Real-time object tracking for event cameras

Zhang, Yexin

2019

Zhang, Y. (2019). Real-time object tracking for event cameras. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/137297>

<https://doi.org/10.32657/10356/137297>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 14:34:22 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

REAL-TIME OBJECT TRACKING FOR EVENT CAMERAS

ZHANG YEXIN

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2019

REAL-TIME OBJECT TRACKING FOR EVENT CAMERAS

ZHANG YEXIN

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

02 February 2020

.....
Date

.....
Zhang Yexin

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

02 February 2020

.....
Date

.....
Goh Wang Ling

Authorship Attribution Statement

This thesis contains material from one paper accepted at conferences in which I am listed as an author.

Chapter 3.1 & 3.4.2 are published as Qinyi Wang, **Yexin Zhang**, Junsong Yuan, and Yilong Lu, “Space-time Event Clouds for Gesture Recognition: from RGB Cameras to Event Cameras”, in *Proc. IEEE Winter Conf. on Applications and Computer Vision (WACV'19)*, 2019

The contributions of the co-authors are as follows:

- A/Prof Yuan provided the initial idea.
- Wang and I co-developed the theory framework.
- Wang designed implementation framework and evaluation process.
- I designed a rolling buffer mechanism to improve real-time performance.
- Prof Lu provided overall guidance to the research.
- Wang and I prepared manuscript together. The manuscript was revised by Prof Lu and A/Prof Yuan.

02 February 2020

.....

Date

.....

Zhang Yexin

Acknowledgment

I would like to express my sincere appreciation to my supervisor, Associate Prof. Goh Wang Ling for her patient guidance and continuous support. I really appreciate her willingness to become my supervisor in the halfway of master candidature. Without her efforts and help, I cannot continue and focus on my research.

I would also like to express my sincere appreciation to my last supervisor, Associate Prof. Yuan Junsong. Thank him for providing the direction of my thesis. It is a pity that I do not have opportunities to learn from him more. However, during the limited time I learned from him, I can feel his sense of responsibility as both a scholar and a teacher.

Finally, I would like to express my sincere, not just appreciation but also respect, to my colleague and girlfriend, Wang Qinyi. She is a competent mentor to share her rich research experience. Thank her for always providing me her wisdom and encouragement.

Table of Contents

Acknowledgment	iv
Abstract	vii
List of Figures	ix
List of Tables	xi
Chapter 1. Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Major Contribution of the Thesis	6
1.4 Organization of the Thesis	6
Chapter 2. Literature Review	8
2.1 Review of Conventional-camera-based Object Tracking Methods	8
2.1.1 Tracking by Detection	8
2.1.2 Tracking by Structured Output	11
2.1.3 Correlation Filter Based Trackers	13
2.1.4 Tracking with Deep Neural Network	16
2.2 Review of Event-camera-based Object Tracking Methods	18
2.2.1 Real-time Particle Tracking by Event Cameras	19
2.2.2 Object Tracking by Cooperation between Frame-based and Event-based Cameras	21
2.2.3 Event-based Tracking by Detection Framework	24
2.2.4 Event-based Motion Compensation Tracking Framework	25
Chapter 3. Spacetime Event Cloud	27
3.1 From Temporal Sequences to Spacetime Event Clouds	29
3.2 Description of Object Tracking Task based on Spacetime Event Clouds	32
3.3 Methodology of Spacetime Event Clouds-based Siamese Regression Network	

3.4	Architecture of Spacetime Event Clouds-based Siamese Regression Network	36
3.4.1	Preprocessing Stage: Search Region Generation	37
3.4.2	PointNet for Spacetime Event Clouds-based Feature Extraction	38
3.4.3	Regression Fully Connected Neural Network Layers	41
3.5	Experiments.....	42
3.5.1	Dataset.....	42
3.5.2	Experiment on the Capability of Tracking Similar Sequences	43
3.5.3	Experiment on the Capability of Tracking New Sequences	46
3.6	Performance Analysis	46
3.6.1	Scale of Data Size Analysis	46
3.6.2	Speed Analysis.....	47
3.6.3	Accuracy Analysis	48
3.6.4	Effect of Bagging Augmentation	49
Chapter 4.	Event-wise Continuous Tracking Framework	49
4.1	Limitations of PointNet-based Siamese framework.....	50
4.2	Nature of Continuity of Object tracking	51
4.3	Event-wise Recurrent Network	54
4.4	Methodology for Event-wise Tracking	55
4.5	Siamese EventNet Tracker	58
4.6	Experiments and Analysis	61
Chapter 5.	Conclusions and Suggestion for Future Work	66
5.1	Conclusions	66
5.2	Future Works.....	68
	Bibliography	69

Abstract

There have been many breakthroughs in computer vision, especially in image processing over recent years. Progressively, the focus of the computer vision community has moved to video processing. There are a few obvious difficulties in video processing. First is the much larger data scale. Videos from conventional cameras are virtually a series of images. To complete a task related to video, e.g. action recognition or object tracking, more than one frame is required to be dealt with at the same time. The second problem is the inherent sequential dependency between frames. All video related tasks require an efficient way to extract, parameterize and learn sequential information.

Object tracking is a fundamental video task that is engaged in many daily life applications. For example, object tracking is one of the core tasks in auto-driving and field surveillance. Also, it plays an important role in tasks like action recognition and human-computer interaction. Object tracking has some obvious challenges to handle, including deformation and rescale, illumination variation, background clutter and blur caused by fast movement. This thesis starts with reviewing the development history of object tracking based on conventional trackers, like generative and discriminative paradigms, tracking-by-detection approaches, correlation filter trackers and how video object tracking takes advantage of recently developed deep neural networks.

Different from conventional cameras, recently developed event cameras do away completely with frame concept. In event cameras, only the local pixels whose intensity changes more than a certain threshold will generate data. To summarize the difference between conventional cameras and event cameras in terms of data format, event cameras create temporal-dense but spatial-sparse data while conventional cameras generate spatial-dense but temporal-sparse data. Event cameras have temporal resolution reaching the microsecond level and do not generate redundant data for static objects. This makes event cameras more consistent with the nature of motion-related vision tasks. Also, event cameras embed a timestamp into the output data to record the triggering time of a certain event. These characteristics provide a huge potential to investigate different ways to complete object tracking tasks. This thesis also reviews

current efforts and achievements related to applying event cameras into object tracking. Researchers have already demonstrated the capability of event cameras in this area.

Based on a novel perspective in [1] to treat the data from an event camera as spacetime event clouds, this thesis proposes a spacetime event clouds-based Siamese regression network to learn both spatial and temporal features simultaneously of a moving object and to regress to predicted target location directly. This work demonstrates that event cameras can complete tracking tasks with a much smaller scale of data and less computing power.

Finally, to be more consistent with the nature of continuity of motion, this thesis also investigates cutting-edge techniques to track objects event-wise continuously. The objective is to provide a prediction when a new event coming. With this idea, the tracking system does not provide discrete bounding boxes anymore. Instead, it will provide a smooth 3D bounding volume. The trajectory and rescale effect should be reflected by the bounding volume.

List of Figures

Figure 1 Definition of Video Object Tracking	1
Figure 2 (a) Hardware-related challenges (b) Motion-related challenges	2
Figure 3 Development trajectories of deep learning and event camera.....	3
Figure 4 (a) Conventional camera's output of a static red ball and a moving blue ball. 4	
Figure 5 Frame data from conventional cameras and event data from event cameras ...	5
Figure 6 Tracking-by-detection approaches generate a set of samples and depending on the type of learner, produce training labels.....	11
Figure 7 Tracking by structured SVM -- avoids a separated labeler and operates directly on the tracking output.	13
Figure 8 Comparison between tracking by detection (left) and KCF (right).....	14
Figure 9 Illustration of the cyclic shift to generate training samples [31]	15
Figure 10 Training and Testing of [18].....	17
Figure 11 Network architecture of [18]	18
Figure 12 Illustration of event data from [34] (Color represents the parity)	20
Figure 13 Check if there are neighboring events within time interval T	21
Figure 14 Architecture of event-frame cooperation framework [16]	22
Figure 15 warping process of events within an interval	26
Figure 16 Illustration of the data format of event camera and conventional camera ...	27
Figure 17 (a) Conventional approach treats event streams as temporal sequences. For an event e_i , it carries the pixel location (x_i, y_i) reporting an intensity change and a timestamp t_i . (b) Proposed approach treats event streams as 3D spacetime event clouds. An event e_i becomes a point in 3D space denotes as (x_i, y_i, t_i)	28
Figure 18 (a)In time sequence approach, accumulation of events over time dilutes spatial sparsity of event streams and lose dense temporal information captured. (b)Spacetime event clouds approach preserves spatial sparsity and temporal density of event data..	31
Figure 19 Event stream of a moving object with changing background	32
Figure 20 Divide event stream into multiple event clouds.	33
Figure 21 Bounding box under spacetime event cloud framework. (a) Projection view.	

(b) 3D view.	34
Figure 22 (a) Using a well-labeled dataset to train a generic spacetime-event-cloud-based tracker. (b) During the test stage, this network can track the new target object.	35
Figure 23 Architecture of Spacetime Event Clouds based Siamese Regression Network	37
Figure 24 Search Region Generation	38
Figure 25 (a) The event cloud of a moving dog in x,y,t-continuum. (b) The point cloud of a rabbit in x,y,z-continuum.	39
Figure 26 (a) PointNet is a neural network architecture that directly consumes point clouds for feature extraction. (b) In this work, for the first time, PointNet is applied to consume spacetime event clouds for feature extraction in event camera-based tracking system	41
Figure 27 Original sequence & event camera version	43
Figure 28 Down-sampling of event clouds	44
Figure 29 Illustration of feature extraction	44
Figure 30 2D bounding boxes to 3D bounding cuboids	51
Figure 31 Tracking system response at each timestamp	52
Figure 32 Bounding Boxes VS Bounding cuboids VS Bounding volume	53
Figure 33 Illustration of decayed rotation of event spatial feature vector	54
Figure 34 Event-wise tracking network	57
Figure 35 Siamese EventNet Tracker	59
Figure 36 Illustration of IoU	62
Figure 37 IoU for each sequence	62
Figure 38 Demonstration of predictions and ground-truths	64

List of Tables

Table 1 Evaluation of tracking capability on similar sequences.....	45
Table 2 Evaluation of tracking capability on new sequences	46
Table 3 Scale size of Data.....	47
Table 4 Speed Comparison	48

Chapter 1. Introduction

1.1 Motivation

In recent years, there have been many breakthroughs in computer vision, especially in image processing. However, the pace of video processing does not catch up well. There are a few obvious difficulties in video processing. Two of them are the much larger data scale and the inherent sequential dependencies between frames.

Video object tracking is a process that locates a moving object continuously in a video. The Video Object Tracking Challenge [2] has the standards for video-based object tracking, which are shown in Fig. 1. First, the bounding box is provided only in the first frame. Then, the tracker should not perform re-detection. Finally, trackers do not use future frames for pose estimation.

Figure 1 Definition of Video Object Tracking

Object tracking is a fundamental computer vision task that is engaged in many daily life applications. For example, object tracking is one of the core tasks in auto-driving and field surveillance. It also plays an important role in tasks like action recognition and human-computer interaction. Besides two common difficulties for all video-related computer vision tasks which are described earlier, there are many other challenges specific to object tracking. These challenges can be categorized into two main classes. Challenges in the first class are due to hardware limitations, e.g. illumination problem,

noise corruption and motion blur, which are shown in Fig. 2 (a). The challenges in the other class are related to some motion characteristics, e.g. occlusion, out-of-plane rotation, and deformation, which are shown in Fig. 2 (b).

Figure 2 (a) Hardware-related challenges (b) Motion-related challenges

Recently developed event cameras get rid of the frame concept in conventional vision systems. In event cameras, only the local pixels whose intensities change will generate data. It means the data scale will be much smaller, from gigabytes level to megabytes level. Another attractive characteristic of the event camera is that the data itself encapsulates the timing information. Due to these features, event cameras have profound potential in many daily life applications, for example, object tracking, gesture recognition, hand language recognition, human-computer interaction, auto-driving, and field surveillance.

Conventional RGB cameras see the world as a synchronous sequence of frames, while event cameras directly sense the changes in the scene as an asynchronous sequence of per-pixel intensity changes. The first design of the event camera [3] was reported in 2005. Pioneer researchers have tried to use event cameras for fast-moving particle

tracking [4], object recognition [5] [6] [7], and visual information processing [8]. However, the lack of synchronous intensity information made event cameras less suitable for most of the multimedia and computer vision applications in the early years.

Meanwhile, as the explosive development of deep learning techniques in the past few years, neural network models to analyze data with different natures were invented, not only for images and videos, but also audio [9] [10], text [11], time-series data [12], point clouds [13] [14] and so on.

These valuable studies offer new opportunities to apply event cameras in computer vision tasks and provide new ideas to handle the unique asynchronous event data generated by the event cameras. Recent explorations achieve amazing results on using event cameras for trajectory estimation [15], object tracking [16], and simultaneous localization and mapping (SLAM) [17]. The development trajectories of both deep learning and event cameras in Fig. 3 indicates that it is time to explore more between deep learning and event cameras.

Figure 3 Development trajectories of deep learning and event camera

1.2 Objectives

As described previously, due to the proprietary characteristics, event cameras can target some challenges just by nature. In terms of video object tracking, scholars have put much effort to deal with background clutters. Many models or algorithms have been proposed to distinguish between background and foreground. Event cameras only generate data in pixel dimension (in contrast to frame dimension in conventional RGB cameras) when intensity change happens locally. This means many static constituent parts in the background never generate data. Therefore, the efforts needed for background clutters are reduced significantly.

Event cameras have a fast response to movements with microsecond temporal resolution. This advantage makes event cameras highly suitable for recognizing high-speed motions. Instead of capturing synchronized frames, event cameras directly sense the motions in the scene as asynchronous streams of events.

Figure 4 (a) Conventional camera's output of a static red ball and a moving blue ball.

(b) Event camera's output of a static red ball and a moving blue ball. [1]

As shown in Fig. 4(a), a conventional camera captures all pixel intensities at a fixed frame rate. It captures a clear shape of the stationary object (red ball) and a blurred shape of the fast-moving object (blue ball). As shown in Fig. 4(b), an event camera only captures an asynchronous sequence of intensity changes caused by the fast-moving blue ball while information of the stationary objects (red ball and background) will not be recorded. Each intensity change is referred to as an event encoding the spatial coordinates (x, y) of the pixel reporting a change and a precise timestamp (t) indicating when that change happened. There is also a polarity bit (p) that denotes the direction of the intensity change, positive or negative.

From the aspect of data, conventional RGB cameras generate spatial-dense but temporal-sparse data. Meanwhile, event cameras generate temporal-dense but spatial-sparse data. This can be illustrated by a real example in Fig. 5. The temporal-dense feature of event cameras is closer to the nature of motion-based applications which are continuous along time.

Figure 5 Frame data from conventional cameras and event data from event cameras

By reducing a large amount of redundant data of background, the data scale is reduced profoundly from gigabytes level to megabytes level. However, without intensity values, it is difficult to apply prevailing computer vision techniques directly to event data. To

take the advantages of event cameras mentioned previously, the objective of this thesis is to investigate an efficient way to make the most use of this kind of sparse data. With the help of cutting-edge machine learning and deep learning techniques, this thesis aims to apply event cameras into object tracking applications.

1.3 Major Contribution of the Thesis

The major contributions of this thesis are:

- 1) This thesis extends our idea of Spacetime Event Cloud in [1] into single object tracking. With this idea, the computation efficiency has been improved significantly.
- 2) This thesis introduces the bounding volume concept instead of the bounding box to track objects with event cameras. It is more consistent with the format of event data and more consistent with the nature of the continuity of moving objects.
- 3) This thesis introduces a Siamese network to extract temporal and spatial information simultaneously from the geometric distributions of two successive spacetime event clouds.
- 4) This thesis investigates on applying cutting-edge techniques to process event data event-wise continuously. The event data will further get rid of the sliding window used in 3). The bounding volumes will be improved from discrete to smooth.

1.4 Organization of the Thesis

Chapter 2 reviews the development trajectory of video object tracking, which includes both research work based on conventional cameras and primary efforts on tracking with event cameras. Several classical concepts and techniques are explained in detail. Further, recent breakthroughs of applying deep neural networks in tracking are also examined.

Chapter 3 starts with the explanation of the Spacetime Event Cloud concept [1]. Based on this concept, a PointNet-based [13] Siamese network [18] is constructed to solve the object tracking problem with only event data. Chapter 3.5 explains several experiments

to evaluate the tracking capability of this network. Chapter 3.6 analyzes experiments result in detail.

Chapter 4 investigates how to process event data event-wise continuously. This thesis refers to the most recent work from [19] and designs a network to process data event by event on the fly based on a smooth bounding volume idea.

Finally, in Chapter 5, a conclusion and some high-level thinking on how to conduct object tracking with event cameras are presented. Some ideas that can be investigated in the future are also described.

Chapter 2. Literature Review

2.1 Review of Conventional-camera-based Object Tracking Methods

Based on conventional frame-based cameras, researchers have done a lot of studies on object tracking. Early algorithms mainly fall into two categories, generative and discriminative algorithms. In generative models, the main idea is to search the best-matched window, e.g. Kalman filter, particle filter and mean shift. Vojir's work [20] in 2014 using a scale-adaptive mean-shift method is an impressive example of generative algorithms.

Discriminative trackers discriminate foreground (target patches) from the background. A discriminative tracker usually contains a classifier, which classifies image patch candidates as foreground or background. As discriminative models can make the use of more information – background knowledge, they usually have better performance than generative models. Most discriminative models are under the tracking-by-detection paradigm.

2.1.1 Tracking by Detection

Tracking-by-detection is a classical tracking paradigm that benefits from the development of object detection. Adaptive tracking-by-detection approaches maintain a classifier updating online [21] to distinguish the target object from its surrounding background. Usually, these approaches use a sliding-window to feed image patches around the predicted bounding box in the previous frame into classifiers. The image patch with the highest classification score is the predicted bounding box in the current frame. The assumption here is a constant motion model, which assumes the target does not move too fast. Therefore, the target position in the last frame is an unbiased rough prediction for the target position in the current frame. The constant motion model is a common assumption in many trackers, no matter generative or discriminative.

Mathematically, an adaptive tracking-by-detection tracker maintains an estimate of the bounding box $\mathbf{p}_t \in P$ of the tracking target. The bounding box $\mathbf{p}_t$ should be within the current image frame. A feature extractor is applied to the bounding box $\mathbf{p}$ to get a feature descriptor $\mathbf{f}_t^{\mathbf{p}} \in F$. There are different choices for feature extractors, e.g. color-based feature, geometry-based feature and haar-like feature [22]. Direct template matching is also a choice for classification.

The classifier is trained with sample pairs $(\mathbf{f}, b)$, where b is the binary label to represent positive samples (foreground) or negative samples (background). The objective of the online training process is to update classification confident function $h: F \rightarrow \mathbb{R}$, which gives each image patch a score to show the confidence of this patch is the tracking target in the current image frame.

Let $\mathbf{p}_{t-1}$ be the estimated bounding box at time $t - 1$. The tracker estimates the target position by maximizing h in a local region around $\mathbf{p}_{t-1}$. Without considering the effect of rescaling and deformation (the bounding box size keeps constant), this can be done by searching a translation $\mathbf{T}_t$, by which the new bounding box of tracking target is translated as $\mathbf{p}_t = \mathbf{p}_{t-1} + \mathbf{T}_t$, which can maximize $h(\mathbf{f}_t^{\mathbf{p}_t})$. The search space depends on the motion model which is assumed for the tracking target. For simple cases, like constant motion model here, a circle search around the center of $\mathbf{p}_{t-1}$ is a good choice. Another choice is to apply a scale factor for both the width and height of the bounding box $\mathbf{p}_{t-1}$. In short, the desired translation is determined by the following formula when spanning in search space.

$$\mathbf{T}_t = \arg \max_{\mathbf{T}} h(\mathbf{f}_t^{\mathbf{p}_{t-1} + \mathbf{T}}) \quad (1)$$

The bounding box is updated as below formula.

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \mathbf{T}_t \quad (2)$$

Given the estimated object's new location, the tracker generates a set of binary labeled samples from the current frame, which are foreground or background, to update the

classifier online. In [22], Sam summarized this process into two phases: the *sampler* and the *labeler*. The sampler generates a set of different transformations $\{\mathbf{T}_t^1, \dots, \mathbf{T}_t^n\}$, resulting in a set of training samples $\{\mathbf{x}_t^{\mathbf{p}_t + \mathbf{T}_t^1}, \dots, \mathbf{x}_t^{\mathbf{p}_t + \mathbf{T}_t^n}\}$. After this phase, the labeler chooses labels $\{b_t^1, \dots, b_t^n\}$ for these training samples.

Traditional labelers use a *transformation similarity function* [22] to determine the label of a sample positioned at $\mathbf{p}_t + \mathbf{T}_t^i$. This function can be expressed as $s_{\mathbf{p}_t}(\mathbf{T}_t^i, \mathbf{T}_t^j) \in \mathbb{R}$. Given a reference bounding box $\mathbf{p}_t$ and two translations $\mathbf{T}_t^i, \mathbf{T}_t^j$, this function determines how similar the resulting samples are. For example, the overlap function is defined as the ratio of intersection in terms of union. This is also called intersection-over-union, which is an important accuracy metric for trackers.

$$s_{\mathbf{p}_t}^o(\mathbf{T}_t^i, \mathbf{T}_t^j) = \frac{(\mathbf{p}_t + \mathbf{T}_t^i) \cap (\mathbf{p}_t + \mathbf{T}_t^j)}{(\mathbf{p}_t + \mathbf{T}_t^i) \cup (\mathbf{p}_t + \mathbf{T}_t^j)} \quad (3)$$

Transformations can include more than simple rigid body translation. For example, if taking into consideration the rescale effect, besides translation, there should be also a scale transformation. Here uses only translation for simplification.

Given a transformation similarity function, the labeler uses a *labeling function* $b_t^i = l(s_{\mathbf{p}_t}(\mathbf{T}^0, \mathbf{T}_t^i))$ to determine the label b_t^i of a sample generated by translation $\mathbf{T}_t^i$.

$$l(s_{\mathbf{p}_t}(\mathbf{T}^0, \mathbf{T}_t^j)) = \begin{cases} +1, & s_{\mathbf{p}_t}(\mathbf{T}^0, \mathbf{T}_t^j) \geq t \\ -1, & s_{\mathbf{p}_t}(\mathbf{T}^0, \mathbf{T}_t^j) < t \end{cases} \quad (4)$$

t is the threshold. $\mathbf{T}^0$ is the zero translation. In labeler phase, researchers have already put many efforts on increasing the robustness to poorly labeled samples, including using robust loss function [23] [24], semi-supervise learning [25] [26], or multiple-instance learning [27] [28].

Fig. 6 illustrated the process of tracking by detection. As shown on the left side, when a new frame comes in, several regions of interest (ROIs) are selected around the bounding box in the previous frame. These ROIs are passed into the classifier. The ROI with the highest classification score is the predicted bounding box (green) in the current frame. As shown on the right side, several samples are sampled around the newly predicted bounding box. These samples are labeled as positive or negative (1 or 0) to update the classifier.

Figure 6 Tracking-by-detection approaches generate a set of samples and depending on the type of learner, produce training labels.

2.1.2 Tracking by Structured Output

Even though the conventional tracking-by-detection paradigm shows reasonable bases and performs well in some tracking tasks. Sam in [22] points out some issues which cannot be ignored.

- 1) There is an assumption in conventional tracking-by-detection approaches, which is that the classification confidence function provides an accurate estimate of object position. However, the objectives of these two tasks are slightly different. The objective of the classifier is to classify an image patch correctly. However, the objective of the tracker is to estimate object coordinates as accurately as possible.

One is a straightforward classification task, while the other one is more like a regression task.

- 2) When updating the classifier online, samples used for training are equally weighted. This means that no matter how wrong the negative samples are, they are treated the same. A negative sample overlapping 50% with the ground truth makes the same effect on the classifier as another negative sample overlapping only 1% with the ground truth.
- 3) Another important limitation is that the labeler is usually chosen base on intuitions and heuristics of the researchers, which may not suitable for the classifier. Many efforts have been put on label noise, for example, using robust loss function [23] [24], semi-supervise learning [25] [26] and multiple-instance learning [27] [28].

No matter which techniques are engaged to improve the robustness against label noises, the real problem is the separation of the labeler from the learner, which is not solved. To address these issues, Sam [22] proposed a framework for adaptive visual object tracking based on structured output, which uses a kernelized structured output support vector machine. This method can completely avoid the need for an intermediate classification step.

For the purpose of better comparison with Chapter 3.1, this section continues to use translation instead of general transformation. Rather than learning a classifier, Sam's approach tries to learn a prediction function $h: F \rightarrow T$ to directly estimate the object translation between frames. The output space becomes the space of all translations T instead of the binary labels. It learns h in a structured output SVM framework [29] [30], and introduces a *discriminant function* $D: F \times T \rightarrow \mathbb{R}$. This discriminant function measures the compatibility of the feature descriptor $\mathbf{f}_t^{\mathbf{p}^{t-1}}$ and the translation $\mathbf{T}$. The target translation can be predicted by the following equation [22]:

$$\mathbf{T}_t = h(\mathbf{f}_t^{\mathbf{p}^{t-1}}) = \arg \max_{\mathbf{T} \in T} D(\mathbf{f}_t^{\mathbf{p}^{t-1}}, \mathbf{T}) \quad (5)$$

The difference can be found between conventional tracking-by-detection approaches and this approach. The discriminant function D includes the translation $\mathbf{T}$ explicitly. This indicates that this algorithm can learn and predict the translation directly. The

prediction function can be updated online by a labeled example based on the new tracker location $(\mathbf{f}_t^{\mathbf{p}_t}, \mathbf{T}^0)$.

It can be found that the structured output approach completely avoids the need for an unreliable labeler and can predict the translation of the target object directly. This feature is consistent with the natural objective of the tracking tasks. As shown in Fig. 7, when a new frame comes in, the translation with the highest discriminant function value is applied to the bounding box in the previous frame. The translated bounding box is the prediction for the target in the current frame. After predicting, the learner is updated by the image patch in this new bounding box with null translation.

Figure 7 Tracking by structured SVM -- avoids a separated labeler and operates directly on the tracking output.

2.1.3 Correlation Filter Based Trackers

As discussed in Chapter 2.1.2, conventional tracking-by-detection approaches treat samples for updating classifier equally weighted. F. Henriques in [31] provided an analytical solution to solve this problem. Instead of labeling samples absolutely with binary labels, F. Henriques assigns a score to each sample based on their translations from the target sample.

Therefore mathematically, the aim of Henriques's tracker is to find a function $f(\mathbf{z}) = \mathbf{w}^T \mathbf{z}$ by training, which minimizes the loss function of Ridge Regression over all samples X and corresponding labels $\mathbf{y}$. Ridge Regression is one of the variants of linear regression to mitigate overfitting and multicollinearity problems.

$$\min_{\mathbf{w}} (||\mathbf{y} - X\mathbf{w}||^2 + \lambda ||\mathbf{w}||^2) \quad (6)$$

Fig. 8 is a comparison between the learning process of tracking by detection and this approach. The learning process of tracking by detection is to learn from binary labeled samples to update a classifier. In contrast, the learning process of this approach is to update the linear relationship between scores and feature descriptors.

Figure 8 Comparison between tracking by detection (left) and KCF (right)

Instead of training Ridge Regression by numerical methods, e.g. gradient descent, this optimization problem has a complex version closed-form solution.

$$\mathbf{w} = (X^H X + \lambda I)^{-1} X^H \mathbf{y} \quad (7)$$

X^H is the Hermitian transpose, where $X^H = (X^*)^T$. X^* is the complex conjugate of X . Normally, computing $\mathbf{w}$ via this equation costs too much computing power (This is also the reason to use gradient descent to solve linear regression instead of the normal equation). This reality prevents the tracker from using this formula in real-time. However, for a special case of X , this problem can be solved efficiently by Fourier Transform manipulation.

Instead of cropping negative examples directly from the current frame, the author uses the cyclic shift to generate numerous samples simultaneously and artificially. The cyclic shift happens in both vertical and horizontal directions. Fig. 9 is an illustration of four samples via a cyclic shift in the vertical direction. The middle image is the target. Left and right samples next to it are generated from a one-step cyclic shift up and down respectively. The most left and the rightest samples are generated from a two-step cyclic shift.

Figure 9 Illustration of the cyclic shift to generate training samples [31]

All samples generated from the cyclic shift can form a circulant matrix. For a target sample $\mathbf{x}_i = (x_1, x_2, x_3, \dots, x_n)$, a circulant matrix can be represented as below [31].

$$X = C(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ x_n & x_1 & x_2 & \dots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \dots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \dots & x_1 \end{bmatrix} \quad (8)$$

The circulant matrix has the following property [32].

$$X = F \text{diag}(\hat{\mathbf{x}}) F^H \quad (9)$$

F is a constant matrix that is independent of $\mathbf{x}$ and $\hat{\mathbf{x}}$ is the *Discrete Fourier Transform* of $\mathbf{x}$. Based on this property, the DFT of $\mathbf{w}$ can be derived.

$$\hat{\mathbf{w}} = \text{diag}\left(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}\right) \hat{\mathbf{y}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda} \quad (10)$$

$\odot$ is the element-wise product. The fraction denotes the element-wise division. The solution is a regularized correlation filter [33]. After that, the $\mathbf{w}$ can be easily recovered from the Inverse Discrete Fourier Transform. From the author's estimation, the calculation in the Fourier domain has effectively reduced the time complexity from $O(n^3)$ to $O(n)$, which is meaningful for a real-time task. Comparing to conventional tracking-by-detection approaches, this method solves a general regression problem without extracting negative samples explicitly. A closed-form solution means there is no need to loop through negative samples explicitly to update the learner. Another advantage is that this method assigns a score for each negative example, which avoids the equally weighted problem and more consistent with the nature of the object tracking task.

Although this method is ingenious and has an attractive high speed, it also has a few limitations. This method does not work well on high-speed moving objects or a low frame-rate camera. The objective is still to update a classifier rather than predict the target location directly, which is not the natural objective of an object tracking task. The biggest problem is that it cannot deal with deformation and rescaling problem. It can only track rigid objects, which is usually not the case in real applications.

2.1.4 Tracking with Deep Neural Network

Neural networks have achieved great success in many computer vision areas because of their ability to learn complex functions from large amounts of data. These tasks, like image classification, object detection, etc., allow offline learning to make the use of large-scale datasets. Due to the purpose to learn more deep features in a more complex

environment, researchers start applying neural networks in object tracking tasks. However, early neural-network-based trackers are trained online. It results in a big issue. Training neural networks is a slow process, which leads to slow tracking.

David in [18] has proposed a neural-network-based object tracking framework that can benefit from offline training. This framework is based on a *Siamese deep regression network*. Siamese networks are networks that join two inputs and produce a single output. The objective is to determine whether identical or similar objects exist.

David's approach is rather intuitively, see Fig. 10. First, a collection of videos and images with bounding box labels (coordinates) are used to train a neural network offline to track generic objects. The at test time, regions of interests (ROIs) of both previous frame and current frame are fed into the well-trained neural network and get an output of the relative coordinates of the target bounding box.

Figure 10 Training and Testing of [18]

The technical details can be summarized as the following steps, which are illustrated in Fig. 11.

- 1) Based on the chosen motion model, enlarge the bounding box in the previous frame to generate ROI.
- 2) The area in the current frame which is corresponding to the generated ROI in the previous frame is the ROI in the current frame.
- 3) Both ROIs of the previous frame and the current frame are fed into their dedicated convolutional layers to extract their own features.
- 4) The output pair of feature vectors are concatenated together and then fed to the

fully connected layers.

- 5) The fully connected layers will regress the concatenated feature vector to the predicted location of the target within the search region.

Figure 11 Network architecture of [18]

The advantages of this approach are obvious. First, it benefits from the superior learning ability of neural networks. Second, it can be trained offline which can make use of large-scale image datasets by creating artificial movements. Offline training enables this tracker can deal with high frame rates videos. Finally, compared with tracking-by-detection approaches, this framework does not have a classifier and output the coordinates directly. This feature is more consistent with the natural objective of object tracking.

2.2 Review of Event-camera-based Object Tracking Methods

Event cameras are designed to record high-speed moving objects with a high temporal resolution, where the scenarios are consistent with real-time object tracking tasks. Researchers have already developed several event-based or event-frame-hybrid frameworks to apply event camera in object tracking tasks. These frameworks demonstrate the capabilities of event cameras in object tracking.

2.2.1 Real-time Particle Tracking by Event Cameras

The first attempt to use event cameras to perform tracking is demonstrated by David [34] in 2011. In this paper, an event camera was used to track particles in high Reynolds number fluid flows. As the author pointed out, before it, optically based measurements in high Reynolds number fluid flows often require high-speed (high frame rate) imaging techniques. These cameras typically record data internally and thus are limited by the size of onboard memory. In contrast, event cameras transmit asynchronous events only when relative changes in the intensity of approximately 10% are encountered. With a microsecond temporal resolution, the storage requirement for an event camera is as small as one percent of a conventional high-speed camera.

For comparison, both an event camera and a Photron Ultima APX (a 2000fps CMOS camera with one-megapixel resolution) were set to monitor the same high Reynolds number fluid flow. The data format of an event is (x, y, t, p) . (x, y) are the coordinates of the responded pixel and t is the timestamp of the event. p is a sign bit, which is the sign of the intensity change.

Therefore, a short burst of events with a positive sign is generated when a particle enters the focus of view. Correspondingly, a short burst of events with a negative sign is generated when a particle leaves. A piece of data is illustrated in Fig. 12. Each cluster of events represents a potential particle. Purple points are events with positive polarity, which indicate potential coming particles. In contrast, blue points are events with negative polarity, which indicate potential leaving particles. As the experiment environment is highly noise-free, Fig. 12 is a good illustration that event cameras get rid of frame concept. It is sparse in the spatial domain with impressive data efficiency.

Figure 12 Illustration of event data from [34] (Color represents the parity)

The Algorithm is rather straightforward, which is a kind of clustering. The main idea can be summarized as the following steps.

1) Particle Initialization and Merge

When a new event comes in, check all 24 nearest and second nearest neighboring pixels for events within the last T microseconds, see Fig. 13. If there are no neighboring recent (within T) events, create a new particle with coordinates the same as the event and mass $m = 1$. Else if there are several neighboring recent pixels and these pixels belong to different particles, merge these particles.

2) Particle Position and Mass Update

A particle's position is updated when a new event with coordinates e joins it. The particle's mass m is decayed by multiplying $e^{-\frac{t}{T}}$ and will increase 1 for each newly joined event. Here t is the time interval since the last particle event. Only particles with a mass larger than a predetermined value (5 is used in the author's paper) are considered alive. The new position is computed as the weighted average of e and the particle's old position $\mathbf{p}_0$, $\mathbf{p}_n = \frac{1}{m} \mathbf{e} + \frac{m-1}{m} \mathbf{p}_0$.

Figure 13 Check if there are neighboring events within time interval T

The author has done a thorough comparison of performances between the event camera and the high-speed CMOS camera under different environment parameter settings. The comparison result indicates that event cameras can achieve performance as good as conventional high-speed camera. Meanwhile, the storage requirement and computation power requirement of event cameras are much smaller than conventional high-speed cameras.

Even though this experiment demonstrates that the event camera can complete high-speed tracking tasks with simple and efficient algorithms, there are still issues cannot be ignored. First, this experiment is conducted in a highly noise-free environment. However, object tracking tasks in real life are much more complicated where noise and background clutter cannot be ignored. Second, particles can use clustering because they do not suffer from deformation and rescale issue. Therefore, when trying to apply event cameras to more complex real-life tracking problems, more powerful algorithms are needed. These algorithms should extract more information from the sparse data generated by event cameras.

2.2.2 Object Tracking by Cooperation between Frame-based and Event-based Cameras

As discussed in Chapter 2.2.1, event cameras can only provide sparse data when there is significant pixel-level intensity change. This means much valuable information is lost, e.g. RGB information, background information and so on. These limitations can profoundly limit the capability of an event camera to track complicated objects, especially when prior knowledge of objects is unavailable, which means the tracker does not know what it will track until the first frame is displayed.

To achieve the purpose of tracking objects in real-life scenarios, Liu in [16] in 2016 provided an algorithm that makes use of both event-based cameras and frame-based cameras. The main idea of this algorithm is to divide tracking tasks into three stages, which are shown in Fig. 14.

- 1) Use a cluster-based tracking method to generate regions of interest (ROIs).
- 2) Use a convolutional neural network to classify ROIs as foreground and background.
- 3) Use a particle filter to infer the target location from the selected ROI.

Figure 14 Architecture of event-frame cooperation framework [16]

The event-based camera only plays a role in the first stage. The cluster-based tracker is solely based on events to generate ROIs. Like the method in Chapter 2.2.1, an incoming event is assigned to the nearest existing cluster or creates a new cluster for this event if no cluster is close enough to it. A cluster is visible only when it contains enough number

of events and dies out when there is no new event to support them within a threshold time interval. The ROIs are the corresponding crops in frame-based images which have the same coordinates as the visible clusters, see green patches in Fig. 15. If the relative motion between the event camera and the object are small, it is possible that too few events are generated to make the target selected as an ROI. Therefore, another ROI centered at the object's position of the previous frame is added, which is the blue patch in Fig. 14.

The core of the second stage is a CNN-based classifier. The network is trained offline with positive samples, which are the foreground, and negative samples, which are the background. The ROIs generated in the first stage are fed into this classifier and each one gets a likelihood score for being foreground.

The advantage of this CNN-based classifier is that it is also an embedded sliding window detector in this system. In the first frame, CNN uses a sliding window mechanism over the whole frame to detect the initial location of the object being tracked. This means there is no need for an external detector to provide an initial bounding box. Using the same idea, this tracker can recover from tracking failure. If the maximum score of all ROIs is below a threshold, it is highly possible that the target is out of the view. In this condition, sliding window detection can be performed over the next frame to generate a confidence map.

The final stage tracking algorithm is based on particle filtering. The author has demonstrated that the particle filter in the last stage can improve the accuracy of the prediction of the location significantly.

The advantages of this combination framework are obvious. First, it makes the use of the strong learning ability of CNN to deal with the complicated environment. Second, it uses an event camera to play a complementary role to the conventional frame-based camera. The event camera proposes a limited number of regions of interest for the CNN classifier to classify, rather than other approaches usually use sliding window mechanism to handle much more image patches. This approach saves computing power significantly, which is critical for many mobile application scenarios, like auto-driving or field surveillance.

However, the limitations of this approach are obvious. As said previously, the event camera only plays a complementary role. This approach still needs the full picture information, which is from a frame-based camera, to distinguish between foreground and background. In this framework, the event camera does not demonstrate its ability to complete visual object tracking tasks solely on its own. This means the tracking system will need more storage to buffer both events data and frames data, which is not desired for the memory saving purpose for many real-time mobile object-tracking tasks. A smaller data scale is one of the most attractive features of event cameras. Another problem is that the classifier is trained offline for dedicated use which can only track specific targets. It contradicts our purpose to track generic objects in real-time.

2.2.3 Event-based Tracking by Detection Framework

Inspired by the spirit of the classical tracking-by-detection paradigm for conventional frame-based cameras, Ramesh in [35] proposes a pure event-based tracking-by-detection framework. This framework is similar to what is illustrated in Fig. 6. However, there are no more frames.

During the tracking process, several regions of interest (ROIs) are sampled from the search region which is around the object location in the previous period. In frame-based tracking-by-detection approaches, the features of an ROI are extracted from the image patch corresponding to the ROI. However, in this pure event-based tracking-by-detection framework, features are extracted from the events within the ROI in a classification period. Each event is encoded as a local descriptor [36]. Then a bag-of-words model is used to summarize these local descriptors into a global feature vector. This global feature vector is fed into an online updated SVM classifier to determine if the object is present in this ROI.

When an ROI is classified as the object, the classifier is updated. Small padding is applied to the ROI to ensure the object will exist in the search region. Events within the ROI form the positive sample. In contrast, the events outside the ROI but within the research region forms the negative sample. However, a single pair of training samples is meaningless to update the classifier. To solve this problem, the Bayesian Bootstrap is

used to augment the training samples. With the augmented training samples, the classifier is updated.

Another characteristic of this framework is that the classification period is dynamic. It is not a fixed time interval but determined by the number of events within the search region. When the number of events received within the search region reaches a predetermined value, the classification will happen. This mechanism copes with different speeds well.

This work takes advantage of the classical tracking-by-detection paradigm and proves that a pure event-based tracking system is possible. However, this framework still relies on an online updated classifier. The online learning mechanism limits the characteristic of the low latency of event cameras. Also, this framework is only suitable for rigid object tracking, which does not take into consideration of rescaling and deformation.

2.2.4 Event-based Motion Compensation Tracking Framework

Recently years, optical flow estimators [38] [39] are used for motion compensation. These approaches are time-consuming for real-time applications. Mitrokhin in [37] proposes a pure event-based featureless motion compensation moving objects detection and tracking pipeline. As shown in Fig. 15, in this work, a warp field is introduced to map any event (x, y, t) within the time interval $(t_0, t_0 + \delta t)$ to its location (x', y') at t_0 . An *event count image* is constructed with the number of warped events at each pixel. A *time image* is constructed with the mean timestamp of all warped events at each pixel.

The warp field is modeled with a four-parameter motion model which consists of shift along the x-axis, shift along the y-axis, expansion and 2D rotation. This motion vector is obtained by a coarse motion minimization on the time image and a fine motion refinement on the event count image. Moving objects are detected based on the inconsistencies with the estimated global camera ego-motion. The tracking is completed by a simple Kalman Filter with a constant acceleration motion model.

Figure 15 warping process of events within an interval

The author declares the detection is robust. The tracker plays a complementary role. It only works when occlusions or wrong detections happen. As the framework is well parallelized, the detection can run in real-time. This framework can describe each moving object in detail. However, the process to obtain the warp field is an optimization problem, which is still time-consuming.

Chapter 3. Spacetime Event Cloud

In a conventional vision system, intensity information of all pixels is sent out at each time. The temporal information captured is discrete and limited by the frame rate (vary from tens of to thousands of frames per second). Thus, conventional video streams are dense in the spatial domain but sparse in the time domain. In an event-based vision system, an event camera outputs an event whenever the intensity change of a certain pixel exceeds the threshold. Each event carries only the spatial coordinates (x, y) of the pixel reporting a change and a timestamp (t) indicating when that change happened [40]. Event streams are asynchronous and have a much higher temporal resolution from microsecond to nanosecond level. Thus, event streams are sparse in the spatial domain while dense in the time domain. Fig. 16 can help understand the data format difference between event cameras and conventional cameras.

Figure 16 Illustration of the data format of event camera and conventional camera

Although there are many different approaches to extract features from the unique asynchronous event streams, they all share a common point to treat the event streams as temporal sequences. The concept of temporal sequences and matrix of pixels inherited from conventional video analysis of frames is applied to event streams in a deeply

intuitive way. However, treating event streams as temporal sequences do not fully utilize the spatial sparsity of event streams. Meanwhile, the dense temporal information captured is diluted. In this work, a novel perspective is provided to think out of the conventional concept of temporal sequences and view the event streams in a three-dimensional spacetime continuum. In conventional 3D space, a point is denoted as (x, y, z) while in the world of event cameras, a point (event) in 3D space is denoted as (x, y, t) . The continuous event stream forms a cluster of points (events) in 3D space and is named as *Spacetime Event Clouds* [1].

Figure 17 (a) Conventional approach treats event streams as temporal sequences. For an event e_i , it carries the pixel location (x_i, y_i) reporting an intensity change and a timestamp t_i . (b) Proposed approach treats event streams as 3D spacetime event clouds. An event e_i becomes a point in 3D space denotes as (x_i, y_i, t_i) .

3.1 From Temporal Sequences to Spacetime Event Clouds

In the human being's vision system, the world consists of three dimensions of space plus one dimension of time. Similarly, camera vision systems see the world as two dimensions of image space plus one dimension of time. In Newtonian view of space and time, time is a measurement separated from the space dimensions and it is an independent variable flowing on its own [41]. In conventional video analysis, algorithms are intuitively developed under the Newtonian idea of time, where time is considered as a measurement of duration, sequential order or frequency of motions. Thus, both frames from conventional cameras and event streams from event cameras are treated as temporal sequences. When treating event streams as temporal sequences, an event i is denoted as:

$$\mathbf{e}_i = ((x_i, y_i), t_i) \quad (11)$$

The event streams are time-series data recording intensity change in image space in chronological order. As shown in Fig. 17(a), an event i is represented by the pixel address (x_i, y_i) reporting that change and a precise timestamp t_i indicating when that change happened.

In modern physics, scientists propose to fuse the three dimensions of space and the one dimension of time into a single four-dimensional continuum, where time is a dimension identical to the other three dimensions [41]. Inspired by the modern physics' understanding of time, here, this paradigm fuses the two dimensions of image space and the one dimension of time into a three-dimension continuum in the event-based vision system. As shown in Fig. 17(b), time t is identical to x, y spaces and measured with a numerical number with a physical meaning. In the proposed approach, event i is denoted as:

$$\mathbf{e}_i = (x_i, y_i, t_i) \quad (12)$$

An event becomes a point in a 3D space and event streams form 3D spacetime event clouds.

In the event-based system, event streams are divided into multiple segments for feature extraction. Hard segmentation divides events by a fixed time interval or a fixed number of events, while soft segmentation adaptively obtains segments according to some certain predefined rules. In the selected time interval T , assume a total number of n events are generated by an event camera with a $N \times M$ resolution. When treating event streams as temporal sequences, a set of events within time interval T is expressed as:

$$S_{temporal}^T = \{\mathbf{e}_i = ((x_i, y_i)|t_i \in T, i = 1, 2, \dots, n)\} \quad (13)$$

When treating event streams as 3D spacetime event clouds, a set of events within time interval T is expressed as:

$$S_{spacetime}^T = \{\mathbf{e}_i = ((x_i, y_i, t_i)|t_i \in T, i = 1, 2, \dots, n)\} \quad (14)$$

In the temporal sequence approach, no matter modeling events as postsynaptic potentials for neural spiking pattern classification or interpreting event streams as a bag of events for joint probability distribution classification, feature extraction is conducted on set $S_{temporal}^T$ and events are accumulated overtime on the image plane. Thus, an $N \times M$ array is required to store the spiking patterns or probability patterns as shown in Fig. 18(a). Processing of event sequences becomes the processing of dense arrays. As event streams are extremely sparse in the spatial domain, only around 10% of the array is occupied by valid data. The advantage of spatial sparsity is largely diluted, and memory cost is increased. The accumulation of events over a short time interval also dilutes the dense time information captured by event cameras. The temporal order among events within that interval is lost during integration.

In the proposed approach, events streams are treated as spacetime event clouds, which is a set of points (events) in 3D space. Spatial features and temporal features are fused in the 3D spacetime continuum as shown in Fig. 18(b). Processing of event streams becomes set operation over $S_{spacetime}^T$. It fully utilizes the spatial sparsity and achieves an effective usage of memories. As each event is a point in 3D space denoted with coordinates (x, y, t) , there is no longer integration over time. Thus, the dense time

information is also well preserved and is transformed into geometric information. As conventional neural network models are no longer suitable to process sparse set data, a network architecture to learn geometric distribution features of event clouds in 3D space is needed.

Figure 18 (a) In time sequence approach, accumulation of events over time dilutes spatial sparsity of event streams and lose dense temporal information captured. (b) Spacetime event clouds approach preserves spatial sparsity and temporal density of event data.

3.2 Description of Object Tracking Task based on Spacetime Event Clouds

Getting rid of the constraint of the frame concept in conventional cameras, the description of the object tracking task should be changed. First, instead of a series of frames, a moving object and corresponding changing background captured by an event camera is a stream of events, see in Fig. 19, which is can be recorded as:

$$S_{Stream} = \{\mathbf{e}_t^i | 0 < t \leq T_{total}, 0 < i < i_t\} \quad (15)$$

T_{total} is the total duration in which the target object is being tracked and i_t is the total number of events that happened in t . The task is how to track the location of the target object in S_{Stream} .

Figure 19 Event stream of a moving object with changing background

Normally, the temporal resolution of the event camera is in the microsecond level. At each moment, there are zero or few events, which do not contain enough information. To extract meaningful spatiotemporal features, the event stream needs to be divided into multiple fragments along the time axis. As mentioned previously, people normally adopt

hard segmentation or soft segmentation to deal with event data. This thesis simply uses hard segmentation to divide the event stream equally along the time axis. As shown in Fig. 20, Events within each time interval form a spacetime event cloud.

$$S_{event-cloud}^n = \{ \mathbf{e}_t^i | t_{n-1} < t \leq t_n, 0 < i < i_t, \mathbf{e}_t^i \in S_{Stream} \} \quad (t_0 = 0) \quad (16)$$

t_{n-1} is the ending timestamp of the $n - 1_{th}$ event cloud and $t_{n-1} + 1$ is the starting timestamp of the n_{th} event cloud. Each event cloud consist of the events of the moving object and the events of the background.

Figure 20 Divide event stream into multiple event clouds.

The next important thing is the bounding box under spacetime event clouds framework. In conventional RGB cameras, the bounding box is the rectangular best fits the object in the current frame. However, using event cameras, there are no frames anymore. Therefore, the bounding box here should be a constraint that can be applied for the whole interval. As illustrated in Fig. 21(a), the bounding box in the n_{th} interval is a constraint that best fits all events belong to the target object in the n_{th} event cloud if projecting them into xy-plane. Also, it can be treated in another way in Fig. 21(b). This is a constant spatial constraint during the n_{th} interval to best fit all events of the target object, i.e. a

3D bounding boxes.

With all the above definitions, the methodology of tracking under spacetime event cloud perspective can be summarized as to extract features from both current event cloud and previous event cloud, and to use extracted features to predict the location of the target object in the current event cloud.

Figure 21 Bounding box under spacetime event cloud framework. (a) Projection view. (b) 3D view.

3.3 Methodology of Spacetime Event Clouds-based Siamese Regression Network

This thesis proposes a Siamese Regression Network to deal with spacetime event clouds to complete object tracking tasks under a complex environment. As illustrated in Fig. 22, this network can be trained offline generically with a well pre-labeled dataset and track online for new target objects.

During the offline training stage, current (n_{th}) spacetime event cloud and previous ($n - 1_{th}$) spacetime event cloud are fed into two identical learners, which are used to extract their features (including target object and background) separately and simultaneously. As explained previously, event clouds encode both spatial and temporal information of the moving object into their geometric distribution. Therefore, the extracted feature vector is assumed to include all spatiotemporal information to describe the target moving object.

These extracted features of both the current and the previous spacetime event clouds are concatenated together and fed into a neural network to regress to the label, which are coordinates of the target bounding box in current interval (current spacetime event cloud).

Figure 22 (a) Using a well-labeled dataset to train a generic spacetime-event-cloud-based tracker. (b) During the test stage, this network can track the new target object.

During the test stage, the current spacetime event cloud, in which the bounding box coordinates are unknown, and the previous spacetime event cloud, in which the

bounding box is known already, are fed into the pair of feature learners. Then, the extracted features are concatenated and fed into the well-trained fully connected neural network to regress to the outputs. These outputs are the prediction of the bounding box coordinates in the current time interval (current spacetime event cloud).

It can be found that this approach is an intuitive end-to-end framework. This framework directly predicts the coordinates of the bounding box, which avoids training a classifier to distinguish between the foreground and the background. Compared with classical tracking-by-detection approaches, it is more consistent with the objective of object tracking to track the location of the target. Another thinking is that due to the concept of spacetime event cloud, the exact value of time is meaningful, some features related to time, e.g. motion, can be encapsulated into the extracted features. Finally, this approach only uses the data from event cameras, which is much more memory-friendly and computation-friendly.

3.4 Architecture of Spacetime Event Clouds-based Siamese Regression Network

The architecture of this spacetime event clouds-based Siamese regression network is simple. As shown in Fig. 23, it consists of a preprocessing stage, a pair of PointNet-based feature extraction networks and a fully connected regression neural network. The preprocessing stage will generate two sub event clouds of interest for the current event cloud and the previous event cloud respectively. These two sub event clouds are fed into a pair of PointNet based feature extraction networks with identical structure. The outputs of these two feature extraction networks are two vectors that summarize the features of the two sub event clouds respectively. Then, the two feature vectors, as explained in Chapter 3.3, are concatenated together and fed into final fully connected neural network layers to regress to the predicated location of the target object.

Figure 23 Architecture of Spacetime Event Clouds based Siamese Regression Network

3.4.1 Preprocessing Stage: Search Region Generation

When trying to predict the location (bounding box) of the target object in the current spacetime event cloud, the already known information is the current spacetime event cloud, previous spacetime event cloud, and the estimated location of the bounding box in previous spacetime event cloud. If taking into consideration the whole current event cloud, the difficulty of tracking the target object is too high because of the low signal to noise ratio (target to background ratio). Also, the computing power will be wasted too much on the uninterested background. Therefore, this thesis adapted an efficient mechanism to generate a restricted search region.

To make the tracker know where to search the target object in the current event cloud in terms of x and y , the estimated location in the previous event cloud should be a good (unbiased) guess for the location in the current event cloud, see in Fig. 25. More precisely, the search region should center at

$$\mathbf{c}_{guess} = (x_{guess}, y_{guess}) \quad (17)$$

$$\mathbf{c}_{guess} = \mathbf{c}_{previous} \quad (18)$$

$\mathbf{c}_{previous}$ is the estimated center of the bounding box (the red box in Fig. 24) in previous event cloud. $\mathbf{c}_{guess}$ is the unbiased mean location of the center of the bounding box in the current event cloud. This estimation is equivalent to a constant position motion model. Other more sophisticated motion models can also be used.

The search region of the current event cloud should center at $\mathbf{c}_{guess}$ with k_1w width and k_2h height. w and h are the width and height of the bounding box in the previous event cloud respectively. k_1 and k_2 are two coefficients which define our search range of target object in the current event cloud in terms of x and y . As illustrated in Fig. 24, the two crops centered at $\mathbf{c}_{previous}$ (as $\mathbf{c}_{guess} = \mathbf{c}_{previous}$) with k_1w width and k_2h height in both current and previous event clouds (which are bounded by blue boxes) are final *sub event clouds of interest*, which will be fed into the pair of PointNet based feature extraction networks.

Figure 24 Search Region Generation

3.4.2 PointNet for Spacetime Event Clouds-based Feature Extraction

After restricting the search region to a certain sub event cloud, the next consideration is how to process the data with the event-cloud format. As explained previously, in spacetime event cloud framework, all events lie into a 3D spacetime continuum with x -axis, y -axis and t -axis. These three dimensions should be treated identically to make the most use of temporal implications of event clouds.

To treat x -axis, y -axis and t -axis identically, dealing with events in the 3D spacetime continuum is equivalent to dealing with normal points in the conventional 3D spatial continuum. Therefore, processing event clouds should share the same idea of processing point clouds when extracting local and global features from them. In Fig. 25, an event cloud in x, y, t -continuum is similar to a point cloud in x, y, z -continuum. The only difference is that an event cloud is an action or a change of an object while a point cloud is the 3D structure of a static object. The idea of this thesis is to benefit from point cloud techniques to extract temporal information directly from event clouds. Furthermore, dealing with point clouds in x, y, z -continuum is virtually processing all points at the same moment in t -axis. Similarly, dealing with event clouds in x, y, t continuum is to process all points at the same point in fourth dimension if thinking of event clouds at the 4D spacetime continuum.

Figure 25 (a) The event cloud of a moving dog in x, y, t -continuum. (b) The point cloud of a rabbit in x, y, z -continuum.

PointNet [13] is a neural network architecture that directly takes point clouds as input

and it learns to summarize the geometric features of the input point clouds. A point cloud is denoted as a set of 3D points $S = \{\mathbf{p}_i | i = 1, \dots, n\}$, where each point $\mathbf{p}_i$ is a vector of its coordinates (x, y, z) . The architecture of PointNet is shown in Fig. 26(a). Each input point $\mathbf{p}_i$ is processed by a shared multi-layer perceptron network and it is trained to capture different properties of the input set. As points in the set are unordered, the input sequence of points should not affect the output. A single symmetric function, max pooling, is applied. The network learns a set of optimization functions to select informative points of the input set of points. The fully connected layers aggregate these learned optimal values into a global descriptor of the entire shape of point clouds. Though simple, PointNet architecture demonstrates universal approximation ability of continuous set functions. A Hausdorff continuous symmetric function $f: 2^x \rightarrow \mathbb{R}$ can be arbitrarily approximated by PointNet as

$$\Phi(S) = \text{MAX}_{\mathbf{p}_i \in S} \{h(\mathbf{p}_i)\} \quad (19)$$

where $\mathbf{p}_1, \dots, \mathbf{p}_n$ is elements in S ordered arbitrarily, h is a continuous function approximated multi-layer perception networks, and MAX is an element-wise max-pooling operator. $\Phi(S)$ is the extracted feature vector of the point cloud.

Considering that PointNet is highly efficient and it demonstrates strong performance in point clouds classification, PointNet is selected and trained for spacetime event clouds feature extraction in this work. As shown in Fig. 26(b), our input is a set of event clouds within time interval T denotes as $S^T = \{\mathbf{e}_i | i = 1, \dots, n\}$, where each event $\mathbf{e}_i$ is a vector of its coordinates (x, y, t) . The universal approximation becomes

$$\Phi(S^T) = \text{MAX}_{\mathbf{e}_i \in S^T} \{h(\mathbf{e}_i)\} \quad (20)$$

where $\mathbf{e}_1, \dots, \mathbf{e}_n$ are elements in S^T ordered arbitrarily, h is a continuous function approximated multi-layer perception networks, and MAX is an element-wise max-pooling operator. PointNet architecture models set functions that directly take a set of events S^T as input and outputs a global feature vector $\Phi(S^T)$ of the input event cloud.

Figure 26 (a)PointNet is a neural network architecture that directly consumes point clouds for feature extraction. (b)In this work, for the first time, PointNet is applied to consume spacetime event clouds for feature extraction in event camera-based tracking system

3.4.3 Regression Fully Connected Neural Network Layers

In Fig. 23, the outputs of the two PointNet-based feature extraction networks are two feature vectors of two cropped spacetime event clouds. They are the feature summarization for the selected search regions of the current and the previous event clouds respectively. These two feature vectors are concatenated together and fed into a three-layer fully connected neural network.

$$\mathbf{F}_{current} = \Phi_1(S_{current}) \quad (21)$$

$$\mathbf{F}_{previous} = \Phi_2(S_{previous}) \quad (22)$$

Φ_1 and Φ_2 are learned functions of two PointNet-based feature extraction networks.

$S_{current}$ and $S_{previous}$ are event sets of generated search regions of the current and the previous event clouds respectively. $\mathbf{F}_{current}$ and $\mathbf{F}_{previous}$ are extracted feature vectors with the same dimension.

This fully connected neural network directly regresses the concatenated feature vector into relative coordinates of the corners of the predicted bounding box. In Fig. 23, in order to represent a rectangular bounding box, two coordinates are needed. Mathematically,

$$\mathbf{B}_{predicted} = r(\mathbf{F}_{current}, \mathbf{F}_{previous}) \quad (23)$$

$$\mathbf{B}_{predicted} = (x_1, y_1, x_3, y_3) \quad (24)$$

r is the learned regression function of the fully connected network. $\mathbf{B}_{predicted}$ is the predicted bounding box. (x_1, y_1) is the top left corner and (x_3, y_3) is the bottom right corner. They are relative coordinates in the generated search region.

3.5 Experiments

3.5.1 Dataset

The dataset used in this experiment is the event camera version Tracking Dataset [42] from Y.Hu [43] [44]. The original Tracking Dataset has 77 single object tracking sequences. The average number of frames per sequence is 605. Hu has included 67 categories when generating event camera version Tracking Dataset. Due to the problem of reading the category “Vid_D_person”, this experiment has excluded it. Therefore, the total number of sequences considered in this experiment is 66.

The original sequence of category “Trans” and corresponding event camera version are shown in Fig. 27. The first row is the original “Trans” sequence. The second row is the sequence replicated by the event camera. For display and comparison purposes, it is reconstructed into frames, but it is virtually an event stream as described previously in this report. The red boxes are ground truth of bounding boxes. The setup details and recording procedures can be referred to from [43]. The important notice is that original

sequences are displayed at 30 Hz and the bounding boxes rate is also set to 30 Hz in the event camera version dataset.

Figure 27 Original sequence & event camera version

3.5.2 Experiment on the Capability of Tracking Similar Sequences

The first experiment is done to verify the capability of this framework to track sequences to which the tracking system is familiar. Each sequence is divided into two parts, the first part consists of the first 90 percent events and the second part consists of the last 10 percent events. The first part of the data is shuffled generally and divided into two parts. 4/5 of them are used as the training dataset and the other 1/5 of them are used as the first test dataset (Test Dataset NO.1). The remaining 10 percent of events form the second test dataset.

The first test dataset is used to verify the generalization capability of this framework to track almost the same sequences. In statistics language, this test dataset shares the same distribution of the training dataset. The thinking behind the second test dataset is that the last 10 percent of the data are from the same sequences which are also used for training, but the movement status may be totally different from the training dataset. Specifically, this test dataset mostly captures the final stage of the movement of sequence.

Another important notice is that both the previous event cloud and the current event cloud are down-sampled to the same pre-defined normalized size before being fed to PointNet-based feature extractors, see Fig. 28. The assumption behind down-sampling is that only a subset of events can capture enough temporal and spatial information of

the full event cloud.

Figure 28 Down-sampling of event clouds

There is also an intuitive explanation for the reasonability of down-sampling. As described in Chapter 3.4.2, PointNet first expands each event locally to a larger size output temporal-spatial feature vector. After that, a channel-wise max-polling layer is applied to summarize the event cloud to a global temporal-spatial feature vector.

For example, if the size of the feature vector is 1024 (the number for output channels of multi-layer perceptron), the summarized global feature can come from 1024 events at most, see in Fig. 29. Therefore, as more and more events provided to the network, the marginal information added to the network becomes less and less. Random down-sampling is a good tool to avoid spending computation power on redundant data.

Figure 29 Illustration of feature extraction

The predetermined normalized size is one of the most important hyperparameters in this framework. First, it illustrates the minimum number of events that can form as a representative event cloud to describe a moving object in a certain time interval. If this number is set small, the event cloud may not be able to contain enough spatial or temporal information, which will result in a poor tracking performance. If this number is set too large, it may contain redundant information, which is a waste of memory and computation resources.

More importantly, this number determines the scale of data to be processed. It is explained in detail in later Chapter 4.1, Analysis and Conclusion. It determines the complexity of the network, thus determines how this framework is superior to conventional camera-based trackers in terms of resource requirement.

Following Table 1 summarizes the accuracy of the prediction of the current bounding box position given the previous bounding box position under three different down-sampling rates. The mean loss is defined by the average deviation from ground truth in terms of pixels.

Table 1 Evaluation of tracking capability on similar sequences

Down-sampling Rate (Event)	Mean Loss (pixels per coordinate)		
	Training Set	Test Dataset NO.1	Test Dataset NO.2
128-point	2.60	3.82	6.40
256-point	2.51	3.35	6.17
512-point	2.34	3.15	6.07

It can be found that along with the increase of the down-sampling rate, both training accuracy and test accuracy improve. When setting the size of the representative event clouds to 512 points, this framework can achieve the best accuracy result for this dataset. Another experiment has done with the down-sampling rate set to 1024, but it does not show significantly more outstanding performance than down-sampling of 512.

3.5.3 Experiment on the Capability of Tracking New Sequences

To evaluate the performance of this framework on tracking thoroughly new sequences (which are different from the sequences used for training), Tracking Dataset is divided into two parts. 5 randomly selected sequences make up the test dataset. The other 60 sequences make up the training dataset. Like Chapter 3.5.2, the first thing to evaluate is the accuracy of current bounding box prediction given previous bounding box. The performance is evaluated under down-sampling rate of 512 because it demonstrates the most optimal performance in the last experiment. Table II summarizes the experiment results.

Table 2 Evaluation of tracking capability on new sequences

	Training Set	Test Dataset
Mean Loss (pixels per coordinate)	2.34	6.72

Comparing Table 2 with the last row of Table 1, it can be found that even though the accuracy on newly-looked sequences shows a little descent than the accuracy on ever-seen sequences (the difference is less than 1 pixel on average), this framework still demonstrates the capability to track novel objects with offline training only.

3.6 Performance Analysis

3.6.1 Scale of Data Size Analysis

The first aspect to be analyzed is the scale size of the data. One reason accounting for the prevalence of event cameras is its potential to generate a relatively small amount of data. However, it is a kind of fake proposition which is illustrated in column 2 and column 5 of Table 4. A 30-frame grey level conventional camera sharing the same spatial resolution with the event camera (which is 180×240 in this thesis), will generate

1.3MB data stably. However, in contrast, with the purpose to record a normal video sequence, an event camera will generate 160KB ~800KB data.

The behind reason is that even though data generated from event camera is sparse in space, the temporal resolution is ultra-high compared with conventional cameras, which reaches 1M per second. In addition, the data size for each pixel is 7 folds more than the pixel of a conventional grey-level event camera. To generate less data than conventional cameras, the target object or field must be relatively static. However, the natural characteristics of event cameras make them mostly be applied in high-speed scenarios.

Fortunately, as explained in Chapter 3.5.2, only a subset of events is needed to fully describe an event cloud. Due to the spacetime event cloud concept has a strong descriptive capability of moving targets, both spatially and temporally. The down-sampling process before the feature extraction stage in this framework reduces the data scale to a promising level, $3 \times 512 \times 30 = 46K$ (in terms of 512-point down-sampling), which is around one thirtieth of conventional cameras. The comparison details can be found in Table 3.

Table 3 Scale size of Data

	Raw events	Treat events as 3D points		Cast events back to Frames / Conventional camera
		Without Down-sampling	With Down-sampling	
Camera Resolution	180×240	180×240	180×240	180×240
Events per second	20K~100K	20K~100K	512×30	
Frames per second				30
Total memory / Total data processed	160KB ~800KB	60K~300K (3×20K~100K)	46K (3×512×30)	1300K (1×180×240×30)

3.6.2 Speed Analysis

One of the most important concerns of a tracker is its speed. In terms of a conventional frame-based tracker, the speed is the frame rate in which the tracker can predict bounding box effectively. Table 5 lists a few trackers which are famous for speed [45].

Table 4 Speed Comparison

Trackers	KCF [31]	GOTURN [18]	ASMS [20]	Spacetime
FPS / (Predictions per second)	65	100	125	≈ 140

For events cameras, there is no frame concept, the speed can be defined as how many predictions can be done in one second, which is equivalent to FPS for conventional camera-based trackers. After a great number of experiments, it indicates that this framework can make around 140 predictions per second with regular hardware resources (a single GTX-1060 GPU), which is attractive to high-speed motion real-time tracking tasks.

3.6.3 Accuracy Analysis

As illustrated in Chapter 3.5, this spacetime event clouds-based regression neural network shows acceptable tracking accuracy on ever-seen moving objects. However, the performance decreases significantly on novel objects, even on the same object but with different moving statuses. This result within the expectations. There are two main reasons.

The first reason is the quality of the dataset. This dataset is an event camera version replication of Tracking Dataset [42]. The quality is strongly limited by the quality of original videos, i.e. noises and low illumination. When using the event camera to record original videos, there is one important problem cannot be ignored. The moving object in real-world is continuous, which highly matches the high temporal resolution characteristic of the event camera. However, the video displayed is discrete, it is limited by the frame rate, which is 30 FPS in this dataset recording setting. Therefore, this recorded event camera version dataset is not as continuous as it is supposed to be.

The second reason is the scale of the dataset. Event-based object tracking datasets of available to the public are limited. In general, neural networks are suitable for tasks with rich training datasets. However, in this thesis, there is only as few as 60 sequences to train the network. The lack of training data is the biggest hurdle to improve the performance of this framework.

3.6.4 Effect of Bagging Augmentation

As explained in Chapter 3.6.3, open accessible dataset for event camera object tracking is limited. One straightforward idea is to find a suitable way to do data augmentation. As demonstrated in Chapter 3.5.2, only a subset of events from an event cloud, generated from random down-sampling, can represent a moving target well. Based on this idea, a reasonable idea based on bagging can be used to augment the training dataset.

From an event cloud, we can repeat picking several subsets with the same size randomly. These subsets will have the same label (same bounding box). To do this, the size of the training dataset is enlarged to a more reasonable level.

However, this bagging augmentation idea does not result in significantly better results. This result indicates that bagging does not bring new information to the network. It again repeating emphasizing the suitability of the down-sampling idea.

Chapter 4. Event-wise Continuous Tracking Framework

In this chapter, the limitations of the framework in Chapter 3 is analyzed first. Then, based on these analyses, this thesis investigates new ideas to improve it further. The purpose of this part is to design a framework that is more consistent with the characteristics of the data format of event cameras. Also, this framework should demonstrate the capability to finish the tracking task in a continuous manner. How should define this continuity, this thesis will explain in detail later. After that, a series of experiments will be designed to evaluate the ideas.

4.1 Limitations of PointNet-based Siamese framework

One of the most specific characteristics of the event camera is that there is no frame concept anymore. As explained earlier, the conventional camera describes a moving object in a video, which is a series of discrete frames. This kind of video only provides information densely in certain predetermined frequency, no matter how the motion status changes. The temporal information between frames is lost by nature. The only temporal information a computer vision system can get is the relative position of frames along the time axis. In other words, the temporal information is suppressed into frames.

This is the true hurdle faced by conventional computer vision systems to do video analytics. In contrast, event cameras generate data based on pixel-level intensity change in an ultra-high time frequency. A moving object is described as an event stream with negligible temporal discontinuity. In other words, compared with conventional cameras, event cameras extract a huge amount of temporal information from frames. It has been illustrated in Fig. 5.

This is also the declaration in this thesis why we should not accumulate or compress event streams into image planes in order to fit current 2D computer vision techniques. When adopting the idea of the spacetime event cloud, it is equivalent to interpret a moving object or an action into a 3D geometric distribution in a 3D spacetime coordinate system. There are two space axes and another time axis as normal. However, in this description, there is no physical meaning difference between three axes. If there is a way to extract geometric information from spacetime event cloud, it can be said that spatial and temporal information of the motion or the action are extracted simultaneously. PointNet [13] is a powerful state-of-art technique to deal with 3D point clouds.

After solving the first problem to extract enough spatial and temporal information from event clouds. The other problem is how to use extracted feature descriptors to track objects. Siamese network is an intuitive way to solve this problem. Siamese is a class of networks that take into two pieces of input vectors and output comparable output vectors. To cope with this idea, Chapter 3 uses a two clouds-based network to work on both previous time interval and current time interval, shown in Fig. 23.

Figure 30 2D bounding boxes to 3D bounding cuboids

Here the limitation occurs. With the spacetime event cloud concept, we get rid of the frame constrain. However, the event stream is still divided into several discrete time intervals. In terms of object tracking task, discrete bounding boxes are only improved into 3D discrete bounding cuboids, as shown in Fig. 30. It is still not as continuous as it is expected.

4.2 Nature of Continuity of Object tracking

Previous chapters have illustrated that event cameras have much higher temporal resolution than conventional RGB cameras. However, the framework in Chapter 3 still fragments event data with a fixed time interval, which implicitly restrains the advantage of high temporal resolution. The ultra-high time resolution makes event data from event

cameras much more consistent with the nature of continuity of a moving object. An ideal event-based tracking system should make the use of this characteristic as much as possible. Based on this kind of spatial-sparse and temporal-dense data, this thesis defines an ideal tracking system based on event camera as below:

There should be a bounding box for every timestamp. This bounding box is a summary of the target looks back a predetermined time interval from current timestamp. Every time a new event comes in, the tracking system should do a corresponding response. If the new event is not related to the target, the bounding box should respond with no update. If the new event is related to the target, the bounding box should update accordingly. Each step, the update is minimal. Along the time axis, the bounding box can form a smooth 3D bounding volume.

This definition emphasizes the event-wise response of the tracking system, as shown in Fig. 31.

Figure 31 Tracking system response at each timestamp

The bounding box here is a cross-section of 3D smooth bounding volume at a timestamp, not something updated according to a frame. The comparison between bounding cuboids and smooth bounding volume is illustrated in Fig. 32. The bounding cuboids in Chapter 3 are discrete based on the length of the selected time interval. In contrast, the bounding volume should start with the first given bounding box and be adapted with minimal change (or no change for background events). It should be smooth from the beginning to the end of a given video sequence.

Figure 32 Bounding Boxes VS Bounding cuboids VS Bounding volume

4.3 Event-wise Recurrent Network

Recently, Sekikawa in [19] proposes EventNet to process event data asynchronously and recurrently. EventNet uses a temporal coding scheme to process raw spatial data (x, y) and raw temporal data (t) separately.

EventNet uses a 2D version Multi-layer perceptron to expand space information (x, y) of each event to a feature vector with the predetermined length. The temporal information of the event is encoded by *decayed complex rotation*. This process is shown in Fig. 33. h is the multi-layer perceptron. The extracted feature vector of the event is rotated by an angle that is related to (t) . τ is a predetermined time interval. $\Delta t = t_r - t$ is the difference between timestamps of the considered event and the reference event. The reference event is the last event within the time interval.

$$\theta = \frac{2\pi\Delta t}{\tau} \quad (25)$$

Figure 33 Illustration of decayed rotation of event spatial feature vector

Meanwhile, the feature vector also takes an element-wise temporal decay also related to (t) . Along with the new events come in, if the considered event moves out of the time interval τ (related to the newly coming event), feature vector of this event decays to 0. Thus, the event will no more have effects on the network. The whole feature extraction process of an event is illustrated in the following formula. $h(\cdot)$ is the multi-layer perceptron in Fig. 33. $\bar{\mathbf{e}}$ is the spatial vector, i.e. (x, y) , of the event $\mathbf{e}$. $e^{-i\theta}$ represents element-wise complex rotation by θ . $\mathbf{a}$ is the final extracted spatiotemporal feature vector.

$$a = \left(1 - \frac{\Delta t}{\tau}\right) h(\bar{\mathbf{e}}) e^{-i\theta} \quad (26)$$

A complex max-pooling layer is designed to apply on the extracted feature vectors of the events within the time interval. These same size feature vectors are summarized into a global spatiotemporal feature vector.

The decaying complex rotation mechanism equips EventNet with the capability to process coming events recurrently. When a new event comes in, the max pooled global feature vector is updated with few marginal computations. There are no more needs to process all events within the sliding window and summarize them again. The only thing that needs to do is to rotate and decay summarized feature vector of the previous window with $\delta t = t_{new} - t_{old}$. t_{old} is the timestamp of the last event in the previous window. t_{new} is the timestamp of the newly coming event. Then, this updated summarized feature vector do an element-wise max-pooling with the feature vector of the newly coming event. As this event is just coming, it does not experience any rotation and decay.

In summary, EventNet has two characteristics. First, EventNet encodes spatial and temporal features in separate phases. For temporal encoding, it has two parts – temporal decay and complex temporal rotation. Second, EventNet can run recursively for every newly coming event without reprocessing all events in the window. The marginal computation for every event is limited.

4.4 Methodology for Event-wise Tracking

This thesis aims to find a framework that can do an event-wise prediction of a moving target. This framework should meet the requirements which are defined for the ideal tracking system based on event cameras in Chapter 4.2.

When the bounding box for the last event is known, based on the constant motion model, the tracking system does not need to process the events too far from the last bounding box. Therefore, when the newly coming event is far from the last bounding box, the prediction for the current bounding box should keep the same with the last one. When

the newly coming event is within the search region around the previous bounding box, the framework should do prediction to update the bounding box, as shown in Fig. 31.

Based on this idea, Network in Fig. 34 is designed based on EventNet [19]. In the initial stage, events within a search region (also within a predefined time interval), which is generated by extending the first given bounding box by a small ratio, are summarized into their own spatial feature vectors. The extraction is done by a 2D multi-layer perceptron, which is h in Fig. 34.

$$h: \bar{\mathbf{e}} \rightarrow \mathbf{f} \quad (27)$$

All the spatial feature vector will experience an element-wise temporal decay according to the temporal distance to the last event within the time interval.

$$\mathbf{f}' = \left(1 - \frac{\Delta t}{\tau}\right) \mathbf{f} \quad (28)$$

The decayed feature vectors are summarized into a global spatial feature vector, which is s_5 in the Fig. 34.

Different from original EventNet, the complex temporal rotation (exponential part in Eq. 26, Chapter 4.3) is not adopted here. The thinking is that the temporal information should be encoded into the feature vector implicitly by the decay. As time going, earlier events will decay until 0 when they leave the window. Therefore, the temporal decay itself is an excellent candidate to do the recurrent event processing. Besides, the Ablation study in [19] shows that without the complex temporal rotation term, the performance does not deteriorate significantly. On the other hand, the complex temporal rotation term affects the speed of event processing.

Figure 34 Event-wise tracking network

When a new event comes in, similar to the process described in Chapter 4.3 (without rotation), the event is extended to its own feature vector. Previous global feature vector will decay by a marginal time interval. As illustrated in Fig. 34, when $\mathbf{e}_6$ comes in, the last global feature vector S_5 decays marginally by $t_6 - t_5$. Similarly, when $\mathbf{e}_7$ comes in, the last global feature vector S_6 decays marginally by $t_7 - t_6$. Thereafter, the updated feature vector is fed into a trained fully connected neural network to regress into the predicted bounding box position.

This network is intuitive to understand and implement. It also shows a pleasant training accuracy, the error for predicted bounding box is around 5 pixels each coordinate on average. However, it has much worse performance on the test dataset. One reason for failure should be the size and quality of the training data, which is described in Chapter 3.6.3. Another more important reason is that this framework only considers events

within the interval and makes no use of information provided by the previous bounding box. Therefore, the prediction is highly related to the appearance of the tracking target. However, the natural characteristics of event cameras do not provide enough information on target appearance.

4.5 Siamese EventNet Tracker

The failure of the network in Chapter 4.4 shows that we cannot use only information from events within a past interval to predict the current bounding box simply. In this chapter, based on the idea explained in Chapter 4.4, a more reasonable network is designed which aims for event-wise object tracking.

Dual spacetime event clouds network in Chapter 3 is a good idea to make the use of past information. Also, experiment results in Chapter 3.5 demonstrate it can keep tracking objects. The idea is to concatenate feature vectors of past and current two event clouds to find the relative relationship between two bounding boxes. However, dual spacetime event clouds network cannot meet the requirement to do recurrent event processing. The network can only process data at event clouds level, instead of event-wise processing.

In Chapter 4.4, the temporal decay scheme enables the network to have recurrent process capability. When a new event comes in, it is extended to a feature vector and iteratively updates the global feature vector. In this scheme, only limited marginal computation is needed, as shown in Fig. 34.

Figure 35 Siamese EventNet Tracker

Combined with the idea in Chapter 3.3 and Chapter 4.4, a Siamese EventNet Tracker is designed in Fig. 35. The lower network delays the upper network by one event. In the initial stage, events within the bounding box b_0 are extended to feature vector by the upper network h_1 . The superscript 1 or 2 indicates the upper or the lower network.

$$\mathbf{a}_0^1, \mathbf{a}_1^1, \dots, \mathbf{a}_n^1 = h_1(\bar{\mathbf{e}}_0), h_1(\bar{\mathbf{e}}_1), \dots, h_1(\bar{\mathbf{e}}_n) \quad (29)$$

Similarly, another set of events is extended by the lower network h_2 .

$$\mathbf{a}_1^2, \mathbf{a}_2^2, \dots, \mathbf{a}_{n+1}^2 = h_2(\bar{\mathbf{e}}_1), h_2(\bar{\mathbf{e}}_2), \dots, h_2(\bar{\mathbf{e}}_{n+1}) \quad (30)$$

Each feature vector does an element-wise temporal decay by the time difference with the last element.

$$\mathbf{f}_i^k = \left(1 - \frac{\Delta t_i^k}{\tau}\right) \mathbf{a}_i^k, \quad k = 1 \text{ or } 2 \quad (31)$$

$$\Delta t_i^1 = t_n - t_i \quad (32)$$

$$\Delta t_i^2 = t_{n+1} - t_i \quad (33)$$

The decayed feature vectors will do an element-wise max-pooling to summarize into a global feature vector ($\mathbf{s}_6$ & $\mathbf{s}_7$ in Fig. 35)

$$\mathbf{s}_n^1 = \max(\mathbf{f}_0^1, \mathbf{f}_2^1, \dots, \mathbf{f}_n^1) \quad (34)$$

$$\mathbf{s}_{n+1}^2 = \max(\mathbf{f}_1^2, \mathbf{f}_2^2, \dots, \mathbf{f}_{n+1}^2) \quad (35)$$

After the initial state, when the new event $\mathbf{e}_{n+2}$ comes in, it will be fed into the lower network. At the same time, last event $\mathbf{e}_{n+1}$ will be into the upper network. They will be extended into feature vectors accordingly.

$$\mathbf{a}_{n+1}^1 = h_1(\bar{\mathbf{e}}_{n+1}) \quad (36)$$

$$\mathbf{a}_{n+2}^2 = h_2(\bar{\mathbf{e}}_{n+2}) \quad (37)$$

Previous feature vectors will be decayed by a small time interval.

$$\mathbf{s}_n^{1'} = \left(1 - \frac{\delta t_{n+1}}{\tau}\right) \mathbf{s}_n^1 \quad (38)$$

$$\mathbf{s}_{n+1}^{2'} = \left(1 - \frac{\delta t_{n+2}}{\tau}\right) \mathbf{s}_{n+1}^2 \quad (39)$$

The $\delta t_i = t_i - t_{i-1}$ is the marginal time difference between two events within regions of interest.

The global feature vectors will be updated by max-pooling between previous decayed feature vectors and newly extracted event feature vector.

$$\mathbf{s}_{n+1}^1 = \max(\mathbf{s}_n^{1'}, \mathbf{a}_{n+1}^1) \quad (40)$$

$$\mathbf{s}_{n+2}^2 = \max(\mathbf{s}_{n+1}^{2'}, \mathbf{a}_{n+2}^2) \quad (41)$$

With the idea of the Siamese network, these two feature vectors are concatenated together to learn the relative relationships by a fully connected network to regress into predicted bounding box translation and rescale.

This framework has several characteristics. First, it makes the most use of previous information by a Siamese structure. Then, it meets the purpose to give an event-wise response, which is consistent with the nature of continuity of event camera and motions. Finally, when processing the newly coming event, only a few marginal computations are needed. Previous events are not needed to process again. They are updated implicitly by decaying the global feature vectors.

4.6 Experiments and Analysis

To evaluate the network, an experiment is designed based on the same dataset in Chapter 3. Each sequence is divided at 90 percentiles. The first 90 percent of each sequence is used as training data, while the last 10 percent of each sequence is used as test data.

Before starting the experiment, there is a need to preprocess the original dataset. The original dataset only gives around 30 ground truth bounding boxes per second. As explained in Chapter 4.1 and Chapter 4.2, the framework is designed to give event-wise responses. Therefore, it is necessary to generate ground truth bounding boxes for events between two successive original given bounding boxes. With the assumption that the marginal translation and rescale between two successive events is minimal, this thesis uses simple linear interpolation to generate event-wise bounding boxes.

In this part, Intersection-over-Unit (IoU) is used to evaluate and analyze the performance of the framework. IoU is defined as a ratio between the overlap and the unit of two bounding boxes, as shown in Fig. 36.

Figure 36 Illustration of IoU

The test result is presented in Fig. 37, the horizontal axis is the sequence name, the vertical axis is the corresponding IoU for the sequence.

Figure 37 IoU for each sequence

There are 6 sequences whose IoUs are quite close to zero. After checking the original sequences, it can be found that the qualities of these sequences are too bad that even humans cannot find out the target. This is because the original dataset is not a record of real motion scenes. The dataset is an event camera version copy of the corresponding original dataset recorded by conventional cameras. Therefore, the quality suffers from

many limitations, which has been described in Chapter 3.6.3 already.

The overall average IoU is 0.3007, which demonstrates the preliminary tracking capability of this framework. However, IoU results in most cases are lower than 50%, which indicates that this framework has not been a mature and reliable solution yet. There are several challenges to be conquered.

Noise should be the biggest problem. Currently, there is not an efficient way to remove noise in event data. Framework in Chapter 3 uses geometric distribution information. This may help reduce the effect of noises. However, in this framework, time is used only for decay. No matter in training and test, there are amount noisy events being treated as events of interest.

After comparing more carefully between the predictions and the ground truth, it can be found that the framework usually can predict the correct direction of translation and rescale (expansion and contraction). However, both predicted translation and rescale tend to be smaller than ground-truth. As explained in Chapter 4.2, the minimal change between two successive events is the desired property of the framework. Considering that the event camera has an ultra-high temporal frequency, there are should not be sharp changes between every two successive events. However, the original dataset labeled bounding boxes still based on frames and this experiment uses linear-interpolation to generate event-wise bounding boxes.

All in all, when the recorded object is clear and does not have sudden shape change between very small time-interval, this framework can provide desired event-wise update of the bounding box. As shown in Fig. 38, several running results are displayed to demonstrate. Red bounding boxes are predictions while green bounding boxes are ground truth.

Figure 38 Demonstration of predictions and ground-truths

Another important topic in evaluation is speed. One of the reasons to do recurrent processing is to reduce the marginal computations when a new event comes in. For example, it is impossible to process a single event in the framework in Chapter 3. When considering a new event, an entire spacetime event cloud must be processed again to

summarize into useful information.

However, in this framework, to deal with a new event, there is not a need to process all events in the interval again. The only thing that needs to do is to extend the new event into a feature vector. Then the previous global feature vector updates with the feature vector based on a decay mechanism. This characteristic gives this framework the capability to predict in a much higher frequency than the framework in Chapter 3. To extend an event into feature vector, a lookup table can be created to decrease the computation further at the cost of some additional library.

Chapter 5. Conclusions and Suggestion for Future Work

5.1 Conclusions

Event cameras successfully break the conventional thinking inertia of using synchronous frames to describe the moving world. But it remains a challenging problem to think out of the conventional view of space and time when trying to use the sparse and asynchronous data to deal with conventional computer vision tasks. The main purpose of this thesis is to investigate new ideas to process event data, especially when applying them in object tracking.

A framework based on PointNet [13] to process events data in (x, y, t) 3D spacetime domain has been described. The assumption behind this is that the spatial and temporal information are embedded in the geometric spacetime event clouds. Then the idea is efficient to extract spatial and temporal features simultaneously.

The experiments and analysis in Chapter 3.5 and Chapter 3.6 show that the proposed framework is an effective representation to characterize the event streams from event cameras. It preserves both spatial and temporal information to analyze the event streams and is end-to-end learnable. As verified in our experiments, event clouds can benefit from down sampling the same as point clouds. It is believed that the spacetime event cloud idea is promising and applicable in object tracking based on event cameras. Also, we demonstrated that the spacetime event cloud concept has huge potential to conquer other real-time multimedia and computer vision tasks with high efficiency, both in terms of hardware resource requirement and computation requirement.

Even though framework in Chapter 3 demonstrates the capability to complete object tracking task with event cameras, can only provide discrete bounding cuboid based on a sliding window mechanism. Therefore, this thesis investigates recurrent event processing in Chapter 4. The reason to do this is that an event-wise response is desired to be consistent with the nature of continuity of a moving object, which is also the characteristic of event cameras. This is also the basis for tracking high-speed moving

objects.

Based on the recent idea EventNet [19], a recurrent bounding box regressor is designed in Chapter 4. The idea is to decay event-wise feature vector along with time. Different from the spacetime event cloud concept, this network only extracts spatial information. The temporal information will be encoded implicitly in the decay process. Even though this idea is not as efficient as spacetime event cloud which can extract spatial and temporal features at the same time, it is superior in providing recurrent processing capability.

With the framework in Chapter 4, only a few marginal computations are needed to update the feature vector. It is also easier to make use of past information. With an offline trained event feature descriptor lookup table, the computation time can be decreased further. The experiments and analysis, this network can successfully learn the trend of translation and rescale of the moving target. However, it has the problem to drift over time. Especially, when there is a strong change within a short period. Even though with this limitation, it is believed that the performance can be improved to a significant level if a larger size with a higher quality dataset is provided.

Chapter 3 and Chapter 4 are two different approaches to make full use of temporal information. Chapter 3 extracts spatial and temporal features jointly from the geometric distribution. There is no difference between time and space axes. Chapter 4 separates the space axes and time axis conventionally. However, the time is used to conduct a decay which embeds temporal information implicitly. The framework discussed in Chapter 3 is more efficient in terms of extracting information, while the framework described in Chapter 4 provides a recurrent process capability. A recurrent way based on spacetime event cloud concept to combine these two ideas for achieving better performance and reliability should be investigated.

This thesis still has a few limitations. The IoUs are still too low to be a mature single-object tracking solution. The noise problem is significant, especially in the event-wise framework. Besides, the proposed approaches are only suitable for surveillance scenarios. If the background is not static, background events will possibly disrupt the

tracking process. To solve this problem, motion segmentation techniques are needed.

5.2 Future Works

The frameworks discussed in both Chapter 3 and Chapter 4 refer to the Siamese network structure in GOTURN tracker [18]. The offline training of GOTURN takes advantage of the huge number of artificially generated from publicly available image datasets. This idea works well for conventional camera-based videos because they are virtually a series of images. In terms of limited dataset for event camera object tracking, it is valuable to investigate some semi-supervised learning techniques to train on limited labeled data. Semi-supervised learning techniques can help the tracker continually improve with unlabeled data.

On the other side, both frameworks are assumed constant motion model, which does not consider too much about object speed. As seen in the experiment, the tracker most likely drifts when there is a big movement in a small time interval. In terms of this limitation, it is valuable to investigate more complex motion models to consider target speed. Even further, it can refer to some ideas in optical flow algorithms to consider the speed effect in object tracking. For the drift problem, some global correction mechanism should be designed to recover from lost tracking.

Finally, a dataset with higher quality can be prepared. The current dataset is recorded from videos taken by conventional cameras, which suffers from the frame rate of the video. Also, the bounding boxes are generated automatically based on some rules, which is not as accurate as expected. Therefore, there is a demand for datasets recorded from real scenes by event camera that can also provide carefully labeled event-wise bounding box.

Bibliography

- [1] Q. Wang, Y. Zhang, J. Yuan and Y. Lu, "Space-time Event Clouds for Gesture Recognition: from RGB Cameras to Event," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.
- [2] M. Kristan, J. Matas and A. Leonardis, "The Visual Object Tracking VOT2016 Challenge Results," in *European Conference on Computer Vision*, 2016.
- [3] P. Lichtsteiner and T. Delbruck, "A 64x64 aer logarithmic temporal derivative silicon retina," in *Research in Microelectronics and Electronics, 2015 PhD*, 2005.
- [4] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schon, H. Kohn and H. Garn, "Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor," in *IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*, 2006.
- [5] S. Chen, P. Akselrod, B. Zhao, J. Carrasco, B. Linares-Barranco and E. Culurciello, "Efficient Feedforward Categorization of Objects and Human Postures with Address-Event Image Sensors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 302-314, 2012.
- [6] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, P. Douglas and T. Delbruck, "A pencil balancing robot using a pair of AER dynamic vision sensors," in *IEEE International Symposium on Circuits and Systems*, 2009.
- [7] B. Zhao, X. Zhang, S. Chen, K. Low and H. Zhuang, "A 64×64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 581-588, 2012.
- [8] R. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs and R. Etienne-Cummings, "A multichip neuromorphic system for spike-based visual information processing," *Neural Computation*, vol. 19, no. 9, pp. 2281-2300, 2007.
- [9] C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski and M. Bacchiani, "State-of-the-art Speech Recognition With Sequence-to-Sequence Models," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

- [10] F. Tao and C. Busso, "Gating Neural Network for Large Vocabulary Audiovisual Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 7, pp. 1290-1302, 2018.
- [11] Y. Wu, M. Schuster, Z. Chen and etl, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine TranslationGoogle's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv, 10 08 2016. [Online]. Available: <http://arxiv.org/abs/1609.08144>.
- [12] B. Oancea and S. Ciucu, "Time series forecasting using neural networks," arXiv, 01 07 2014. [Online]. Available: <https://arxiv.org/abs/1401.1333>.
- [13] C. Qi, H. Su, K. Mo and L. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Conference on Computer Vision and Pattern Recognition*, 2017.
- [14] C. Qi, L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Conference on Neural Information Processing Systems*, 2017.
- [15] E. Mueggler, G. Gallego, H. Rebecq and D. Scaramuzza, "Continuous-Time Visual-Inertial Trajectory Estimation with Event Cameras," arXiv, 10 06 2018. [Online]. Available: <http://arxiv.org/abs/1702.07389>.
- [16] H. Liu, D. P. Moeys, G. Das, D. Neil, S. Liu and T. Delbrück, "Combined frame- and event-based detection and tracking," in *International Symposium on Circuits and Systems*, 2016.
- [17] H. Kim, S. Leutenegger and A. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *European Conference on Computer Vision*, 2016.
- [18] D. Held, S. Thrun and S. Savarese, "Learning to Track at 100 FPS with Deep Regression Networks," in *European Conference on Computer Vision*, 2016.
- [19] Y. Sekikawa, K. Hara and H. Saito, "EventNet: Asynchronous Recursive Event Processing," in *Conference on Computer Vision and Pattern Recognition*, 2019.
- [20] T. Vojir, J. Neskova and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, 2014.

- [21] S. Avidan, "Support Vector Tracking," *IEEE Trans. on PAMI*, vol. 26, pp. 1064-1072, 2004.
- [22] S. Hare, A. Saffari and P. H. S. Torr, "Struck: Structured Output Tracking with Kernels," in *International Conference on Computer Vision*, 2011.
- [23] C. Leistner, A. Saffari, P. M. Roth and H. Bischof, "On Robustness of On-line Boosting - A Competitive Study," in *International Conference on Computer Vision*, 2009.
- [24] H. Masnadi-Shirazi, V. Mahadevan and N. Vasconcelos, "On the design of robust classifiers for computer vision," in *Conference on Computer Vision and Pattern Recognition*, 2010.
- [25] H. Grabner, C. Leistner and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *European Conference on Computer Vision*, 2008.
- [26] A. Saffari, C. Leistner, M. Godec and H. Bischof, "Robust multi-view boosting with priors," in *European Conference on Computer Vision*, 2010.
- [27] B. Babenko, M.-H. Yang and S. Belongie, "Visual Tracking with Online Multiple Instance Learning," in *Conference on Computer Vision and Pattern Recognition*, 2009.
- [28] B. Zeisl, C. Leistner, A. Saffari and H. Bischof, "Online Semi-Supervised Multiple-Instance Boosting," in *Conference on Computer Vision and Pattern Recognition*, 2010.
- [29] M. B. Blaschko and C. H. Lampert, "Learning to Localize Objects with Structured Output Regression," in *European Conference on Computer Vision*, 2008.
- [30] I. Tsochantaridis, T. Joachims, T. Hofmann and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *Journal of Machine Learning Research*, vol. 6, p. 1453–1484, 2005.
- [31] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
- [32] R. M. Gray, "Toeplitz and Circulant Matrices: A Review," in *Now Publishers*, 2006.
- [33] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual Object

- Tracking using Adaptive Correlation Filters," in *Conference on Computer Vision and Pattern Recognition*, 2010.
- [34] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," *Experiments in Fluids*, vol. 51, pp. 1465-1469, 2011.
- [35] B. Ramesh, S. Zhang, Z. W. Lee, Z. Gao, G. Orchard and X. Cheng, "Long-term object tracking with a moving event camera,," in *British Machine Vision*, 2018.
- [36] B. Ramesh, H. Yang, G. Orchard, N. A. L. Thi and X. Cheng, "A neuromorphic descriptor for pattern recognition," in *IEEE Biomedical Circuits and Systems Conference*, 2017.
- [37] A. Mitrokhin, C. Fermüller, C. Parameshwara and Y. Aloimonos, "Event-based Moving Object Detection and Tracking,," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.
- [38] F. Barranco, C. Fermuller and Y. Aloimonos, "Contour motion estimation for asynchronous event-driven cameras," *Proceedings of the IEEE* , vol. 102, no. 10, pp. 1537-1556, 2014.
- [39] F. Barranco, C. Fermuller and Y. Aloimonos, "Bio-inspired motion estimation with event-driven sensors," in *International Work-Conference on Artificial Neural Networks*, Springer, 2015, pp. 309-321.
- [40] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, F. M and D. Modha, "A Low Power, Fully Event-Based Gesture Recognition System," in *Conference on Computer Vision and Pattern Recognition*, 2017.
- [41] L. Zyga, "Scientists suggest spacetime has no time dimension," 25 04 2011. [Online]. Available: <https://phys.org/news/2011-04-scientists-spacetime-dimension.html>.
- [42] O. Bichler, D. Querlioz, S. J. Thorpe, Jean-Philippe Bourgoin and Christian Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks*, vol. 32, no. <https://doi.org/10.1016/j.neunet.2012.02.022>, pp. 339-348, 2012.

- [43] Y. Hu, "Generation of Benchmarks for Visual Recognition with Spiking Neural Networks," University of Zurich, Zurich, Swizerland, 2016.
- [44] Y. Hu, H. Liu, P. Michael and D. Tobi, "DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition," *Frontiers in Neuroscience: Neuromorphic Engineering*, 2016.
- [45] M. Fiaz, A. Mahmood and S. K. Jung, "Tracking Noisy Targets: A Review of Recent Object Tracking Approaches," 14 02 2018. [Online]. Available: <https://arxiv.org/>.
- [46] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid and S. Roth, "Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking," arXiv, 10 04 2017. [Online]. Available: <https://arxiv.org/abs/1704.02781>.
- [47] H. Possegger, T. Mauthner and H. Bischof, "In Defense of Color-based Model-free Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2015.