Bayesian Computation in Recurrent Neural Circuits - a Neuro-Control Approach

MASTER THESIS

submitted by
Thomas Larasser, B.Sc.

born 29.04.1992
Gabelsbergerstraße 70
80333 München
Tel.: 0173 1877728

Chair of
NEUROSCIENTIFIC SYSTEM THEORY
Technische Universität München
Univ.-Prof. Dr. sc. nat. Jörg Conradt

Supervisor: M.Sc. Mohsen Firouzi
Start: 15.06.2016
Submission: 07.02.2017
Bayesian Computation in Recurrent Neural Circuits, a Neuro-Control Approach

01-Feb-2016

Problem description:
A large number of human sensorimotor control tasks are well explained successfully using Bayesian mathematics [1]. In addition to Bayesian approach to estimate the state of the world, Dynamic Bayesian Networks (e.g. HMM) are also widely used to model and control dynamic properties of the environment. The benefit of this probabilistic approach compared with deterministic models is its robusticity to deal with uncertainty and its superiority where the system is partially observable [2].
The challenging question of this thesis is how a recurrent cortical-like circuitry can perform such a probabilistic computation.

Method:
The method we chose to cope with this challenge is associative recurrent dynamics which is known as a well-known existing dynamics in cortical circuits. We would like to make a bridge between the dynamic Bayesian models (specifically HMM) and neural dynamics principles within a control approach.

In the last phase of this work we would like to demonstrate and validate the proposed neuro-controller using a robotic control experiment. The purpose of this experiment is to reproduce the human behavior during a saccadic eye movement [3]. The robotic platform is a 6-DOF robot-head equipped with stereo cameras and IMU units (Fig.1).

This work needs a motivated student familiar with probability theory, differential equations, and neuro-computing, with a basic knowledge of control theory. Programming skills in C/C++ or MATLAB are strongly necessary.

Task:
- Studying the literature about the application of the Dynamic Bayesian Network (HMM) in control theory, and associative recurrent networks.
- Simulating a basic Bayesian Dynamic Network using the associative networks’ dynamics:
  - Parameter identification of the neural dynamical system with respect to DBN
- Scaling up the basic network to solve the experimental problem.
- Demonstration and validation of the results.
- Documentation and report.

Bibliography:

Supervisor: Mohsen Firouzi (M.Sc)
Start: 15.06.2016
Delivery: 26.01.2017

(Jörg Conradt)
Professor
Abstract

Over the past years, Bayesian probability theory became a popular tool for modeling human sensorimotor control and behavior. It is commonly used in the creation of optimal controllers due to its robustness in the presence of uncertainty and noise which is also a pervasive property within human sensory measurements and dynamic control tasks like saccadic eye motion. The Hidden Markov Model is an explicit example for a dynamic Bayesian network. The idea of this work is to implement the latter as a state estimator of velocity, location, and movement direction of a moving object within a robotic controller in order to recreate human-like saccadic motion. The visual input is given by a neuromorphic dynamic-vision sensor (DVS). This is fed into the Bayesian network which is built as a recurrent neural model inspired by visual processing in the brain. The motion detection network proves to be robust against noise, detects multiple stimuli, and is able to react to motion perturbations. A winner-takes-all policy, with additional movement cost considerations, then generates output initiating a saccadic movement in order to center the moving stimulus in the foveal area of the visual field. Moreover, further model extensions are addressed which could enable the controller to produce smooth pursuit behavior - a more complex eye movement for continuous tracking.
I would like to thank my supervisor, professor, and colleagues for guiding and inspiring me during the making of this thesis. Special thanks go to my dear girlfriend for being a big support throughout every step and to my family for always having my back in six years of studies.
# Contents

## 1 Introduction

## 2 Basics

2.1 Understanding Bayesian statistics ........................................... 7
2.2 Hidden Markov Model (HMM) .................................................. 9
2.3 Model transformation - HMM within a neural network ................. 10
   2.3.1 Neuronal architecture - Leaky-integrate-and-fire neurons (LIF) ... 11
   2.3.2 Modeling the Markov Dynamics ..................................... 11
2.4 Eye motion - Saccades and smooth pursuit ............................. 13
2.5 Model for a sensorimotor controller ..................................... 15

## 3 Implementation

3.1 Tools and Material ............................................................ 19
   3.1.1 Robotic head setup .................................................. 19
   3.1.2 Robot Operating System ROS ...................................... 20
3.2 State estimation using a Bayesian estimator ........................... 22
   3.2.1 Framing the sensor input .......................................... 22
   3.2.2 Network topology - Forward and recurrent connectivity ... 23
   3.2.3 Generating a comparable posterior over layers ............... 33
   3.2.4 Evaluation of the network ........................................ 33
3.3 Utility function - Signal modulation depending on expected cost ... 48
3.4 Final target selection and controller output ........................... 49

## 4 Discussion

4.1 Ideas on smooth pursuit and potential framework extensions ....... 53
4.2 On the network performance - From serial to parallel processing ... 57
4.3 Neural principles - Using logarithmic signals and learning ........ 59

## 5 Conclusion

## List of Figures

## Bibliography
Chapter 1

Introduction

Brain and neuroscience research is one of the most trending research topics nowadays. According to an analysis of the worldwide and cross-disciplinary database Scopus between 2009 and 2013 nearly 16% of all published articles are related to this field of research - next to an annual publication growth of approximately 4% [35]. The incentives are clear. Aside from the medical advantages of an thorough understanding of the brain research, progress is closely monitored by researchers in hardware-software system theory as the brain often outperforms robotic or software applications, especially when it comes to adaption and learning tasks. Using neurophysiologic and psychologic findings and concepts as a guideline, may bring faster solutions to problems like machine learning or robotic movement coordination and vice-versa will help to deepen the understanding of brain models in neuroscience.

Unfortunately, it is still mostly a mystery how the human brain processes the high flow of information and acts according to it in a close to optimal way. Localization of the responsible brain areas is done mainly through lesion or stimulation research and psychological experiments supported by modern measurement and imaging technologies (fMRT, EEG, EMG, etc.). But exact processing mostly remains hidden and more importantly the extracted data is subject to the researchers reasoning. As Schall [33, p.78] stated,

“Responses to stimuli depend on the context in which stimuli are presented.”

Clearly the localization and understanding of brain functionality and of underlying processes are tedious tasks. Nonetheless, brain research is going on for more than 100 years and especially modern tools have speed up research progress significantly. Most of the proposed models try to mimic very specific areas or cognitive tasks, and thus can only successfully explain a subset of the rich experimental data[11]. Models on the visual processing, for example, are very often tested for their plausibility simply looking at their reflectance of certain ill functions in the brain e.g. when dealing with illusions or lesions as those are the most apparent sources of functionality. From the richness of those approaches a bigger picture begins to form. Hence uniting the different models and principles into more extensive ones starts to explain fairly well some of the processes the human brain is underlying.
In this work the focus will be on well studied neural models for sensorimotor tasks with emphasis on saccadic motion and smooth pursuit. The corresponding research topics spread from visual analysis, target selection, task discrimination, top-down vs. bottom-up processing pathways, up to various transformation models for motor output [37]. According to this we are looking into the results made so far in cortical visuo-motor processing and try at first to define what parts need to be realized for an artificial "neuro-controller". Furthermore a large number of human sensorimotor control tasks can be successfully modeled using Bayesian mathematics [13] (see figure 1.1) and specifically Dynamic Bayesian Networks like the Hidden Markov Model are widely used to model and control dynamic properties of the environment. Compared with deterministic models in general its robustness to deal with uncertainty and its superiority where the system is partially observable is outstanding. Thus, we pick those up in the hope to benefit of this probabilistic approach and evaluate their compatibility within neural models. The challenging question of this thesis is how a recurrent cortical-like circuitry can perform such a probabilistic computation within a robotic controller.

After explaining the underlying theory, the main focus will be on the implementation and evaluation of a Bayesian estimator within a neural network for the detection of motion. Aside from that, the whole control-loop from the visual input from neuromorphic hardware to the final target selection will be addressed considering corresponding models in neuroscience.

Figure 1.1: Basic model for human sensorimotor control. The controller, an optimal decision maker, takes into account both the output of the Bayesian estimation process as well as the utility function. The Bayesian estimator combines inputs from the sensors (e.g. limb or object positions) with prior knowledge in addition to the efference copy. A more extensive model and further explanations can be found in chapter 2.5. Picture adopted from Körding and Wolpert (2006)[14].
Chapter 2
Basics

2.1 Understanding Bayesian statistics

Bayesian statistics provide a systematic way of solving problems in the presence of uncertainty and define a strategy of how new information should be combined with prior belief which both helps to estimate states (ours or the worlds) or make decisions.

Bayes also seems to fit to human behavior. Looking at one simple example of ball sports then a combination of the experience/prior knowledge of how far a person normally can throw or shoot a ball with sensory feedback of the likelihood of the movement is how we estimate the future location of the ball with more certainty. Thus the two main properties of a Bayesian estimator are: (1) the likelihood is combined with the prior and (2) all quantities are weighted by their uncertainty. Let us for example assume we are dealing with univariate Gaussian distributions. Then $p(o|x)$ is describing the likelihood for some observations $o$ given a certain, likewise Gaussian distributed, prior $x$. We know the observations are noisy i.e. incorrect to some extent and we want to estimate the correct data in form of the posterior using the prior which forms our experience e.g. from previous trials. For the normal distributions this uncertainty can be expressed by their variance value $\sigma_o^2$ and $\sigma_x^2$ where usually the uncertainty for the prior is bigger than for the measurement ($\sigma_o^2 << \sigma_x^2$). The corresponding mean values are $\mu_{prior}$ and $\mu_{likelihood}$.

Using Bayes rule we get the following expression for the posterior:

$$p(x|o) = \frac{p(o|x)p(x)}{p(o)} \quad (2.1)$$

Figure 2.1 illustrates the described equation. In order to get the optimal estimate $\hat{x}$ from the resulting posterior distribution we use the maximum-a-priori method (MAP):

$$\hat{x}_{MAP} = \arg\max_x p(x|o) \quad (2.2)$$
Figure 2.1: Procedure of Bayesian integration/estimation in a statistical optimal fashion using Gaussian distributions. Green represents the resulting posterior after the combination of the likelihood (yellow) and the prior (purple).

which resolves to:

$$\hat{x} = \lambda \mu_{likelihood} + (1 - \lambda)\mu_{prior} \quad ; \quad \lambda = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_o^2} < 1 \quad (2.3)$$

Therefor -if the likelihoods and the prior probability are Gaussian distributions- the MAP estimate has a very simple form and reduces to a convex combination of the two means where the weights are inversely proportional to the variances. Furthermore we can calculate the posterior variance as $\sigma_p^2 = \lambda \sigma_o^2$ where $\lambda < 1$. Hence, the uncertainty for the correct estimate according to the posterior is less than for the estimate solely based on measured data. This is an outcome which was expected looking at the example of the human estimation process from before. Consequently, the Bayesian framework seems to explain fairly well human performance in some multi-modal and especially perceptional tasks\[13\] [41]. Moreover, Bayes offers an approach to an "optimal controller" and already has a long history in probabilistic robotics (eg. Kalman filters). Similarly, Körding and Wolpert (2006) reason that the Bayesian theory is applicable on sensorimotor control tasks.

In general, Bayesian statisticians argument one can make a Bayesian model do anything by designing a sufficiently complex prior [41]. There is increasing evidence in research on the role of Bayesian framework in cognition. But it has to be carefully evaluated if and especially how and where the brain itself is Bayesian. There is a problem with the idea of a "Bayesian brain" and that everything might fit into a Bayesian framework. It is very useful for designing systems but generally the functionality of a cortical system is over-simplified and biased by a more or less subjective prior which is the main point of critic from the side of researchers following the strictly objective frequentist concept of probability. Thus postulating that
the brain is Bayesian is kind of far-fetched. Nevertheless, it still provides valuable characteristics helping to understand the world and some brain processes.

To conclude, Bayesian integration provides a sophisticated yet easily applicable way to deal with the uncertainty in the world. Consequently, it is very recommended to consider when, for example, developing a controller. Especially in our case for the realization of a “neuro”-controller as noise is ubiquitous in cortical-like signals and measurements.

2.2 Hidden Markov Model (HMM)

Hidden Markov Models (HMM) provide a potent tool for modeling temporal and sequential data [25]. Their specific methodology makes them computationally very efficient offering themselves for applications with close to real-time performance. Typical state of the art speech recognition systems and artificial intelligence applications make use of them. They are understood to be dynamic Bayesian networks [7] and several algorithms solving various inference problems exist. The Viterbi-algorithm for instance finds the most likely sequence of past states given a series of observed data. This characteristic is exploited in handwriting recognition. The Backward-algorithm gives information about future observations given the current state of the process which is producing such observed data and the Forward-algorithm tells what is the most probable state corresponding to the most recent observations. Nevertheless, those are no optimal solutions. But they provide a good and efficient approximation due to some assumptions we make. Figure 2.2 illustrates the defining properties and dependencies between states and observations according to the Markov model. Exploiting those characteristics the joint probability can be factored to a common form as:

\[
P(S_1,\ldots,T; S_1,\ldots,T) = P(S_1)P(Y_1|S_1) \prod_{t=2}^{T} [P(S_t|S_{t-1})P(Y_t|S_t)]
\] (2.4)

Figure 2.2: Graph showing dependencies of a dynamic probabilistic network - the Hidden Markov Model
CHAPTER 2. BASICS

Hidden states: Visible to the observer is only the data input $Y_t$ generated by some internal process at time $t$ whose state is described by the state variable $S_t$. The form of a handwritten letter which is varying due to the individual motor characteristics over different writers, for example, is the output of a hidden process with the intention of graphically depicting the letter in order to transfer its meaning. The observation hereby can be fully explained by the modeled state at the corresponding time where the observation experiences a certain degree of variance from the ideal state.

Markov property: The sequence of hidden states satisfies the property of a Markov chain. It describes a memoryless process saying that probabilities of future states only depend on the current one or that past states depend on the preceding step respectively (see equation \(2.5\)). This is also known as a first-order Markov process.

$$P(S_{t+1}|S_1, S_2, ..., S_t) = P(S_{t+1}|S_t)$$  \(2.5\)

Closed property: One state can represent one or several parameters which values are part of a finite set of discrete values. This reduces complexity but as well decreases accuracy. Depending on the model this does not play a big role and can be neglected. Nevertheless, it leads to the fact that a corresponding state-space model like the HMM is closed under their respective state transition probabilities and emission probabilities of observations. Saying that with a transition from any state the next state is part of the state space described by these probabilities and moreover every state transitions follows the same principles.

These properties make Markov models peculiarly favorable as they lead to fast recursive algorithms like the ones mentioned before. They are understood to be closely related to Kalman filters to the effect that changing the state-space from discrete to continuous domain is enough to transform a HMM to a Kalman filter.

2.3 Model transformation - HMM within a neural network

Neuronal models help to investigate human brain mechanisms but often at the expense of computational efficiency. On the contrary hybrid neuromorphic systems might exploit synergistically advantages from both fields, high speed parallel processing of biologically inspired and well performing algorithms. That’s a good incentive for the realization of Bayesian networks in the form of actual neuronal networks and vice-versa could result in a more thorough understanding of the neural principles.
2.3.1 Neuronal architecture - Leaky-integrate-and-fire neurons (LIF)

The Neurons in this approach are modeled like leaky integrate-and-fire (LIF) neurons. A simple and popular neuronal model which dynamics are described by:

\[ \tau \frac{dv}{dt} = -v(t) + WI(t) \]  

(2.6)

where \( \tau \) is the membrane time constant, \( v \) represents the output, and \( I \) the input potentials, while \( W \) can be taken as weighting factor or neurophysiologically spoken the membrane resistance. For the controller, the implementation of a recurrent network is necessary in order to include the time dependencies of a dynamic system. Therefor, we extend the simple LIF-neuron equation by another term of now recurrent connections \( r \) with weights \( M \):

\[ \tau \frac{dv}{dt} = -v(t) + WI(t) + Mr(t) \]  

(2.7)

The discrete integral form of this equation (with a constant integration rate assumed to be 1 iteration per time step) for any neuron \( i \) in the network then results to be:

\[ v_i(t + 1) = \sum_k w_{ik} I_k(t) + \sum_j m_{ij} v_j(t) \]  

(2.8)

Now, after the general neuronal architecture is clear, the question is how this can help us to implement the actual Bayesian network with Markov dynamics. For this we follow the model transition approach of Rajesh (2004)[27].

2.3.2 Modeling the Markov Dynamics

In order to solve the state estimation step of the proposed controller model with a Bayesian network one has to solve an inference problem of the following kind. In humans the sensory system provides direct observations \( I(t) \) and data about the current state \( \theta_i(t) \) of the body or the environment. Hence, the possible states \( \theta_1, ..., \theta_N \) are not directly observed and form the hidden states of the Markov model which are producing such data. Acting accordingly necessitates to decide on the most probable state for the observed sensory input. Consequently we have to know the posterior probabilities \( P(\theta(t) = \theta_i|I(t), ..., I(1)) \) for all the states.

The Forward algorithm provides a solution for the joint probability distribution \( p(\theta(t), I(t), ..., I(1)) \). Using the Product rule i.e. factoring in two different ways we rewrite this term as:

\[ p(\theta(t), I(t), ..., I(1)) = p(\theta(t)|I(t), ..., I(1))p(I(t), ..., I(1)) \]

\[ = p(I(t)|\theta(t), I(t-1), ..., I(1))p(\theta(t), I(t-1), ..., I(1)) \]  

(2.9)

This shows that the joint probability is proportional to the posterior probability we are looking for. In the second part of the equation the likelihood for data \( I(t) \)
only conditioned on the current state due to the Markov property. Consequently, the dependencies on previous data cancel out. Before we extract the posterior probability distribution we marginalize the last term over the previous state \( \theta(t - 1) \) and factorize as follows:

\[
p(\theta(t), \mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) = \sum_{\theta(t-1)} p(\theta(t), \theta(t - 1)|\mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) \\
= \sum_{\theta(t-1)} p(\theta(t)|\theta(t - 1))p(\theta(t - 1)|\mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) \tag{2.10}
\]

Now, inserting equation 2.10 in equation 2.9 we finally can solve for the posterior probability and get a formula which is recursive over the posteriors from the previous state:

\[
p(\theta(t)|\mathbf{I}(t), \ldots, \mathbf{I}(1)) = kp(\mathbf{I}(t)|\theta(t)) \sum_{\theta(t-1)} p(\theta(t)|\theta(t - 1))p(\theta(t - 1)|\mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) \tag{2.11}
\]

Here, \( k = \frac{1}{p(\mathbf{I}(t), \ldots, \mathbf{I}(1))} \) is a normalization constant. Furthermore, apart from the recursive term this equation consists of the transition probability distribution \( p(\theta(t)|\theta(t - 1)) \) and the emission probability distribution \( p(\mathbf{I}(t)|\theta(t)) \). As equation 2.11 is purely multiplicative there is yet no way of finding a model transition between the Bayesian model and the neural architecture. But looking at it in the log domain gives us:

\[
\log p(\theta(t)|\mathbf{I}(t), \ldots, \mathbf{I}(1)) = \log k + \log p(\mathbf{I}(t)|\theta(t)) + \\
\log \left[ \sum_{\theta(t-1)} p(\theta(t)|\theta(t - 1))p(\theta(t - 1)|\mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) \right] \tag{2.12}
\]

Thus, comparing with equation 2.8 we can find the following correspondences:

\[
v_i(t + 1) = \log P(\theta_i(t)|\mathbf{I}(t), \ldots, \mathbf{I}(1)) \tag{2.13}
\]

\[
\sum_k w_{ik} I_k(t) = \log P(\mathbf{I}(t)|\theta_i(t)) \tag{2.14}
\]

\[
\sum_j m_{ij} v_j(t) = \log \left[ \sum_{\theta(t-1)} p(\theta(t)|\theta(t - 1))p(\theta(t - 1)|\mathbf{I}(t - 1), \ldots, \mathbf{I}(1)) \right] \tag{2.15}
\]

The only term missing is the normalization term \( \log k \). Without it the networks output represents the joint probability. Rajesh here-for suggests adding the negative log normalization term after each integration step in the form of an log-sum-exp
of the according network evaluation period. In simple terms this acts as a recurrent inhibition layer where the global maximum (log-sum-exp performs the argmax operation) is subtracted preventing instabilities. This results in the final network form:

\[
v_i(t + 1) = \sum_k w_{ik} I_k(t) + \sum_j m_{ij} v_j(t) + \log \sum_j e^{u_j(t)}
\]

(2.16)

where \( u_j(t) = \sum_k w_{ik} I_k(t) + \sum_j m_{ij} v_j(t) \). As the network produces logarithmic probabilities a sum-of-log equals a log-of-sum in equation 2.15. For this reason we can not choose directly recurrent weights from our transition probabilities but have to approximate them for equation 2.15 to hold true. The standard pseudo-inverse method is the way to go. Here \( M \) is the recurrent weight matrix, \( T \) the matrix of log sums and if \( L \) is a matrix of random log probabilities \( \log x(t) \) we get the following representation for equation 2.15:

\[
ML = T
\]

(2.17)

Solving for the recurrent weights affords an inversion of \( L \). But \( L \) might be a sparse matrix and therefor not invertible. For this reason the pseudo-inverse of \( L \) is calculated:

\[
M = TL^T(LL^T)^{-1}
\]

(2.18)

More about how to choose the transition probabilities for different network approaches is presented later in chapter 3.2. At this point, with a few simple steps (Forward-algorithm, log-domain-transformation, recurrent weight approximation), a feasible Bayesian Markov computation can be performed in a cortical-like network.

### 2.4 Eye motion - Saccades and smooth pursuit

Two main classes of eye movements are saccades and smooth pursuit. Their purpose is to bring external objects of interest into the high resolved foveal region of the visual field for high-acuity examination. This is done via gaze shift. Meaning that either the eyes or the head or both are moving in order to deploy the object in the fovea. Reasoned on their distinct phylogenies and different functional constraints, corresponding cortical pathways are considered to be mainly disparate [40]. Furthermore, the pursuit task is usually split in two phases. First an open-loop initiation period is needed which is then seconded by a closed-loop steady-state phase. We also refer to them as Pre-pursuit and On-pursuit phases. The first phase of movement detection and stimuli selection is mainly relying on retinal motion signals but after pursuit initiation also extra-retinal signals become important as retinal output experiences degradation in form of increased motion blur and pseudo-dynamic stimuli from the static environment [21].
Looking at the timescales the pursuit onset i.e. saccade-programming lags approximately 100 ms behind of the beginning of motion activity with certain variability. Delay originates mainly from visual processing pathways. This timing can be prolonged up to 150 ms if one out of several moving stimuli has to be chosen or reduced when behavioral or cognitive factors like attention, motion anticipation, or learned motion trajectory are considered. When dealing with perturbation during smooth pursuit reaction latency is about 120 ms.

However, there has to exist a link between both kinds of eye movements in order to coordinate such a complex motion sequence. Figure 2.3 shows the schematic diagram for the basic anatomical substrates forming the pursuit circuit (more on the individual role of the distinct areas in chapter 2.5). Research on the cerebellum and the brainstem areas far downstream of the processing chain postulate their involvement in both saccades and smooth pursuit and show that they have tied relations shortly before the realization of the actual movement. Naturally, they also use the same signals from early visual processing areas. Consequently, following the physiological model we intent to realize a joint controller for both motion tasks although saccades and pursuit should be treated separately to some extent. Now, from the neural circuit schematic a controller model is derived (see figure 2.4).
2.5 Model for a sensorimotor controller

In the robotic control experiment using the mentioned resources a controller beyond the simple actor-sensor model will have to be implemented. The additional steps necessary are the use of sensor data in order to estimate the (hidden) state and its evaluation according to the control policy and a weighting utility function, up to finally making a decision on the best command. Furthermore, processing the active feedback of the motor command to predict sensory consequences plays an important role in order to achieve a fast control which is compensating for sensory delays. Sensorimotor control of somatic movements within a human are reasoned to follow a similar model - from pursuit to reaching - and a lot of research was done identifying the corresponding brain areas taking over the functionality of the different controller parts [36]. Figure 2.4 depicts the controller and those assumed equivalences on a rough scale. Note that for different sensorimotor tasks like reaching and smooth eye pursuit additional or distinct cortical pathways or areas have to be considered in varying relevance for the task at hand. Due to the experimental setup the focus will be on relevant control elements for saccadic eye movement and smooth pursuit.

**Sensory system:** The neuromorphic hardware used (compare chapter 3.1) gives us visual information about the dynamic environment and represents the retina and the first steps in the visual pathway retina-V1-V2-MT for motion detection. Pro-
prioceptive sensory signals like muscle strain and vestibular sensation can be taken from the load or positional encoding of the motors and the inertial measurement units respectively.

**State estimation:** After taking in the data, the sensory signals have to be correctly encoded for further processing. This step is called state estimation as the actual state is only represented by the sensor data, which is assumed to be imperfect i.e. noisy or faulty to a certain amount. Here we will realize a neurally inspired Bayesian estimator combining inputs from sensors with prior knowledge. In the brain this role overtakes the parietal cortex areas e.g. the middle superior lobe (MST) where visual motion signals are primarily processed in order to produce commands for eye movement [22].

**Utility function:** Mathematically the utility function defines the value which is prescribed to each possible outcome of our decisions. Different names are reward function or in the negated sense loss or cost functions. Mostly their goal is representing a scale with which minimizing the error, and effort or needed energy becomes possible. In general, saturation and other non-linearities are present in these functions which become more and more complex considering ever more factors effecting the cost. For tasks involving perception humans, however, are believed to use a relatively small number of utility functions in an adaptive manner [14]. This makes it promising to model the cost with just a few simple but important parameters while still being approximately consistent with the human behavior in sensory-motor tasks. On the other hand, non-perceptional cost functions may automatically play a part in every control task biasing the decision but overall the outcome should always be close to optimal. In the brain the ability to associate reward to stimuli depends on the basal ganglia and to associate reward to a spatial location depends on the medial temporal lobe [36]. However, if those areas are centers responsible for the cost in motor control or just gateways for such remains unclear.

**Feedback controller:** At this step the estimated state is chosen and transformed into a motor command weighted by a certain gain depending on the state and cost for the desired achievement. Other terms for this step are ”decision maker” or simply ”motor controller”. Depending on the motor task different transformations for varying goals have to be applied where a certain hierarchy is implied splitting the step in high level and low level feedback controllers responsible for the integration of state estimation signals, the ”cost-to-go” or direct sensory inputs. Mostly those pathways are assumed to be independently integrated. The sensory signals received as feedback for example take distinct routes e.g. from visual to (premary)motor cortex via posterior parietal cortex [12] and the proprioceptive feedback propagates to the (primary)motor cortex via the thalamus [34]. Also perceptual top-down modulation of the signals should be considered (e.g. from experience or expectation) for an comprehensive neuro-controller. An advantage is the compensative idea of this
2.5. MODEL FOR A SENSORIMOTOR CONTROLLER

approach. If one signal fails or is missing the others still might keep up functionality although less optimal depending on the task. In the cortical pathways the frontal eye field (FEF) and nucleus reticularis tegmenti pontis (NRTP) make main contributions corresponding to final saccade target selection [33] or setting the gain for the eye movement velocity and acceleration [22], pursuit initiation [40] and other motor-planning related functions. Subsequently, those signal are controlling the output activity in motoneurons.

**Forward model:** This step influences the state estimation as the forward model continues to predict sensory outcome of the chosen motor command like future target position. Taken into account in the estimation step those predictions form part of the prior belief providing sort of an efference copy as an internal feedback signal after initiating a movement. This feedback could have a simple corrective role in state estimation or play a major part in prediction of future states [36]. This has an even bigger weight as we assume visual sensory input during perfect pursuit to be close to zero [21]. For this an internal model has to be provided in order to keep up the functional drive for the controller output i.e. predict the sensory consequences. The brain area conjectured to be assigned such a role is the cerebellum. There is a clear relevance of this area for smooth pursuit which is shown by the consequences of lesions to the cerebellum which lead to a total impairment of this task [42]. Furthermore, the parametric adjustment for eye movements, for example ”mid-flight” when reacting to perturbations, and pursuit/saccade adaption or learning are considered to have their source here as well [6][40]. The importance is clear but the underlying processes are still undergoing intensive research. Also in a conventional controller this step can be seen as the most complex to implement unless this step functions as a mere switch between models regarding the task e.g. switching from the initial saccadic movement to smooth pursuit.

Clearly, more complex or detailed models might fit even better with experimental data (see for example the ocular pursuit model of Barnes and Collins [4]) Nevertheless, the theory always will differ from the neural mechanism to some extend and this model provides a good framework for the functional anatomy of different movement tasks formulated as an optimal feedback controller which was built on fundamental research in lesion patients. Note also that there are other strong models with nonsensory frameworks focusing on world centered representation (intrinsic or extrinsic space) which are said to facilitate on the one hand integration of relative movement e.g. head + eye + target and on the other hand ease the coordination of perception and action [11]. But they mainly postulate on explaining the transformation of retinal to world-coordinate representations on a perceptual level. Thus other auto-motor movements and reflexes like saccades may be better explained by the fast low-level integration of sensory input on which we focus in our model.
Chapter 3

Implementation

3.1 Tools and Material

3.1.1 Robotic head setup

Validation of the implementation requires a robotic setup with sensors and actuators desirably close to those of humans. The robotic head (figure 3.1) used for the experimental part of this work tries to find a good compromise between technical feasibility and equivalence. It origins from a previous work of the chair with similar intentions of functionality [20].

The basic skeleton is made of 3D printed solid parts where the printer deployed an opaque white photo-polymer (Objet VeroWhitePlus). A first compromise lies in the interpupillary distance which is 230 mm due to the size of motors compared to human eyes with about 63 mm.

Dynamixel servomotors from Robotis [3] are installed in a daisy chain configuration and communicate with the computer over a serial protocol. Because of the different loads the head uses two MX-64 (63rpm@12V; 6Nm) and the eyes four smaller MX-28 (55rpm@12V; 2.5Nm). Supported rotational movements are pitch and yaw.

Visual input is recorded with neuromorphic dynamic vision sensor (DVS) with a resolution of 128 x 128 pixels. This kind of sensor features pixels with responsiveness to temporal contrast at a resolution of about 15 µs and a dynamic illumination range of about 120 dB. The encoded scene reflectance change (ON/OFF-event for higher/lower-intensity) per pixels is propagated asynchronously to an on-board microcontroller which is connected to the computer using serial communications allowing for high timing precision and data rates up to 1 M-events/sec. Also called silicon retinas [16], they work in a way very similar to the human eye. Their characteristic to inherently detect information about edges already includes a valuable preprocessing step for the latter network by filtering for image activity or in our case moving edges of objects similar to contour cells in visual cortical area V2 [23].
This reduces calculation time and information redundancy especially compared to conventional cameras.

Furthermore the embedded DVS boards (eDVS) comprise inertial measurement units (IMU) providing information about 3-axis rotational velocities and accelerations representing the vestibular system of the robotic head. The boards are positioned appropriately so that the rotational axes of the eyes match those of the sensors. Correspondingly an additional IMU for the head is attached. An adapter board with a four port FTDI chip allows interfacing the whole setup with the computer via one single USB cord and only an additional 12 V power supply for the motors is necessary [28].

![Figure 3.1: Left: Setup of 6-DOF robotic head with silicon retinas, inertial measurement units, operated by servomotors. Right: 3D model with sketched degrees of freedom corresponding to the axes of rotation, 2 for each eye and the head respectively. Adopted from [20].](image)

### 3.1.2 Robot Operating System ROS

For a fast integration of the robotic head system from the sensory input over signal processing and eventually motor output the open-source Software Development Environment (SDE) ROS [2] is the system of choice. It provides a framework for multi-threading, an infrastructure for synchronous or asynchronous communication and hardware abstraction while keeping the implementation simple which enables fast development and easy access for non-professional users.

Supported programming languages are LISP, C/C++, and Python. Additionally XML and YAML are used for structural purposes e.g. as in YAML configuration.
files or XML package-dependency descriptions. Programs or groups of programs addressing all kind of applications are modularly organized in packages comprising the same architecture. Simply put, running a program is realized as a processing node which can publish or subscribe to so called topics in order to exchange data/messages with other independently working nodes as figure 3.2 depicts. This is organized in a hierarchical name-space structure. Diagnostic tools (e.g. rqt, rosbag) facilitate debugging, logging, and simulations without actually using real hardware.

The artificial neural network is implemented using MATLAB. Although there are ways to generate C code in MATLAB which would be necessary for the robotic operating system program a different approach is taken in order to keep the advantages of the clear and well understood MATLAB code. There exists support from the Robotics System Toolbox enabling communication between a ROS network and MATLAB emulating the key functionality.

Figure 3.2: ROS communications concept
3.2 State estimation using a Bayesian estimator

3.2.1 Framing the sensor input

As mentioned before the input of the silicon retinas are asynchronous events detecting changes in illumination and producing a binary ”retinal image” of local contrast. For our network we use a whole image, framing events every $t_{\text{update}}$ over a small period of time $t_{\text{disp}}$ (compare figure 3.3). Pixels with multiple events of the same polarity within one frame are neglected or otherwise subtracted. Doing this we simulate the scene framing of human perception and generate connected lines of moving objects. By using only fresh input of one polarity neglecting pixels with accumulated events the input becomes pattern invariant to a certain degree and the displayed edges are more likely to represent the actual object boundaries. Moreover, the used DVS-sensor pixels produce events for constantly moving objects that are ”by no means homogeneously distributed, ... a certain pixel may fire very quickly after an adjacent pixel, while another pixel of the same edge fires much later” (Rückauer, 2016) [30]. With this kind of preprocessing and further filtering this problem for event based algorithms is mostly avoided as the influence of single pixels on the resulting contour of an edge decreases. Nevertheless, edges of different broadness are generated depending on the rotational speed, the pattern of the object, and especially the duration of the displayed time window which is setting the lower limit of the dynamic range detectable by the network. To prove the concept only ON-events are processed but OFF-events could be used likewise. The difference would be a shift in the movement localization as moving objects produce edges of opposite polarity on opposing sites where the polarity depends on the contrast to the background. For our purpose we consider the information of ON- and OFF-events to be equivalent. On this created input additional preprocessing steps or filters can be applied. The big advantage is that the effort of designing a costly image movement detection and selection algorithm like in [41] is already overtaken by the used neuromorphic hardware in form of the DVS-sensors.

Figure 3.3: Framing of DVS input. $t_{\text{disp}}$ is the duration of the framing window and $t_{\text{update}}$ is the sampling period needed in order to get shifted retinal activity.
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

3.2.2 Network topology - Forward and recurrent connectivity

The initial phase of a pursuit task leads to an saccadic movement centering the object in the fovea. For this information mainly visual information is used and for the network to implement this kind of motion a detection or estimation of the parameters \textit{position}, \textit{direction}, and \textit{velocity} is necessary. Target position might suffice as relevant information for an infinitely fast moving system but the delay from movement initiation to reaching the desired position affords us to implement the calculation of at least the first order dynamical movement vector consisting of velocity and direction. With information about the acceleration this task could be performed even more accurately but for this the network would have to implement second order dynamics (second order Markov chain) which goes beyond of the scope of this work.

![Network layer example for place code neurons representing velocity sensitivity of one pixel per time step and in rightward direction realized by recurrent weights and input connections activating neurons depending on position of retinal activity. Neuron colors from cold to warm illustrate the increasing output activity over time for layer neuron sensitivity matching the target motion.](image)

Figure 3.4: Network layer example for place code neurons representing velocity sensitivity of one pixel per time step and in rightward direction realized by recurrent weights and input connections activating neurons depending on position of retinal activity. Neuron colors from cold to warm illustrate the increasing output activity over time for layer neuron sensitivity matching the target motion.

We find similar parameter sensitivities in the middle temporal visual area (MT) which is one of the responsible areas for retinal image processing in the brain and the main source for signals guiding pursuit eye movements [17]. So called ”place code” neurons appear to be tuned for different speed and direction of image motion where the neurons encoding actual target speed and direction have the largest response. In this model one neuron represents the state tuple of the motion vector as well as the retinal position of activity [18] in a topographical manner. Such topographic representations are found throughout the visual and oculomotor system [33],[17]. In other words, there is one neuron per pixel, direction, and speed. Tuning happens by correctly choosing the values for the recurrent weights. Figure 3.4 illustrates an exemplary layer for such ”place-code” neurons sensitive to right moving input at the velocity one pixel per processing period.
Role of input weights

Where the recurrent connections tune the neurons according to the mentioned dynamic parameters, the retinal input can undergo some changes, too, before being used in the detection network. Sensory, visual input from the used silicon retinas is noisy on the one hand and as the network works on frames rather than single retinal events it also greatly depends on the used framing method. Some of those disadvantages can be counteracted smartly choosing the connections and corresponding weights on the input path to the detection network. This way simple filters can be applied. Mathematically, a filter which is uniformly applied on a selected set of neurons on the retina performs a two-dimensional spatial convolution in the form:

\[ c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2)b(n_1 - k_1, n_2 - k_2) \]  

In practice the convolution is computed for finite intervals which are the sensor image matrix dimensions in this case. Here \( a \) is the filter matrix and \( b \) the input producing the output \( c \) which is usually of the same size as the input. This weighted input is understood to be the likelihood of the Bayesian model (first term of equation 2.16). As image input is binary so would be the probability for activity. However the framing method leads to ambiguous movement edges of different broadness. Using a directed gaussian weighting we bias the likelihood of the edge to be centered. This may shift the actual movement position but at this stage of detection no statement about the direction of the movement can be done and this expression of uncertainty is important in order to get meaningful separation of neighboring velocities later. As we are dealing with log-likelihoods, an arbitrary but small value is chosen replacing the zero-probability before performing the transformation into the log-domain. Neurally speaking a certain non-zero base-level activity of neurons is always present.

Noise filter

Similar to conventional cameras noisy pixel activity is existent in silicon retinas. Noise can easily be filtered out to a certain level using a natural Gaussian-like weighting on neighboring pixel and thresholding the result. Thus, only pixel in areas of higher activities are forwarded to the detection layers as they experience additional excitation from the surrounding pixels. The discretized filter base is shown in Figure 3.5. The values sum up to one, fitting the probabilistic framework. In order to inhibit the propagation of noise and the blur produced by this kind of filter the minimal threshold has to equal the maximal filter value. This is valid for sparse noise with no activity in the direct neighborhood. The vicinity considered by the filter is given by the non-zero width of the filter. Consequently the threshold value has to be increased for higher noise densities. Aside from its potential for noise inhibition the application of a Gaussian filter helps to stress the ”center of mass” of the input stimuli which is favorable for motion detection algorithms in order to detect the center of the input stimulus. As we deal with the edges of the target object due to the DVS-sensor no actual object center will be
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.5: Discrete Gaussian filter with broadness of 3 pixel. Values summing up to 1. Minimal noise filtering threshold should be equal to peak value.

detected. Fast stimuli however introduce broader edges because of the framing over fixed time periods (cmp. chapters 3.2.1 and 3.2.4). Here a clear "center of mass" makes detection more concise. Taking a small part of the edge in movement direction would be more accurate but before the actual detection (without considering the event timestamps) we have no information about this most recent pixel event. Slow stimuli on the other hand might only produce sparsely filled edges mainly because of the asynchronous triggering of pixel leading to second order motion detection. Orientation selective Gaussian filtering can smooth out such interrupted lines. Thus, the application of such a filter is advantageous in various ways and easy to realize in a simple feed-forward network.

Another way of noise filtering is downscaling the input with the threshold depending on the interpolation method. Decreased size of input for further processing is computationally more efficient (compare chapter 4.2 on performance) while we still use non-parallel computing and will also decrease the number of neurons on the detection layer. For this approach we used \textit{bilinear} interpolation only including directly neighboring pixel. The threshold is chosen to be 0.25. Furthermore stimuli input broadness will be downscaled as well by the factor of scale improving detection of moving edges. Albeit the width reduction of the stimuli choosing the right threshold guarantees preservation of single pixel width lines. Being less physiologically consistent performance considerations make this kind of filter more suitable for a fast controller. Unless otherwise noted the former kind of noise filter is used for later simulations.
CHAPTER 3. IMPLEMENTATION

Figure 3.6: left: DVS-Frame of ON-events of slow moving pen stimuli with noise. right: Noise-free frame filtered with thresholded Gaussian-filter.

Centroid filter One way to optimize the input for the movement detection network even further is to apply a centroid filter. Ideally this way all inputs are reduced to edges with pixel broadness of one moving at their corresponding velocity. Figure 3.7 shows the iterative filter procedure we designed only using neural principles. At the first iteration the noise-filtered DVS-input is taken and convolved with the "inner contour" - filter enhancing most single edges which are then preserved by using the high-threshold binarization and by differencing this information from the output of the low-threshold binarization which incorporates information about the inner contour to be reduced. As the size of the edges is reduced by one pixel for both sides, several iteration steps have to be taken in order to thin out the edges further. The advantage of this approach is the use of only convolution, thresholding and substraction/suppression which can be easily implemented with cortical like circuits. However, this necessitates a fixed number of iterations. Moreover, these steps are computationally performant because the convolution is implemented in MATLAB as a fast built-in function which is needed for a real-time controller.

This is a custom approach but there are plenty of other possible parallel, sequential or iterative methodologies for centroid filtering from the area of pattern recognition in computer vision known as "thinning" or "skeletonization" [15]. Our method is similar to an erosion while preserving the smallest lines. Nevertheless, those are mostly based on morphological operations where a structuring element/matrix is translated over the image masking the underlying image structure with an logical "Hit-and-Miss" transformation or applying special rules. While being computationally performant those algorithm to our knowledge pend to be implemented in neuronal circuits. Outputs of such an morphological thinning operation are compared with the custom one in figure 3.8.

Where the thinning operation offered by the MATLAB Image Processing Toolbox reduces pixel broadness for all orientations shrinking the resulting object, the cus-
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.7: Custom centroid filter algorithm based on signal substraction and convolution. Top: Pre-detection signal flow of custom centroid filter. Bottom: Convolutional filter matrix for vertical edges, applied as ”inner contour” filter.

tom method preserves orthogonal edge information and reduces the edge broadness down to 1 or 2 respectively. A resulting size of two happens for ”even” initial broadness. Here the custom method performs better in preserving the geometric properties. While they both have similar processing time below milliseconds on the used system the morphological operation is orientation independent. For different orientations the filter matrix of the customized method has to be rotated accordingly and thresholds have to be designed properly. In order to get a larger number of orientation a more complex filter matrix including more neighbors is needed. Every orientation selection would need such a customized network. Although less neurally representative the inherent MATLAB thinning is used in the later controller due to the orientation independence and therefor reduced design effort. The custom approach to a more neurally inspired method seems valid but might be a research topic on its own. Especially its robustness against noisy input has to be evaluated.
Figure 3.8: Centroid filter method comparison. Custom thinning reduces edge broadness down to 1 or 2 pixel depending on initial size. Morphological thinning shrinks orthogonal edges, too. All outputs were produced by three iterations of the respective algorithm. Both are almost equally performant regarding processing time.
Recurrent connections

So far we covered the general topology of the network and the input preprocessing. The recurrent connections hereby are responsible for the actual speed detection. In general, the recurrent connection approximately represents the transition probability from the previous state to the current state. Figure 3.9 shows the chosen probability values for the network layer detecting right- and leftward motion with velocity of one pixel per frame. The transformation from equation 2.18 is used to approximate the recurrent weights for the respective transition probabilities. The transition probabilities are chosen according to our prior knowledge about how a moving object generates activity patterns on the retina. Consequently, the signals on corresponding neuronal layers are enhanced, while proceeding over frames, when the estimate produced from the recurrent connection reflects the current retinal activation. Thus the neurons experience spatio-temporal frequency tuning. In terms of computer vision a "translation" happens. The difference to the procedure in computer vision is that here the geometric transformation is realized via recurrent connections and not via a transformation of indices. For rotational movement, scaling, or shearing a more complex connectivity regarding features and whole patterns instead of simple pixels will be necessary [9].

Fusing both directions into one layer expresses the opponency or cross-orientation inhibition of MT neurons [38]. Where detection of movement in the "wrong" direction suppresses the actual output for the opposite one i.e. the probability is decreasing (cmp. chapter 3.2.4). Consequently, unidirectional moving input is preferred over counter-directional movement e.g. when the stimulus corresponds to mere scaling from nearing or receding objects it will be suppressed. The transition probabilities near the boundaries (in this case first/last row and columns in the middle) where chosen to be uniformly distributed small random values.

In table 3.1 we illustrate the computation. Using the weight matrix from figure 3.9 the movement estimations are performed mathematically by multiplying the weight matrix \( M \) with the activity from the previous time step \( R_{t-1} \) (cmp. Equation 2.16) where \( M_1 \) is the upper-left quadrant for rightward motion and \( M_2 \) is the lower-right quadrant for leftward motion of the recurrent weight matrix.

Under the assumption that every pixel translation of the same sensitivity layer is based on recurring connections with similar values another more generic computational way would be the use of spatial convolution with a simple filter performing the corresponding translatinonal shift. The use of such filters will also be necessary when dealing with event-based input instead of frames.

Multiple velocities The figure displays the weights of one layer for horizontal movement but more layers for different velocities and directions could be added choosing the transition probabilities accordingly. Rajesh simply shifts the weight matrices for rightward and leftward motion downwards and upwards in order to encode higher velocities. By doing this the size of the middle column boundaries
stay the same but this leads to the effect that noise or additional input from the opposite layer directly influences the other one and leads to unwanted signal perturbation. This is avoided keeping the upper-right and lower left quadrant free from big transition probability values i.e. keeping the direct influence from the opposite layer low.

Furthermore, with this methodology the detectable speed is limited by the size of the input image or the angle of the sensor lens respectively. For the values between about half of the size and the overall input size the moving object might already have left the visual field before the probability for the respective layer becomes strong enough when integrating over frames. Here -among other parameters- the transition weights determine how fast the probability increases. For a more robust detection these weights should have smaller values. Faster detection on the other hand needs bigger weights (compare evaluation chapter 3.2.4). The discrete encoding of velocities demands to enable activation through neighboring pixel. Non-zero weights next to the actual transition weight allow for this side-activation. Otherwise, small oscillations in speed could lead to no detection at all, as decrease and increase of probabilities will oscillate as well. Nonetheless, the output will always be an integer velocity close the actual speed and discrete object velocities will be preferred.

Figure 3.9: Left: Transition probabilities \( P(\theta(t)) = \theta_i | \theta(t-1) = \theta_j \) for \( i,j = 1,\ldots,32 \). Probability values are proportional to pixel brightness. Right: Recurrent weights \( m_{ij} \) computed from the transition probabilities using Equation 2.18. Layout adopted from [27].
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Direction | Matrix Multiplication | Convolution filter
--- | --- | ---
horizontal | $R_{h,t-1} \ast M^T$ | left = \[
\begin{bmatrix}
x & x & x \\
x & x & m \\
x & x & x \\
\end{bmatrix} \quad \text{right} = \[
\begin{bmatrix}
x & x & x \\
m & x & x \\
x & x & x \\
\end{bmatrix}
\]
vertical | $M \ast R_{v,t-1}$ | up = \[
\begin{bmatrix}
x & x & x \\
x & x & x \\
x & m & x \\
\end{bmatrix} \quad \text{down} = \[
\begin{bmatrix}
x & x & x \\
x & x & x \\
x & x & x \\
\end{bmatrix}
\]
right-down | $R_{rd,t-1} \ast M_1^T \ast M_1^T$ | \[
\begin{bmatrix}
m & x & x \\
x & x & x \\
x & x & x \\
\end{bmatrix}
\]
right-up | $R_{ru,t-1} \ast M_2^T \ast M_1^T$ | \[
\begin{bmatrix}
x & x & x \\
x & x & x \\
m & x & x \\
\end{bmatrix}
\]
left-down | $R_{ld,t-1} \ast M_1^T \ast M_2^T$ | \[
\begin{bmatrix}
x & x & m \\
x & x & x \\
x & x & x \\
\end{bmatrix}
\]
left-up | $R_{lu,t-1} \ast M_2^T \ast M_2^T$ | \[
\begin{bmatrix}
x & x & x \\
x & x & x \\
x & x & m \\
\end{bmatrix}
\]
others | : | :

Table 3.1: Mathematics for recurrent computation. Matrix multiplication with weight matrix $M$ similar to figure 3.9 for different directions where $M_1$ is the upper-left quadrant for rightward motion and $M_2$ is the lower-right quadrant for leftward motion of the recurrent weight matrix $M$. $R$ is the recurrent activity. On the right exemplary convolution filters for velocity of one pixel per frame where $m$ is the strong connection performing the translation and $x$ are small random values ($x \ll m$). The latter becomes relevant for an implementation on neuromorphic hardware and when dealing with events instead of frames as the input.
Multiple directions Additionally, when dealing with a discrete visual field of pixels the combinations of orientation and velocity which can be implemented by this method are finite. For example one pixel only has 8 direct neighbors where the diagonal distance is slightly higher than the orthogonal one. This results in only 8 directions with an approximate pixel distance of one. For more, extra inter-layers trying to interpolate the pixels would be necessary. Moreover, the non-orthogonal directions of movement are again only linear combinations of vertical and horizontal movements which can also be seen looking at the mathematical computation for diagonal movement in table 3.1.

Furthermore, if the ratio of the pixel length of an edge and its normal velocity is bigger than one there will be an output not only on the actual diagonal detection layer but also on both the horizontal and vertical one. This is an illusionary effect called aperture problem which is as well present in human motion perception. Generally most of the movement detection algorithms suffer from this problem [30]. Here, the effect is produced due to the local pixel-wise detection which makes a correct statement on pixel movement but does not automatically consider the overall feature/edge.

However, this also means that with only orthogonal detection and knowledge about the features of a moving object most movement vectors can be expressed. When our network is detecting horizontal rightward movement with velocity 1 and with similar probability a nearby vertical upward movement with velocity 2, if those pixels happen to be on the same feature/edge, combining them gives us the actual diagonal right- and upward movement. But in order to do so a feature detection network extension would be necessary.

Nevertheless, an object pursuit based only on orthogonal detection with an high update rate would already be functionally sufficient as the output movement vector initiating the pursuit will be fully valid with respect to the object location and partly valid in the sense of the orthogonal velocity element. This is not a smooth pursuit but high update rates could overcome the impression of jittering movement.

Rectification and firing rate encoding model

So far the model outputs log posterior probabilities in the possible range from 0 to $-\infty$. The negative log term in equation 2.16 responsible for the normalization of the posterior ties the signal to a scope of 0 to $x$ with $x \in [0,-\infty]$ . However the continuous reduction of the probabilities for non-active neurons i.e. missing cell response saturation is on the one hand physiologically not plausible and on the other hand the selectivity of the network for new motion patterns suffers as the signal recovery from very low log-probabilities close to $-\inf$ takes essentially longer. Range limitation in the form of a simple encoding model including a rectification step resolves this problem:

$$f_i = [c \cdot v_i + m]^+$$  \hspace{1cm} (3.2)
where \( -\frac{m}{c} \) determines the minimal log value (base level activity) propagated between neurons and \( \exp^{-\frac{m}{c}} \) the minimal posterior probability respectively. With values of \( m = 100 \) and \( c = 10 \) we get \( \exp^{-10} = 0.000045 \) and see that bigger ratios become negligible very quickly. If not otherwise stated, we use the parameters \( m = 100, c = 15 \) in the following. \( [\,]^{+} \) is a half-wave rectification responsible for dropping smaller posterior values than given by the mentioned ratio. From the neuronal point of view extracellular response measurements (firing rates in our case) are, by definition, positive [38]. Thus, this encoding model additionally provides a way to represent the probability values with a neural firing rate of maximal m-Hertz and accounts for cell response saturation setting the dynamic range due to the rectification step. In general, various encoding models even for spike-based neuronal representations are possible [27].

### 3.2.3 Generating a comparable posterior over layers

Except for opposite movement directions the layer responses sensitive over different directions and velocities are independent. The divisive normalization affects only neurons of the same layer. Consequently, peak responses of log-probabilities over different velocities are the same and therefore no direct decision on the correct state can be made. But an estimate of target speed emerges if we normalize the values by the overall activity over populations of neurons. The firing rates over different layers as in figure 3.10—linearly encoding the log probabilities—illustrate the problem of comparing the signals directly. Normalizing over the on-layer activity results in comparable signals where the correct one is strongest. A comparison over all velocities brings similar results (compare chapter 3.2.4). Without implicating a similar implementation in the brain we profit again from using the log-domain as divisive normalization is performed by substraction like in equation 2.16. But now in order to compare the signals over layers we use division by the L1-norm (sum). This is done over rows for horizontal movement sensitivity, over columns for vertical detection layers and in the same way within other layers. This is a statistically motivated decoding where the overall signals over neural populations should sum up to 1 but there are a number of other mechanisms with potential neural implementations [17].

### 3.2.4 Evaluation of the network

After setting the mathematical framework inspired from neural principles we now evaluate the model for different kinds of input from well controlled inputs to real input given by the silicon retina.
Figure 3.10: Left: Firing rate and normalized posterior probability for motion detection layer sensitive to actual object motion (sensitivity = 3 pixel/frame; velocity = 3 pixel/frame). Right: Same plots as left side but with input velocity = 4 pixel/frame. Plots are over one row of the corresponding layers. Input is a rightward moving edge with horizontal broadness = 1 pixel.

**Testing signals over multiple velocities and layers**

Different detection layers encode discrete directions and velocities. Assuming no connectivity between such layers they have to generate comparable output signals. Coincident layer sensitivity and input patterns over frames should produce the strongest output in comparison. Testing on a simple input like a downward moving horizontal bar with a width of one pixel lets us examine the performance of the network at different velocities. The simulated input is shown in figure 3.11.

The maximal posterior probability of two layers at different input velocities after ten integration steps is depicted by bar plots showing also the difference over opposing directions which are encoded on the same layer (blue = downward, yellow = upward). The correct motion is detected on each as indicated by the highest probability present.

Both layers show some sensitivity to velocities close to their most sensitive one. This can be explained by the Gaussian input weighting and the Gaussian-like distributed recurrent weights which express the sensory uncertainty and results in weak side activation of neurons close to the real retinal activity. The last row of the figure tells us that the maximal probability found corresponds to the actual position of the bar input. Starting at position 1 this means ending at coordinate 31 after ten steps for 3 pixels per time step and ending at position 51 for 5 pixels per time step respectively. Hence the network is able to correctly detect state tuple of velocity,
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.11: Comparison of network response over different velocities with simulated binary input being a vertically moving bar with width of one pixel. First: 100x100 pixel input. Middle: Maximal posterior of layer sensitive to 3 (left) and 5 (right) pixel/time step over different velocities. Last: Posterior distribution for neural layers sensitive to 3 (left) and 5 (right) pixel/time step for moving input at 3 and 5 pixel/time.
direction and position for the used singular input.

The response from other layers detecting different directions of movement is not shown but posteriors from those are low compared to the most sensitive one. However, note that elongated input will lead to strong posteriors on certain diagonally selective layers as well due to the pixel-wise detection as mentioned before which does not differentiate the origin from the induced activity over frames (aperture problem). Directed and layer-selective Gaussian pre-filtering helps to combine edge orientation with movement detection in order to avoid resembling strong output over several layers which are meeting the input constraints.

**Behavior during integration over frames**

The framework is based on computations on time-series patterns. For the signal correctly encoding the observed state it has to manifest over time with increasing certainty. Evaluating the performance of the network over input frames we see an integration of relevant neural signals. Different parameter dependencies and their influence on the network behavior are investigated in the following.

The upper graph from figure [3.12] shows us the development of a strong posterior probability for the neural layer sensitive to velocity 3 pixel/time step. The result after 10 time steps is the same as in the left plot from figure [3.11]. After about 5 time steps the output settles on a certain level. A final probability value of one for a single neuron is not possible as the probability for the other neurons included in the divisive normalization would have to be zero which is never the case. As previously mentioned they are assumed to have at least a minimal output response i.e. a base level activity which is set by the rectification step. In general, we identify the following parameters to be crucial for the integration rate and network stability:

- $-\frac{m}{c}$ is the main parameter used for saturation of the negative log-probability. Figure [3.12] clearly shows that a decreasing of this ratio increases the maximal posterior value whereas an increase of the same makes it smaller. Due to the normalization the level of the posterior is closely connected as well to the number of neurons we are using for the normalization. Thus, varying sizes of input images provided by silicon retinas with different resolutions could be accounted for by the variation of the variables responsible for the rectification step. While there is an influence on the maximum probability value the integration rate stays about the same.

- In comparison, the initially arbitrarily chosen constant replacing the "zero"-probability when transforming the input image into the log-domain (s. chapter [3.2.2]) seems to have a significant impact on the integration rate. Increasing this parameter delays the saturation but seems also to provide a more robust integration by the means of a clearer separation from the non excited layers. At least for the tested simple input of a moving bar with a width of one
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

In general, this effect is explained by the sum of the likelihood with the prior in the log-domain. Smaller values for non-active pixel input added with the estimation from the signal of the previous time step means a more rapid degradation of the output signals down to a minimal value.

- Similarly, the estimation before the addition is determined by the recurrent weights. Consequently, this parameter also must have an effect on the integration rate as shown by the last plots of figure 3.12. The maximal posterior value is reached faster for high transition-probabilities. However, lower values lead to an eventually more stable performance compared over layers. An additional small influence of the recurrent weight on the saturation value is observed.

Altogether, for the current configuration we further choose the parameter variations on the values used in the upper plot. With \( \text{const} = 0.1, -\frac{m}{c} = -6.66 \) and the main recurrent weight = 0.8 fast integration with saturation within 5 steps and good separability over different layers is reached.

In the former sections we showed the capability of the network to successfully detect continuous movement. Yet, in the real world we encounter various perturbations in the motion of an object such as alternating velocity or the occlusion of the stimulus over short time periods. Figure 3.13 presents the network output for two alternating stimuli locally separated and with two different velocities. The network reacts accordingly, detecting the correct stimuli within expected time windows. An important effect can be seen at time step 20 when the object moving with 5 pixels/frame leaves the visual field. The signals from the relevant layer immediately drops to a negligible value but interestingly the previously active layer for the other stimulus (4 pixels/frame) stays at a certain value up to time step 25 before dropping as well.

The first finding is that with no stimulus present anymore the signal experiences slower degradation down to a probability of about 0.3 - 0.4. Secondly, the remaining relatively strong output comes from the previous input virtually propagating over the estimation layer until reaching the layer boundaries. Consequently, when the stimulus is occluded the probability for it decreases but is kept in its memory so to say encoding the virtual location over time without ”seeing” input. Similarly while the probability for a different stimulus increases the former signal degrades not promptly but at a specific rate similar to the integration rate. This also leads to faster integration for the re-appearing stimulus shortly after its disappearance. A memory effect like this is a valuable property for motion detection algorithms. The same behavior was observed for oscillating input.

**Test on broadness of input stimuli**

The framing method of chapter 3.2.1 is mainly responsible for the appearance of the stimuli on the retina i.e. it also chooses the range of detectable velocities. Objects moving with a speed of less than 1 pixel/frame (with one frame/t_update) will produce
Figure 3.12: Test of dependency of signal integration to different parameters. Graphs showing trend of maximal posterior probabilities over time. Input downward moving horizontal line with width = 1 pixel and speed 3 pixel/frame. On top: Reference plot with basic parameters used throughout experiments. Rest: Plots for parameter variations decreased (left) and increased (right). For a description of the respective parameters refer to the text.
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.13: Network reaction to changing object speed. Input again downward moving horizontal line with width = 1 pixel where only one line is visible for certain periods of time and both are located on different areas in the visual field. Graph is showing the maximal posteriors of the layers which sensitivity is corresponding to the stimuli.

no significant output as the objects motion is slower than the minimally detectable velocity of the network. Hence, the network only reacts to input of certain dynamics without being able to represent the static environment. On the other side, the upper limit of the dynamic range is given by the size of the visual field as input might vanish before integrated signal becomes strong enough.

Moreover, the ratio between $t_{\text{update}}$ and $t_{\text{disp}}$ determines the broadness of the activity field elicited by the motion of (homogeneous) objects. For example, an input moving with a speed bigger than this ratio is likely to be seen by two neighboring pixels within the framing window. If the speed is double the ratio the certainty for this is high i.e. the object boundary will be represented by two active pixels or neurons. Similarly, tripled speed will produce 3 active pixels per frame given that the activation of pixels is uniformly distributed. Thus the relation between stimuli width and speed is about directly proportional. Nevertheless, for velocities in between the discrete numbered ratios the broadness varies over frames due to the asynchronous nature of pixel output for the given hardware. Moreover, depending on the area of the visual field the object spreads over, it is likely to produce continuous input at least for some locations on the contour. Therefore bigger objects might be more easily or faster detected because continuous motion over frames is favored by the network as mentioned before.

The graphs in figure 3.14 show how the network in the current configuration (same as in previous evaluation chapters) reacts to different widths of input. The broadness is varied from 3 to 5 and assumed to be constant over frames. The bar plots show the maximal posteriors after 10 time steps for one layer at different input velocities and the graphs on the right show the posteriors of all layers over neurons of one matrix column also after 10 steps. It can be seen that increasing the width
Figure 3.14: Test on input width. From upper to last row different stimuli broad-
nesses where tested from 3 to 5 pixel. Left column shows maximal posterior prob-
abilities after ten integration steps for the layer sensitive to the object speed of 5
pixel/time step where certainty decreases with increasing width. Right column de-
picts the posterior probabilities over all layer for one exemplary column of neurons
where (only) the maximal value (green) has a correspondence with respect to the
maximal posteriors of the bar plots on the left.
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.15: Time plot of maximal posteriors over layers for a stimuli of width = 5 and speed = 5 pixel/frame. After an initial signal overshoot of layers sensitive to velocities close to the actual one the signal of the correct layer becomes strongest after about 7 integration steps.

augments the uncertainty of the input and therefore also the maximal value of the output encoding the correct state tuple becomes smaller. This originates from the Gaussian input weighting and the normalization step. Note also that while variance in velocity detection increases that the most likely speed is still close to the real one and more importantly that with increased width the position encoded by the strong signals on the false layers seems to be more correct than for edges with only one pixel in width. For this, compare the peaks of the plots over neurons of figures 3.11 and 3.14.

Looking at the corresponding behavior over time steps in figure 3.15 (input width = 5) endorses the importance of a correct localization of the input over exact speed detection. With the higher variance the separation over layers becomes less significant. Especially during the first steps of the integration an overshoot of neighboring velocities is present. Again the normalization is responsible for this trend. However, the continuous integration leads eventually to a correct distinction of signals in form of distinct posterior probabilities.

After all, we can assess that the network favors edges of a broadness of one pixel per frame which is produced by object motion meeting the constraints which are set by the framing method and the hardware. A different configuration or preprocessing might increase the capability of the network to treat inputs of different broadness equally. One such operation could be the application of a centroid filter mentioned in chapter 3.2.2. Another solution might be extending the network for parallel detection of several dynamic ranges i.e. feeding in with different framing parameters. In general, more complex input patterns from inhomogeneous objects will certainly effect the network performance negatively. This remains to be examined. Yet the tests on real input from the DVS-camera (see chapter 3.2.4) show promising results.
Detection during motion of multiple objects

Arising from the normalization over rows or columns for one neural layer we expect a splitting of probabilities for multiple objects moving at the same speed. The first couple of images of figure 3.16 correspond to such an input. Both horizontal lines are correctly localized by the layer sensitive to their speed. Like anticipated the signal saturates at an probability close to 0.5. These values stay consistent over time steps. The same happens for movement in opposite directions and equal velocities which is depicted by the second row of plots. Consequently, for more objects of the same speed their maximal posterior value on the respective layer can be expected to be approximately \( \frac{1}{\#\text{objects}} \). However, if the stimuli are starting from the periphery or one begins his motion earlier, the first appearing stimulus is preferred i.e. expresses higher probability over time. Despite the previous presumption that they will approach the same probability over time this is less of a problem for simple pursuit tasks as both stimuli stay in the visual field because of their equal velocity while only one is pursued. Starting from a location closer to the center, the network behaves as expected and splits the probability almost equally between inputs. The variations for signals moving simultaneously can be explained by the deviations in weights coming from the recurrent weight approximation of transition probabilities. This represents as well a more natural form of neural connections which can not be expected to be exactly equal despite having the same role in the network.

On the contrary, objects moving with different velocities are detected independently expressing the same maximal probability as earlier observed for single input (compare last row of figure 3.16). The reason for this is the missing connectivity between layers as we compare the neuronal signals locally which already provides us with a good separation of probabilities according to their certainty encoding the current observed state. For the final decision on simultaneously detected varying velocities additional weighting or slightly changed recurrent weighting for preferred speed might be implemented e.g. for choosing faster objects over slow ones as they would be more volatile. Similarly, the transition probability on the same layer could be adapted. This way the "attention" of the network could focus on certain areas like the foveal region as a form of visual attention selectivity similar to human vision.

When it comes to a pursuit task where both the sensors and the object(s) are moving this property of inhibition of same velocity output could be used for suppression of background movement which is induced by motion of the detection system itself (eye, head, etc.) as a lot of edges of the same velocity would be produced by the environment. If wanted, the separation of the detection over multiple patches of the visual field could help for simultaneous detections of objects of the same speed. Overall simple homogenous objects will be preferred as they tend to produce DVS-input only at the boundaries with different polarities on opposite sides.
3.2. STATE ESTIMATION USING A BAYESIAN ESTIMATOR

Figure 3.16: Test of input from several objects. First row: Input of two vertically moving bars moving at the same velocity (3 pixel/frame). Normalization over columns leads to a splitting of the maximally expected posterior about equally between inputs. Second row: Same as in first row but with opposite directions. Third row: Now second input moves with 5 pixel/frame. Due to the detection on different layers both show values close to the maximally expected posterior. Right plots show posteriors after 5 integration steps exemplary for image column 3.
Figure 3.17: Test of noisy input. First row: exemplary noisy input where density is given in % of overall pixel number. Plots are the averaged maximal posterior over all layers (1st row), detection quality \( \frac{\text{#correct signals}}{\text{#expected correct signals}} \) (2nd row), and ratio of wrong detections over correct detections (3rd row). Signals are detected if their posterior exceeds 0.5 and counted beginning with the 4th integration step up to the 7th where at first a 100% signal detection for zero noise is assumed for reference.
Robustness evaluation at different noise levels

For the previous evaluations, the quality of the input signal was assumed to be ideal, meaning also no noise was present during computations. However, the sensor itself which is providing information about the dynamic environment exerts some level of noise coming from various sources. Fast illumination changes excited indirectly by reflectance or directly by artificial light-sources, and the sensor hardware itself produce unwanted signals. The silicon pixel might for example generate signals simply due to cross-talk or broken circuit parts. Thus, this chapter shall provide an analysis of network performance at different noise levels.

From the tests on the width and on multiple input we can already state that more input signals will decrease the detections posterior probability. This is backed by the behavior of the average maximal posterior like we see in the first two graphs of figure 3.17. Note that where in this case only the average output is shown from a long bar input with configurations like in former tests there will be detection outliers with maximal posterior values. These are similar to those in the simulations without noise and also in the other direction with very small posteriors. In order to evaluate the detection over the whole input range we take for reference the noise-free detection signals over a reasonable threshold of 0.5 and calculate how many valid detections we get with increasing noise density (s. detection quality in 2nd row). As we know about the ground truth also the ratio of falsely enhanced signals over 0.5 compared to correct detections is shown (s. 3rd row). All plots are averaged over five simulations each with inputs over all test velocities (8) for 7 frames.

For a simple input of one pixel in width the average maximal posterior drops already with only little noise added and further decreases as expected. The threshold line of 0.5 is passed at a noise density of about 10%. At this point still about half of the expected signals are detected but soon more wrong detections than correct detections are made. On the other side a signal with width 3 has a much smaller decline in posterior probability and a good detection quality over most of the tested noise density range. This can not only be explained by the inherently lower signal-to-noise ratio but originates as well in the side-activation for recurrent connections making the signals stronger.

On the other hand, most wrong detections come from layers close to the tested one due to the side-activation property discussed earlier. But some are produced by noise patterns which seem to induce continuous motion activity over frames. Depending on the quality of this second-order motion the resulting posterior might have a stronger value than the ones encoding the correct state but this value is in most cases refined to only one pixel. Therefor, considering the output over whole layers would bring a solution for this signal separation problem as longer edges (more outputs with less potent values) might be preferred over single strong pixel output. Nonetheless, this adds more tuning for the preference of different object properties which might be unwanted for a universal detection algorithm.
Working with the DVS-sensors we expect only very low consistent hardware noise-density of a small percentage (comp. fig. 3.6) and other noise sources to be rarely affecting the input. But in order to account for the noise we use simple noise filters which could be easily applied at an early stage of the detection like already described in chapter 3.2.2. With this the detection quality is held over 80% for up to a noise density of 15% for one-pixel-width input and up to 30% for three-pixel-width input while the wrong detections are negligible. Note that the threshold-filtering (threshold = 0.75) starts to slowly fail for very high but unlikely noise densities. A higher threshold could be used to compensate this effect. Furthermore, aside from the noise-canceling property of this method, the input benefits also from the Gaussian weighting, stressing the center and smoothing the input edges. In general, we can say the network is already inherently robust to at least some noise and simple pre-processing steps further improve the performance.

Testing on real sensor input

The results so far with simulated input allow us to look optimistically at the test on real DVS-data. For this, the motion of a pen is recorded which is moved rightwards across the visual field of the sensor with oscillating speed while being attached on a string. The ground-truth for this motion was taken by comparing the mean pixel-position of the pen over frames resulting in discrete velocity values. Figure 3.18 shows the corresponding input exemplary for the first and the last frame. Furthermore, the ground-truth and the output signals of the state estimation network which posterior exceeds a threshold of 0.65 are plotted over frames. The hatched area illustrates the areas where the uncertainty between neighboring velocities was high. This originates from the elongated form of the input and the source of the motion-drive. The pen is moved by hand transferring the motion over a flexible string. This produces the oscillation of the speed magnitude and also the orientation of the pen alternates over frame. Thus, the upper part of the pen could have a discrete velocity different from the lower part which explains the uncertainty in the mean ground truth as we do not plot values in between.

The disk size of the thresholded network output depends on the number of detections where the signals detected in the correct direction are colored green and the signals encoding wrong directions are red. The position of the detections always agreed with the location of the pen on the retina. After some integration steps, needed for the signal levels of the according layers to rise, the detection mostly fits the ground truth. Even though continuous motion would be preferred the signal level of layers of which the sensitivity is close to the real one is high enough to react fast and produce strong signals within a few frames.

Interesting is the wrong detection at frame 20 where the pen halts for a short moment due to the pendulum motion. Here a downward movement is detected at the location of the pen. This comes from a second order motion i.e. pattern induced motion.
Figure 3.18: Ground-truth test on recorded DVS-input. The input is a pen moved rightwards across the visual field of the sensor with oscillating speed (between 0-3 pixel per frame) while being attached on a string. The first (a) and the last frame (b) of the resulting ON-events is shown. In (c) the corresponding discrete ground-truth velocity over frames (blue) is plotted against the network signals which exceed a threshold of 0.65. Hatched areas roughly depict uncertainty in the ground-truth values. The disk size of the thresholded network output depends on the number of detections where the signals detected in the correct direction (green) are differently colored than the signals encoding wrong directions (red).
CHAPTER 3. IMPLEMENTATION

The pen does not produce ideal bars like in the simulations. Moreover, when the velocity of the pen is close to zero only a few pixel along the dimension of the object will be active over frames. This noisy input could then result in a detection of an motion orientation parallel to the object. This effect is not present for a perfectly still environment. Increasing the threshold can avoid early detection of such volatile activity. Additionally a suppressive or enhancing weighting from previously chosen directions could be used as a form of an efference copy. But this would also result in a slower detection of changed movement directions.

Nevertheless the state estimation step correctly encodes the state of the object and provides us with enough information for the initiation of a saccadic movement using the framed input of the DVS-sensor. Following the underlying model from chapter 2.5 we now look with more detail into the still missing parts for a holistic controller.

3.3 Utility function - Signal modulation depending on expected cost

The visual analysis of the state estimator gives us an output response with the possible states of the stimuli having varying likelihood. Before transforming those signals into a motor command we consider another weighting step to be necessary in order to express the preferences or constraints of the used apparatus according to the general goal of minimizing the effort or energy consumed by the final movement. For the human such an effort can for example be the strain which is put on the muscles which finally indicates also the boundaries of the active range of motion. Other relevant factors are force magnitudes and force durations [14]. Moreover, the influence of some cognitive factors cannot be excluded for human motion selection. Compare for example psychogenic disorders [10]. Overall a selection of those factors might already sufficiently approximate the optimal behavior.

For the scope of this thesis we decided to limit the number of influences. Using motors instead of muscles, the force magnitude is fixed at a constant level choosing a reasonable value within the capable range of the motors. Theoretically, the small eye-motors attain a maximal value of rotational speed of 330 deg s\(^{-1}\) and the head-motor about 380 deg s\(^{-1}\). This varies by means of the effective load. However, smaller values are chosen in order to guarantee for the stability of the mobile non-fixated set-up when performing sudden movements. A possible force-ramp approach is neglected as this would eventually also result in a similar mean velocity but the data rate for the communication with the robot will be unnecessarily increased.

The most important factor we considered is the muscle strain which is also effected by the force duration and magnitude. It can also be understood as a decreasing weighting with increasing distance from the relaxation state. We simplify this desired state only regarding the visual field i.e. we define this weight \(w\) to be inversely proportional with the radial distance of the stimuli from the focal area which coin-
cides with the center of the visual field:

\[ w_i = \frac{1}{\sqrt{x_i^2 + y_i^2}/2}; \quad \text{with } x, y \in [0, 1] \]  

Consequently as the input images are quadratic the \( w_i \) can take on values between 0 and 1 where 0 represents one of the corners (\( i \) being the index the individual neurons/pixels of the retina). Note that using such a linear model may be oversimplifying. Shadmehr and Krakauer [36] for example uses a quadratic distance cost function for his model of a reaching task. Whereas Körding and Wolpert [14] tried to reverse engineer the utility functions related with force magnitude and duration. In our model we use the result from this function mainly as support for the later decision process on target position selection which is for now enough regarding the motor system because motors don’t suffer from constraints like muscle fatigue. For the ON-pursuit phase a more complex approach also considering feedback like the motor command might be necessary.

3.4 Final target selection and controller output

A matter of ongoing discussion is the final target selection considering the output from the state estimator, the utility function and other feedback sources. From firing rate or average neural response amplitude alone no unambiguous estimate of the target velocity vector can be made. A first step generating comparable input was already addressed in chapter 3.2.3. This resolves in comparable output over different velocity layers for a singular stimuli (compare evaluation results in chapter 3.2.4). Now, simple thresholding gives us a number of most probable states where the peak values are close to the MAP estimate. This is similar to a Winner-takes-all policy which seems also to be existing in cortical processes [8].

Another common approach for quantifying the signals of the neural layers MT over population is ”vector averaging”. Here, a center-of-mass of the population is computed dividing the weighted sum of neural activity across the population by the non-weighted sum where the weights are the preferred velocities. Within our method we can not use this computation directly. Looking for example at the firing rates of neurons over two different layers in figure 3.10, we see averaging over the activity would have the inverse effect as the noisy activity over population in the wrong layers is stronger. The model leads to strong suppression of activity of neurons encoding the correct speed but not the location of the stimulus. Hence, the relative firing rates over one layer are significant which is considered by the former mentioned normalization approach. This can be conceived as an indicator that the proposed estimator in its form might be less relevant in the sense that it can be used to explain processes in the brain because models like vector averaging seem to be closer to physiological signals. But whether exactly this kind of model finds its implementation in the brain is also still not clear [17] and functionally vector averaging and our method produce
the same result. Nonetheless, the first approach proves to be robust according to the evaluation conducted up to now and the underlying normalization might even be realized with neural connections. Thus, it is our method of choice for producing an unambiguous estimate of target speed.

Despite the fact that it allows for good comparability over velocity layers we still have distinct posterior values for different input widths when we neglect the neural validity of a preceding centroid filter. For two stimuli the one with broader retinal activation is put at a disadvantage by the normalization although it can be assigned a bigger speed value. A corrective action is the normalization of the Gaussian-like weighted input with the maximum of the frame or individual patch areas. This enhances the broad edges relative to the thin ones and final maximal posterior values after some integrating steps converge. Using the thresholding for maximal posteriors this adaption could also bias the final selection towards preferred velocities.

For a final comparison of positional peaks we use the influence of distance weighting without knowledge about the affiliation of neighboring active pixels to the same stimulus. in order to select according to the size of stimulus we could additionally weight considering neighboring posterior signals realized by simple neural connections. This way the object size could be considered as another cue for the final decision with a certain but still constrained capability for feature detection. This remains theory so far.

The current implementation of a motor output policy generates the motor command as follows:

\[
\vec{t}_{\text{target}} = \vec{t}_{\text{retinal}} + \Delta \vec{t}_{\text{dynamic}} \\
\vec{t}_{\text{retinal}} = \max\left(p_i, \text{estimate} + \alpha g_i, \text{cost}\right) \\
\Delta \vec{t}_{\text{dynamic}} = \vec{v}(\vec{t}_{\text{retinal}})\tau
\]

where \(\vec{t}\) is a position vector, \(p\) is the posterior probability output of the state estimation, \(g\) is the gain from the utility function, \(\vec{v}\) the target velocity vector (direction + speed), \(\tau\) is a motion delay constant and the variable \(\alpha\) an additional weighting factor modulating the influence of the distance cost function in the target selection compared to the posterior probability. Note that the equations are predicting the future object position considering the gaze shift delay.

Simply put, after deciding for a state the controller puts out a motor command incorporating a kinematic equation assuming constant object velocity. This also elucidates why the detection of the velocity compared to simple positional activity is desired. The biggest advantage of such a system compared to activity based tracking is a dynamic estimation of future state. This is conform with the human behavior in pursuit unveiled originally by the "step-ramp" test on stimuli motion showing that human pursuit is rather driven by target motion than position [29].
However, the used equation is an approximate and could be extended among other factors with variable motor speed or delay constant $\tau$ in order to reproduce more exact human pursuit behavior. With the final motor output transformation we certainly differ from the theoretical neural framework as the motor-system consists of non-neuromorphic motors instead of muscles. Gain considerations might be similar but transformation to actual commands is simply conventional. A compromise which had to be taken already in the utility function as we lack direct muscle strain information. Nonetheless, now all the necessary information for movement initiation is available. But as noted by the attentive reader the proposed approach so far addresses mostly the first gaze shift after target detection and doesn’t yet cover a follow-up smooth pursuit control phase. Necessary steps for changing from the Pre-pursuit phase to the On-pursuit phase will be addressed now. But an underlying implementation is pending. Still, a neurally-inspired control approach trying to reproduce saccadic eye movement on a robotic platform was successfully implemented.
Chapter 4

Discussion

4.1 Ideas on smooth pursuit and potential framework extensions

It is possible to close the control loop with a repetitive sequence of motion detection and target selection initiating a catch-up saccade in a stop-and-go like fashion. For human pursuit this would be a very ineffective pursuit method especially when considering the long timing delays within the visuo-motor pathway. This latency is critical as any sequential reflex or action is prolonged as well. The presented computational model is certainly faster in processing but has to be peculiarly fast and flexible in order to emulate smooth pursuit motion. Thus, it is recommendable to involve additional methods and ideas.

The essential component for maintaining the pursuit motion and also for reacting fast to target motion perturbation is the so called efference copy which can be understood as a feedback signal forming an extra inner loop with a smaller synaptic length compared to the sensor-actor control loop. Such a signal is necessary because -for perfect target pursuit- visual input from the target is absent or at least strongly degraded. The existence of such a signal is not debated but the implementation is not yet explained. Some researchers interpret this signal as a copy of the motor command. Others postulate it "allows the brain to remember the location of future targets in a spatial coordinate frame in the face of ongoing saccades" (Mays, 1980) [19]. Summarizing, its main purpose is to keep responsiveness high for any new visual target inputs while creating a representation of target motion that is present throughout initiation and steady-state intervals independent of retinal motion activity.

According to this we try to formulate a theoretical approach how such a feedback could be incorporated in our framework in order to realize smooth pursuit. For this 3 questions have to be cleared. (1) How is noisy visual motion activity of the static background suppressed while target stimuli detection capability is unchanged? (2) How is active pursuit maintained without visual feedback on the target (3) What is
necessary to rapidly react to target motion perturbations?

First we look at two different ideas: actively detecting background motion or simply suppressing responsive neural activity determining the relevant layers using extra-retinal signals like eye-motion or a copy of the motor command encoding the pursuit velocity. Pack et al. [23] reason that in the human brain 3 channels for driving pursuit are present. This is the combination of target motion, background motion, and efference copy in a mutually supportive manner. For the different kinds of motion signals the ventral and dorsal MST area respectively are hypothesized as cortical substrates. Naturally concurrent background and target motion have opposing direction and thus connectivity of corresponding detection layers is needed to realize the supportive influence (compare figure 4.1). According to this pursuit is maintained visually detecting gaze motion without extra-retinal signal and when no such activity is present, due to an homogeneous background, the efference signal steps in as a substitute. Including other sensory input like vestibular sensation (for the robot in the form of IMU data) could make this approach even more robust.

Realizing this in an implementation extra background motion detection layers would have to be created with an overall topology similar to the target motion detection layers but now in order to detect unevenly distributed background motion we recommend the accumulation of patch-wise detected moving edges. Thus, suppressive activity from same velocity input of multiple objects is avoided which happens in the previous composition. Adding the proposed excitatory interlayer connectivity will then help to maintain the pursuit even in the absence of target motion activity. Note that the main influencing factor should still be the target motion signals in order to trigger termination of pursuit. Optionally, realizing the support from extra-retinal signals encoding inertial movement should also pose no problem. This theoretically answers the second question.

Now there is an extra layer sensitive to background motion but at the same time this form of retinal activity results in a disturbance of the target detection. Therefore, corresponding ”noisy” signals have to be suppressed somehow. The brain actively suppresses visual processing during saccades to reduce perception of motion [39]. For our framework one potential approach is simply reusing the background movement detection with inhibitory connectivity to target motion layers encoding the same speed and direction. Additionally switching from PRE- to ON-pursuit might result in a general suppression of the peripheral visual field. The purpose of the initiating saccade is to shift the target stimuli on the foveal area. Thus the signal-to-noise ratio for this area is naturally the biggest and simply neglecting or just degrading peripheral input increases the detection quality. However, a good compromise has to be found in enhancing signals from the focal area while still being attentive to new stimuli. We conclude both initial principles are necessary: active detection and suppression of background motion.

The last question to clarify addresses the generation of modulating signals in order to avoid the target to drift off relative to the point of fixation. On the one hand we
Figure 4.1: A leftward eye movement channel with driving signals. All connections are excitatory. The retinal image is processed by two types of cells in MT. MT cells with inhibitory surrounds (MT - ) connect to MSTv cells, with MT cells preferring greater speeds weighted more heavily. MT cells with excitatory surrounds (MT + ) connect to MSTd cells. MSTv cells have excitatory connections with MSTd cells preferring opposite directions. MSTv cells drive pursuit eye movements in their preferred direction, and the resulting eye velocity is fed back to MSTv and MSTd cells (thick arrows). Leftward eye rotation causes rightward retinal motion of the background. The MT and MST cells are drawn so as to approximate their relative receptive field sizes. Image adopted from Pack et al. [23].
conjecture that the existing estimation network could be reused. The retinal activity corresponds to deviation of continuous gaze movement from target motion. But now instead of detecting the absolute speed the relative velocity between target and eye motion is encoded. If the estimation is performed fast enough this might be a valid method. Again, concentrating on the foveal area avoids conflict with background motion to some degree. Note that this approach only makes sense for abrupt changes in velocity which then is constant for the integration to work properly. Another or additionally supportive way which also allows for constant acceleration of the target could be simply checking for retinal activity relative to the center of fixation. Thus, the pursuit gain can be modulated according to the deviation of the moving stimuli from the center of the visual field. This is valid at least for homogeneous stimuli which produce retinal activity mainly at the object boundaries. In both methods an error signal is generated to drive the pursuit correction. In general the network adaptions to all three questions have to perform concurrently without inhibiting each other.

Summing up, for the PRE and ON-pursuit phases distinct processes are necessary which is propped by the cortical differences we mentioned before. That is why we postulate that the efference copy can also be understood as kind of a switch between the corresponding processes or models in order to handle the visual signals differently and activate manipulative top-down signals performing this change.

The steps treated until now referred mainly to signals regarding the existing visual processing and motor output control. The perceptual component of pursuit is so far neglected. Nevertheless, cognition is an important factor in pursuit especially considering how the human influences the choice of target, or termination and initiation of pursuit. Another non-retinal cue could be the comparison of the visual target with a representation of the target in extrapersonal space coordinates which allow us to coordinate movements just based on mnemonic or cognitive feedback. Furthermore, multi-modal cues could have an influence on target selection as multi-modal integration of for example auditory with visual input might be more pervasive compared to unimodal stimuli. Also the perception of features is an important factor. Throughout the thesis we made several remarks about the potential gain which could be made from feature-based instead of pixel-based tracking. Especially color and form features seem essential for target localization and guided visual attention. Moreover, the aperture problem could be avoided and pre-target selection might be improved based on continuous feature evaluation of the objects in the visual field. Finally, without learning and adaption the system is "dumb" and prone to repeat misbehavior. The switching between internal models as a form of conditioning of the cortical circuits on different tasks could be a first step in the direction of a more intelligent system. However, as long as those factors are not considered in a pursuit model no actual human-like and smooth pursuit will be possible. Although this thesis cannot provide a direct implementation for "smooth" pursuit we wanted to address it for comprehensiveness and as a support to future work on the framework or smooth pursuit by itself.
4.2 On the network performance - From serial to parallel processing

The implementation so far uses only serial computation and corresponding tests are executed on a *Ubuntu 16.04 LTS 64bit* Linux distribution and an *AMD Phenom II X4 945* quad-core processor. Additionally, the computer comprises a *GeForce GTX 650* graphics processor. For the potential use in parallel processing we evaluate the computational complexity of the current network and compare the performance when executed with CPU and GPU.

Looking at first at the serial processor, a profiling of the execution time identifies the recurrent calculations for the state estimation as the most time intensive part. This is due to the full matrix multiplications used in order to calculate the prior for the state estimation (compare chapter 3.2.2). The currently employed visual sensor produces events based on a 128x128 pixel matrix. The sensor layers are topographically equivalent and have the same number of neurons per direction and velocity sensitivity. Thus one layer over two anti-parallel directions needs $256 \times 256^2$ multiplications which results in roughly 1.4 ms execution time per layer depending on the CPU-load. On the other hand when executed on the graphical processor the same calculation takes about 0.6 ms. As expected the matrix operations are much faster on the graphical unit.

The execution time scales up directly proportional when adding more layers encoding additional direction sensitivity. But how does the resolution of the input image effect the timing profile? In figure 4.2 the time is shown which is necessary for the forward and recurrent calculations regarding one integration step over 10 layers. The data is averaged over 5 trials. Beyond an input matrix of 75 pixel the serial processor takes increasingly longer than the GPU. We identify an about quadratic dependency between execution time and matrix dimension for both processing units where the leading coefficient of the curve-fit for the GPU-data is smaller. Thus, it certainly depends on the input resolution which processing unit should be used.

Despite that, the overall execution time of one controller step which results in a motor command is about the same when using an input of the dimensions of the DVS-sensor. This can mainly be attributed to the high cost when transferring and retrieving the data to/from the GPU which is not reflected by the figure. Hence, for the current state of development it does not matter on which processing unit the state estimation is conducted. Yet, the silicon-retinas are further developed, the image resolution will increase more, and eventually parallel processing on the framed input is the better option, also because of the design of the estimator which was designed for parallel processing in a cortical-like network in the first place.

However, upscaling the input dimensions or the number of detection layers of the network will augment the necessary number of neurons significantly as the current static implementation is not economical. The future research on this kind of networks should therefore also consider a more plastic design approach using
Figure 4.2: Average CPU and GPU execution time over different input resolutions. Tracked is the time needed for forward and recurrent calculations regarding one integration step over an exemplary network encoding ten different velocities. The upper graph shows the direct comparison where the computation time of the CPU exceeds the respective on of the GPU at a dimension of $n = 75$ of the input matrix (dotted line). Also depicted is the linear curve-fit on the data after taking the square root. This shows the about quadratic dependency of execution time and input image dimensions due to a corresponding increase in matrix multiplications.

for example learning or adaption methods. This on the one hand may enable to create close to optimal weighting beyond the capability of the designer who is choosing the best weight with respect to his prior knowledge. Moreover, the exploitation of the used resources may be optimized as learning could avoid redundancy or elicit implementation flaws.
4.3 Neural principles - Using logarithmic signals and learning

One of the main advantages seen in the implementation of algorithms in neural networks is their aptitude as universal function approximators based on relatively simple physiological processes. Next to excitatory and inhibitory connections performing addition or subtraction neurons have inherent non-linearities enabling them to perform more complex functions. Furthermore, their architecture allows for parallel processing and this often makes them especially suitable for the implementation of some computationally more expensive operations like convolution. This chapter addresses why neural models benefit from a logarithmic interpretation of signals and in how far neural learning principles are present in the current framework.

Log domain In this work the neural firing rates reflect logarithmic probabilities. This signal encoding model is chosen due to several benefits that can be drawn from it [27]:

- The Hidden Markov model used to realize a bayesian estimator mainly employs multiplicative combination of probabilistic signals. A common problem that rises when implementing multiplicative probabilistic models is the risk of "underflow". This means a value becomes too small for the used machine or programming language to be represented. This problem can be bypassed by applying the log. Hereby very small values become representable with non-positive values with magnitudes bigger than 1. In a similar way this transformation allows neurons - which also have a limited dynamic range - to represent probabilities with reasonable levels of activity.

- As can be seen by the model derivation in chapter 2.3.2 the log-domain transformation is the main reason why it is possible to implement the Markov Model with a network of leaky-integration neurons. Furthermore, it enables us to realize multiplication and division in the from of excitation and inhibition which is easily performed by neural connections. Especially the divisive normalization steps which are necessary several times throughout the state estimation benefit from this.

- Finally there seems to be growing evidence that logarithmic likelihood ratios can be found in different areas or task of the brain like decision making.

On the contrary, the implementation of some basic operations like addition, subtraction or logical OR is more complex in the log-domain. Rajesh [27] states that in this case the brain can resort to approximations. We propose also a potential alternation between domains within cortical circuits depending on their suitability for the specific tasks they are performing.
**Learning and Adaption**  During the evaluation of the network we already made assumptions of how the synaptic strength between neurons i.e. the weights could be adapted in order to express different preferences. Also the recurrent weights in general are based on long-term prior knowledge of expected inputs. Similarly, we said a smart system could be realized with some "hard coded rules" which lead to switching between models/networks. What’s resounding is that we are missing the learning ability of cortical networks. The brain has not the luxury of extending its mass over and over again in order to realize more and more networks which are fitting different tasks or preferences. It mostly reuses existing substrates and adapts them in a mostly optimal way using learning mechanisms.

In order to do so within our framework (HMM) a neural adoption of optimization techniques like Baum-Welch algorithm [25] might be the way to go. It estimates the local optimal parameters by coupling the Forward-Backward algorithm with the EM (expectation-maximization) method [26]. Another approach could be the identification of how more conventional neural learning methods like Hebbian learning might be applied to the current network. However, this is not a trivial task and to our knowledge no direct solution for both approaches exists so far.

For a more detailed information on related work and further thoughts on a generalization of the used model we recommend the reader to take a look into the work of Rajesh and al. [27][31].

The design of the network should allow an implementation of the controller not only on parallel processing but also on neuromorphic hardware. The network topology is treated extensively and should be easily realized although the number of necessary neurons might exceed the capabilities of current processors. Depending on the hardware, spike based or firing-rate based encoding models have to be used. Some additional work is necessary to evaluate single event based processing with the current network. So far, whole frames are processed and full matrix calculations are employed. Nevertheless, all those operations could be replaced by neurally feasible operations like convolution, addition and subtraction showing the potential for a realization on a neuromorphic processor.
Chapter 5

Conclusion

This thesis approaches the realization of a dynamic human-like motion detection network for sensorimotor control based on Bayesian statistics. Beyond the concept of using HMM in a cortical-like network we identified the necessary parameters for an estimation of the state of moving stimuli with different dynamics and embedded the network in a robotic controller model which fits the basic principles and structure of sensorimotor control in the brain. The final implementation is fast despite mostly using serial computation during the development. It is robust against noise and has potential for a realization on parallel or neuromorphic hardware. The network is capable of detecting multiple objects within the visual field, reacts to perturbation during a continuous motion and has potential to be used for smooth pursuit where both the sensor and the stimuli are moving.

Although not covered in this thesis, a functional real-time tracking application using a robotic head was set up in order to investigate the further optimization of the system. The controller so far correctly detects motion and executes a saccadic motion towards the stimuli. The presented work supports research on Bayesian sensorimotor processing in the brain by providing a thorough model evaluation beyond experiments confined to artificial noise evaluation and mere computational simulations. First ideas were discussed how to extend or adapt the controller for more complex tasks like smooth pursuit. Additionally a more human-like behavior could be realized regarding more idiosyncrasies of eye and head motion like the vestibular ocular reflex or considering muscle-like actuators.

Future work on this framework might also include the direct implementation of the controller on neuromorphic processors. Another interesting option for future research is a look into network learning methods in order to find the optimal parameter settings or intelligently adjust the network depending on the context. In general, there is a lot of movement in this field and we will keep our eyes open looking forward to see new milestones made in neuroscientific system theory.
### List of Figures

1.1 Basic model for human sensorimotor control ............................................. 6
2.1 Bayes rule illustrated with Gaussian functions ........................................ 8
2.2 Hidden Markov Model Graph ................................................................. 9
2.3 Schematic of brain substrates for ocular pursuit ..................................... 14
2.4 Controller model ....................................................................................... 15
3.1 Robotic head ............................................................................................ 20
3.2 ROS communications concept ................................................................. 21
3.3 Framing-method of DVS-input .................................................................. 22
3.4 Network layer example with feedforward and recurrent connections ........ 23
3.5 Gaussian-filter base .................................................................................. 25
3.6 Noise filter output ..................................................................................... 26
3.7 Custom centroid filter ................................................................................ 27
3.8 Centroid filter method comparison ........................................................... 28
3.9 Recurrent weights matrix .......................................................................... 30
3.10 Firing rate comparison ............................................................................ 34
3.11 Multiple velocities .................................................................................... 35
3.12 Tests on integration parameters ............................................................... 38
3.13 Network reaction to changing object speed ............................................. 39
3.14 Tests on input width ................................................................................ 40
3.15 Test on integration over frames for broad input ...................................... 41
3.16 Tests with multiple objects ....................................................................... 43
3.17 Tests with noise ....................................................................................... 44
3.18 Ground-truth test on recorded DVS-input ................................................. 47
4.1 Schematic of neural model for signals driving steady-state eye pursuit ...... 55
4.2 Comparison of performance on CPU and GPU ......................................... 58
Bibliography


[2] robotic operating system ROS.


License

This work is licensed under the Creative Commons Attribution 3.0 Germany License. To view a copy of this license, visit http://creativecommons.org or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.