.

$$s(e.x - c(i).x, e.y - c(i).y) = e.s \tag{3.92}$$

Similar to the above definition an additional map t is used to store the timestamp of the last change of the state for each position in the map. The formula is shown in 3.93. The timestamp is updated whenever the new event changes the stored state in the map s at the corresponding relative position.

$$t(e.x{-}c(i).x, e.y{-}c(i).y) = \begin{cases} e.t & \text{if } s(e.x - c(i).x, e.y - c(i).y) \neq e.s \\ s(e.x - c(i).x, e.y - c(i).y) & \text{otherwise} \end{cases} \tag{3.93}$$

To compute the intervals between an on- and the next off-event and vice-versa the two maps can be used. The map s is just used to determine if there is a change of the state while t is used to compute the interval. To compute the time span between two events $\triangle t$ the algorithm uses the map t as shown in formula 3.94.

$$\triangle\, t = e.t - t(e.x - c(i)x, e.y - c(i).y) \tag{3.94}$$

Transition History

In order to have a stable measurement of the different intervals the transition history presented in section 3.6.1 can be used. The usage of this representation helps to reduce fluctuations caused by noise since it combines the information of all assigned events. In this approach the algorithm has to take every pair of transitions and compute the difference in their timestamps. This is shown in equation 3.95. Here $c(i).th$ represents the clusters $c(i)$ ordered transition history.

$$\triangle\, t = t_{i+1} - t_i,\ t_i, t_{i+1} \in c(i).th \tag{3.95}$$

The interval $\triangle t$ represents the time span between an on to an off event and vice-versa. This method uses the same representation of the blinking pattern as shown in section 3.6.5. The usage of the cluster's transition history $c(i).th$ as input allows to reduce the noise within the measured time spans.

Discussion

Although the approaches using the histograms to identify the LED's blinking patterns worked in the algorithms first version it is not implemented in the algorithms final version. The reason for this are the already discussed problems with histograms in section 3.2.

3.7. Classification of a Blinking LED

This section describes how the LED's blinking pattern is classified. We assume now that the classification algorithm knows some blinking patterns $r(j)$ where $r(j)$ represents the j-th registered blinking pattern. The registered blinking patterns $r(j)$ are either represented by a temporal pattern as presented in section 3.6.4 or by a histogram as shown in section 3.6.5. The cluster classification's goal is to find for a cluster $c(i)$ the best matching registered blinking pattern $r(j)$.

Figure 3.24 illustrates the process to assign a cluster to one of the predefined patterns. For each unclassified cluster the algorithm has to find the best matching predefined blinking pattern.

1. For each unclassified cluster the classification algorithm computes the quality of the features used for the classification. If the quality is not satisfied the algorithm has to collect more data before a classification can be performed.
2. If the features quality is satisfied the algorithm computes for each predefined pattern j a cost to assign the cluster i to the pattern j.
3. If there is a satisfying solution the algorithm assigns the cluster to the best matching pattern j^*, otherwise the algorithm registers the clusters pattern as a new observed pattern which was unknown before.

The matching based on cost functions can either be done by using histograms as described in section 3.6.5 or by the blinking pattern itself as presented in section 3.6.4.

3.7.1. Evaluate Feature Quality

Before a classification can be performed the algorithm has to evaluate the quality of the features used for the classification.

If the algorithm uses histograms as defined in section 3.6.5 to perform the classification the evaluation of the histograms quality is straightforward: the quality is satisfied if the histogram contains a certain amount of samples. The mathematical notation is shown in formula 3.96 where $N_{threshold}$ defines the minimal number of required samples and $c(i).h$ the clusters histogram.

$$\sum_{j=1}^{N} c(i).h(j) > N_{threshold} \tag{3.96}$$

On the other hand if the classification is based on the LED's blinking pattern as presented in section 3.6.4 the algorithm uses a more sophisticated method to evaluate the features quality. In this case the algorithm correlates the blinking pattern with the observed transition history. If the cross-correlation is above a certain threshold $Q_{threshold}$ the quality is considered as sufficiently high. This is shown in equation 3.97 where $c(i).s$ represents the clusters blinking pattern and $c(i).th$ its transition history. The threshold $Q_{threshold}$ is set to 0.8.

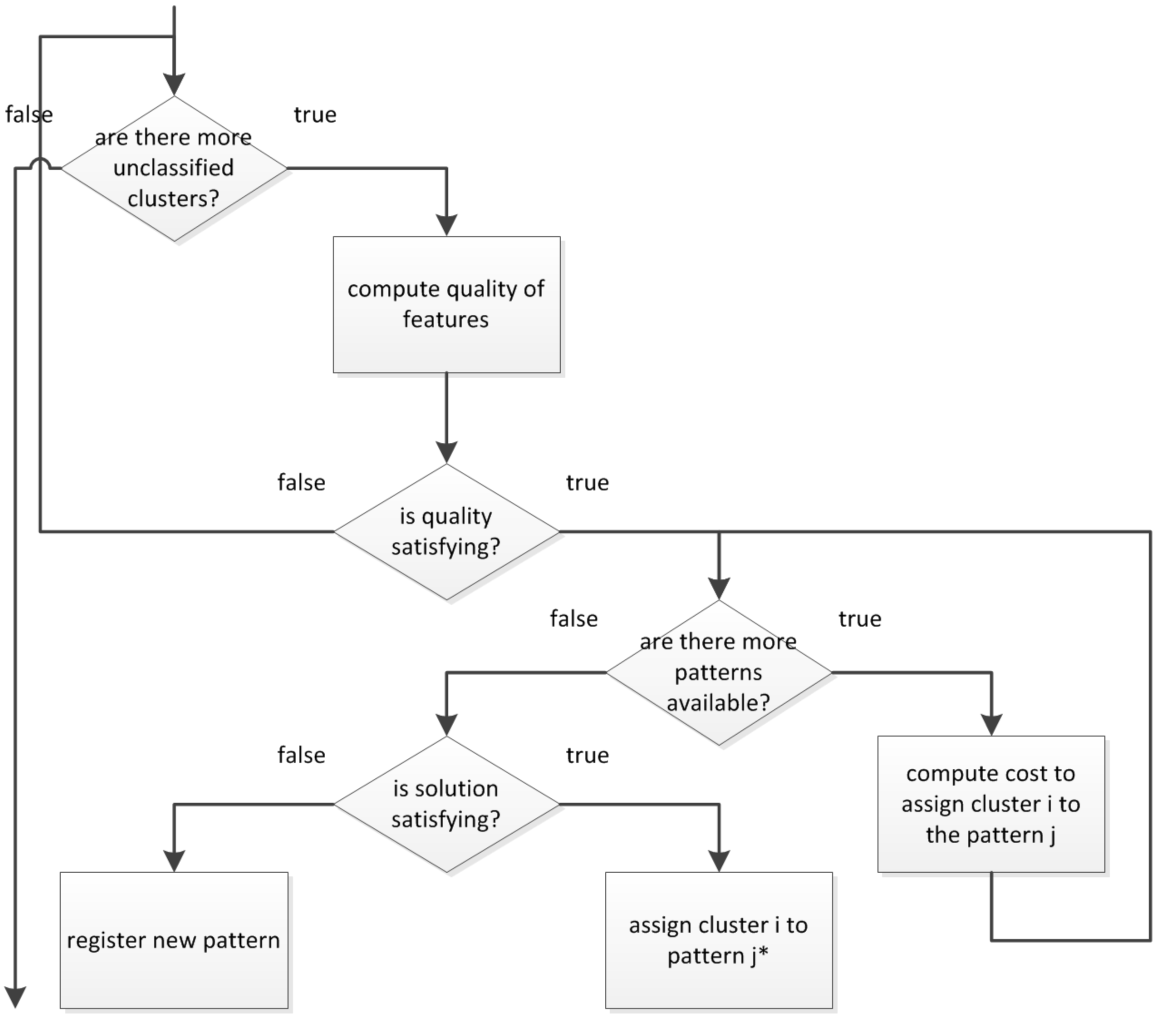

false
true
are there more unclassified clusters?
compute quality of features
false
true
is quality satisfying?
false
true
are there more patterns available?
false
true
is solution satisfying?
compute cost to assign cluster i to the pattern j
register new pattern
assign cluster i to pattern j*

Figure 3.24.: *The figure shows the process to find the best assignment for a cluster $c(i)$ to a predefined blinking pattern $r(j)$ using a cost functions.*

$$\max_{\tau} \int_{-\infty}^{\infty} c(i).s(t+\tau)c(i).th(t)dt > Q_{threshold} \tag{3.97}$$

Considering the approach presented in section 3.6.3 the computational effort of the correlation function can be reduced by using formula 3.62.

3.7.2. Different Methods for the Classification

The classification of a cluster is based on the same principle as described in section 3.4. To have a generic approach allows evaluating different metrics for the comparison of the LED's blinking pattern the principle of cost functions is again used to have an ordering on the possible classifications.

The classification uses the threshold based approach defined in section 3.4.1. If the best possible classification does not satisfy a rejection function the algorithm registers the blinking pattern as a new one. The approach is chosen because the quality of the feature used for the classification is continuously monitored. It is therefore not necessary to have a sophisticated method to weight the different assignments since the classification algorithm reduces by its design the influence of noise.

3.7.3. Cost Functions to Evaluate Classifications

Again we want to apply the concept of cost function to compare the observed blinking pattern with the registered ones. As defined already in section 3.4.2 the concept of cost function requires that the best solution minimizes the cost. This section presents now different approaches to compare the pattern of a blinking LED. The input for the cost function could either be a histogram as described in section 3.6.5 or the blinking pattern itself as presented in section 3.6.4.

Kullback-Leiber Divergence

First of all we describe a possible cost function to compare histograms. The Kullback-Leibler (KL) divergence is a fundamental equation of information theory that quantifies the proximity of two probability distributions. It quantifies in bits how close a probability distribution $p = \{p_i\}$ is to a model (or candidate) distribution $q = \{q_i\}$ [SK07]. The KLs definition is shown in formula 3.98. The KL divergence is only defined if p and q both sum to 1 and if $p(i) > 0$ for any i such that $q(i)$. If the quantity $0\ log_2(0)$ appears in the formula, it is interpreted as zero.

$$D_{KL}(p\|q) = \sum_i p_i \log_2 \left(\frac{p_i}{q_i} \right) \tag{3.98}$$

Adapted to our problem situation where the algorithm has to compare the clusters transition histogram $c(i).h$ with the histogram of a registered blinking pattern $r(j).h$ we get the following definition for the cost function visible in formula 3.99.

$$R(j|i) = D_{KL}(c(i).h\|r(j).h) = \sum_b c(i).h(b) \log_2 \left(\frac{c(i).h(b)}{r(i).h(b)} \right) \tag{3.99}$$

Mixture Model

In statistics, a mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data-set should identify the sub-population to which an individual observation belongs. This concept is visualized in figure 3.25.

The feature values of the candidate distribution depends on the Gaussian mixture model and are computed as represented in formula 3.100. In the formula g_α corresponds to a Gaussian density and y to a feature which is in our case a value of the histogram. The probability $p(\alpha|v)$

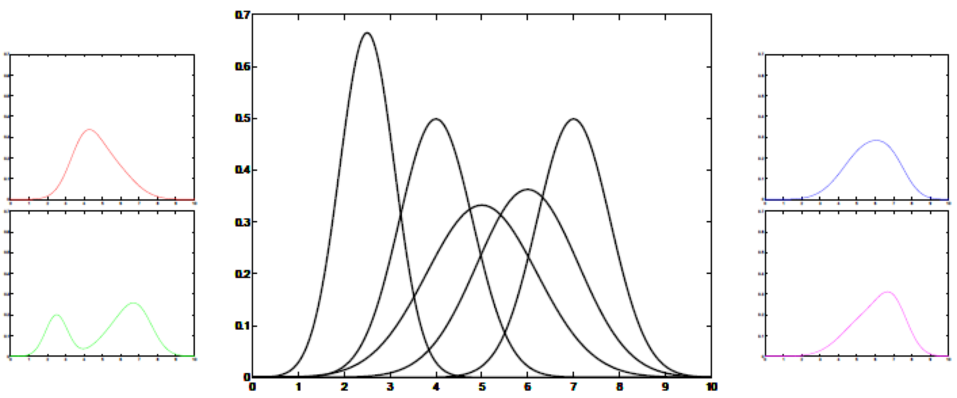


Figure 3.25.: *The figure illustrates a Gaussian mixture model. The Gaussian distributions in the middle are the candidate distributions. The other figures represent the mixture densities composed of the candidate distributions using different weights.*

is the mixture weight υ [Buh11]. A Gaussian mixture model is therefore a weighted sum of m component Gaussian densities.

$$p(y|\upsilon) = \sum_{\alpha=1}^{n} p(\alpha|\upsilon) g_\alpha(y) \tag{3.100}$$

We wish to estimate the parameters of the Gaussian Mixture Model, which in some sense best matches the distribution of the feature vectors. There are several techniques available for estimating the parameters of a Gaussian Mixture Model. By far the most popular and well-established method is maximum likelihood (ML) estimation. The aim of ML estimation is to find the model parameters which maximize the likelihood of the Gaussian Mixture Model given the data [Rey08].

The corresponding cost function based on the mixture models uses the negative log likelihood. The mixture model minimizing the cost function represents best the observation and therefore has to be selected by the algorithm. Formula 3.101 shows the cost function where $c(i)$ represents the given cluster and h its histogram.

$$R(j|i) = -\log \prod_b c(i).h(b) \left(\sum_{\alpha=1}^{m} p(\alpha|j) g_\alpha(b) \right) = -\sum_b \log \left(c(i).h(b) \left(\sum_{\alpha=1}^{m} p(\alpha|j) g_\alpha(b) \right) \right) \tag{3.101}$$

Cross Correlation

The cross correlation is used to measure the match or similarity between two different waveforms. It also has applications in pattern recognition [WN96] [Chi03]. For functions f and g the cross-correlation for an offset τ is defined as shown in formula 3.102.

$$(f \star g)(\tau) = \int_{-\infty}^{\infty} f(t)g(t+\tau)dt \tag{3.102}$$

This cost function uses as input the LED's identified blinking pattern as presented in section 3.6.4. The cost function looks at every point in time t if the observed blinking pattern of the cluster is the same as the registered blinking pattern used as the reference model.

Formula 3.103 represents the cost function to compute the cost to assign a cluster i to a registered blinking pattern j. The function uses an indicator function $\mathbb{I}$ to increase the value of the cost function it the two states $c(i).s(t)$ and $r(j).s(t)$ are not equal. The variable o is an offset in time to synchronize the two signals. Since the true offset o is unknown the algorithm has to compute the correlation for all possible offsets $candidate$ and taking the one that minimizes the cost function.

$$R(j|i) = \min_{o \in candidate} \left(\int_{0}^{end} \mathbb{I}_{\{r(j).s(t) \neq c(i).s(t+o)\}} dt \right) \tag{3.103}$$

The computational effort of this correlation can be reduced by using formula 3.62 as presented in section 3.6.3. The algorithm computes the correlation only at offset o's allowed by the temporal patterns. This is shown in formula 3.104 where $c(i).s$ represents the clusters temporal pattern. The temporal pattern is an ordered set of transitions t_i with timestamps $t_i.t$.

$$candidate = \{t_i.t : t_i \in c(i).s\} \tag{3.104}$$

Discussion

According to the discussion in section 3.6.4 we came to the conclusion that it makes sense to classify the blinking LED's based on their blinking pattern and not on their histograms. The only cost function developed to measure the similarity of a blinking pattern is the cost function based on the cross correlation presented in 3.7.3. This is therefore the cost function used to classify a blinking LED.

A cluster is classified if the cost of an assignment is below 0.2. Therefore the threshold t_{cost} defined in formula 3.26 is set to 0.2.

3.7.4. Experiments for the Classification of a LED

In this section we analyse the algorithms ability to classify a blinking LED. Since the classification process relies fully on the identified blinking pattern the experiments have to quantify the impact of different parameters on the quality of the clusters extracted blinking pattern. This section describes the experiments executed to estimate the impacts. Besides the quality of the algorithm the experiment summarizes also results about its performance.

The goal of the experiment is to determine the best method to extract the period of the LED's blinking pattern out of the methods presented in section 3.6.3 and the best method to extract the blinking pattern itself as presented in section 3.6.4. The methods used in the experiment are listed in the following enumeration.

1. **Voting:** The method uses an election to extract a discretized version of the blinking LED's temporal pattern. The method is presented in more detail in section 3.6.4.
2. **Transition:** Extracts a continuous version of the LED's blinking pattern. More details for this approach are available in section 3.6.4.
3. **Correlation:** The method estimates the blinking LED's period using the autocorrelation function as described in section 3.6.3.
4. **Fourier:** Uses the Fourier Transformation to estimate the blinking LED's period. The approach is presented in section 3.6.3.

To evaluate the different approaches a set of criterions has to be defined. All approaches are evaluated by considering the following criterions:

1. **Recognition time:** The time in microseconds it takes to identify the LED's blinking pattern. As start time we take the timestamp of the clusters creation. The recognition time is an important metric because the blinking LED has to be identified before it leaves the DVS's field of view.
2. **Relative error / Signal mismatch:** The quantity measures the error relative to the length of the LED's identified blinking pattern $c(i).s$. The error is defined as the difference between the temporal positions of the transitions with the expected position. This is shown in formula 3.105 where the extracted transitions are represented by the set $c(i).s$. For each transition t in the set the metric computes the difference of its timestamp $t.t$ with the expected temporal position of the transition with the help of a function t_{true}. This error is then divided by the extracted period $c(i).p$. This division is important since a signal with a small period has to be more accurate than one with a large period if we want to compare them using the cross correlation.

$$e_{relative} = \frac{\sum_{t \in c(i).T} |t.t - t_{true}(t.t)|}{c(i).period} \tag{3.105}$$

As presented in section 3.7.1 the classification algorithm monitors the quality of the features used for the classification. As consequence the experiments have shown that the blinking pattern's relative error is negligible since the algorithm searches as long as there is no satisfying solution. The recognition time on the other hand contains implicitly information about the noise

Parameter	*Quantity*
Distance [cm]	50-400

Table 3.18.: *Lists the parameters for the experiment with a stationary blinking LED at different distances.*

within the blinking pattern identification process since the recognition time increases with an increasing noise level.

Effect of Varying Marker Distance on Recognition Time

In this experiment the distance between the DVS and the blinking LED was continuously increased from 50cm to 400cm. The experimental setup corresponds to the one for a stationary LED presented in section 3.1.1. The changed parameters of the experiment are shown in table 3.18.

First of all we analyse the effect of an increasing distance on the recognition time. An increasing distance means essentially that the blinking LED is observed by fewer pixels. This decreases of course the signal to background ratio and reduces the redundancy within the events. Both factors decrease the algorithm's ability to handle noise. Therefore we would expect that the algorithm needs more time to recognize the blinking LED's blinking pattern which leads to an increasing recognition time. Figure 3.26 illustrates the decreasing size of the LED cluster radius with an increasing distance.

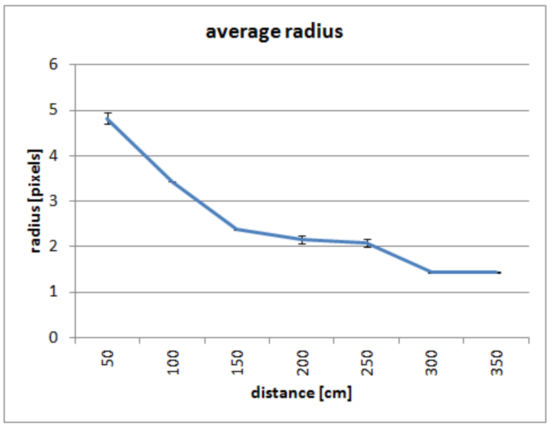
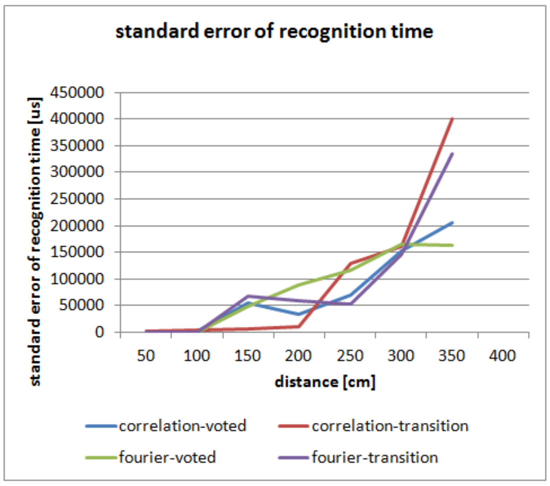


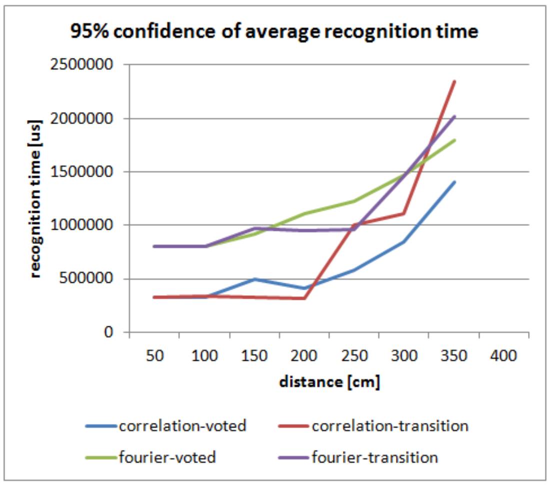
Figure 3.26.: *In this figure the radius of the observed blinking LED is shown in relation to the distance.*

Figure 3.27 summarizes the results for the measured recognition times. It is clearly visible that the standard error of the recognition time increases with an increasing distance and the corresponding confidence interval as well. Furthermore the approach using the Fourier transformation to estimate the period of the blinking pattern always takes longer time than the one using the autocorrelation function.

To conclude we can say that an increasing distance increases the noise level as expected. This leads to a higher recognition time since the algorithm has to handle this noise. The Fourier



(a) *The figure shows the standard error of the measured recognition times with an increasing distance.*



(b) *Shows the 95% confidence interval for the average recognition time. If we would repeat the experiment we would have a chance of 95% to measure a recognition time smaller than the one visible in the figure.*

Figure 3.27.: *The figures illustrates the statistics of the measured recognition times with an increasing distance.*

transformation has the problem of the sampling process. Before the transformation is performed the algorithm has to sample the signal for a fixed amount of time. Since the autocorrelation function is able to work with a very small time window it leads to a shorter recognition time.

To explore the effects of the different methods a $2^k\ r$ factorial design with replication was used [Jai91]. This method helps to determine the effect of k factors, each of which has two alternatives or levels. It helps in sorting out factors in the order of their impact. For the factorial design the results of the experiment at a distance of 250cm was used since all approaches work well at this distance. The experiment shows that a significant amount, 21.12%, of the recognition times variation is explained by the method to extract the period. It has therefore an impact on the measured recognition times whether we use the Fourier transformation or the autocorrelation function to estimate the period of the observed blinking pattern. According to the experiment the method using the autocorrelation function leads to shorter recognition times.

Effect of Varying Duty Cycle on Recognition Time

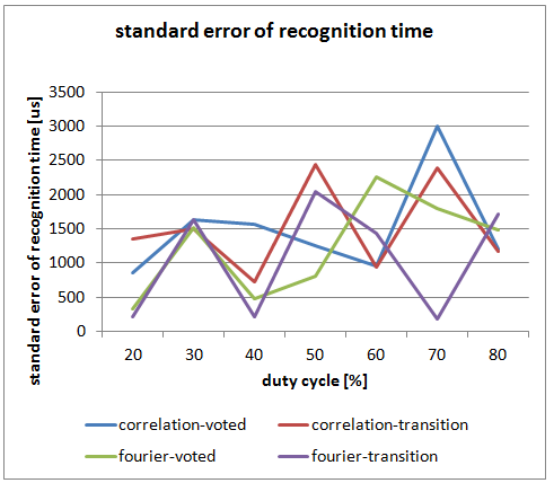
The next experiment should answer the question whether the duty cycle of the signal (here a square wave) has an impact on the algorithm. To answer this question the duty cycle was continuously increased during the experiment. The experimental setup corresponds to the one for a stationary LED presented in section 3.1.1. The changed parameters of the experiment are shown in table 3.19.

First of all we will analyse the impact of the duty cycle on the recognition time. Since the duty cycle only changes the temporal position of the transition and does not affect the noise level in the data we expect that the duty cycle has no significant impact on the recognition time.

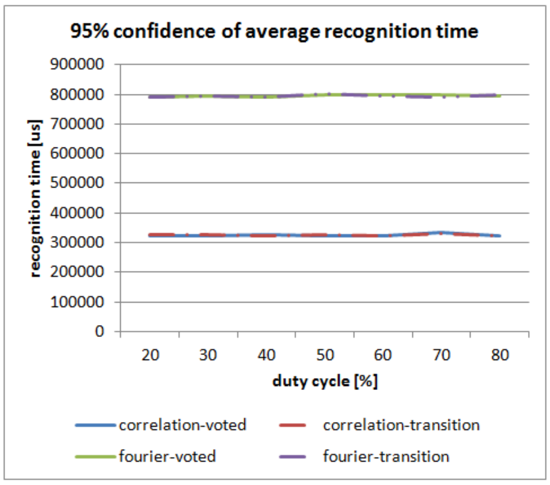
Parameter	*Quantity*
Duty Cycle [%]	20-80

Table 3.19.: *List the parameters for the experimental setup with an increasing duty cycle.*

The results of the experiment are shown in figure 3.28. It is clearly visible that the standard error is very small for all duty cycles and the corresponding 95% confidence interval is almost constant.



(a) The graph shows the standard error of the measured recognition time with an increasing duty cycle.



(b) Shows the corresponding 95% confidence interval for the average recognition time.

Figure 3.28.: *The figures illustrate the statistics of the measured recognition times with an increasing duty cycle.*

As expected the duty cycle has no impact on the recognition time of the algorithm. In this experiment we can also clearly see the different measured recognition times for the two approaches to estimate the period of the blinking pattern. The Fourier transformation is always higher due to the sampling period.

Effect of Varying Frequency on Recognition Time

The purpose of this experiment is to analyse the performance and accuracy of the algorithm for different frequencies. The experiment is based on the setup described in section 3.1.1 using a stationary LED. The frequency was continuously increased during the experiment as shown in table 3.20.

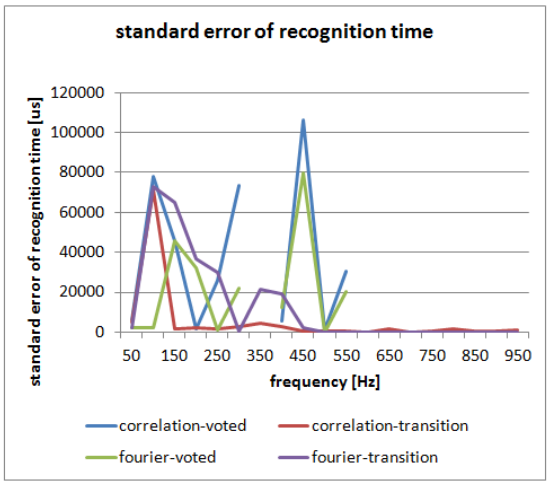
For the experiment we expect that the recognition time decreases with an increasing frequency because the periodicity of the signal. There will be more repetitions of the LED's blinking pattern if we observe a constant time window.

Figure 3.29 shows the standard error of the measured recognition times and the corresponding

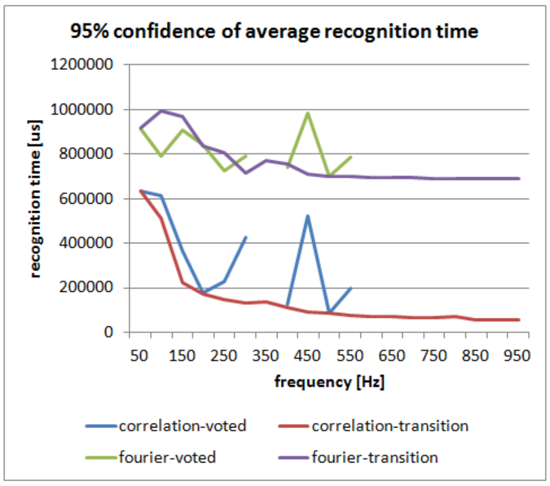
Parameter	*Quantity*
Frequency [Hz]	50-950

Table 3.20.: *Lists the default values for the experiment with a stationary blinking LED and an increasing frequency.*

confidence interval for the average recognition time. It is obvious that the transition based approach to extract the signal works even for very high frequencies while the method using the voted approach seems to be unstable with an increasing frequency. Furthermore the voted approach has several problems at particular frequencies like the one at 350Hz. The overall tendency for the recognition time is clear. It decreases for an increasing frequency.



(a) Illustrates the standard error of the recognition time with an increasing frequency of the LED's blinking pattern.



(b) Represents the corresponding 95% confidence interval for the average of the measured recognition time.

Figure 3.29.: *The two figures represent the standard error and the corresponding 95% confidence interval of the measured signal mismatch with an increasing frequency.*

The source of the voted approach problems lies in the discretization of the time. The discretization produces a quantization error which increases the signal mismatch. This is not a problem if the observed signal has a low frequency since the signal mismatch measures the error relative to the period of the signal. But if the frequency is getting higher the corresponding period is getting smaller. Because of the shorter period the signal mismatch increases for a constant error caused by the quantization. Since the signal mismatch is a criterion to estimate the blinking patterns quality as described in section 3.7.1 the condition will never be satisfied and the algorithm therefore tries to further optimize the signal. To overcome this problem the resolution of the time has to be adapted.

The recognition time decreases with pattern rate because a high frequency leads to much more repetitions of the signal in the same time interval than a low one. Because of that the algorithm is able to extract the LED's blinking pattern significant faster.

To determine the impact of the chosen methods the factorial design model is used at a frequency

Parameter	*Quantity*
Velocity [pixel / sec]	54.15 - 795.64
Frequency [Hz]	200

Table 3.21.: *The table lists the values for the experiment to observe a moving LED with an increasing frequency.*

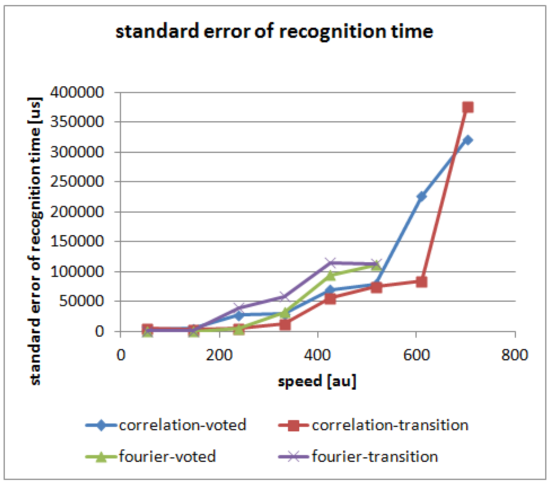
of 200Hz. According to the design model 94.42% of variation in the measured recognition times is explained by the method to extract the period. It is therefore highly recommended to use the autocorrelation based approach to estimate the period.

Effect of Marker Velocity on Recognition Time

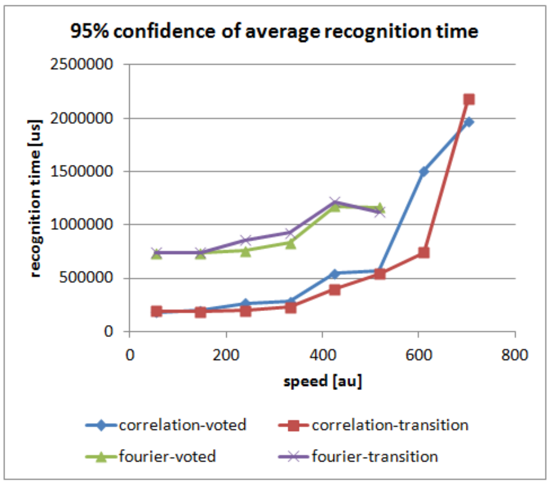
This experiment analyses the impact of noise caused by a movement of the observed blinking LED. The experimental setup corresponds to the one presented in section 3.1.2 with the robot-arm. The LED was observed with an increasing velocity of the arm in this experiment. The changed parameters for the experiment are shown in table 3.21.

For the experiment we assume that the recognition time increases with an increasing velocity. This is because of the decreasing signal to background ratio described in section 3.2.2.

Figure 3.30 shows the standard error of the measured recognition times and the corresponding 95% confidence interval for the average recognition time. As visible the recognition time increases with an increasing velocity as well as the corresponding standard error.



(a) The graph represents the standard error of the measured recognition times with an increasing velocity of the blinking LED.



(b) The figures shows the 95% confidence interval for the measured average recognition times.

Figure 3.30.: *The two figures show the statistics of the measured recognition times with an increasing velocity of the blinking LED.*

The experiment has shown that the algorithm needs more time to identify the blinking pattern

Parameter	*Quantity*
Velocty [pixel / sec]	54.15 - 795.64
Frequency [Hz]	100, 200, 300

Table 3.22.: *Lists the default values for the experiment to observe a moving LED at different blinking frequencies.*

because the signal to background ratio decreases. The reason for the higher background activity is due to the events generated by the moving blinking LED. As faster the LED moves as more events are generated by the movement while the events generated by the blinking of the LED remain constant. This explains also the increasing standard error since the higher noise level makes the algorithm unstable.

Since the autocorrelation function to extract the period of the signal provides such better results in terms of recognition time than the Fourier transformation we have to further investigate the robustness of the method to extract the LED's blinking pattern. The parameters for the experiment are shown in table 3.22.

For this experiment we expect that an increasing frequency would improve the signal to background ratio. This means essentially the recognition time decreases with an increasing frequency since the algorithm is much more robust to background activity. Furthermore the algorithm should also be able to recognize the blinking pattern at very high velocities if the frequency is sufficiently high.

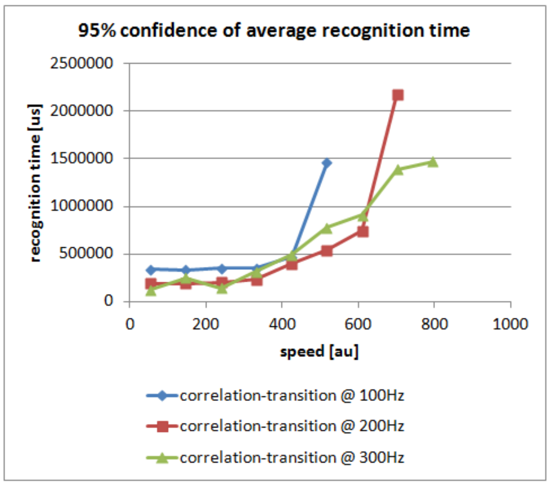
The statistics of the measured recognition times in the experiment are shown in figure 3.31. The two graphs are showing that for both approaches an increasing frequency leads to a decreasing recognition time. It is also visible that the algorithm is not able to identify the blinking pattern for a fast moving LED if the LED's frequency is too low. If the LED's frequency increases the algorithm is also able to identify the blinking pattern for these fast moving LEDs.

The experiment has shown that the algorithm behaves as expected. A high frequency allows identifying the LED's blinking pattern also for fast moving LEDs. It is obvious that there is a relation between the frequency and the LED's velocity.

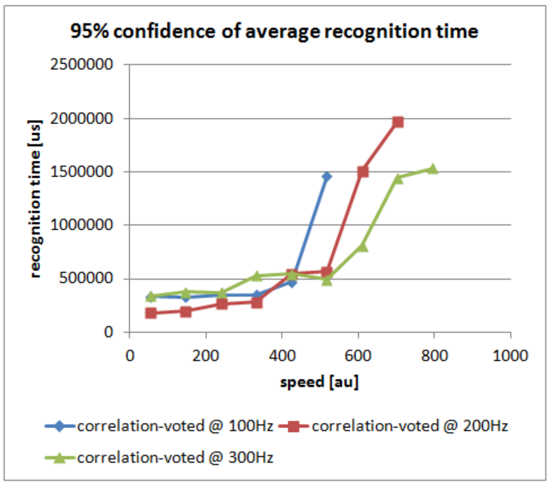
Discussion

In the experiments we have seen that all approaches work. The correlation function seems to be the better method to extract the period of the signal since the experiments have shown that it is more robust to noise.

The methods to extract the blinking pattern of the blinking LED works as expected. There was an exception at the experiment with an increasing frequency. There we have seen that the transition based approach works much better since there is no discretization of the time domain. With a proper adaptation of the discretization the voted approach should be able to achieve similar results like the one observed for the transition based. Such an adaptation of the discretization can be done if we would consider the extracted period of the LED's blinking



(a) Shows the 95% confidence interval for the average recognition time using the correlation function to extract the period of the signal and the discrete voted approach to extract the blinking pattern.



(b) Shows the same confidence interval as in figure 3.31(a) using the transition based approach to extract the LED's blinking pattern.

Figure 3.31.: *The two figures show the confidence interval for the average recognition times using the correlation function with different frequencies of the blinking pattern.*

pattern for the division of the time domain.

Another factor that significantly influences the quality of the signal extraction process is the extraction of the transition history. Since the signal is based on the results of the transition history the signal is basically as good as the quality of the history although the algorithm is able to handle a certain level of noise within the transition history. Therefore a tracker to track the blinking LED should additionally improve the quality of the transition history. There are two factors that have to be considered.

1. **Assignment** Of course the assignment of events to a cluster is a major source of noise. If there is a wrong assignment of an event the event can directly be seen as noise. Therefore a tracker should improve the reliability of the assignment of these events. This can either be done by adding new cost functions considering new measurements or by improving the existing cost functions.

2. **Size** The size of the object is used to determine how many pixels have to change the state in order to change the state of the cluster. A large size of the cluster means that there are a lot of pixels assigned to the cluster which leads to the conclusion that to change the state of the cluster a lot of the assigned pixels have to change the state at the same time. In contrast to that a small cluster will change its state just because of a small amount of changing pixels. This will lead directly to a higher noise level. A tracker should therefore extract the correct size of the cluster to improve this decision.

According to the experiments the algorithm will use the approach using the correlation function to find the blinking LED's period and the transition based approach to find the corresponding blinking pattern.

3.8. Tracker Combining Spatio-Temporal Information

This section describes the final implementation of the tracker. The goal of the tracker is to improve the assignment of events to the correct cluster by considering all available information. The tracker has to combine the spatial and temporal information of an event. If the blinking pattern of an LED is known the algorithm is able to use the pattern to make additional temporal constraints. An algorithm to track a blinking LED has therefore to distinguish the two cases whether the algorithm knows the temporal pattern or not. This situation is illustrated in figure 3.32.

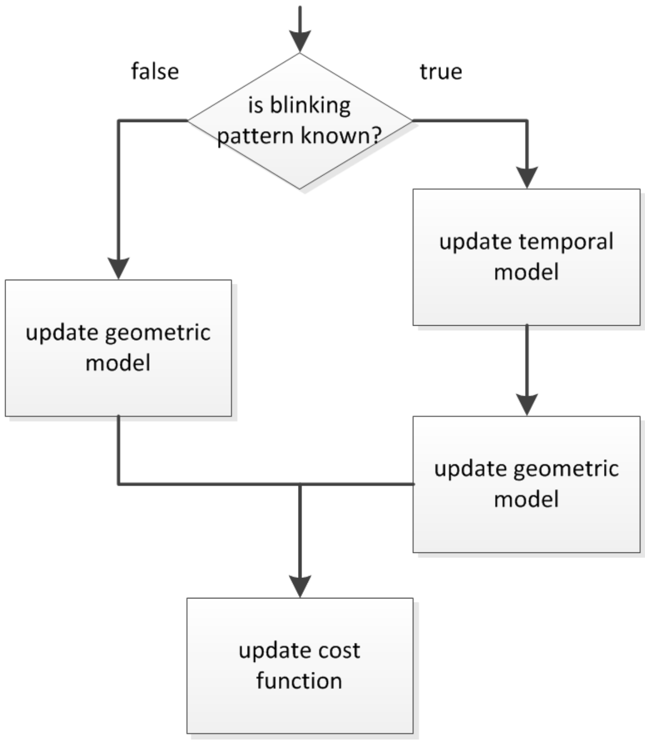


Figure 3.32.: *The diagram shows how the tracking algorithm updates its cost function for the event assignment depending on whether the LED's blinking pattern is known or not.*

The geometric model of the tracker can be updated in any case. This model includes the prediction of the position of the cluster, its velocity and other geometric properties. The model representing the temporal pattern can only be used if the tracker knows the pattern of the observed blinking LED. Having the temporal information the tracker is able to use other cost functions considering also the temporal occurrence of events.

3.8.1. Spatio-Temporal Cost Functions for Event Assignment

The tracker uses a different approach for the assignment of events as presented in section 3.4. Since the assignment is based on cost function the tracker just has to define a new cost function while the rest of the assignment process remains the same. These new cost functions used by the tracker are presented in this section.

Spatial Prediction

The tracker extends the cost function for the spatial closeness presented in section 3.4.2. Formula 3.106 shows the cost function using the clusters predicted position $c(i).x(e.t), c(i).y(e.t)$ at the events timestamp $e.t$. Here D is representing a function to measure the dissimilarity.

$$R_{spatial}(i|e) = D\left(\sqrt{(e.x - c(i).x(e.t))^2 + (e.y - c(i).y(e.t))^2}\right) \tag{3.106}$$

To measure the dissimilarity D a quadratic function is used.

Temporal Prediction

Once the algorithm was able to identify the LED's blinking pattern the tracker is able to use the pattern to make additional constraints. The algorithm is able to predict the occurrence of events generated by a particular LED. This information can be used to define a robust cost function by using this additional information to have a reliable assignment of events to the clusters. In order to use the identified blinking pattern of the LED the algorithm has to synchronize the observed cluster in order to track the phase of the blinking LED.

This approach uses the tracked phase to compute a temporal distance between the predicted temporal position of an event and the measured one. The temporal distance can then be computed by computing for an event e the minimal temporal distance by considering its type $e.type$. This would correspond to the search of the best matching transition. The mathematical notation is shown in formula 3.107 where $e.t$ represents the measured timestamp of the event, $c(i).s$ the clusters blinking pattern, t a transition of the pattern $c(i).s$ and $t.t$ the transitions timestamp. The properties $c(i).phase$ and $c(i).period$ are the phase and period of the observed blinking pattern $c(i).s$.

$$d_{temporal} = \min_{t \in \{t : t \in c(i).s \wedge t.s = e.type\}} |(e.t - c(i).phase \; \% \; c(i).period) - t.t| \tag{3.107}$$

This temporal distance $d_{temporal}$ can directly be used to measure a dissimilarity which is then used as a cost function. Formula 3.108 illustrates the cost function considering the temporal distance $d_{temporal}$ where D represents a function which represents a measure for the dissimilarity.

$$R_{temporal}(i|e) = D(d_{temporal}) \tag{3.108}$$

Spatio-Temporal Closeness

The trackers goal is to combine the events spatial and temporal information provided by the DVS. Using different cost functions the algorithm is able to compute a spatial cost $R_{spatial}(i|e)$ as well as a temporal cost $R_{temporal}(i|e)$ to assign an event e to a cluster i.

To combine the spatial and temporal costs the algorithm computes probabilities out of the two costs. As probability model an exponential distribution was used for the computed costs. The reason for this choice is that the exponential distribution maximizes the entropy for a given mean μ if there are no further information about the distribution [Buh11]. The model is shown in formula 3.109. Here μ represents the distributions mean, x a particular value and $p(x)$ the probability of value x.

$$p(x) = \frac{1}{\mu} e^{\left(-\frac{x}{\mu}\right)} \tag{3.109}$$

The probability density function $p(x)$ defined in formula 3.109 is used to derive the cumulative distribution function $F(x)$ (CDF). The CDF computes the probability that a real-valued random variable with a given probability distribution will be found at a value less than or equal to x. The definition is shown in formula 3.110.

$$F(x) = \int_{-\infty}^{x} p(t)dt = \begin{cases} 1 - e^{\left(-\frac{x}{\mu}\right)}, & \text{if } x \geq 0 \\ 0, \text{otherwise} \end{cases} \tag{3.110}$$

The CDF is used to compute the probability that a measured value x is found in the interval $[x, \infty]$. By inserting the spatial cost $R_{spatial}(i|e)$ into the cumulative distribution function $F(x)$ the algorithm directly gets the corresponding probability that the event e belongs to the cluster i considering only the spatial cost. This is shown in formula 3.111 where $\mu_{spatial}$ corresponds to the parameter for the expected average spatial cost and is set to 2.

$$p_{spatial}(i|e) = 1 - F_{spatial}(R_{spatial}(i|e)) = 1 - \begin{cases} 1 - e^{\left(-\frac{x}{\mu_{spatial}}\right)}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.111}$$

The same formula can be adapted to compute $p_{temporal}(i|e)$ by using the temporal cost $R_{temporal}(i|e)$ and the corresponding expected average of the costs $\mu_{temporal}$. The average cost $\mu_{temporal}$ is set to 200. To combine the two probabilities a straightforward approach was used. The two probabilities are multiplied together. This is shown in 3.112.

$$p(i|e) = p_{spatial}(i|e) * p_{temporal}(i|e) \tag{3.112}$$

The assignment for the event requires a cost rather than a probability. Using the negative log-likelihood the cost can directly be computed out of the probability as shown in formula 3.113.

$$R(i|e) = -\log p(i|e) \tag{3.113}$$

Method	*Formula*
Euler	$x(t+h) = x(t) + h * v(t)$
Heun	$x(t+h) = x(t) + \frac{h}{2}\left(v(t) + v(t+h)\right)$
Symplectic-Euler	$v(t+h) = v(t) + h * a(t)$ $x(t+h) = x(t) + h * v(t+h)$
Runge-Kutta	$k_1 = v(t)$ $k_2 = v\left(t + \frac{1}{2}h\right)$ $k_2 = v\left(t + \frac{1}{2}h\right)$ $k_4 = v(t+h)$ $x(t+h) = x(t) + \frac{1}{6}h * (k_1 + 2k_2 + 2k_3 + k_4)$

Table 3.23.: *The table lists different integration schemes to compute a position $x(t)$ based on a velocity $v(t)$ where t represents a timestamp.*

3.8.2. Position Predictor

The ultimate goal is the prediction of a position for each cluster using a predicted velocity as input. The purpose of predicting the clusters center is to avoid missing the object in the next event data packet when the object moves too fast. If the clusters center is out of the spatial cost functions range the tracking will be terminated. Therefore, a proper prediction is necessary for tracking fast moving object [DxYx10]

A popular method to predict an objects position in the future is the Kalman filter. It is used in various tracking algorithms [HH11]. The Kalman filter addresses the general problem of trying to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation [GW06]. The output of the DVS is an asynchronous stream of events and not a discrete-time process. But a Kalman filter requires this discretization. In order to preserve the tracking algorithms event-driven behaviour the prediction of the positions in the future is based on a parametric model where a predicted velocity is used to approximate an integral.

This section describes different methods to predict the clusters position in the future. Table 3.23 shows different integration schemes which can be used to predict the objects position $x(t)$ based on a velocity $v(t)$ where t represents a timestamp [Tho10].

Although Runge-Kutta is the most popular method used as an integration schema the algorithm uses the Huen-Method since the acceleration is constant over time which does not require an expansion of the Taylor series up to a degree of 4 as provided by the Runge-Kutta method. In contrast to Runge-Kutta the Huen-Method expands the Taylor series up to a degree of two.

3.8.3. Velocity Predictor

This section describes different methods to predict the clusters velocity. The predicted velocity can be computed either based on the clusters measured velocity or by using a prediction of the acceleration.

Constant Velocity

The simplest approach to predict the velocity is by assuming that it stays constant. Having this assumption the velocity can directly be calculated by using the last measured velocity. By using a low pass filter it would be possible to predict a velocity that reduces the influence of noise. To set up this filter we have first of all to define a weight γ. Formula 3.114 shows the definition of this term which describes how much of the new measured velocity has to be added to the current velocity. It depends on the time $\triangle t$ between two measured velocities and the temporal resolution $t_{resolution}$ of the filter. This method is used in rectangular cluster tracker [Del12b].

$$\gamma = \min\left(1, \frac{\triangle t}{t_{resolution}}\right) \tag{3.114}$$

To compute the new velocity vector the above defined coefficient is used to bound the maximum change of the velocity. This is shown in formula 3.115.

$$v(t+h) = v(t-\triangle t) + (v(t) - v(t-\triangle t)) * \gamma \tag{3.115}$$

Constant Acceleration

By estimating the acceleration of the observed object the algorithm is able to derive more reliable predictions for its velocity. In this approach the acceleration of the observed object is assumed to be constant for a certain time interval. Formula 3.116 shows the computation of the predicted velocity where $a(t)$ represents the acceleration at time t and h the time difference used for the predication.

$$v(t+h) = v(t) + a(t) * h \tag{3.116}$$

For smooth velocities an approach considering a changing velocity would improve the accuracy of the whole prediction process. Therefore the algorithm uses this approach assuming a constant acceleration.

3.8.4. Acceleration Predictor

Since the acceleration is defined as the change of the velocity over time it can be used to correct a predicted velocity over time. This section describes an approach to compute an acceleration which can be used as a prediction for the acceleration.

This approach has as a first input argument the reference velocity $v_{reference}$ which represents the predicted velocity used by the prediction. The second input argument is the measured velocity $v_{measured}$ of the observed object which corresponds to the true velocity which the algorithm has to approximate.

To compare these two velocities the algorithm rotates the velocity $v_{reference}$ around an angle α to align $v_{reference}$ with the x-axis. The velocity $v_{measured}$ is rotated around the same angle so that the difference between them is preserved. Formula 3.117 shows the computation of the angle between the x-axis and the reference velocity $v_{reference}$.

$$\alpha = \cos^{-1}\left(\left\langle \frac{v_{reference}}{\|v_{reference}\|}, x_{axis} \right\rangle\right) \tag{3.117}$$

Since the rotation of vectors using matrices is counter-clockwise we have to distinguish the two cases whether we are above the x-axis or not. The angle has to be adapted depending on the orientation as shown in formula 3.118.

$$\alpha = \begin{cases} 2\pi - \alpha, & \text{if } v_{reference}.y > 0 \\ \alpha, & \text{otherwise} \end{cases} \tag{3.118}$$

In order to rotate the vector around the computed rotation angle α the rotation matrix for 2-dimensional rotations is used. The definition of the matrix is illustrated in formula 3.119.

$$r(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.119}$$

To rotate the two vectors around an angle α we multiply them with the rotation matrix defined in 3.119. This is shown in formula 3.120 where v is the original vector and v' the rotated one.

$$v' = r(\alpha)v \tag{3.120}$$

Figure 3.33 shows the effect of the rotation. Using this representation of the velocity vectors it is straightforward to distinguish between the velocity acting sideways and the one acting forward. This allows determining the corresponding accelerations.

The algorithm first computes the difference of the magnitudes $\triangle m$ of the two velocity vectors. The definition of the difference is shown in formula 3.121.

$$\triangle\, m = \|v_{measured}\| - \|v_{reference}\| \tag{3.121}$$

In order to determine the acceleration of the magnitude of the velocity vector the algorithm uses the difference to decide whether the acceleration has to be increased or decreased. Formula 3.122 shows the definition of the magnitude's acceleration $a.m$.

$$a.m = \triangle m \tag{3.122}$$

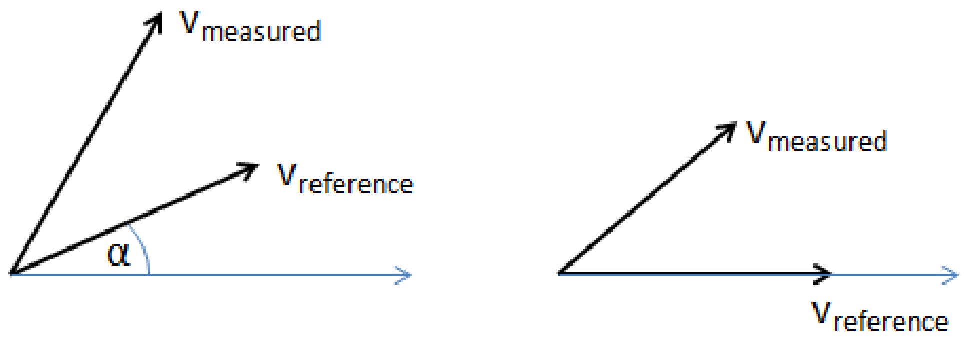


Figure 3.33.: *Shows the rotation of the vector around an angle α to align $v_{reference}$ to the x-axis. After the rotation the reference velocity is aligned with the x-axis while the angle between the two vectors is preserved.*

The computation of the angle between the two vectors $\triangle\alpha$ is done by using the dot product. The mathematical definition is shown in formula 3.123.

$$\triangle\,\alpha = \cos^{-1}\left(\left\langle \frac{v_{reference}}{\|v_{reference}\|}, \frac{v_{measured}}{\|v_{measured}\|} \right\rangle\right) \tag{3.123}$$

To determine the sign of the angle the algorithm has to check if the velocity $v_{measured}$ is above or below $v_{reference}$. Formula 3.124 shows the condition which has to be satisfied to change the sign the angle α.

$$\triangle\,\alpha = \begin{cases} \triangle\alpha, & \text{if } v_{measured}.y > 0 \\ -\triangle\,\alpha, & \text{otherwise} \end{cases} \tag{3.124}$$

The angular acceleration has to be increased if the angle between the two velocity vectors is positive and decreased if the angle is negative. The mathematical notation is shown in formula 3.125 where $a.a$ represents the angular acceleration.

$$a.a = \triangle\alpha \tag{3.125}$$

Experiments have shown that the approach delivers good estimations of the acceleration of an object since it provides a separation of the magnitude and angular acceleration. Especially for smooth changes in the velocity this approach finds good approximations for the true acceleration.

Two low pass filters are used to handle the noise within the measured data for the magnitude's acceleration $a.m$ and the angular acceleration $a.a$. The definition of the low pass filter is discussed in section 3.8.3.

3.8.5. Phase

The phase of a signal is the amount of time which has been elapsed from an arbitrary point in time to the signals start. This corresponds to a timestamp of the start of the LED's blinking pattern.

The blinking pattern has to be extracted by the classifier as described in section 3.6.4. The only thing the tracker needs to do is to synchronize the observed blinking pattern with the extracted transition history of section 3.6.1. Since the extraction and identification of the temporal pattern is part of the classifier the tracker can use the temporal pattern provided by the classifier. The synchronization of the cluster with the temporal pattern of the blinking LED is achieved by tracking the phase of the signal.

A straightforward approach to track a phase of the signal is by using a correlation function. The correlation of a function g with a function f is shown in formula 3.126 where τ corresponds to an offset.

$$C_{gf}(\tau) = \int_{-\infty}^{\infty} g(t)f(t+\tau)dt \tag{3.126}$$

This equation has to be adapted to our problem situation where we have the clusters identified blinking pattern $c(i).s$ and its observed transition history $c(i).th$. The phase of the blinking pattern is determined by correlating the given temporal pattern of the blinking LED with the observed transition history. This is shown in formula 3.127 where $c(i).s$ is representing the temporal pattern and $c(i).th$ the observed transition history. The parameter τ determines the phase of the signal in the observed time interval $[w_l, w_u]$.

$$C_{c(i).s,c(i).h}(\tau) = \int_{w_l}^{w_u} c(i).s(t)c(i).th(t+\tau)dt \tag{3.127}$$

In order to track the phase of the signal the algorithm has to choose the phase τ maximizing the correlation function. Since we are interested in an actual estimation of the phase we have to setup further constraints in order to restrict the phase to be in a certain time window. This helps to reduce the number of possible phases which reduces the computational effort. Additionally it ensures that the phase is an actual estimation and not locked by a very good estimation in the past. The mathematical notation is shown in formula 3.128 where t illustrates the current timestamp used by the algorithm and $t_{threshold}$ a threshold to specify the interval for the search.

$$phase = \arg_{\tau} \max C_{gf}(\tau), \quad t - t_{threshold} < \tau \tag{3.128}$$

Considering the approach presented in section 3.6.3 the computational effort of the correlation function can be reduced by using formula 3.62.

3.8.6. Temporal Pattern Improver

This section describes a method to continuously improve the blinking LED's temporal pattern. The pattern is used as input in the temporal cost function presented in section 3.8.1. The accuracy of the cost functions and therefore of the assignment depends fully on the accuracy of the temporal pattern. This shows the necessity of some kind of temporal pattern improver.

A Phase Locked Loop (PLL) causes a particular system to track with another one. More precisely, a PLL synchronizes an output signal (generated by an output device) with a reference or input signal. In the synchronized - often called locked - state the phase error between the output signal and the reference signal is zero, or remains constant [Bes03].

In our problem situation we synchronize the blinking LED's temporal pattern with the incoming transition history. The pattern corresponds therefore to the output signal while the transition history is represented by the input signal.

Our implementation of the PLL involves three steps:

1. Find the best matching transition t_i in the temporal pattern for an incoming transition t of clusters transition history. This is shown in formula 3.129 where the signals $phase$ and $period$ is used to find the transition t_{i^*} minimizing the temporal distance.

$$i^* = \arg_i \min |t_i.t - (t.t - phase) \,\%\, period| \tag{3.129}$$

2. Compute the error and add it to the transitions sum of errors s_{i^*}. The error corresponds to the temporal distance between the transition t and its best matching transition t_{i^*}. The mathematical notation is shown in formula 3.130.

$$s_{i^*} = s_{i^*} + (t_{i^*}.t - (t.t - phase) \,\%\, period) \tag{3.130}$$

3. The sum of errors s_i defined in 3.130 gives then an indication in which direction a transition has to move. If the sum is positive the transitions temporal position has to be moved forth, otherwise back. Formula 3.131 shows the mathematical notation where a is a predefined factor to control the correction. Here t_i represents a particular transition and s_i its sum of errors.

$$t_i.t = t_i.t + a * s_i \tag{3.131}$$

4

Benchmarking

The section describes different experiments to investigate the limitations for the developed tracking algorithm combining the spatial and temporal information of the events.

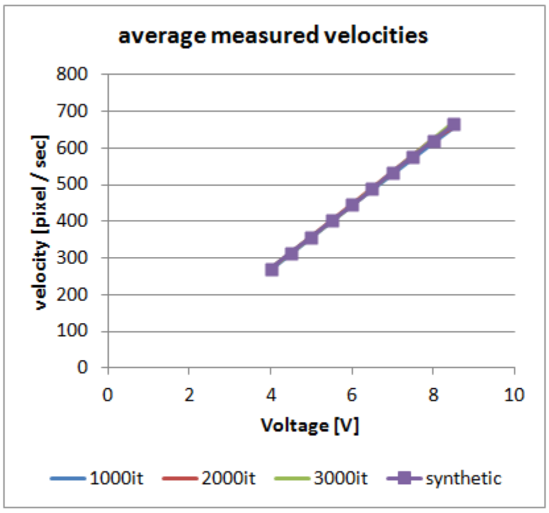
For the experiments the setup with the rotating LED presented in section 3.1.3 was used. The changed parameters for the experiments are listed in table 4.1.

The blinking LED's velocity is controlled by changing the drum motors input voltage. The relation between voltage and the resulting velocity is shown in figure 4.1(a). The micro controller is used to turn the LED on and off. It is programmed with a C program. The program has a while loop running infinitely long. After a certain amount of iterations in the loop the program changes the state of the blinking LED. Figure 4.1(b) shows how many iterations by the micro controller are required to get a specific period of the blinking LED.

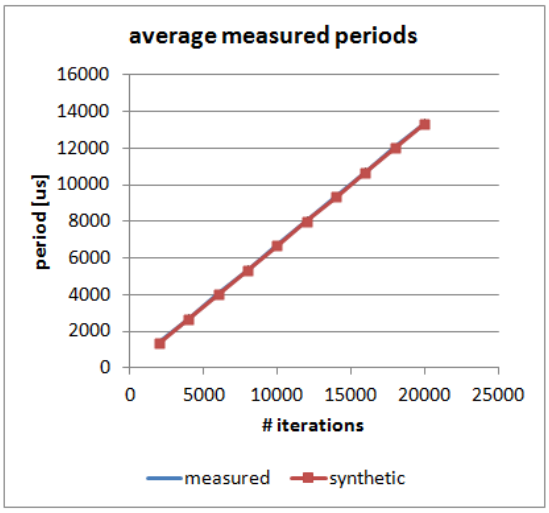
The synthetic curves in figure 4.1 are computed using a linear regression for the measurements. The results are listed in table 4.2. The interpretation of table 4.2 is straightforward. The parameters p_1 and p_0 are the coefficients for a polynomial of degree one: $f(x) = p_1 x^1 + p_0 x^0$.

Parameter	*Quantity*
Distance [cm]	80
Velocity [pixel / sec]	270 - 670
Frequency [Hz]	50 - 750

Table 4.1.: *The table list the default values for the experiment with a moving blinking LED mounted on a cylinder.*



(a) Shows the relation between the drum motor voltage and the LED's resulting velocity on the DVS.



(b) The figure shows the relation from the micro controllers number of iterations to the resulting period of the signal.

Figure 4.1.: *The two figures are showing how the voltage affects velocity and the number of iterations the resulting period.*

Model	p_1	p_0
Velocity	87.68	-80.84
Frequency	0.67	29.12

Table 4.2.: *Parameters for a prediction of the velocity and the frequency using a linear regression.*

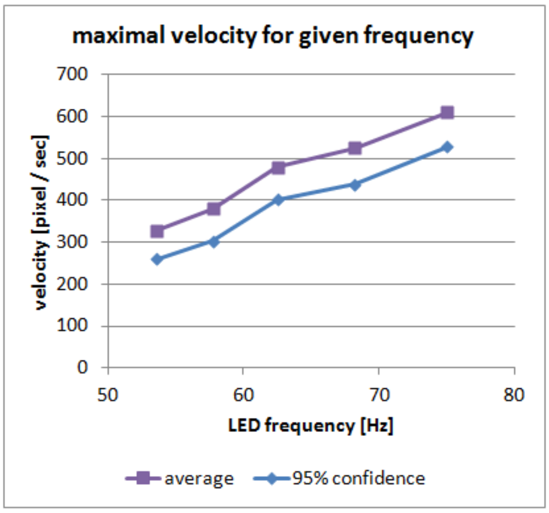
4.1. Maximum Velocity for Tracking

This experiment has to answer the following question: What is the LED's maximum velocity on the DVS that the tracking algorithm is able to cope, given the LED frequency and LED distance? In the experiment different frequencies were tested while the LED's velocity was continuously increased. The maximum velocity is the fastest measured velocity by which the tracking algorithm was able to track the blinking LED.

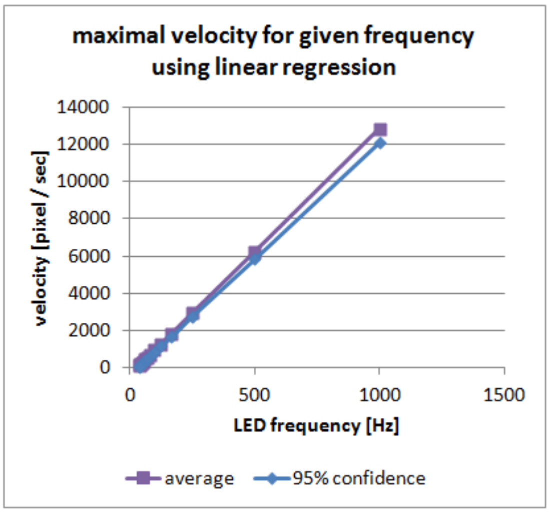
For the experiment we expect that the maximum velocity increases with an increasing frequency. An increasing frequency leads to more events induced by the blinking LED and therefore improves the signal to background ratio presented in section 3.2.1.

Figure 4.2(a) represents the average measured maximum velocities for a given frequency as well as the corresponding 95% confidence interval. It is evident that the maximum velocity increases with an increasing frequency. Using a linear regression for the measured maximum velocities we can predict the tracking algorithms behaviour for other frequencies as shown in figure 4.2(b).

As expected the increased signal to background ratio improves the tracking algorithm's ability to track an object at high velocities.



(a) *The figure shows the average measured velocities for a given LED frequency and the corresponding 95% confidence interval.*



(b) *Shows a prediction of the maximum velocity for a given frequency based on the measured maximum velocities.*

Figure 4.2.: *The figures are showing the maximum velocity for a given frequency the tracking algorithm is able to handle.*

4.2. Accuracy of Measured Positions

This experiment shows how a changing velocity and frequency influence the tracking algorithm's accuracy. The accuracy is defined as the standard error of the measured positions.

To compute the standard error the experiment has to find a reference position, which can be used to compare the tracking algorithm's measured positions. The process to find this reference position is described below.

1. Compute a binary image using the tracking algorithm's measured positions $P = \{p_i\}$ where p_i represents a particular position. A pixel is equal to one, if there exists a corresponding position in the set P and zero otherwise. This is shown in formula 4.1 where i represents the binary image.

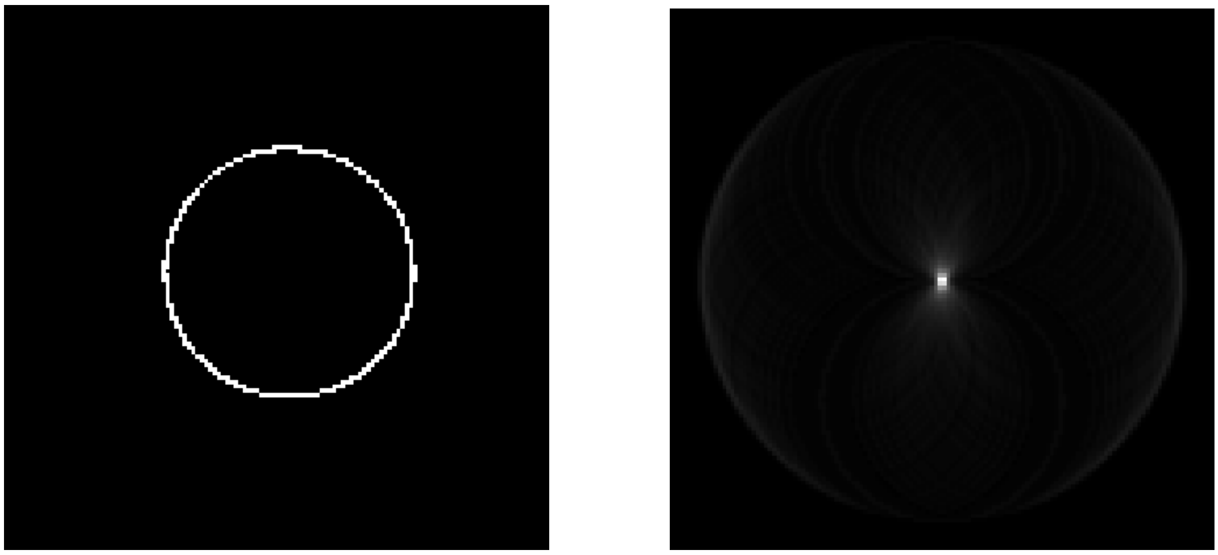
$$i(x, y) = \begin{cases} 1, & \text{if } \exists p_i \in P \wedge p_i.x = x \wedge p_i.y = y \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

   The resulting binary image is shown in figure 4.3(a).

2. Perform a Hough circle transform to find the circle's center c. The result is shown in figure 4.3(b) using the binary image in 4.3(a) as input.

3. We are interested in those positions in P crossing the vertical line at $c.x$ above $c.y$ where c represents the circle's center. The lines mathematical definition is shown in formulas 4.2 and 4.3.

$$l.x(t) = c.x \tag{4.2}$$
$$l.y(t) = c.y + t,\ t \geq 0 \tag{4.3}$$



(a) Shows the binary image used as input for the hough circle transform.

(b) The resulting hough circle transform at a radius of 29 pixels. Figure 4.3(a) was used as input.

Figure 4.3.: *The figure shows the Hough circle transform for a binary image.*

By using the line l defined by the circle's center c, we are able to compute the cluster position whenever it crosses the line. To simplify the notation we define an ordering on the set of measured positions $P = \{p_i\}$. The ordering is based on the positions timestamp $p_i.t$ and shown in formula 4.4.

$$p_i.t < p_{i+1}.t,\ p_i, p_{i+1} \in P \tag{4.4}$$

The LED's rotation on the cylinder is clock-wise. To find the two positions p_i and p_{i+1} crossing the line the condition in formula 4.5 is used.

$$p_i.x < l.x(0) \wedge p_{i+1}.x > l.x(0) \wedge p_i.x > l.y(0) \wedge p_{i+1} > l.y(0) \tag{4.5}$$

An interpolation is used to compute the clusters position on the line p_{line}. The corresponding formulas are shown in 4.6 and 4.7.

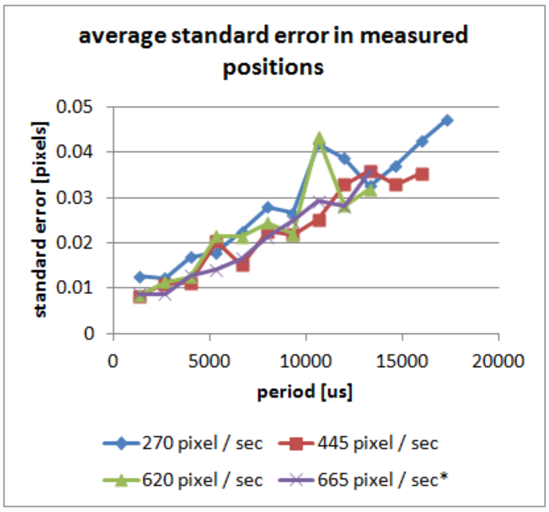
$$p_{line}.x = l(0).x \tag{4.6}$$
$$p_{line}.y = \frac{(l(0).x - p_i.x)p_i.y + (p_{i+1}.x - l(0).x)p_{i+1}.y}{p_{i+1}.x - p_i.x} \tag{4.7}$$

To illustrate the standard error in the measured positions the standard error for the interpolated $p_{line}.y$ was used.

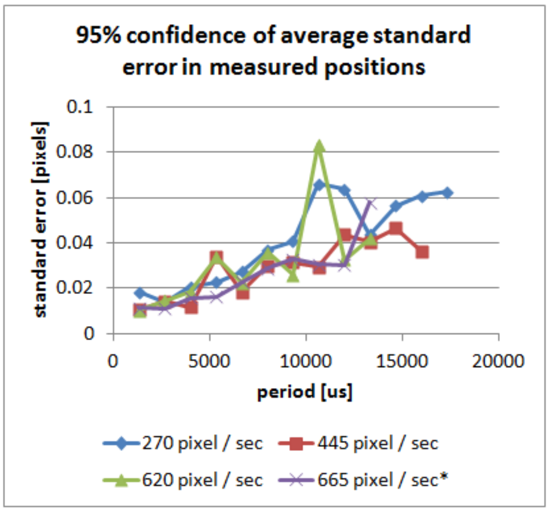
4.2.1. Accuracy with an Increasing Period

In this experiment the LED's period was increased which directly decreases its frequency. We expect therefore that the standard error in the measured positions increases since a decreased frequency leads to a decreasing signal to background ratio as mentioned in section 3.2.1.

The resulting standard error in the measured positions is shown in figure 4.4. The standard error clearly increases with an increasing period of the blinking LED.



(a) The figure shows the average standard error in the clusters measured positions.



(b) Shows the corresponding 95% confidence interval of the standard error. If we repeat the experiment for a particular period and velocity there would be a 95% chance to be below the corresponding line.

Figure 4.4.: *The figures are showing the average standard error in the clusters measured positions and the corresponding confidence interval. The * symbol indicates the maximum velocity measured in the experiment.*

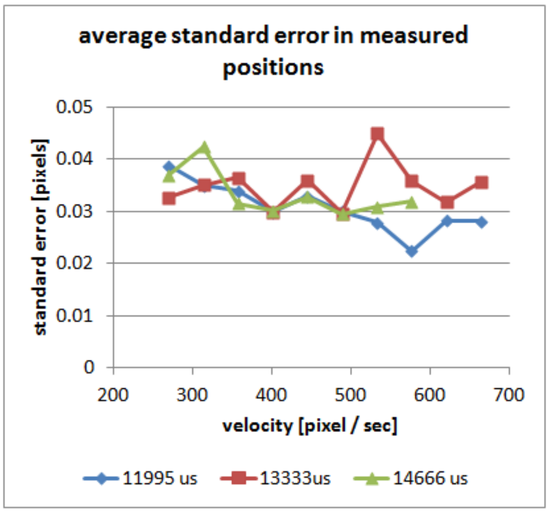
As expected the standard error increases with an increasing period because of the decreasing signal to background ratio. In conclusion, the experiment indicates that a sufficiently high frequency allows achieving any accuracy the user might want to achieve. Of course there are other limitations like the DVS's cutoff frequency as well as the algorithm's ability to distinguish the events induced by the blinking LED and the ones induced by noise as described in section 3.2.4. These observations clearly limit the maximum frequency recognizable by the DVS.

4.2.2. Accuracy with an Increasing Velocity

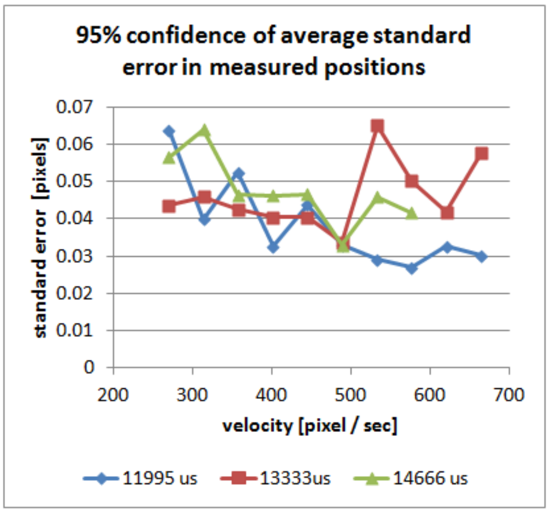
This section presents the resulting standard error in the measured positions using a fixed LED period. During the experiment the LED's velocity was continuously increased.

For the experiment, we expect that the standard error in the measured positions increases with an increasing velocity. The decreasing signal to background ratio described in section 3.2.1 influences the accuracy of the clusters measured positions.

Figure 4.5 shows the results for the experiment with an increasing velocity. There is no clear trend visible and it seems that the standard error in the measured positions remains almost constant.



(a) The figure shows the average standard error in the clusters measured positions.



(b) Shows the corresponding 95% confidence interval for the average of the standard error.

Figure 4.5.: *The figures show the average standard error in the clusters measured positions and the corresponding confidence interval for an increasing velocity.*

A final conclusion why the increasing velocity does not affect the tracking algorithm's accuracy is not possible. A possible explanation could be that the cylinder described in the experimental setup has an imbalance causing a significant jitter for low velocities.

5 Conclusion and Outlook

The thesis has first of all shown that the DVS is able to record the blinking pattern of an LED at frequencies above 1kHz. The DVS with its high temporal resolution has in addition a sufficiently high accuracy to reliably identify the blinking pattern of an LED. The experiments performed in the thesis have proven two fundamental characteristics of the DVS: Its high temporal resolution and temporal accuracy. Using this conclusion as a base, the development of a tracking algorithm for blinking LEDs is justified.

The tracking algorithm presented in the thesis is able to identify the blinking patterns of multiple LEDs. The tracking algorithm combines the spatial and temporal information of the events to enhance the algorithm's detection precision. The events temporal information is used as additional information to compensate the DVS's absence of any color or intensity information. The combination of the events spatial and temporal information enables the algorithm to handle temporary object disappearance, object trajectory crossing and camera movements in a robust way. The thesis has therefore shown that a tracking algorithm using the events temporal information as additional constraints enables a robust marker tracking based on blinking LEDs. As expected, the experimental results show that the detection accuracy and the speed-limit of a moving LED can be straightforward increased by simply increasing the blinking frequency. Corresponding graphs are derived and allows estimating the required blinking frequency at a given object speed.

The algorithm presented in the thesis is able to identify and track an LED based on its blinking pattern. It is therefore possible to track multiple LEDs over time and to distinguish them according to their blinking patterns. The usage of the LED's blinking pattern as a feature for the identification constraints the number of traceable LEDs. Aiming at demonstrating the basic capability of tracking multiple objects, the developed algorithm was able to identify and track 5 objects. The algorithm can certainly cope with more objects - but this has not been exploited further.

In general, to further extent the number of trackable objects, it would be most interesting to investigate distinct blinking patterns that are well distinguishable. The unknown phase for the blinking patterns is an additional constraint for the definition of the set. This is especially important if the markers are not synchronized to each other. The correlation function used for the classification is defined as shown in formula 5.1 where f represents the LED's blinking pattern and g a possible candidate for the classification. The integration is done over the interval $[0, end]$ where end represents the period of the longer signal.

$$\int_0^{end} f(t)g(t)dt \tag{5.1}$$

An ideal set of blinking patterns would maximize the dissimilarity between the patterns which corresponds to minimizing the result of the correlation function. Therefore the resulting similarity computed using the correlation function defined in formula 5.1 has to be equal to zero. This is exactly the definition of orthogonality in signals over an interval $[t_1, t_2]$ [WN96]. Because the markers are not synchronized an additional constraint is introduced. The mathematical notation is shown in equation 5.2 where we consider an arbitrary offset τ.

$$\max_{\tau \in \mathbb{N}} \int_{t_1}^{t_2} f(t+\tau)g(t)dt = 0 \tag{5.2}$$

Since a signal can be interpreted as a vector the definition of an orthogonal set of blinking patterns is straightforward. By using a signal with 50% duty cycle we just double the frequency as shown in formula 5.3.

$$\{ab, aabb, aaaabbbb, ...\} = \{a^{2^i} b^{2^i}, i \in \mathbb{N}\},\ a = -1, b = 1 \tag{5.3}$$

To use only orthogonal signals clearly limits the number of LEDs the tracking algorithm is able to handle. It would therefore be interesting to investigate the required dissimilarity between signals to distinguish them and the definition of a set of signals satisfying this dissimilarity.

A

User Guide

The user guide helps the user to get the tracking algorithm started. It also describes how the user is able to start the blinking light emitting diod (LED)e described in the experimental setup of a rotating LED in section 3.1.3.

A.1. Start Spatio-Temporal Pattern Recognition

jAER is a java project that allows you to do real time processing spike-based processing with address-event representation (AER) systems on PCs [Del12b]. The algorithm for spatio-temporal pattern recognition developed in the thesis is part of jAER. To install and run jAER please follow the instructions available at `http://sourceforge.net/apps/trac/jaer/wiki`.

A.1.1. Activate LabelingFilter

The `LabelingFilter` is used to reduce the number of events that have to be processed by the tracking algorithm. The filter is described in section 3.3.

1. Press the button `Filters` in the main window.
2. A window appears with a list of all added filters.
3. If the list contains the entry `LabelingFilter` activite the filter using the corresponding check box.
4. Otherwise the filter has to be added to jAER.

a) Select in the tab `View` the entry `Customize...`.

b) Use the filter to search `LabelingFilter`.

c) Select the filter and add it by a click on the Button `Add`.

d) Close the window.

e) Activate the `LabelingFilter` by using the corresponding check box in the list of all added filters.

A.1.2. Activate SmartSpatioTemporalTracker

The tracking algorithms name in jAER is `SmartSpatioTemporalTracker`. The tracker can be activated using the same procedure as described in section A.1.1 to activate the `LabelingFilter`.

A.1.3. Customize Filer Parameters

To change the parameters of the two algorithms `LabelingFilter` and `SmartSpatioTemporalTracker` the user can use the graphical user interface provided by jAER.

1. Press the button `Filters`.
2. A window appears with a list of all added filters.
3. Press the corresponding button `Controls` to customize the filters parameters.
4. A list with the filters parameters appears which contains a description of the different parameters.

A.1.4. Choose Input for jAER

As soon as `LabelingFilter` and `SmartSpatioTemporalTracker` are activated as described in section A.1.1 and A.1.2 jAER waits for input.

For live input the user has to plug in a Dynamic Vision Sensor. To play a logged data file the user has to open a file of the format `AER raw binary data file`.

A.2. Start Blinking LED Marker

To start the blinking LED described in the experimental setup of a rotation LED in section 3.1.3 one has just to add a power supply like three AAA batteries if the micro controller is programmed.

The Atmel USB chip is a high-performance Atmel 32-bit CPU optimized for small code size [Del11]. To setup the environment to program the micro controller please follow the description at `http://www.ini.uzh.ch/~tobi/wiki/doku.php?id=dig:uc`. The

C program used in the experimental setup of a rotating LED is part of jAER and can be downloaded from its repository.

A.3. Live Demonstration

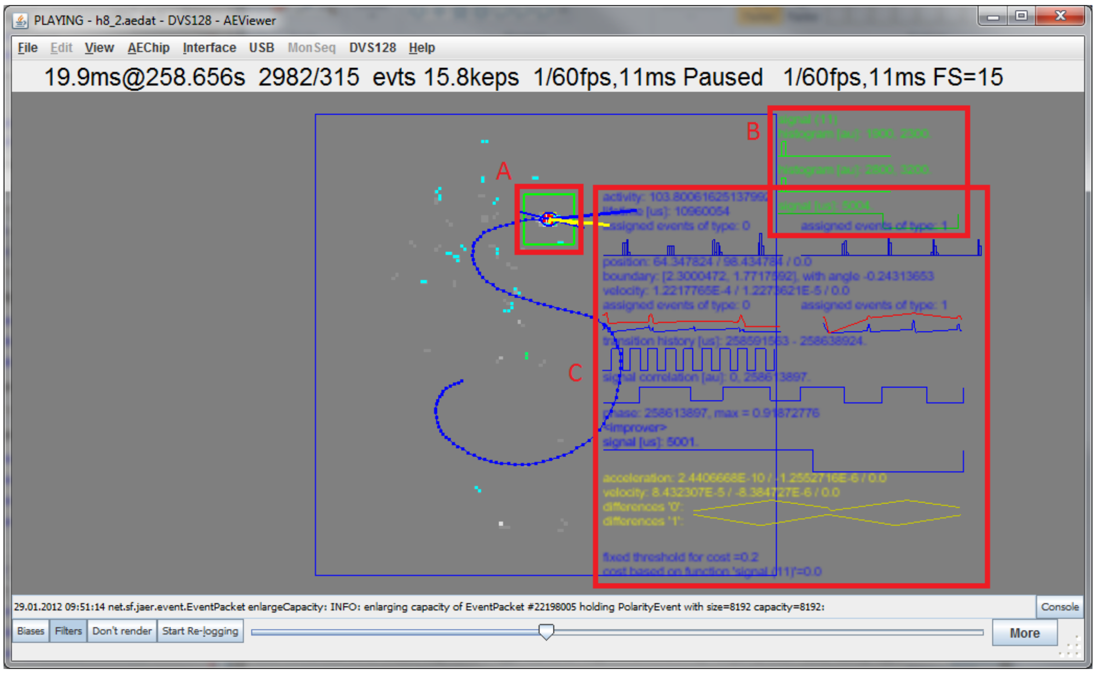
This section shows some pictures to illustrate a live demonstration of the tracking algorithm. It explains the tracking algorithm's information visualized in jAER.

Figure A.1 shows a screen shot of the developed marker tracking algorithm running in jAER. The enumeration below describes the most important information visible.

A: Shows the tracked blinking LED. The LED is surrounded by a blue circle indicating its spatial position and contour. The green box represents the LED's classification. The LED is assigned to the green blinking pattern visible in **B**. The blue dotted line behind the cluster shows the path of the tracked blinking LED.

B: The sector shows the list of registered blinking patterns.

C: The area illustrates various features extracted for the cluster representing the blinking LED. These features are used to track and classify the blinking LED.



```
19.9ms@258.656s  2982/315  evts 15.8keps  1/60fps,11ms Paused   1/60fps,11ms FS=15

signal (11)
histogram [au]: 1900, 2300.
histogram [au]: 2800, 3200.
signal [us]: 5004.
                                         assigned events of type: 1

activity: 103.80061625137992
lifetime [us]: 10960054
assigned events of type: 0
position: 64.347824 / 98.434784 / 0.0
boundary: [2.3000472, 1.7717592], with angle -0.24313653
velocity: 1.2217765E-4 / 1.2273621E-5 / 0.0
assigned events of type: 0               assigned events of type: 1
transition history [us]: 258591563 - 258638924.
signal correlation [au]: 0, 258613897.
phase: 258613897, max = 0.91872776
<improver>
signal [us]: 5001.
acceleration: 2.4406668E-10 / -1.2552716E-6 / 0.0
velocity: 8.432307E-5 / -8.384727E-6 / 0.0
differences '0':
differences '1':

fixed threshold for cost =0.2
cost based on function 'signal (11)'=0.0

29.01.2012 09:51:14 net.sf.jaer.event.EventPacket enlargeCapacity: INFO: enlarging capacity of EventPacket #22198005 holding PolarityEvent with size=8192 capacity=8192:
```

Figure A.1.: *A screen shot of the running tracking algorithm tracking one blinking LED.*

As discussed above, sector **B** in figure A.1 shows the list of registered blinking patterns. Figure A.2 shows the same sector in more detail. This screen shot contains a single registered blinking pattern. The two histograms represent the time spans between on- to off-events and vice-versa as discussed in section 3.6.5. The lower representation shows the registered temporal pattern as discussed in section 3.6.4.

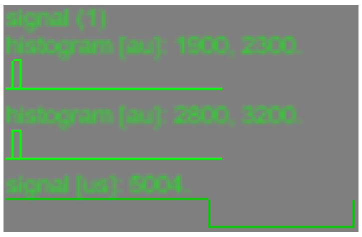


Figure A.2.: *Shows the registered blinking patterns in more detail.*

Figure A.3 shows the cluster's basic features. These features include the cluster's activity, its lifetime, position, contour and velocity. These basic features are discussed in section 3.5. A description for the area highlighted by the red line follows below.

A: Additionally the cluster visualizes its assigned events over time. The x axis corresponds to the time axis while the y axis represents the number of events assigned to the cluster at a particular time.

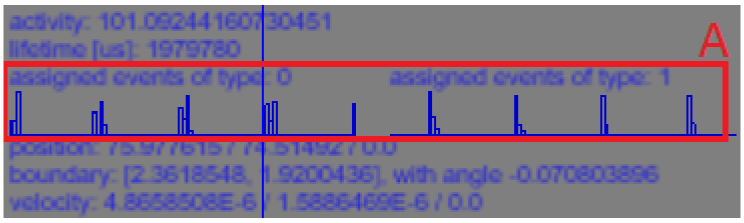

activity: 101.09244160730451
lifetime [us]: 1979780
A
assigned events of type: 0
assigned events of type: 1
position: 75.9776157 / 74.514927 / 0.0
boundary: [2.3618548, 1.9200436], with angle -0.070803896
velocity: 4.8658508E-6 / 1.5886469E-6 / 0.0

Figure A.3.: *The figure shows the cluster's basic features.*

Figure A.4 shows the cluster's features for the identification process in more detail. The first visualization shows the result of the convolved stream of events discussed in section 3.6.1 and the resulting transition history. The next features are showing the estimation of the blinking LED's period presented in section 3.8.5 and the phase locked loop to improve the blinking pattern's temporal accuracy as discussed in section 3.8.6.

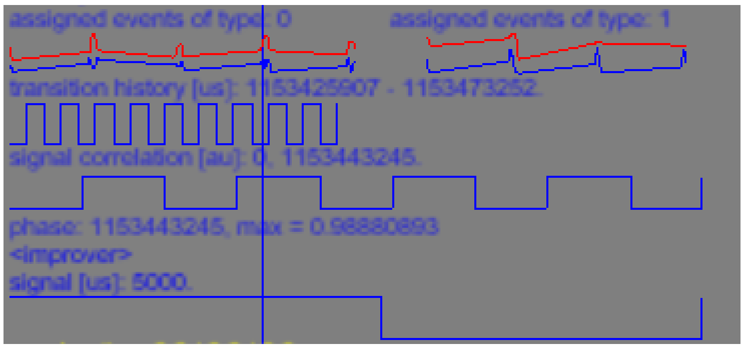
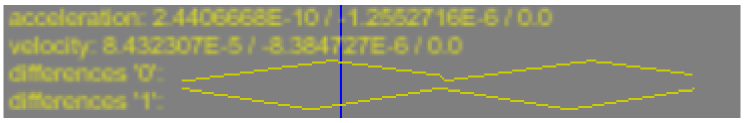

assigned events of type: 0
assigned events of type: 1
transition history [us]: 1153425907 - 1153473252.
signal correlation [au]: 0, 1153443245.
phase: 1153443245, max = 0.98880893
<improver>
signal [us]: 5000.

Figure A.4.: *The figure shows the cluster's features for the identification process in more detail.*

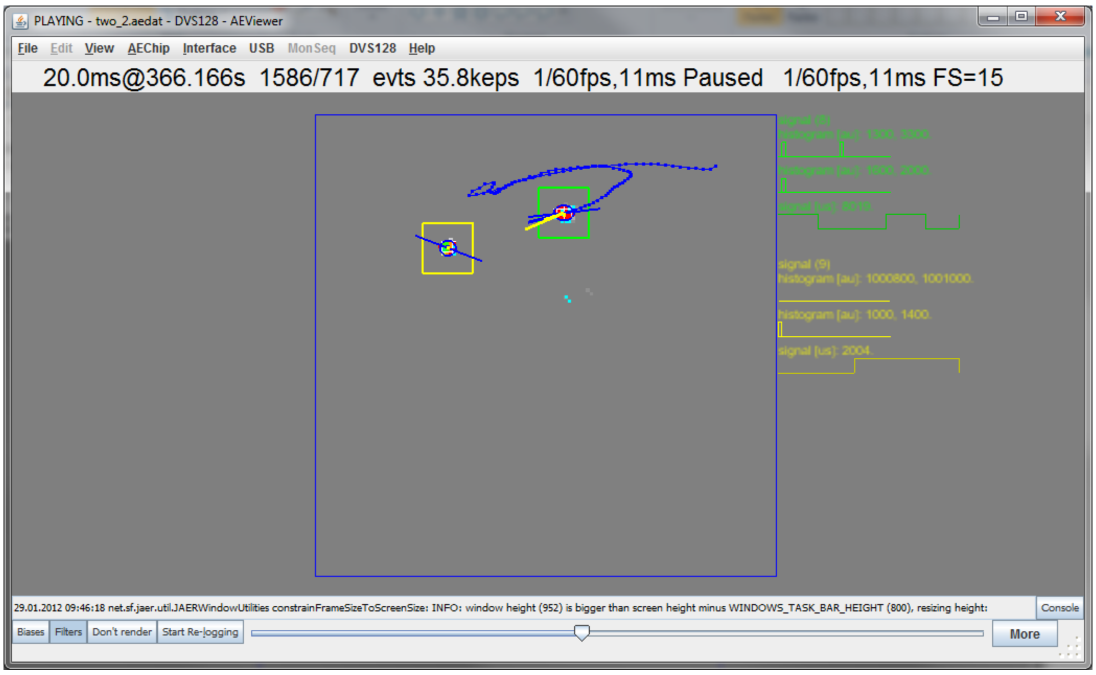
The last close-up in figure A.5 shows the cluster's features used for the assignment of events. The figure shows the predicted acceleration and velocity. The last feature illustrates the occurrence of the next off and on event. The features are discussed in section 3.8.



```
acceleration: 2.4406668E-10 / -1.2552716E-6 / 0.0
velocity: 8.432307E-5 / -8.384727E-6 / 0.0
differences '0':
differences '1':
```

Figure A.5.: The close-up shows the cluster's prediction for the spatial position and the temporal occurrence.

Figure A.6 shows a screen shot of the running tracking algorithm. Two blinking LEDs are tracked. The colors (green and yellow) indicate again the classification to one of the registered blinking patterns visible on the left side of the figure.



PLAYING - two_2.aedat - DVS128 - AEViewer
File Edit View AEChip Interface USB MonSeq DVS128 Help
Biases Filters Don't render Start Re-logging
More
Console

```
20.0ms@366.166s 1586/717 evts 35.8keps 1/60fps,11ms Paused 1/60fps,11ms FS=15
```

```
signal (8)
histogram [au]: 1300, 3300.
histogram [au]: 1600, 2000.
signal [us]: 8019.

signal (9)
histogram [au]: 1000800, 1001000.
histogram [au]: 1000, 1400.
signal [us]: 2004.
```

```
29.01.2012 09:46:18 net.sf.jaer.util.JAERWindowUtilities constrainFrameSizeToScreenSize: INFO: window height (952) is bigger than screen height minus WINDOWS_TASK_BAR_HEIGHT (800), resizing height:
```

Figure A.6.: The screen shot shows the running tracking algorithm with two tracked LEDs.

A.4. Impression

This section gives an impression about the tracking algorithm's live demonstration. Figure A.7 shows a picture of me at a demonstration using a single blinking LED as marker. The two regions highlighted by the red lines are described below.

A: The blinking LED controlled by the AVR32 micro controller.

B: The tracked blinking LED visualized in jAER.

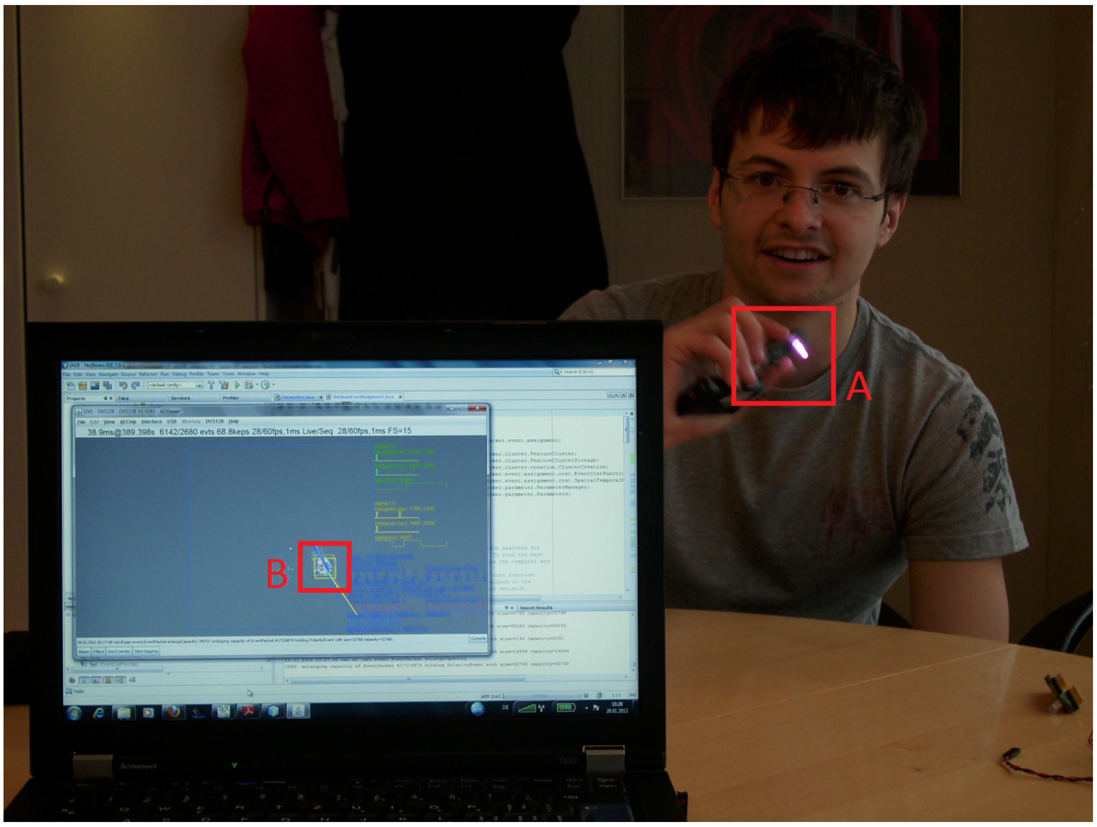


Figure A.7.: *The picture shows me at a live demonstration of the track algorithm.*

Bibliography

[Agr09] Dušan Agrež. Period estimation of the modulated signal. In *Fundamental and Applied Metrology*. IMEKO, 2009.

[BB82] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice Hall, first edition edition edition, May 1982.

[Bes03] Roland E. Best. *LoopPhase-locked loops: design, simulation, and applications*. McGraw-Hill Professional, 5 edition, 2003.

[Buh11] Joachim Buhmann. Statistical Learning Theory. Course at ETH, Zurich, 2011.

[CD09] J. Conradt; R. Berner; M. Cook and T. Delbruck. An embedded AER dynamic vision sensor for low-latency pole balancing. In *Computer Vision Workshops (ICCV Workshops)*. IEEE, 2009.

[Chi03] J.S. Chitode. *Communication Systems - I*. Technical Publications Pune, 3 edition, 2003.

[Del08] T. Delbruck. Frame-free dynamic digital vision. In *Advanced Electronics for Quality Life and Society*. University of Tokyo, 2008.

[Del11] Tobias Delbruck. Electronics for Physicists II (Digital). Course at ETH, Zurich, 2011.

[Del12a] Tobias Delbruck. Dynamic Vision Sensor (DVS) - asynchronous temporal contrast silicon retina. Website, 2012. `http://siliconretina.ini.uzh.ch/wiki/index.php`.

[Del12b] Tobias Delbruck. jAER Open Source Project. Website, 2012. `http://sourceforge.net/apps/trac/jaer/wiki`.

[DL07] T. Delbruck and P. Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system . In *Circuits and Systems*. IEEE, 2007.

[DxYx10] Zhao Ding-xuan and Cui Yu-xin. Research on the Method of Tracking and Measuring Moving Object Based on Machine Vision. In *Computational Intelligence and Software Engineering (CiSE)*. IEEE, 2010.

[FP98] A.J. Lipton; H. Fujiyoshi and R.S. Patil. Moving target classification and tracking from real-time video. In *Applications of Computer Vision, Forth IEEE Workshop*. IEEE, 1998.

[GW06] Gary Bishop Greg Welch. An Introduction to the Kalman Filter. Website, 2006. `http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf`.

[HH11] Shengluan Huang and Jingxin Hong. Moving object tracking system based on Camshift and Kalman filter. In *Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 2011.

[HX10] Qiang Chen; Quan-Sen Sun; Pheng Ann Heng and De-Shen Xia. Two-Stage Object Tracking Method Based on Kernel and Active Contour. In *Circuits and Systems for Video Technology*. IEEE, 2010.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis*. Wiley, first edition, 1991.

[Kil01] Johannes Kilian. *Simple Image Analysis By Moments*. Cranfield University, version 0.2 edition, March 2001. `http://public.cranfield.ac.uk/c5354/teaching/dip/opencv/SimpleImageAnalysisbyMoments.pdf`.

[KR06] R. Kapela and A. Rybarczyk. The Neighboring Pixel Representation for Efficient Binary Image Processing Operations. In *Parallel Computing in Electrical Engineering*, pages 396 – 404. ELEC, IEEE, September 2006.

[Kre05] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 9 edition, 2005.

[LK07] Andreadis L. Kotoulas, I. Accurate Calculation of Image Moments. *Image Processing, IEEE Transactions*, 16(8):2028 – 2037, August 2007.

[LS08] D. Wagner; T. Langlotz and D. Schmalstieg. Robust and unobtrusive marker tracking on mobile phones. In *Mixed and Augmented Reality, 7th ACM International Symposium*. IEEE, 2008.

[MY11] Eun Yeong Ahn; Jun Haeng Lee; T. Mullen and J. Yen. Dynamic Vision Sensor Camera Based Bare Hand Gesture Recognition. pages 52–59, 2011.

[Oly] Olympus. *i-Speed FS*. `http://www.olympus-ims.com/en/hsv-products/i-speed-fs/`.

[Opt] Optronis. *CamRecorder CV*. `http://www.optronis.com/en/products/high-speed-cameras-cv/camrecord-cv.html`.

[Rey08] D. Reynolds. Gaussian Mixture Models. *Encyclopedia of Biometric Recognition*, February 2008.

[Ror10] C. Britton Rorabaugh. *Notes on Digital Signal Processing: Practical Recipes for Design, Analysis and Implementation.* Prentice Hall, first edition, 2010.

[Sci] ScienceMedia. *Phatnom Flex.* `http://www.highspeed-camera.net/D/camera/phantom_flex.html`.

[Seg07] Toby Segaran. *Collective Intelligence.* O'Reilly, 2007.

[SK07] R.A. Leiber S. Kullback. *On Information and Sufficiency Annals of Mathematical Statistics*, volume 22. Institute of Mathematical Statistics, 2007.

[Smo10] Aljoscha Smolic. Multimedia Communications. Course at ETH, Zurich, 2010.

[SN09] S.P. Mathew; P. Samuel; J. Varghese; S. Arumugaperumal; S. Subhash and K. Nallaperumal. Enhanced Morphological Contour Representation and Reconstruction using Line Segments. In *Nature and Biologically Inspired Computing*, pages 1400 – 1405, April 2009.

[ST94] Jianbo Shi and C. Tomasi. Good Features to Track. In *Computer Vision and Pattern Recognition, Computer Society Conference*. IEEE, 1994.

[Sun01] D. Sundararajan. *The Discrete Fourier Transform. Theory Algorithms and Applications.* World Scientific Pub Co Inc, 2001.

[Tho10] Bernhard Thomaszewski. Physically-based Simulation. Course at ETH, Zurich, 2010.

[WN96] Alan V. Oppenheim; Alan S. Willsky and Syed Hamid Nawab. *Signals and Systems.* Prentice Hall, 2 edition, 1996.