Developing a Recurrent Neural Network solution for motion estimation to enhance attention-based tracking network

Project Laboratory

submitted by
Mingpan Guo, Takuro Takezawa, Christoph Allig

NEUROSCIENTIFIC SYSTEM THEORY
Technische Universität München

Prof. Dr Jörg Conradt

Supervisor: M.Sc. Mohsen Firouzi
Final Submission: 07.07.2015

TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR STEUERUNGS- UND REGELUNGSTECHNIK
ORDINARIUS: UNIV.-PROF. DR.-ING./UNIV. TOKIO MARTIN BUSS

26.09.2014

PRACTICAL COURSE

Developing a Recurrent Neural Network solution for motion estimation to enhance attention-based tracking network

Problem description:

We have developed a neural solution for event-based Visual Attention Control in tracking application using eDVS sensor [1]. The problem is: in case of collision of two moving objects, the attention network would lose the attended object where two objects collide. One promising solution is biasing attention field neurons using motion cues so that attention network can still keep tracking target object but first we should estimate direction of motion using a neural computing machinery. To construct a neural solution for such mechanism seemingly Recurrent Neural Networks (e.g. Elman RNN or ESM) is a suitable architecture. that we want to use and evaluate in this project. Expected RNN should estimate the flow of motion (direction of motion) given the momentary location of attended object (activity of attention network neurons), and then bias attention network accordingly through feedback connection. Students need to have background on Artificial Neural Networks, motivation in neuro-computing and MATLAB/C programming ability. Basic mathematical knowledge about Integrate-and-Fire neuron model would be helpful.

Tasks:

- review current work on stimulus-driven visual attention network [1].
- read literature review on Recurrent Neural Networks architecture and dynamics.
- developing a recurrent neural solution in order to estimate motion cue.
- evaluation of the network for real scenarios.
- documentation and report.

Bibliography:

[1] M. Truong Le, "Stimulus-driven visual attention control for object tracking" In *Bachelorarbeit, LSR - Technische Universitat Munchen*, Aug 2014

Supervisor: M.Sc. Mohsen Firouzi

(J. Conradt)
Professor

Abstract

A neural solution for event-based visual attention control in tracking application using embedded dynamic vision sensor (eDVS) is developed in recent years. However, it encounters a problem in case of collision between two objects. The correct object can not be successfully tracked after collision. To solve this problem, this paper presents solutions which biases attention field neurons using motion cues for recognizing and tracking the correct object. The main task of this paper is therefore to apply a motion estimation model based on eDVS data, to enhance the existing system.

A thoroughly analysis of the problem is presented. Two possible methods are evaluated in this paper. The designed back-propagation neural network with a 'Elman network' alike structure provides satisfactory results, while the other method of Spike-Timing-Dependent Plasticity learning is also analyzed, which turns out to be more suitable for the asynchronous eDVS data but more challenging to be applied. Using the back-propagation motion estimation network, the system can solve the collision problem. Different evaluations of this model are performed, both in simulation and real-world scenarios. The result shows that it can be applied to enhance a system with motion information of objects in tracking application with a real-time performance.

Contents

Chapter 1

Introduction

There are a lot of studies on motion estimation method. Most of these methods are frame-based, where frames taken in are processed one by one through the entire processing chain. These methods cost a lot in terms of computational and time complexity because the system needs to process all the pixels in every frame. And these frames often have a lot of redundant information because two frames next to each other have nearly the same information, since not many things can change during the short time interval. For motion estimation, changes of the pixels alone contain most of the needed informations. In recent years, another device and method for processing images and estimating motions has drawn attention. With Dynamic Vision Sensor (DVS) only the information of changing pixels are taken in and processed, which provides a great potential for fast estimation of motion with limited computational resources.

The problem of the second method is, changing pixels could also be a result of unrelated events other than a moving target. In order to filter the unnecessary information off, an attention network can be applied [Mic14] for the purpose of focusing on target. However, the problem of traditional attention network is that it would lose the attended object when it collides with other distractors.

In order to solve this problem, this paper employs recurrent neural network such as Elman neural network, which uses both the current information and past information as input. This paper employs also the previous motion of the object as a cue to predict the following future motion. The reason why this paper chooses the previous motion among various possible cues, for instance color or shape, is because motion is the most peculiar and invariant character to a specific object and it can be easily applied to any other scenarios. If one chose the color as a cue for tracking, one would lose the object after collisions with distractors whose color is the same as that of the attended object. Recurrent neural network can store the previous motion just like biological mechanism does as memory for the system to predict the next motion.

This paper employs mainly two approaches: one is the neural network with firing neurons, where neurons fires synchronously when the sum of the signals that a neuron gets reaches its threshold, with weights learned by back-propagation. The detail is explained in 3.3. In this paper it employs two focus maps which are current location and time shifted location so that the network store the previous information. The other approach is a neural network with spiking neurons where a neuron accumulates spikes from the neurons below but is leaky. Neurons spikes asynchronously transmit when the accumulation level reaches its thresholds, and weights between neurons is learned in a way of Spike-Timing-Dependent-Plasticity (STDP). The learning mechanism of STDP is explained in 3.2.

Attention network with motion estimation can be used for many applications such as pedestrians tracking of autonomous car driving system, or assisting people to pick up moving objects in nursing.

Chapter 2

Related Work

This chapter introduces related works, one of which is about stimulus-driven visual attention using firing network developed in [Mic14] and the other is about spiking neural network with STDP learning developed in [BQT+12].

2.1 Stimulas-driven visual attention control for object tracking

This section introduces spiking neural network proposed in [Mic14]. Unlike firing neuron that fires synchronously, spiking neuron which is inspired by biological retina mechanism, integrates all the stimulus neurons get from the neurones below and spikes when the accumulated state reaches a threshold. Because the state is not solid but leaky, it takes into account time not only the number of spikes needed to reach a threshold, but also time interval between neurons. Input of the network is not conventional frame-based information, but stimulus-based information using eDVS sensor, which this paper uses also. This work tries to extend the network so that it can deal with collision problems and it suggests two approaches; non-neural based algorithm and neural-based algorithm.

The first approach, non-neural based algorithm predicts the next location of the object by adding Gaussian probabilities $N(x)$ of the next location, which is formulated as following.

$$N(x) = D \cdot \exp\left(-\frac{|x - x'|}{d^2}\right) \quad \text{with} \quad x' = \beta \cdot (x_{t-1} - x_{t-2}) + x_{t-1} \tag{2.1}$$

where x_t is the location of the object at time t and d is the distance and D, β are constants. Since this method is very fast and does not take much memory to store x_{t-1} and x_{t-2}, but this is not neural-based approach.

The second approach, neural-based algorithm uses another focus map F' besides input focus map F. In another focus map F', time-shifted input is stored which is connected to neural network. Basic structure is the same as that of Elman network. However, this paper only suggests this approach because of complexity of the network and its high computational cost.

2.2 Extraction of temporally correlated features with STDP

This section introduces spiking neural network with STDP learning developed in [BQT+12]. This network is inspired by biological retinas model, which uses Address-Event representation (AER) to asynchronously transmit spikes in response of pixel changes. The weights between the neurones of the network is learned by simplified STDP, through which neurones become sensitive to particular patterns of pixels by depressing neurones that do not contribute to the post-synaptic activation. Although this work is not about motion estimation, it extracts patterns of motion trajectories that can be used for motion estimation. In order to apply this system for motion estimation, however, another focus to store the last location needs to be added as mentioned section 2.1, because this extraction of temporally correlated features employs feed-forward network.

Chapter 3

Neural Network Implementations

3.1 Combination of a Recurrent Neural Network with Gaussian probability

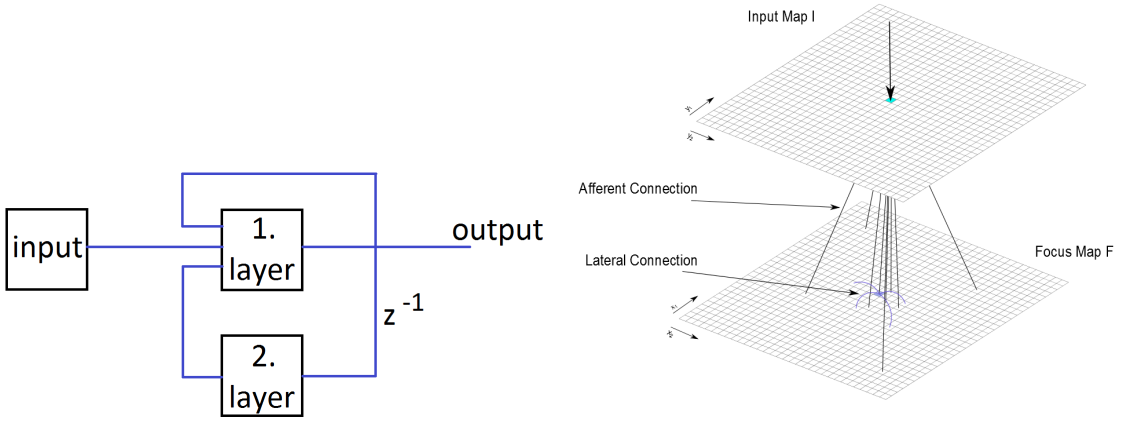
The basic principle of the neural network considered in this chapter was developed by [RV06]. [RV06] and [Mic14] evaluated the network in terms of the ability to track an object and neglecting distractors and noise. The results indicate that the network is capable of tracking an object in a noisy environment, even combined with distractors. But as soon as the target object and a distractor are colliding, the network fails. In this chapter we expand this model to a Recurrent Neural Network (RNN) combined with gaussian probability for the predicted next position.

The general structure of the network is shown in Fig. 3.1. It consists of an input, a first and a second layer. The input is connected to the first layer. The output of the first layer is equivalent to the networks output. The input of the second layer is this time-shifted output. Also the output of the first and second layer are input of the first layer. These two additional inputs of the first layer are employed for predicting the next position of the target object. All maps consists of nxn neurons. At first we consider the common features afferent connection and lateral connection of both layer. The vector $\vec{x}\,\epsilon\,\mathbb{R}^2$ and the vector $\vec{y}\,\epsilon\,\mathbb{R}^2$ denote the position in the considered layer, respectively the position of its input. C and c represents some constants. The afferent connections of the input to the layer is described by:

$$s(\vec{x},\vec{y}) = s\left(\begin{pmatrix}x_i\\y_j\end{pmatrix},\begin{pmatrix}x_k\\y_l\end{pmatrix}\right) = Ce^{-\frac{|\vec{x}-\vec{y}|^2}{c^2}} \tag{3.1}$$

The summation of all the connections connected to the neuron of the layer at position $\vec{x}$ multiplied with the state of the input neuron at position $\vec{y}$, yields in:

$$afferent(\vec{x}) = \sum_{k,l} s(\vec{x},\vec{y})input(\vec{y}) \tag{3.2}$$



(a) (b)

Figure 3.1: (a) structure of expanded RNN: The input is connected to the first layer. The output of the first layer is equivalent to the networks output. The input of the second layer is the time-shifted output. Also the output of the first and second layer are input for the first layer. (b) The neurons of a layer are connected by afferent connections with the input neurons and by lateral connections with the neighboring neurons [Mic14]

Additionally to the afferent connections, the layer also has its own lateral connections, which are described by a difference-of-gaussians function. This characteristic is being utilized to eliminate distractors and noise around the target. The connection between the positions $\vec{x}$ and $\vec{x}'$ is computed as follows:

$$w(\vec{x} - \vec{x}') = w\left(\begin{pmatrix} x_i \\ y_j \end{pmatrix} - \begin{pmatrix} x'_k \\ y'_l \end{pmatrix}\right) = Ae^{-\frac{|\vec{x}-\vec{x}'|^2}{a^2}} - Be^{-\frac{|\vec{x}-\vec{x}'|^2}{b^2}} \tag{3.3}$$

where A, a, B and b denotes some constants. The summation of all the connections connected to the neuron at position $\vec{x}$ multiplied with the state of the neuron at position $\vec{x}'$ is computed as follows:

$$lateral(\vec{x}) = \sum_{k,l} w(\vec{x} - \vec{x}')output(\vec{x}', k-1) \tag{3.4}$$

The following computation is only performed for the second layer. The estimated position of the first layer $\vec{x}_{o1}$ and the second layer $\vec{x}_{o2}$ is computed by the summation over the position multiplied with the state of the neuron at this position and subsequently divided by the sum of all neuron's states:

$$\begin{pmatrix} x_{om} \\ y_{om} \end{pmatrix} = \frac{\sum_{i,j} \begin{pmatrix} x_i \\ y_j \end{pmatrix} output_m(\vec{x})}{\sum_{i,j} output_m(\vec{x})} \tag{3.5}$$

We compute the estimated velocity by a first-order Taylor approximation:

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}x_{o1}\\ y_{o1}\end{pmatrix} - \begin{pmatrix}x_{o2}\\ y_{o2}\end{pmatrix} \tag{3.6}$$

Now we are capable to estimate, respectively predict the next position of the target object $\vec{\mu}$.

$$\vec{\mu} = \begin{pmatrix}x_{predict}\\ y_{predict}\end{pmatrix} = \begin{pmatrix}x_{o1}\\ y_{o1}\end{pmatrix} + \begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} \tag{3.7}$$

Hence the underlying model - the input as well as the output of both layer - has gaussian shape, we model the predicted next position by a gaussian probability:

$$f(\vec{x}, \vec{\mu}, \vec{\Sigma}) = \frac{1}{\sqrt{|\Sigma|(2\pi)^2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})} \tag{3.8}$$

Σ depends on the size and shape of the target object. The larger the object is, the bigger shall be Σ, respectively Σ indicates the extension of the target object. Subsequently we are able to update the state of each neuron of the first layer according to:

$$\begin{aligned} output(\vec{x}, k) = & \frac{afferent(\vec{x}) + lateral(\vec{x}) + df(\vec{x}, \vec{\mu}, \vec{\Sigma}) - output(\vec{x}, k-1)}{\tau} \\ & + output(\vec{x}, k-1) \end{aligned} \tag{3.9}$$

The variables d and τ denote a weighting factor for the probability and the neuron's time constant. In order to achieve normalized neuron's states, the states are constrained to the interval from 0 to 1. For updating the second layer, the gaussian probability term is omitted. Except for d and Σ the parameters shown in table 3.1 are taken from [Mic14].

Table 3.1: Parameters for the network

n	α	A	a	B	b	C	c	τ	Σ	d
30	12.5	$\frac{2.27}{\alpha}$	$\frac{3.25}{n}$	$\frac{0.9}{\alpha}$	$\frac{13.25}{n}$	$\frac{1.5}{\alpha}$	$\frac{17.75}{2n}$	0.6	$\begin{pmatrix}2 & 0\\ 0 & 2\end{pmatrix}$	2

Table 3.2: Parameters for the network

	I	$\vec{\mu_0}$	w	$\vec{v}$
target	1	$\begin{pmatrix}5\\ 5\end{pmatrix}$	1.6	$\begin{pmatrix}0.15\\ 0.15\end{pmatrix}$
distractor	1	$\begin{pmatrix}5\\ 25\end{pmatrix}$	1.6	$\begin{pmatrix}0.15\\ -0.15\end{pmatrix}$

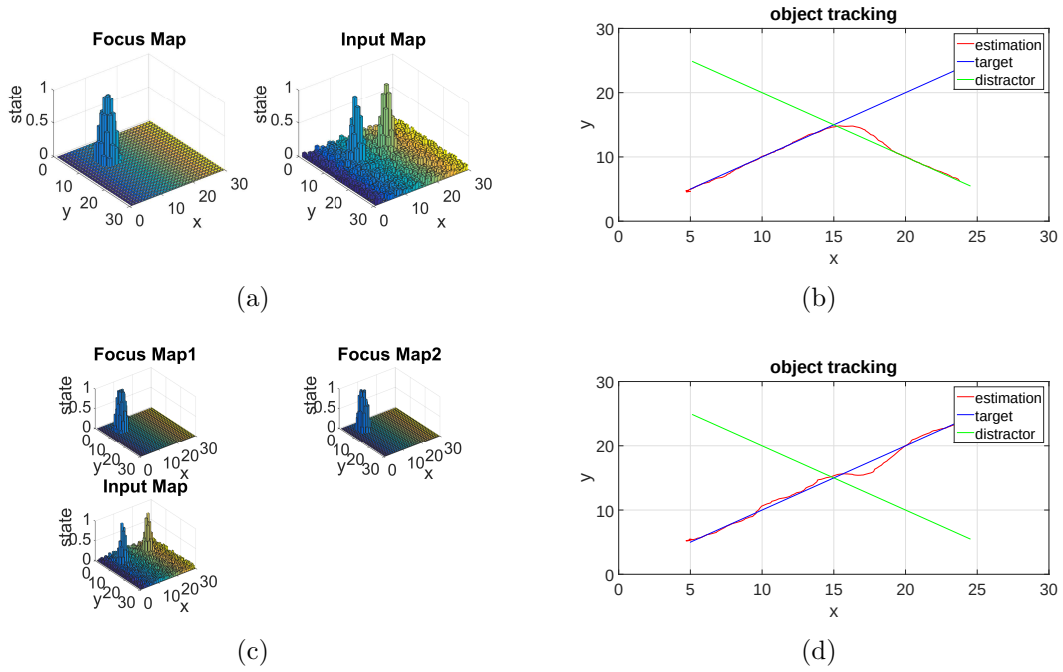


Figure 3.2: The target object is moving from the bottom left to the top right. The distractor is moving from the top left to the bottom right. Focus Map (1) represents the estimated position of the target object. (a) focus map and input map of usual network (b) result of a) (c) focus map1, focus map2 and input map of expanded RNN network (d) result of c)

In order to demonstrate the ability of the specified network, input data is generated for simulation. The chosen input for the target and the distractor is a gaussian function:

$$F(x,y) = Ie^{-\frac{|x-\mu_x|^2}{\sigma^2}} e^{-\frac{|y-\mu_y|^2}{\sigma^2}} \tag{3.10}$$

I, $\vec{\mu}$ and σ denote the height, the position of the center and the standard deviation σ, which controls the width. The movement of the target and distractor is specified according to:

$$\vec{\mu} = \vec{\mu_0} + \vec{v}t \tag{3.11}$$

The target object is moving from the bottom left to the top right. The distractor is moving from the top left to the bottom right. Table 3.2 shows the respective parameters. The target appears 10 time steps earlier than the distractor and the beginning of the movement. After further 10 time steps uniformly distributed noise between 0 and 0.1 emerges. Fig. 3.2 shows the result using the originally neural network and our expanded one. Once the target and the distractor collide the original neural network fails, whereas our expanded one still works properly.

Even if the network is working properly, we have not performed further evaluations on this network, because the predicted next position is obtained by a differential equation and adding a gaussian probability. However, the aim is to receive a recurrent neural network that learns the connections between all the neurons properly and is therefore capable to predict the movement direction, respectively the next position.

3.2 Spike-Timing-Dependent Plasticity Learning

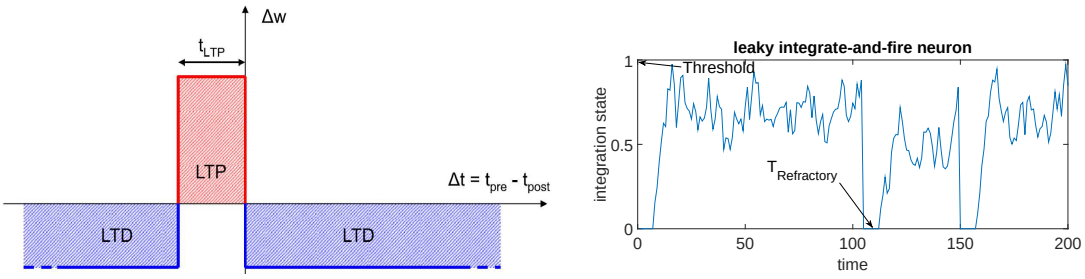
The aim is developing a neural network consisting of leaky integrate-and-fire neurons. The state of all neurons in a layer is updated according to 3.12, as soon as one of the neurons in the input layer is spiking at t_{spike}. t_{last_spike} denominates the time this neuron has spiked recently. u and τ denote the integration state of a neuron and the time constant. When the integration state reaches the neuron's threshold, a new spike event is created and sent to every output synapse. The integration state is then reset to zero and does not increase again until the end of a refractory period T_{refrac}. The principal is shown in Fig. 3.3 (b). When a neuron spikes, it disables all the other neurons during a period $T_{inhibit}$, during no incoming spike is integrated. $T_{inhibit}$ provides lateral inhibition, which is fundamental to allow multiple neurons to learn multiple patterns. Without lateral inhibition, all the neurons end up learning the same pattern. Generally it has to hold that $T_{inhibit} < T_{refrac}$. Otherwise only one neuron would spike the whole time and depress all the other neurons. So the other neurons would never have a chance to fire and learn something. The reason employing a spiking neural network is the concept that the eDVS sensor delivers also spiking data. Hence no preprocessing is required.

In order to obtain a satisfactory unsupervised learning process for the previously described neural network, we employ the spike-timing-dependent plasticity (STDP) rule. STDP is based on Hebbian theory [Heb68]. According to [Sch92], the theory can be summarized by "Cells that fire together, wire together". Additionally, STDP considers causality. This implies strengthening the connection if a pre-synaptic spike occurs a short time before a post-synaptic. Otherwise the connection will be weakened. According to Fig. 3.3 (a) the connection is strengthen, if $0 \leq t_{post} - t_{pre} \leq T_{LTP}$. A general form for the Long-Term Potentiation (LTP) case is 3.13. Whereas Long-Term Depression (LTD) 3.14 weakens the connection. We use respectively some constants.

$$u = u \cdot \exp(-\frac{t_{spike} - t_{last_spike}}{\tau_{leak}}) + w \tag{3.12}$$

$$\Delta w_+ = \alpha_+ \cdot \exp(-\beta_+ \cdot \frac{w - w_{min}}{w_{max} - w_{min}}) \tag{3.13}$$

$$\Delta w_- = \alpha_- \cdot \exp(-\beta_- \cdot \frac{w_{max} - w}{w_{max} - w_{min}}) \tag{3.14}$$



(a) (b)

Figure 3.3: (a) STDP learning rule: the synapse undergoes Long-Term Potentiation (LTP) when $0 \leq t_{post} - t_{pre} \leq T_{LTP}$ and Long-Term Depression (LTD) otherwise [BQT+12] (b) leaky integrate-and-fire neuron

In the following we describe the structure of the implemented network that is based on the above mentioned principals. The input of the network is obtained by eDVS128, respectively artificial eDVS data. A dataset consists of a list of events, where each event includes the address of the emitting pixel, the time-stamp of the event and its type. A pixel generates an event each time the relative change of its illumination intensity reaches a positive or negative threshold. Depending on the sign of the intensity change, events can be either type On or type Off. The input size may be reshaped to any size. Thereby we obtain lower resolution on the one hand and reduced computational complexity on the other hand. We use a size of $16x16$. Then we divide the input into nxn fields to be capable of general learning. So as to consider as well the type of event, the input data is expanded to three dimensions $nxnx2$. Our network employs independently and iteratively each of the divided and reshaped inputs. For the purpose of having enough time to learn the parameters, the input is stringed together multiple times. At first we considered a converging criteria that stops the learning process if the parameter changes lies below a certain threshold. Due to the adjustment of the parameters by constant, this approach fails. Consequently the learning process is stopped after a certain number of parameter adjustments.

Each of the neurons in the first layer are fully connected to the input neurons. If there occurs a spike in the input and the preconditions concerning T_{refrac} and $T_{inhibit}$ are fulfilled, each of the neurons in the first layer is integrated with the weight w_{ijklm} according to 3.12. i and j address the emitting pixel in the input data. k corresponds to the type of the spike. l and m denote the particular neuron in the first layer. As soon as one of the neurons in the first layer reaches a threshold, it is reset to zero. Additionally all the weights connected to the l, mth-spiking neuron w_{ijklm} are updated according to 3.13, respectively 3.14. Hence, if $0 \leq \Delta t = T_{1.layer\_spike}(l, m) - T_{input\_spike}(i, j, k) \leq T_{LTP}$ the connection is strengthened. Otherwise the connection is weakened. Once the stopping criteria is reached and all connections between the

input and first layer have been learned, the second layer is switched on. The second layer shall represent the current direction of the tracked movement. We consider to employ four neurons in the first layer for each neuron in the second layer. Each neuron of the second layer shall correspondent to one direction. Assumed we want to detect four direction, this results in a second layer of size $2x2$ and a first layer of size $4x4$. This structure is shown by Fig. 3.4. Our neural network is implemented in a way that all sizes can be modified straightforwardly. The integration of the states of the neurons in the second layer and the learning of the connections between the first and second layer follow the above-mentioned principles.

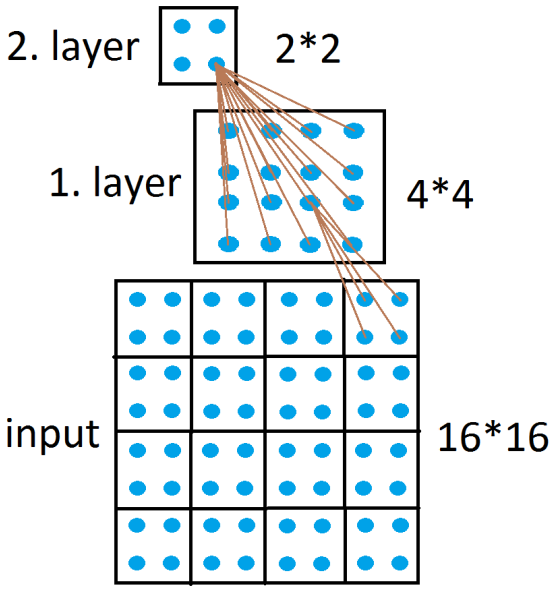


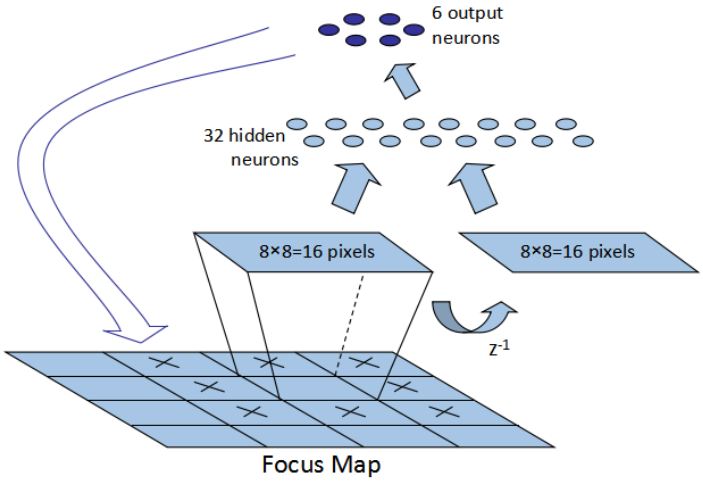
Figure 3.4: structure of STDP-Learning network

Finding convenient values for the parameters I_{thres}, T_{LTP}, T_{refrac}, $T_{inhibit}$, Δw_+, Δw_- and τ_{leak} is a challenging task. We were not able to find convenient values. We guess that this is the problem why the network is not capable to estimate the direction. In the following is described the approach for estimating appropriate parameters. We count the numbers of events of input data within a certain time frame. Depending on the size of the regarded map it possible to compute the average time that is gone between two successive spikes of a position. τ_{leak} has to be grater than this average time. I_{thres} is set to 1. T_{LTP} shall be chosen dependent on the speed of the object. It has also to hold: $T_{LTP} \gg \tau_{leak}$. As mentioned before is $T_{inhibit} < T_{refrac}$. Furthermore it has to hold $0 < \Delta w_+ < I_{thres}$ and $-I_{thres} < \Delta w_- < 0$.

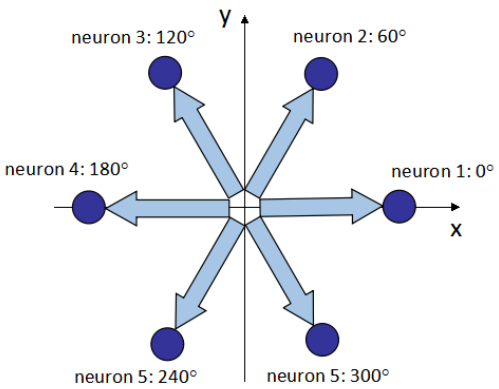
3.3 Back-propagation Neural Network

Another possible solution for motion estimation is to use feed forward neural network with back-propagation(BP) supervised learning algorithm [RHW88]. In con-

trast with the STDP approach, firing neuron model is preferred in this case, because the feed forward network model is based on analog activation level of neurons. Here, the events detected from DVS camera are integrated with fixed time frame, therefore these information can be compatible with the network presented by [Mic14]. Here we change the original event informations to time frames, because the sampling rate of DVS is so high, that a real-time reaction for every event is not applicable under limited computational resources.



(a) Full network structure for motion estimation. The segments crosses represent the recurrent connection from the six output neurons to the focus map



(b) Direction neurons with respect to 6 directions

Figure 3.5: Basic structure of BP neural network

The structure of the model introduced in Fig. 3.5 is extended from focus map of [Mic14], shown in Fig. 3.1 (b). The focus map is separated into several segments. Activation level of neurons in each segment together with their time shifted version are treated as input of an 3-layer neural network to detect active motion. The output of each network are 6 neurons corresponded to 6 directions of motion. These output neurons are then connected to the neighborhood of the original segments. Once an active motion in a direction is detected, these neurons send a small activation δ to the corresponding neighbor in that direction, to *help* the focus map find the most likely position of the object in the next time frames. The sum of all predicted activation is normalized in every step, i.e. $\sum_{allsetments} \delta \leqslant \Delta$ holds for every update.

Some structure parameters must be defined before training the network. These parameters are: number of layers, number of input/hidden/output neurons, regularization coefficient λ and recurrent connection weight from output neurons to neighboring segments. Some of them are fixed due to the limitation of project time and computational cost. The fixed parameters are number of layers 3, number of output neurons 6. The rest of the parameters are flexible and evaluated through simulation tests presented in 3.3.2.

3.3.1 Network Training

BP learning treats the training of the network as an optimization problem of cost function 3.18 [Ng]. Assume that we have K output neurons, N input neurons, L layers i.e. $L-1$ weighting matrix $\mathbf{W}$ and m training examples shown in Fig. 3.6. The input and output of i_{th} example $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ is defined as $[x_1, x_2, \dots x_N]^T$ and $[y_1, y_2, \dots y_K]^T$. The resulting output of the network is defined as $h_{\mathbf{W}}(\mathbf{x}^{(i)})$. Every step of the feed forward process is shown in 3.15, 3.16 and 3.17. According to BP algorithm the errors of every output neurons and hidden neurons are calculated. With these obtained errors, it is feasible to calculate the partial derivative of every weighting value. A gradient descend method is used to find a minimum of the cost function. Details of BP algorithm can be referred to [Ng].

$$\mathbf{a}^{(1)} = \mathbf{W}_1 \mathbf{x} \tag{3.15}$$

$$\mathbf{a}^{(l)} = \mathbf{W}_l \mathbf{a}^{(l-1)}, \text{ for } l > 1 \tag{3.16}$$

$$h_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}_{L-1} \mathbf{a}^{(L-2)} \tag{3.17}$$

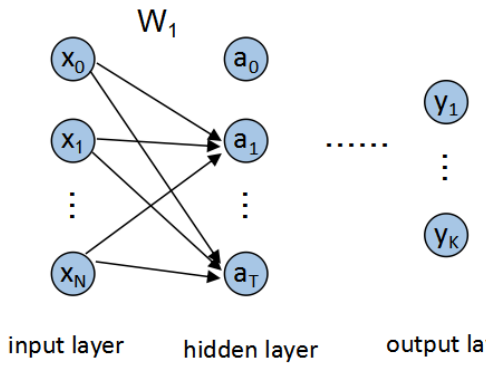


Figure 3.6: The structure for motion estimation network

$$\min_{\mathbf{W}} \left\{ -\frac{1}{m}\left(\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_{\mathbf{W}}(\mathbf{x}^{(i)})_k\right) + (1-y_k^{(i)}) \log\left(1-h_{\mathbf{W}}(\mathbf{x}^{(i)})_k\right)\right) + \frac{\lambda}{2m}\sum_{l=1}^{L-1} ||\mathbf{W}_l||_2^2 \right\},$$
$$\text{where } h_{\mathbf{W}}(\mathbf{x}^{(i)})_k \text{ is the } k_{th} \text{ element in } h_{\mathbf{W}}(\mathbf{x}^{(i)}) \quad (3.18)$$

As the BP learning needs to be supervised, off-line training before application is necessary. The introduced network for direction detection is trained with simulated activations. However, the full-trained network performs well when provided with real DVS data, which will be presented in the next section.

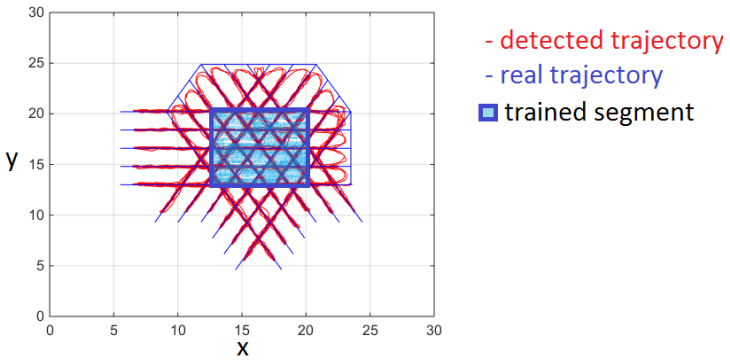


Figure 3.7: The trajectory of activation for training the neural network

Because all segments are assumed to perform identically when detecting directions, it is sufficient to train a single segment, and apply the trained parameters to all others. The trajectory of the simulated activation are shown in Fig. 3.7. Motion of 6 different direction are generated with same amount of time frames. All settings for

Table 3.3: Parameters for training the network

Moving Velocity (Pixels/Frames)	Time Frames in total	Training Set	Cross Validation Set	Activation Threshold
0.3	10400	8270	2130	2

network training are listed in Tab. 3.3. It is designed to cover as many cases as possible when an object is found in the segment. Noted that not all time frames are used for training or cross validation. It is counted as a valid training/cross validation example only when the total activation in the segment is greater then a fixed threshold.

The trained parameter are all resulted from training set. The performance of the network with cross validation set will be used to evaluate the quality of the network structure parameters, in order to avoid the over-fitting problem. The size of training set and cross validation set are designed to have a ratio of $4:1$. In order to maintain the training speed, an fast implementation of gradient descend is applied, which is **Edward Rasmussen's FminCG** [Ng].

3.3.2 Parameter Evaluation

There are 3 parameters to be determined for the motion estimation network, i.e. the number of input/hidden neurons and the regularization factor λ. In this section, different evaluations are performed to determine the optimal value of these parameters. The tests employ the input data, shown in Fig. 3.7. The maximal iteration number for gradient descend is fixed to be 1000.

The original attention network model by [Mic14] has a size of 30×30, in the evaluation it is slightly changed to 32×32 for computational convenience as $2^5 = 32$. Because the change is small, other parameters for attention network introduced by [Mic14] can be applied here.

The first parameter is the segment size, which also determines the input size of the motion estimation network. Since the segment size has a significant impact on the timing-performance of the network, a small size of segment is preferred. However, the size can not be reduced without restriction, otherwise the accuracy of the network will be affected. To solve this trade-off problem, an evaluation is performed, using different size of segment, while the other two parameters are fixed. Fig. 3.8 shows the result of evaluation. In order to keep a higher accuracy on training set and maintain lower computational cost, the size of 8×8 is selected.

The second parameter is the hidden layer size. A similar trade-off problem and evaluation is performed. The result is shown in Fig. 3.9. The size of 32 is selected,

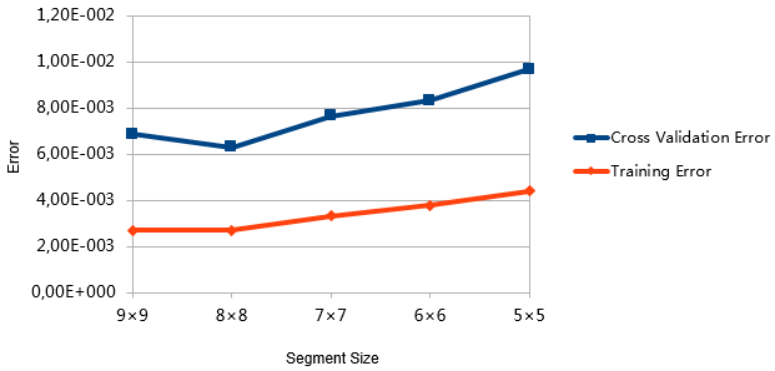


Figure 3.8: Evaluation for different input segment size (regularization factor $\lambda = 0.1$, hidden layer size $T = 32$, max iteration 1000). The Error represents the mean of euclidean distance between normalized direction vector of actual motion and estimated motion.

Table 3.4: Parameters of motion estimation network

number of layers	3	size of network in total	32×32
size of segment	8×8	number of segments	4×4
number of input neurons	128	number of hidden neurons	32
number of output neurons	6	regularization factor λ	1

in order to keep the accuracy on the training set.

The purpose of evaluating the last parameter λ is to avoid the over-fitting problem. The test results are shown in Fig. 3.10. The lambda is selected as $\lambda = 1$ in order to keep a low difference between training error and cross validation error.

According to these tests and former assumptions, the parameters for the network are defined as Tab. 3.4

An important requirement for the motion estimation network is, when no active neurons are found in the area, the estimated motion should be near to zero. This can be achieved using all-zero training data. Assume the 'no activation' situation is as important as one of the six direction, 1378 ($\approx 1/6$ of training set 3.3) examples of all-zero data can be generated and trained. In these training data, both the inputs and outputs are set to be zeros.

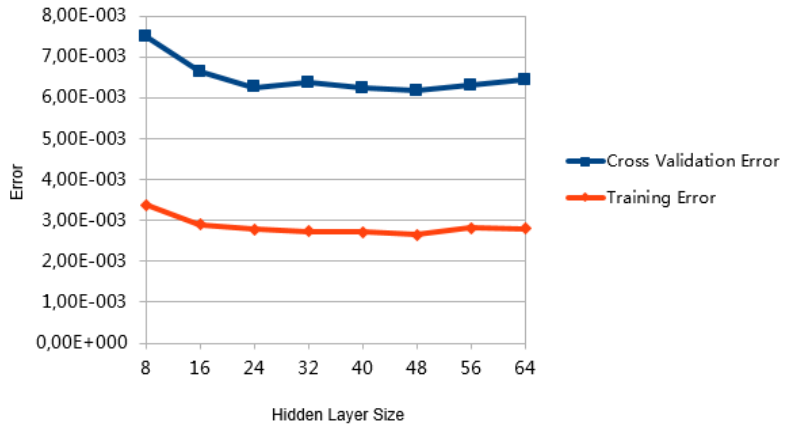


Figure 3.9: Evaluation for different input hidden layer size (regularization factor $\lambda = 0$, segment size 32×32, max iteration 1000). The Error represents the mean of euclidean distance between normalized direction vector of actual motion and estimated motion.

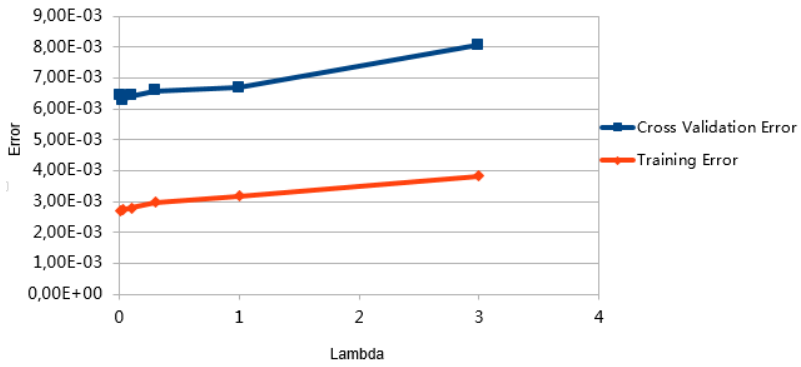


Figure 3.10: Evaluation for different regularization factor λ (segment size 32×32, hidden layer size $T = 32$, max iteration 1000). The horizontal axis represents $\log(\lambda)$. The Error represents the mean of euclidean distance between normalized direction vector of actual motion and estimated motion.

Chapter 4

Evaluation of Back-propagation Neural Network

In this chapter, different evaluations for performance of the network introduced in 3.3 are performed. The most important features are:

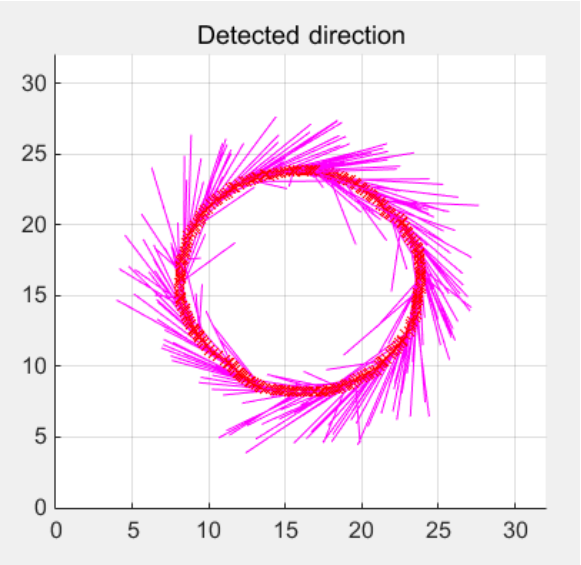
a). Robustness with complex trajectory,
b). Performance with collision case and
c). Performance with collision case in real DVS data

A simulated circle test, simulated collision test and person tracking test are performed to evaluate the performance of the network in the above criteria. General settings of all evaluations are: simulated object speed $0.3 pixel/frame$, predicted value $\delta = 0.01$, normalized predicted value $\Delta = 7$.

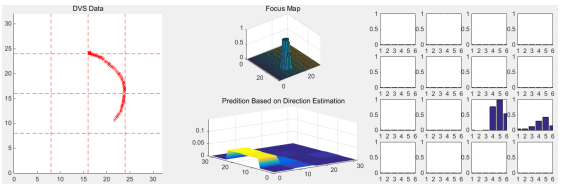
4.1 Simulated Circle Test

In this section, the performance of the network supplied by data of simulated object moving in a circular trajectory is evaluated. The purpose of the circle test is to evaluate the performance of the network when processing data with changing direction.

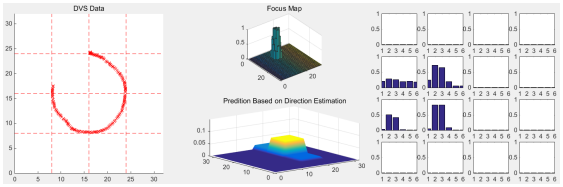
In this test Fig. 4.1, a moving object with clock-wise circle trajectory is simulated, and the direction of motion is estimated using BP neural network. The error in this test is defined as the mean euclidean distance between normalized direction calculated from middle point trajectory and the network-detected direction. If the motion estimation network is only performed for the most active segment, a error of 0.3688 is observed, corresponding to a angle difference of 20.9015 degree. If the motion of every active segments are all estimated, the error can be reduced to 0.3011, 17.1251 degree. From the simulation above it is obvious that the network is able to detect changing direction of the target object.



(a) The red crosses represent the middle point of the moving object while the magenta lines represent the detected direction using BP network.



(b) In 95_{th} time frame



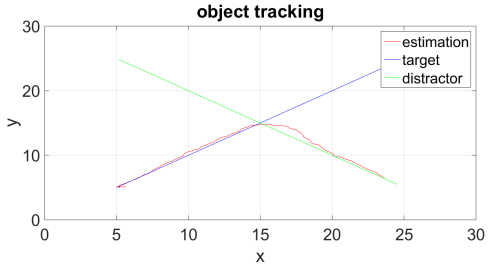
(c) In 193_{th} time frame

Figure 4.1: Evaluation of network when the object has a changing direction of motion.

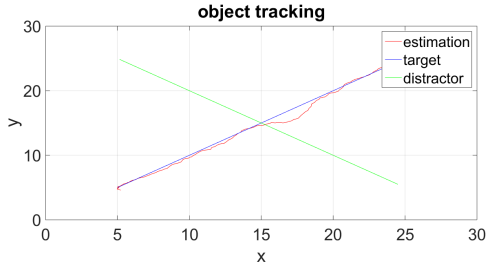
4.2 Simulated Collision Test

In this section, the performance of the network supplied by simulated data of two moving objects is evaluated. The two objects are overlapping in a specific time frame, in order to test the ability of the network in case of collision. Similar to the Gaussian probability model, the trajectory of the two objects are shown in Fig. 4.2.

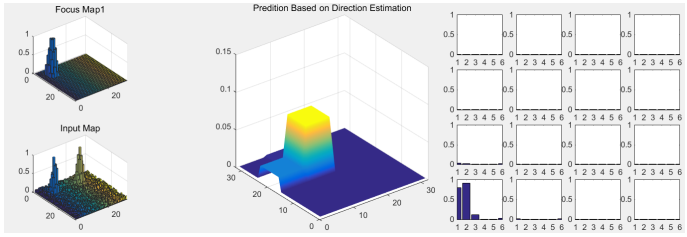
During the tests it is also to be observed, that when the collision point is away from edge of the segments, stronger prediction/direction estimation needs to be performed in order to keep the attention. When the collision happens far away from edge of the segments, the attention can fail. This safe distance needs to satisfy $d < 2\sqrt{2} \approx 2.8284$ pixels, which is determined through different tests.



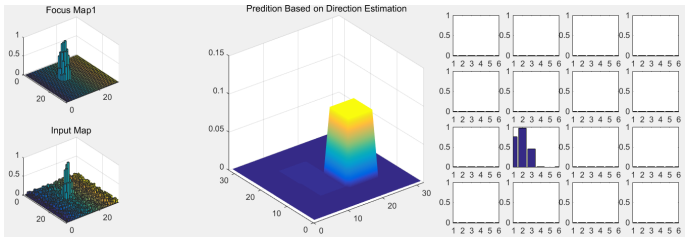
(a) without motion estimation



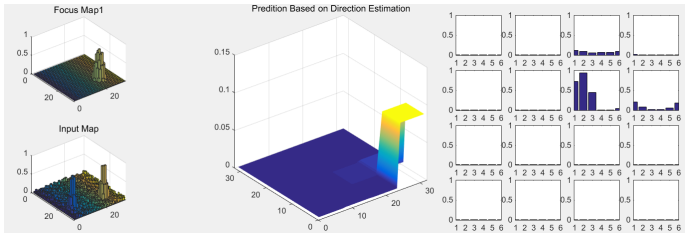
(b) with motion estimation



(c) before collision



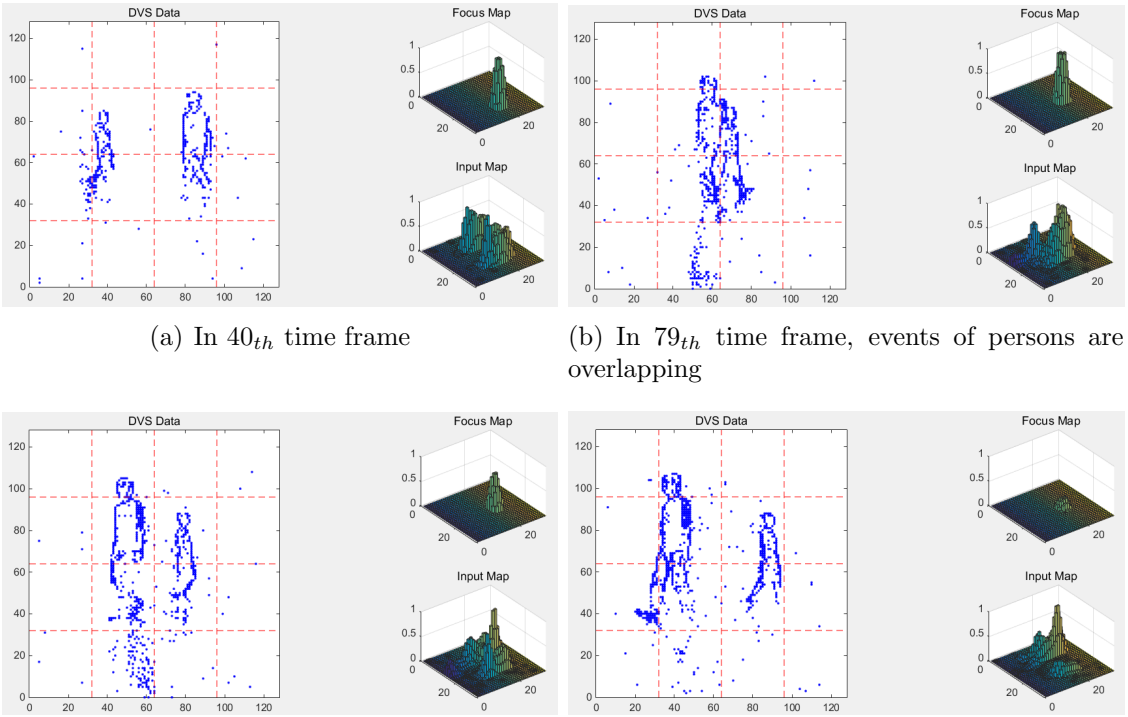
(d) during collision



(e) after collision

Figure 4.2: Evaluation of network when two object collide on the edge of segments.

4.3 Person Tracking Test Using Real DVS Data



(a) In 40_{th} time frame

(b) In 79_{th} time frame, events of persons are overlapping

(c) In 90_{th} time frame, the attention is dropping

(d) In 100_{th} time frame, tracking failed

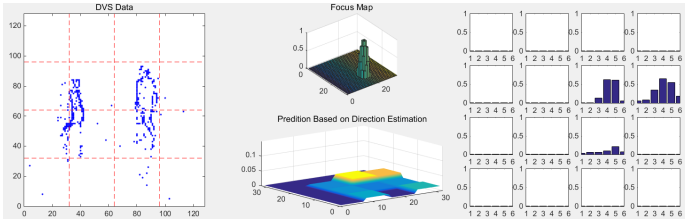
Figure 4.3: Evaluation of network when tracking one of two moving persons without motion estimation.

In this section, the performance of the network supplied with recorded DVS data of two moving persons is evaluated. Similar to last test, the moving persons have a crossing point. One person is moving to the right, the other to the left. Performance of the network with motion estimation Fig. 4.4 and the original network Fig. 4.3 are compared, so that the benefit of the embedded network can be concluded.

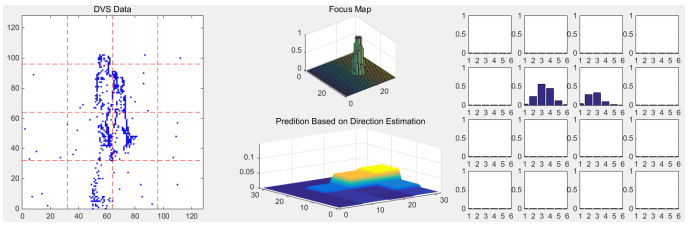
Fig. 4.3 show the recorded eDVS data with a resolution of $128x128$. The input map represents the probability for the position of the two moving persons. The focus map represents the output of the network, respectively the probability for the position of the tracked object. Focus map and input map has a reshaped resolution of $32x32$. The images are recorded at different time steps. At around 80_{th} time steps the two persons are colliding. Fig. 4.3 (c) and (d) indicate that the original network fails after collision. It is not capable to track the regarded person any longer.

Fig. 4.4 show the result of the same recorded eDVS data employing our neural network. Again different time steps are presented. The right side of each image represents the detected motion direction of each segment. Depending on the detected

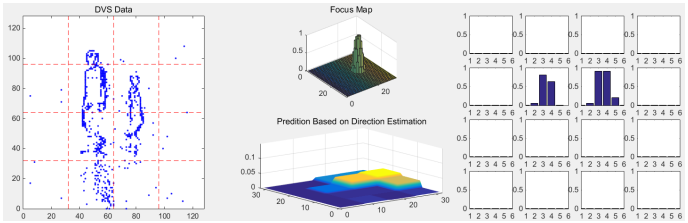
motion direction in each segment, a prediction value is added at the corresponding segment. Thus our network is capable to keep tracking the regarded person after collision.
The timing performance of the motion estimation network when processing real DVS data is also evaluated. A single time frame needs $30ms$ to be executed on a PC-laptop (Intel I7-4510U, 8GB of memory). When processing the same data without motion estimation, the demanding time is $25ms$. So the computational cost for the motion estimation part can be calculated, which is $5ms$ for 16 of 8×8 segments.



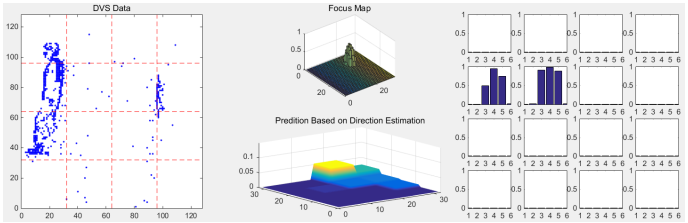
(a) In 41_{th} time frame



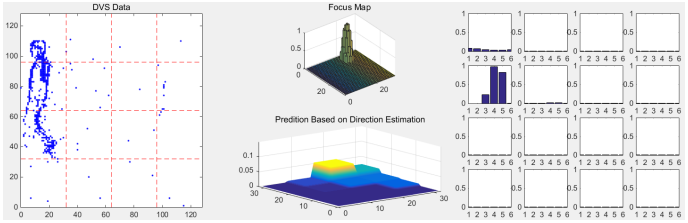
(b) In 81_{th} time frame, events of persons are overlapping



(c) In 92_{th} time frame, the motion of target person is estimated ($\approx 180 degree$, the yellow region in the prediction map.)



(d) In 118_{th} time frame, the movement of attention point is accelerating, to keep up with the moving target person. The attention level is therefore slightly lower.



(e) In 125_{th} time frame, tracking succeeded

Figure 4.4: Evaluation of network when tracking one of two moving persons with motion estimation.

Chapter 5

Conclusion

This paper employs mainly two approaches for motion estimation: one is spiking neural network and the other is firing BP neural network, both of which use event-based data provided by DVS sensor. The reason for choosing DVS data is that it provides enough information of changing pixels, thus the computational cost can be reduced compared to traditional frame-based methods.

In spiking neural network learning by STDP, it was not able to get the expected results. Here it analyses the possible causes. The most probable cause is parameter adjustment. As mentioned in 3.2, this system has 5 parameters to be determined (I_{thresh}, T_{LTP}, T_{refrac}, T_{inhibit}, Δw_+, Δw_-, τ_{leak}). Although there are some rules to help with determining the values of them (such as $T_{\text{inhibit}} < T_{\text{refrac}}$ or $\tau_{leak} \ll T_{\text{LTP}}$), it is still difficult to find out appropriate values. This paper tried to find out the desired values under the assumption that the potential state decreases exponentially and by fixing some of the parameters estimate the rest of the parameters. A related work [BQT+12] suggests genetic algorithm as a solution to parameter adjustment problem, which seems the most plausible solution.

In firing neural network learning by BP, the problem of collision case is solved in both artificial data and real scenario, and the primary goal of the paper is solved. The structure of the network is shown in 3.5. It has a similar structure as Elman network in a point of having time-shifted informations as input. Also several differences between the structure and Elman network can be observed, e.g. it has the output (detected 6 directions) recurring into the focus map, and only propagate to the last input information but not further. Each output neurons are connected to the neighbour segments of the most active segment. This recurrence provides information of which segment the object is moving to. This paper used MATLAB for developer environment for convenience. As shown in 4.3 the algorithm in this paper is able to be applied in near real-time applications.

The possible future work can be building up spiking neural network with DVS data

learning by STDP, which this paper failed to apply for real world scenarios. The biggest obstacle to overcome would be parameter adjustment that differs greatly depending the environment in which the object is, such as the speed of the moving object. Another obstacle would be tracking multiple moving objects. In this case more complicated structure, such as multiple hidden layers, is needed as shown in [BQT+12]. When there are two or more than two hidden layers, the system can learn by two methods, which are global learning and layer-by-layer learning, so future works will examine them.

The 3.3 structure can also be extended in 1) output neuron numbers, ideally 8. 2) re-parametrizing attention network to the size of 128×128. 3) re-parametrizing motion estimation network to further improve the performance of the system.

List of Figures

Bibliography

[BQT+12] Olivier Bichler, Damien Querlioz, Simon J Thorpe, Jean-Philippe Bourgoin, and Christian Gamrat. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32:339–348, 2012.

[Heb68] Donald Hebb. 0.(1949) the organization of behavior, 1968.

[Mic14] Le Michael. Stimulus-driven visual attention control for object tracking, 2014.

[Ng] Andrew Ng. Online course of coursera: Machine learning.

[RHW88] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

[RV06] Nicolas P Rougier and Julien Vitay. Emergence of attention within a neural population. *Neural Networks*, 19(5):573–581, 2006.

[Sch92] Carla J Schatz. The developing brain. *Scientific American*, 267(3):60–67, 1992.