Active Binocular Vision Stabilization

with a Cerebellum Inspired Model

K.N. McGuire

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Active Binocular Vision Stabilization

with a Cerebellum Inspired Model

by

K.N. McGuire

in partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Delft University of Technology,
to be defended publicly on 12th of December, 2014 at 2:30 p.m.

Chair:	Prof. dr. ir. P.P. Jonker	BioMechanical Engineering, 3ME, TU Delft
Thesis committee:	Dr. B. Lenseigne	BioMechanical Engineering, 3ME, TU Delft
	Dr. ir. A. C. Schouten	BioMechanical Engineering, 3ME, TU Delft
	Dr. G. Lopes	Systems and Control, 3ME, TU Delft

TU Delft Delft University of Technology

Preface

This report is the master thesis made for the conclusion of the degree of Master of Science in Mechanical Engineering at the Delft University of Technology. The process which has resulted in this report, has started more than 1.5 years ago, when prof. Pieter Jonker asked me to study the human cerebellum. This was to gain inspiration for a new robust control algorithm for humanoid robotics. For my literature survey, I started my journey at the Erasmus MC, at the Department of Neuroscience. Here, with the help of Opher Donchin and Jos van der Geest, I have emerged myself into the physiology of the cerebellum and what its functionality is. Although I was very interested into going more into depth within the neuroscience side of it all, I quickly realized that the gap between what I was doing at the time and implementation into robotics was getting too wide. I therefore choose not to go in too much detail of the workings of the cerebellum's cells structure, but more into its functionality within the brain. For my literature study, I then came to the conclusion, that the gap between neuroscience and the robotics field was even more extensive than first anticipated. A need exists to pull those two together, as both fields can really benefit from each others work. Robotics engineers will receive inspiration for new robust methods for their implementations and neuroscientist can test their ideas on robots as validation of their research. It is hoped that this will eventually be the standard within their collaboration.

After a 6 months internship in Japan, and finishing my literature survey, I was officially able to start my research the beginning of May 2014. PhD student Xin Wang was working on a robot-head with movable cameras, which was able to imitate eye movements. Since the neuroscience department is very focused on clinical research of the cerebellum by investigating eye movements, it was a logical choice for me to start working with this system. However, the plan was quite ambitious from the start. I wanted to use the physiology and functionality of the cerebellum, as inspiration to build a learning cerebellar model, which learns based on the motion it detected. Neuroscience, machine learning and computer vision are three separate research fields by themselves. Fortunately, I had experience with all of them during the curriculum of my Masters'. However, combining them within a robotic setup was something I did not have experience with. During my research, I had to learn about the Robotic Operating System, and go through the differences between simulated and real life robots. I also had to start out first with building a simulated robot, as the real binocular vision system was not available to me at at the beginning. Usually, one first builds a simulation and then builds the robot, but since I did it the other way around, it has given me a lot of knowledge about the dynamics of its movements and the control of the real system. Although I would have liked to have worked with the robot-head from the start, I do not see it as wasted time at all.

At the end of my Master thesis, I was able to achieve most of my previously set objectives. More than three months before completion I was able to make a schedule and stick with it for the most part. I therefore was able to complete most of my experiments on the robot-head with the implemented cerebellar models. I did achieve some good results, but the workings of the models were not completely as I hypothesized. However, I do appreciate the valued insights that I got on the functionality of the cerebellum at the end, which is something I hope to see implemented one day in a robotic setup. Throughout my research I had many moments where I had to think and philosophize about the human brain itself and its wonders. It would be a privilege to see, when we will be able to translate part of its functionality within a robot and to be able to really understand the human brain itself. Although I tried to do it myself within this master thesis, there is still a long journey ahead in up-following research and I hope that my perspectives fit within that outcome.

I would like to acknowledge some people who helped me through this process of writing the report. First of all, I would like to thank Dr. Opher Donchin from the Ben-Gurion University of Negev and Dr. Jos van der Geest from Erasmus University of Rotterdam, for helping me to understand the cerebellum's functionality from a Neuroscience point of view. I would also like to thank Dr. Boris Leseigne from the University of Delft and Dr. Cagatay Soyer from the NATO Communications and Information Agency, for giving me valuable feedback throughout my work on my literature survey and this thesis. Of course,

a lot of gratitude is expressed to PhD candidate Xin Wang, who had built the system that I was allowed to use during my research. I would also like to thank my boyfriend, Christian, which has endured and helped me through the emotional roller-coaster ride during my thesis. And last but not least, I want to express many thanks to Prof. Pieter Jonker, as he has started and guided me on this journey on the cerebellum, and also taught me to be more confident in the quality of my own work.

K.N. McGuire
Delft, December 2014

Abstract

The need for vision guided mobile robotics is becoming more apparent, however, it is proven to be troublesome to stabilize its vision. For two legged robots, it is, during motion, even more difficult than for wheeled robots to be able to perceive the world around it. Finding a stabilization method which is able to reduce the motion blur while walking is essential. Inspiration can be found within the human body itself. The cerebellum is a brain area which is highly involved with stabilization in general, and numerous neurological research has been done by studying stabilizing eye movements. The cerebellum is responsible for adaption within the vestibular ocular reflex and the optokintetic reflex, compensatory eye movements which directly act on signals from the vestibular organ and the detection of retinal flow. An active binocular vision based robotic setup was used to implement this cerebellar model. During the previous literature survey, several cerebellar models were found to be suitable for implementation, however, they do not match the recent neurological discoveries. During this research, two kinds of cerebellar models have been implemented, one existing model and one extended to recent neurological research. Both models have been built using a self learning neural network and are evaluated by the detected movement within an image. While experimenting with these models, the robotic head will receive both rotational and translational disturbances to simulate the sensation of ego-motion. The cerebellar model will learn to predict the sensory consequence (the optical flow) of the combined control of the inertia sensor and the visual tracking system and creates a motor input signal which can counter that movement. This results in the robotic head to stabilize its vision better, while it is on the move.

Contents

Nomenclature

List of Symbols

M_{int}	Intrinsic Camera Matrix
M_{ext}	Extrinsic Camera Matrix
u	Horizontal coordinates image[a]
v	Vertical coordinates image
u_e	Horizontal off-center pixel distance
v_e	Vertical off-center pixel distance
(X, Y, Z)	3D coordinates of object in real world
(X', Y', Z')	Relative 3D coordinates of object aligned to image lens/sensor
ω	Current Speed of actuator
ϕ	Current Position of actuator
ω^*	Goal Speed of actuator
ω^{IMU}	Gyroscope rate of IMU
a^{IMU}	Accelerator meter value of IMU
K_p	Proportional Gain
K_i	Integral Gain
K_d	Differential Gain
w_{ij}	Weight value
Δw_{ij}	Weight Update Value
Δw^*_{ij}	Temporary Weight Value
β	Learning Rate
E	Correlation function
e	Error signal
g	Activation Function
$\boldsymbol{u}$	Motor input actuators
$m_u,\ m_v$	Horizontal and vertical Marker position on image
$o_u,\ o_v$	Horizontal and Vertical Optical Flow on image
$\boldsymbol{p}$	Vector value parallel fibers
z	Output cerebellar model
N	Number of tapped delays
f	Focal length Lens
R	Rotation matrix
k	Discrete Time step
e_{motion}	Motion error detected in video
S	Degree of variation in intensity
W	Neighboring pixels window
M	Harris Matrix
$I_u,\ I_v$	Image gradients in horizontal and vertical direction
λ	Eigenvalue
g	Gaussian function
L	Gaussian convolved image
DoG	Difference of Gaussians
σ	Standard deviation
$L_{uu},\ L_{uv},\ L_{vv}$	Image with a convolution of second order derivative of the Gaussian filter
$\Delta u,\ \Delta v$	Optical flow per frame in horizontal and vertical direction
$\boldsymbol{\omega}^*_{pursuit}$	Vector of goal speeds for pursuit systems
$\boldsymbol{\omega}^*_{cereb}$	Vector of goal speeds for predicted optical flow
$\boldsymbol{\omega}^*_{robot}$	Vector of final goal speeds for the robotic setup

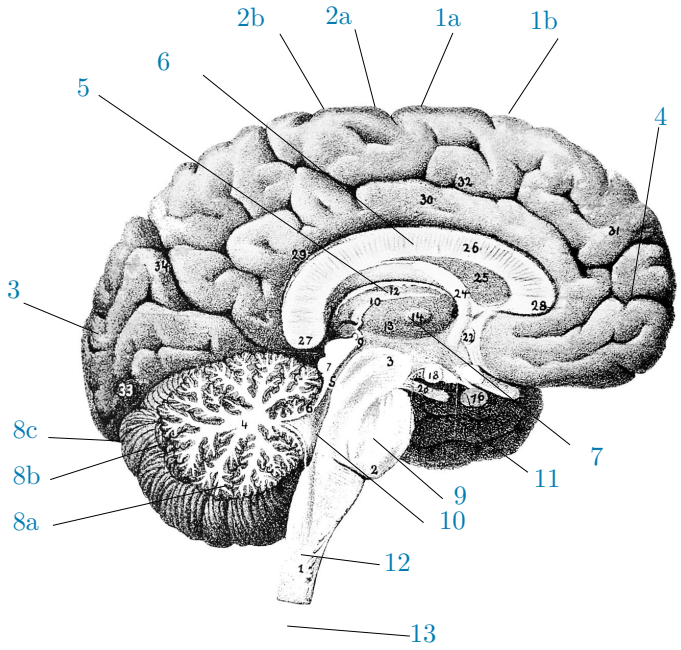
[a] Not to be confused with motor input $\boldsymbol{u}$

ABBREVIATIONS

ANN	Artificial Neural Networks
BP	Back Propagation
CNN	Convolutional Neural networks
DoG	Difference of Gaussians
DT	Dead Time
ER	Encoder Resolution
FOV	Field of View
GR	Gear Box Ratio
MLP	Multi layered perceptron
RBFN	Radial Basis Function Network
RNN	Recurrent Neural Network
RPROP	Resilient Back Propagation
ROS	Robot Operating System
SNN	Spiking Neural Network
TDNN	Time Delay Neural Network
tVOR	Translational Vestibulo Ocular Reflex
tVOR	Rotational Vestibulo Ocular Reflex
VOR	Vestibulo Ocular Reflex
IMU	Inertia Measurement Unit
FB	Färneback method
HS	Horne-Schunck method
LS	Lukas-Kanada method
OKR	Optokinetic Reflex
LTD	Long Term Depression
LTP	Long Term Potentiation
GC	Granule Cells
PC	Purkinje Cells
MF	Mossy Fibers
MSE	Mean Squared Error
PF	Parallel Fibers
DCN	Deep Cerebellar Nuclei
IO	Inferior Olive
CF	Climbing Fiber
CMAC	Cerebellar Model Articulation Controller
PAM	Pneumatic Artificial Muscles
DOG	Degrees of Freedom
FANN	Fast Artificial Neural Networks
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
DBL	Delft Biorobotics Lab

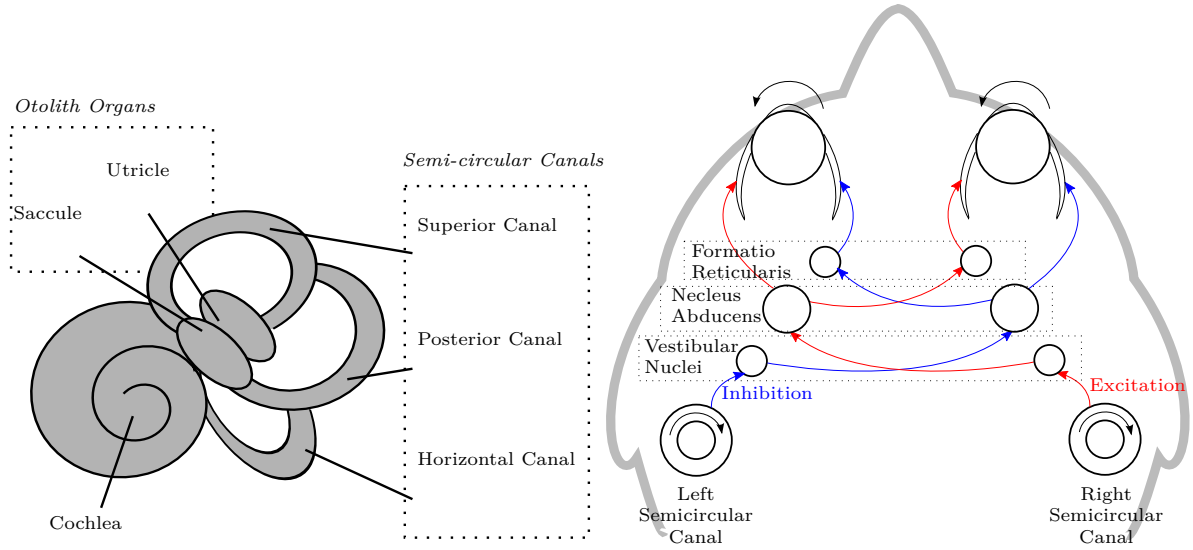
TERMINOLOGY

1. Motor Cortex
   (a) Primary Motor Cortex
   (b) Secondary Motor Cortex
2. Sensory Cortex
   (a) Primary Sensory Cortex
   (b) Secondary Sensory Cortex
3. Visual Cortex
4. Prefrontal cortex
5. Basal Ganglia
6. Limbic System
7. Thalamus
8. Cerebellum
   (a) Vestibular cerebellum
   (b) Spinocerebellum
   (c) Cerebrocerebellum
9. Inferior Olive
10. Vestibular Nucleus
11. Abducens nerve
12. Brain stem
13. Spinal cord



1

The anatomy of the human brain



(a) (b)

(a) The physiology of the vestibular organ and (b) the neurological pathways of the vestibulo ocular reflex.

1

Introduction

Combining both mobility and vision is bound to be troublesome in terms of stability for locomoted robots. Given a humanoid robot with an active vision system, as in Fig. 1.1(a), it will be difficult to stabilize its gaze during walking. If the image stream is corrupted with motion blur or the robot is not able to keep the object within its field of view, it is not able to process it and recognize what the object is. It is necessary for the robots' performance to be mobile and to make use of its cameras at the same time, not having to stand still every time to perceive its environment.

Motion blur can be compensated for by decreasing the shutter time, however this will mean that less light from the scene will reach the image sensor. This means that a trade-off must be made between those two factors: less image blur or the ability to work in low light conditions. Image blur can also be post-processed by deconvolution techniques, resulting again in a sharp image. However, this does not counter the fact that a lot of information is lost from that scene. If the cameras would be actively compensated for external forces, their image stream could be used immediately by the robot, without any post-processing to increase its quality.

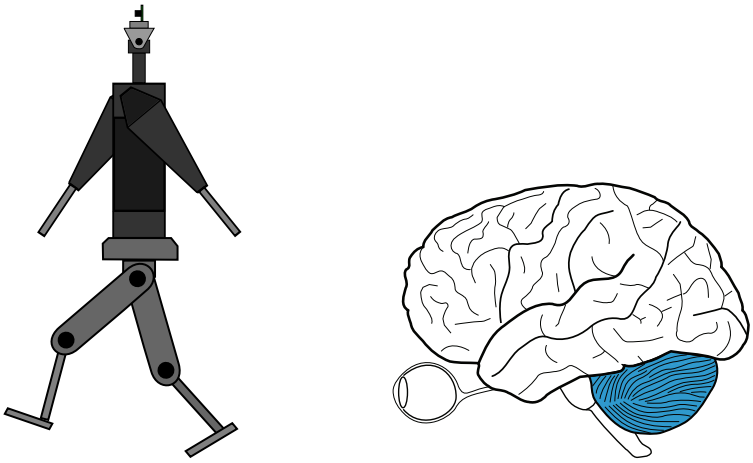
Techniques exist to determine motion using a video stream retrieved from a camera. These algorithms are able to detect the change of pixels from one frame to another and determine the flow of them over time: *The apparent motion of brightness patterns observed when a camera is moving relative to the objects being imaged is called optical flow.* (Horn (1986), page. 278)

Optical flow can be used to determine the speed of other objects, to segment them based on their movements or to ignite a sense of depth perception. This technique is therefore widely used in the automotive industry, as cars encounter other moving vehicles and people.

This master thesis' research is done at the BioMechanical Engineering Department of the Faculty of 3ME at the Delft University of Technology. One of the important research topics of the department, is to find better solutions, for mechanical problems existing today, using inspiration from nature. Humanoid robotics is a branch of biologically inspired robotics, and therefore, to find solutions, one needs to look at the human himself for inspiration. The human body is highly capable to stabilize and even adapt itself to changing environments and surfaces. If a walking robot would encounter a floor structure, which it is not familiar with or not programmed for, chances are high it will fall. The same goes for the stability of its vision system and therefore a stabilization technique needs to be implemented. E.g. based on how the human body solves these kinds of problems.

There are many important processes and mechanisms in the human body that we are not aware of and/or just take for granted. One of them is the ability to stay upright and stable on our feet, handling various types of external disturbance. Within the human brain, there is an area that is usually mentioned in the same sentence as stability and posture maintenances: The cerebellum. It is located just behind the brainstem (Fig.1.1(b)). People with a defect in the cerebellum, do experience serious stability problems. The symptoms include the impairment of locomotion and stance stabilization, limb co-coordination, compensatory eye movements, speech and motor learning.

The cerebellum receives an extensive amount of signals from the human body, which carries sensory feedback of the current state of the body and the external world. This can be deciphered from various senses, as the state of one's body can be monitored by proprioception (sensors in the muscles), but



(a) (b)

Figure 1.1: (a) A walking robot with an active vision system and (b) a representation of the human brain, and the location of the cerebellum in blue.

it can also be seen by the eyes in what configuration the limbs are. The orientation of the body relative to the outside world can be seen visually, but can also be sensed by the vestibular organ. The cerebellum takes into account all these different representation of the same state, and combines them to create a better sense of the current state of the body. As it has a big role in regulating stability and posture maintenance, the cerebellum's effect is directly noticeable in one's eye movements. Learning from experience on previous, similar encounters of tasks, environments and/or tools, it acts like a signal regulator within the brain and is therefore responsible for smooth motion. In order to understand how the cerebellum is capable of these matters, one needs to look into the structure and organization of its nerve cells, their assigned role within the brain and the connections with the other brain areas.

The objective of this research is to implement a model based on the working of the cerebellum in an active binocular vision robot head. This system contains two cameras and is therefore called binocular. It is able to actively focus on an interesting object using actuators, in order to be able to better recognize its 3D environment with its vision. This robotic head can be used for a variety of purposes, wherever autonomous mobility is needed and high speed localization and recognition within its vision are needed. The cerebellar-inspired model will inhibit machine learning, which is driven by the visual detection of self-motion by the cameras. With these aspects, this method will be able to help the robotic setup to better stabilize its vision during external disturbances.

The setup of this chapter is as follows: First some preliminaries will be given about the physiology of the human vision system. The visual processing and gaze shifting will be briefly addressed, to give a general overview for the motivation of human eye movements. Afterwards, the compensatory eye movements will be explained, going into reflexes that act on the sensations of the vestibular organ and the optical flow registered on the retina.

1.1. Preliminaries about the Human Vision System

The human vision system is one of the most sophisticated processes within the human body. Without vision, one becomes totally dependent on the help of others to be able to function within society. Although visual processing is a valuable aspect, the ability to detect and follow interesting objects smoothly can not be overlooked. Unstable eye movements will result in motion blur and therefore decrease the ability to recognize. This section will explain briefly the visual processing within the brain and mostly the nature of human eye movements.

1.1.1. Visual Processing and Gaze Shifting

The eyeball contains a light sensitive layer called the retina, which consists of two types of photo-receptive cells: rods and cones (Fig. 1.2(a)). Rods take up the majority of this layer, as they are the most sensitive to light. However, they are only able to distinguish intensity and not color, as cones do. Cones are mostly centered within the center of the retina, called the fovea. This is where one can see

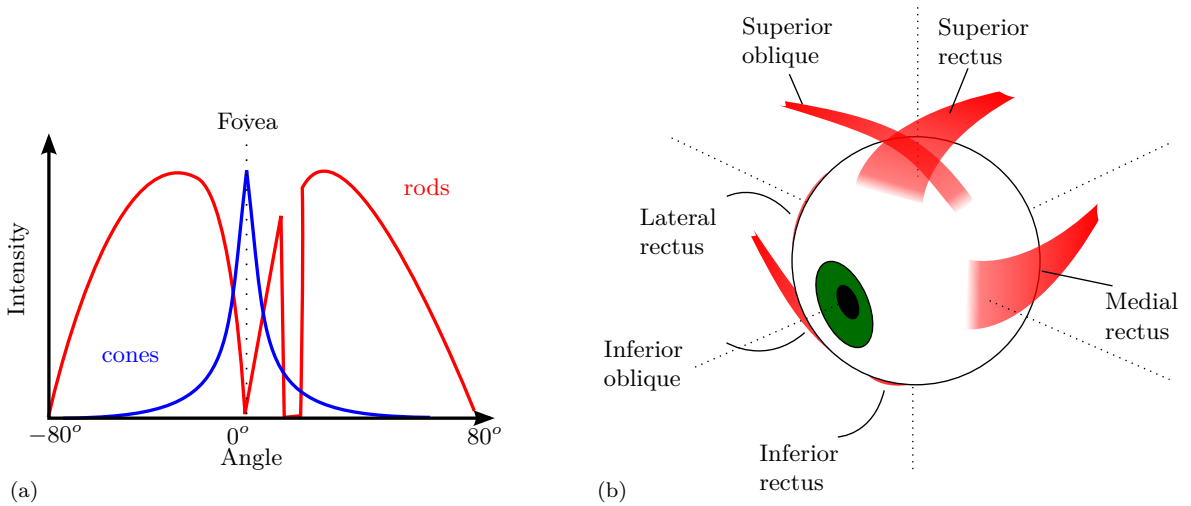


Figure 1.2: (a) The distribution of the photo-receptive cells on the retina: rods and cones. The location of the fovea is where the distribution of cones is mostly centered.(b) A schematic drawing of the eyeball with the extra-ocular muscles: The superior and inferior rectus for tilt movements, the lateral and medial rectus for pan rotations and the superior and the superior and inferior oblique for roll movements.

the sharpest, even though it is just about 0.3 mm in diameter and covers about 2^o degrees of the entire field of view (FOV).

The detected intensity by the rods and cones is transported by the optical nerve to the rest of the brain. Most of these signals are processed by the visual cortex. Exactly how the human brain processes optical information is still not known, but it is eminent that one is not conscious of every single detail within an observed scene. The brain selects interesting features from a scene and connects those seamlessly, in order to deal with the excess of visual information.

The eyes are connected to other parts of the brain as well, where the visual information can be used for more simple, rudimentary tasks. In case the pathways to the visual cortex are interrupted, which is called cortical blindness or blindsight, people afflicted by it can still avoid obstacles or recognize simple shape. However, they are not aware of it and cannot recall seeing or responding to such stimuli.

In order to do the visual processing correctly, one's gaze must be directed toward whatever needs to be processed. Due to the human's foveal vision, eye movements are necessary in order to keep the object of interest within the center of the eye. Rabbits for instance, see evenly sharp over its entire retina and therefore do not have to move their eyes as extensively. The human eye ball is connected with several extra-ocular muscles within the eye socket (Fig. 1.2(b)), which enable the eye to perform pan, tilt and roll rotations.

Two types of gaze shifting movements can be categorized, which are saccades and smooth pursuit (Leigh and Zee). Saccades are quick rapid movements of the eye, usually meant to redirect the fovea's orientation as quick as possible. It is important to be able to scan features of an environment quickly, in order to make a good representation of it within the brain. Reading a book for instance, where the eye makes tiny skips from syllable to syllable. When an object is within the fovea, then smooth pursuit is necessary to keep it there while following it. Humans are able to follow up on to the speed of 30 ^{o}/sec, and it will perform catch up saccades if faster following speeds are necessary. Saccades can be significantly faster at up to 600 ^{o}/sec.

1.1.2. Compensatory Eye Movements

In any mammal's body, mechanisms exist to ensure the stabilization of its head and eyes. A fine example would be the chicken, which is able to keep its head stable in specific position in 3D space. The rest of its body can be moved around in all directions, which can be seen in Fig.1.3. The chicken hardly uses his eye muscles, therefore compensates for external disturbances with its head mostly. Humans stabilize their vision with their neck as well, to some degree, but due to the limits in maneuverability, most of the compensatory movements are done by the eyes. However, why is it necessary for us and the chicken to perform these movements?

Figure 1.3: Footage of the Mercedes-Benz commercial, starring a chicken to emphasize their stable rides (*Daimler AG, 2014*).

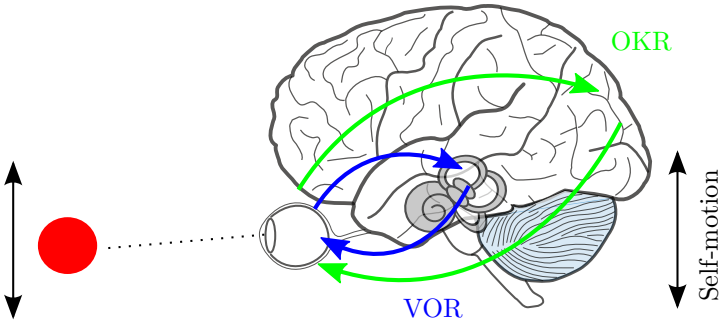


Figure 1.4: (a) The feedback loops of the OKR and VOR during self-motion.

Any kind of disturbance that the body endures is directly translatable to our eyes. If the eyes are not compensated for this by the muscles, the image on the light sensitive layer of the eye, the retina, will be in motion. This phenomenon is called retinal slip and it has an effect on how the world around us is perceived. The photo-receptors, the light sensitive cells, on the retina have a certain sluggishness to them, limiting our vision. This implies that fast moving objects can not be correctly detected and are perceived as a blur. Also, in order to approximate the speed of other moving objects, it would be useful if the whole image is not moving as well.

Compensatory eye movements are necessary in order to keep images stable on the retina, to retain the quality of it for visual processing. The vestibulo ocular reflex (VOR) uses the vestibular organ's signals and the optokinetic reflex (OKR) uses the detected motion of the retinal image as reference to that stabilization. The response of the VOR and OKR are projected to the eyes muscles to compensate for self-motion, and also adjusted by the cerebellum if the signal does not match the outcome of the eye's movement (Fig.1.4). The OKR has a latency of about 75 milliseconds (Schweigart et al. (1997)), therefore able to cope with self motion disturbances of a relatively low frequency. The VOR, however, has a reaction time of about 15 ms and is more suitable for high frequency disturbances. Combining these two types of movement results in the capability to keep the image stable on the eye and prevent retinal slips.

The VOR's neurological connections between the semicircular canals of the vestibular organ are well known and the vestibular information is almost directly linked to the eyes' muscles. If the rotational speed detected in the vestibular organ does not excite the same rational speed in the eyes directly, that signal needs to be modified to match the muscles' dynamics. This adaption happens within in the cerebellum.

There are two kinds of VOR: rotational and translational. The rotational VOR (rVOR) is the most studied from the two and its mechanism is straight forward (Fig. 1.5(a)). The rotation that the vestibular organ registers in its canals has to be exactly replicated by the eyes muscles. Even during

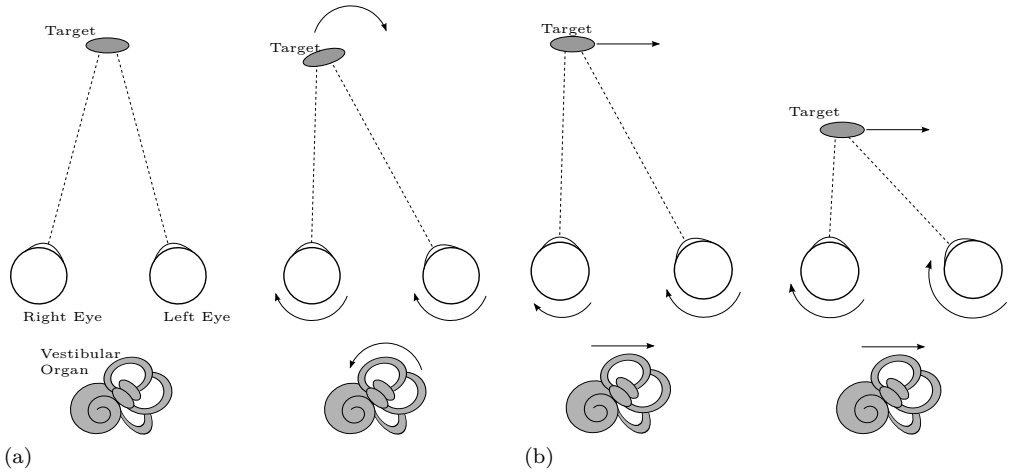


Figure 1.5: (a)The mechanism of the rotational and (b) translational vestibulo ocular reflex.

a pursuit task, the distance between the eyes and the target does not have any consequence on this gain. With translational VOR (tVOR), the distance of the target does have an effect (Fig. 1.5(b)). The closer the object of interest is, the more the eyes have to counter rotate to account for the translation of the head. Although the tVOR is not studied as extensively as the rVOR, it is known that they use the same pathways and are also modulated by the vestibular nuclei. However, the difference is that the gain of the signal must be modified depending on the position of the tracked object (Walker et al. (2010)).

1.2. Objectives

The main objective of this master thesis is formulated as follows: *Implementing an optical flow based stabilization algorithm that handles external disturbances for an active, binocular robot, which is inspired by the functionality of the cerebellum.* To specify this objective, it is divided in four subgoals:

- Building a cerebellar model, which is able to adapt itself by means of supervised learning with optical flow.
- Implementing the cerebellum inspired model and optical flow in a real active binocular robot head.
- Improving the robot-head's vision, by reducing the optical flow and/or motion blur detected, with help of the cerebellar model.
- Comparing two different implementations of the cerebellum inspired model, to be able to draw conclusions on their functionality for the robot.

For the first sub objective, we have to study in several cerebellar learning models currently existing in the field. This model is required to adapt itself by the error of its actions, which is detectable as optical flow by cameras. The combination of machine learning of low level controllers and vision is a challenge by itself. It has become more popular recently, since robots need to adapt themselves more and more to cope with man built environments. However, since this is a new research topic, it has not been extensively studied yet.

The second goal states that the system should work on a real life robot, which imposes another big challenge. Some could argue that in theory the difference between a simulated robot and a real life robot is relatively small. However, many who have experience with this, know that this is not the case. A real life robot has aspects like friction, motor and sensor noise and delays, which can cause the system to show non-linear behavior. With a simulated robot, one can choose to ignore and simplify these matters. Problems like communication delays between the robot and the computer are also an issue. However, on the other hand, with a real robot with cameras, one does not need to simulate exactly what it should see. Capturing a natural, non simulated scene, is necessary for the optical flow generation.

The third sub-objective stands for the effectiveness of the cerebellar model. Until the whole model has been implemented, we will not know if the implementation will be able to sufficiently run on the robot. Many tests of the cerebellar model beforehand, on simulation, and the optical flow detection are needed. Afterwards, the improvement must be validated with various experiments, to see if the performance will hold for different situations.

The last objective implies that two different implementations of the cerebellar model will show a difference in performance of the vision stabilization of the robot. The challenge lies in the fact that a conclusion can be drawn on their performances alone. For this, a estimation can be made of where one would think that one model will perform better than the other.

1.3. The setup of this thesis

This section explains the structure of the chapters within this thesis. It will start with chapter 2 which will explain the physiology of the cerebellum in section 2.1, in terms of its anatomy and its nerve cell structure. Afterwards, the computational models of the cerebellum are mentioned in section 2.2, where the learning within the cerebellum is explained and its role within the scheme of motor control of the human body. Also, previous implementations of the cerebellar-inspired control in a robotic setup are presented, which are evaluated in the preamble of 2.3. The remainder of that section explains the adaptive filter theory of the cerebellum model, as well an adaptation of it, called the information filter and the implications of using these cerebellar learning models on the available equipment for this research.

Chapter 3, presents some technical specifics of an active binocular robotic head, which we used for the implementation of the cerebellar models. Some previous work on similar systems are explained in section 3.1. The details about the hardware, including the calibration information of the cameras, motors and inertia sensor, can be found in section 3.2. The control architecture of the active binocular setup and the visual processing, all actions done mostly from an external computer, are explained in section 3.3. A comparison between the equipment used in this research, and the equipment in a similar experiment is available as well.

Chapter 4, discusses the theory of artificial neural networks, which are the building blocks for the cerebellar learning models. Section 4.1 presents several types of machine learning in general. Section 4.2 explains the different types of artificial neural network models, the basic theory of supervised propagation learning and some alternatives to that. The chapter ends with section 4.3, which presents how the cerebellar model is built up by neural networks and how some choices have been during that process.

In chapter 5, the visual detection of motion in an image stream is researched . The theory about feature recognition will be explained in section 5.1, where both feature detection and description are addressed. Section 5.2 will pick up from there by explaining how optical flow can be approximated, from both dense and sparse features in a scene. Finally, in section 5.3, several optical flow techniques will be compared with each other on the active binocular vision setup itself, determining which one to use for the final implementation.

The experiments and validation the final implementations will be explained in chapter 6, where first section 6.1 will explain the implementation of the cerebellar models and optical flow within the current control architecture of the robotic head. In section 6.2, the experiments conducted to validate the cerebellar model, and the results are presented next to each other. A summary of the performance of the cerebellar models can be found in section 6.3.

Chapter 7 gives an evaluation about research done for this master thesis, where in section 7.1 a discussion is given on the results as well as every step in the process done to acquire those. Section 7.2 provides a conclusion on the degree of fulfillment of the research objectives given in the introduction. From this, some future perspectives and recommendations for up-following research projects of the same topic can be found in section 7.3. problems

2

Cerebellum Inspired Learning and Control

This chapter explains the physiology of the human eye movement and the functionality of the cerebellum within that framework. The cerebellum has an important role in regulating and adapting compensatory eye movements. Researchers therefore believe that the cerebellum contains an internal model containing the dynamics of the human body, and/or every object that needs handling. With this, it fine-tunes and smooths out body movements for specific tasks, based on experience. It is known that stability problems occur when the cerebellum contains a defect. As this has an immediate effect on eye movements, this is an important aspect to investigate, in order to create a bio-inspired binocular stabilization mechanism for a robot.

2.1. Physiology of the Cerebellum

Patients with cerebellar damage show a high deterioration in the performance of direct (on-line) control as well as planning (off-line) of movement. The cerebellum is said to be an integral part of the regulation of posture maintenances and stabilization mechanisms, but what is its functionality within these movements? And how are the anatomy, structure and connections of the cerebellum linked to this specific role? These topics will be discussed in this section, first by explaining the anatomy of the cerebellum and its surrounding areas. The network structure of its nerve cells and its signal processing capabilities are explained afterward.

2.1.1. Anatomy

The cerebellum is located underneath the human brain and behind the brainstem, which can be seen in Fig. 2.1. It actually consist mostly of a long thin plane of tissues, which is called the cerebellar cortex. This layer is folded tightly, so that by appearance, the cerebellum has fine parallel grooves, which are distinguishable from the rest of the brain. The cerebellum can be divided into three sub-anatomical parts. The vestibular cerebellum plays a role in regulating balance and eye movements. It also receives information of the visual cortex. The spinocerebellum plays a role in the body and limb movements. The cerebro-cerebellum is connected with planning and evaluates sensory information for action movements and cognitive functions (Mottolese et al. (2013)).

There are three types of input signals to the cerebellar cortex. The first are the somato-sensory signals, which are redirected from the spinal cord to the cerebellum. Another is the input of the vestibular nuclei, which receives information of the vestibular organ. From the Inferior Olive (IO), it receives input by the climbing fibers into the cerebellum. From the cerebral (neo) cortex, it receives information through the pons with its pontine nuclei. These signals originate from wide spread parts of the neocortex, including the frontal, parietal, temporal and occipital lobes. It was initially thought that the cerebellum is mostly in contact with the primary and secondary motor cortexes (from the frontal lobe), however, since the influence of the cerebellum is noticeable in cognitive processes as well (Ito (2006)), it has been established that it actually receives information from more sections of the neocortex.

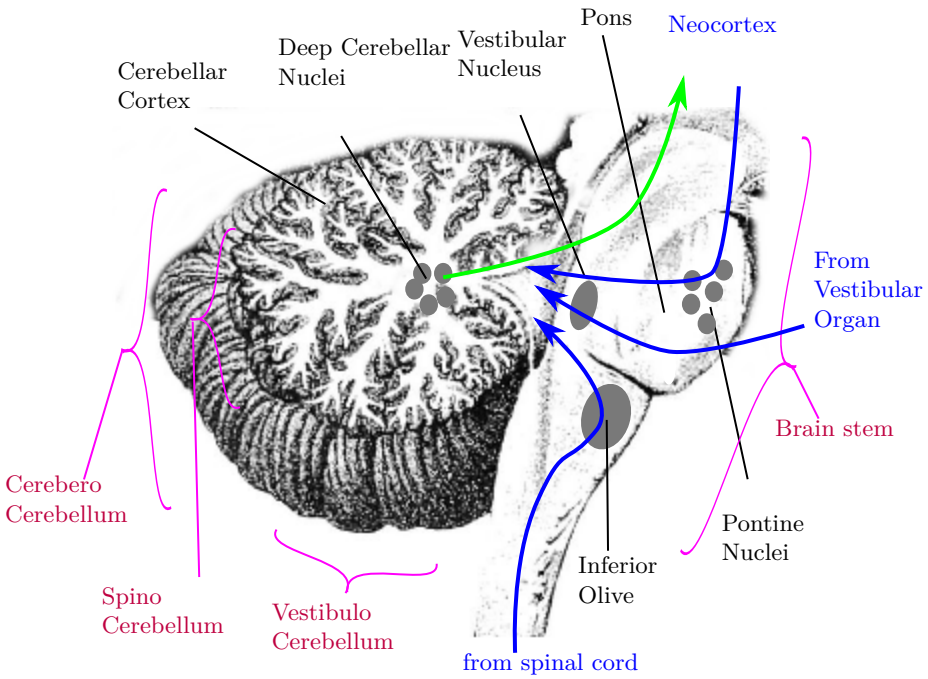


Figure 2.1: The anatomy of the cerebellum and surrounding areas.

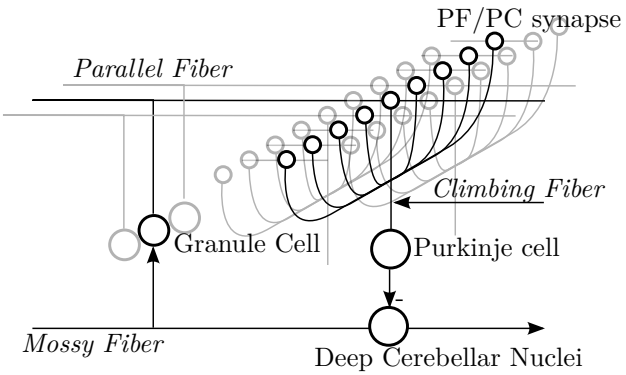


Figure 2.2: The structure of the neuron cells of the cerebellum.

The inputs to the cerebellum are processed by the cerebellar cortex and are than given to the rest of the brain through its deep cerebellar nuclei. These are sent to different sections of the thalamus, which is connected to different parts of the neocortex. Also with the outputs, it was initially thought that it was mostly connected to the primary and secondary motor cortex. By virus transneural tracers, where modified virus strains were used to trace the projections of the cerebellum to the brain. It could be seen that the cerebellum makes connections to the frontal, parietal areas and the basal ganglia as well (Bostan et al. (2013)).

2.1.2. Cell Structure and Signal Processing

If one zooms into the structure of the cerebellum, as can be seen in Fig. 2.2, its composition cannot be compared to the rest of the neurons of the brain (D'Angelo (2010)). The two main neurons that exist in the cerebellum are the granule cells (GC) and the Purkinje cells (PC). GC are the smallest neurons, which make it possible for the cerebellum to contain more than 2/3 of the total number of neurons in the brain (40 billion in total). They receive input from the mossy fibers (MF), which consist of information originating from all sensory organs (Proprioception, vision, vestibular etc.).

The PC are the largest type of neurons within the brain. They contain a large amount of dendrite branches which are connected to many parallel fibers (about 200 000 connections). As they receive a single spike signal from different parallel fibers at their branches, they generate a simple spike signal as

output. This output has an inhibitory effect on the deep cerebellar nuclei (DCN), which also receive an excitatory effect from the mossy fibers (sensory information).

The climbing fibers (CF) originate from the Inferior Olive located in the brain stem, and it is said to be the teaching signal for the plasticity (learning) of the Purkinje cells, carrying the error signal of the predicted state and the actual state Ito (2006). The adaption within the cerebellum is done by changing the sensitivity between the PF/PC synapses, which is possible due to synaptic plasticity. As the signals are being processed forward through the cerebellum, the PC (the post-synapses) will become less sensitive to the signals that come through the PF by means of Long Term Depression (LTD). LTD stands for the reduction in the efficacy of the synapses, which is caused by an external performance measure which is activity depended. Long Term Potentiation (LTP) occurs within the cerebellar cortex as well, which is the opposing mechanism to LTD. Motor learning within the cerebellum must be reversible to prevent saturation.

Eccles et al. (1967) has established early on, that there are four principles identified from the structure of the cerebellum and its signal processing. This principle still hold today:

1. The first is feed-forward processing, as the cerebellum has almost no recurrent connections within its nerve cell system. Although it does have many recurrent communications with the other brain parts, it does not apply for communication within itself. This enables fast signal processing within the brain, which is an asset for smooth motor control.

2. The second is divergence and convergence, as 200 million MF are diverged to 40 billion GC, which then converges to 15 million PC and finally to just 50 DCN. All this processing is done parallel to each other, it also contributes to its high computational speed.

3. Modularity is the third principle, as the cerebellum can be divided in thousands of modules, which all have a exact same nerve cell network structure but receive different input or outputs from the other brain parts.

4. Finally, the last principle is plasticity, were many synapse combinations with the cerebellum can be adapted by strength, which fine-tunes the relation ship between the inputs of the MF to the output of the CF.

The next section will go more into this principle of plasticity, as it discusses several cerebellar learning models.

2.2. Computational Models of the Cerebellum

The cerebellum has a unique and well known structure, therefore giving researchers the chance to develop computational models of its capabilities. First there will be a focus on learning/performance models of the cerebellum itself, where different mechanisms within its signal processing are linked to its learning capabilities. Then, some examples are shown of how the cerebellum is connected to other brain areas in terms of motor control.

2.2.1. Cerebellar learning models

In the course of years, several models have been developed, that simulate the learning process of the cerebellum. In Fig. 2.3(a), a simple representation of this principle can been seen, as formulated by Kawato (2009). In the previous section, it was explained that the cerebellum receives two kinds of input, MF and CF, and gives out its output by the PC. The learning models presented here will look mostly into the input signals, which are used to induce the supervised learning within the cerebellum.

One of the earliest learning models comes from the research done by Marr (1969) and Albus (1971) a few decades ago. They have established that the changes in sensitivities with the PF/PC synapses is due to the correlated firing of the PF and CF. It will occur when both the excitatory inputs of the PC's, from the PF and CF, will fire together within a certain time window. The effect will be the highest if the CF are activated about 50-200 ms after the PF (Ogasawara et al. (2008)). Since the PC output is inhibitory to the DCN, it will mean that the desensitizing of one synaptic output at a PF, will actually increase its influence. Afterwards, the output of the cerebellum will be the weighted combination of its input. Albus (1975) also created the cerebellar model articulation controller (CMAC) based on his cerebellar learning model (see Fig. 2.3(b)). This neural network based controller is considered the basis of reinforcement learning, which currently has its uprise within the machine learning community.

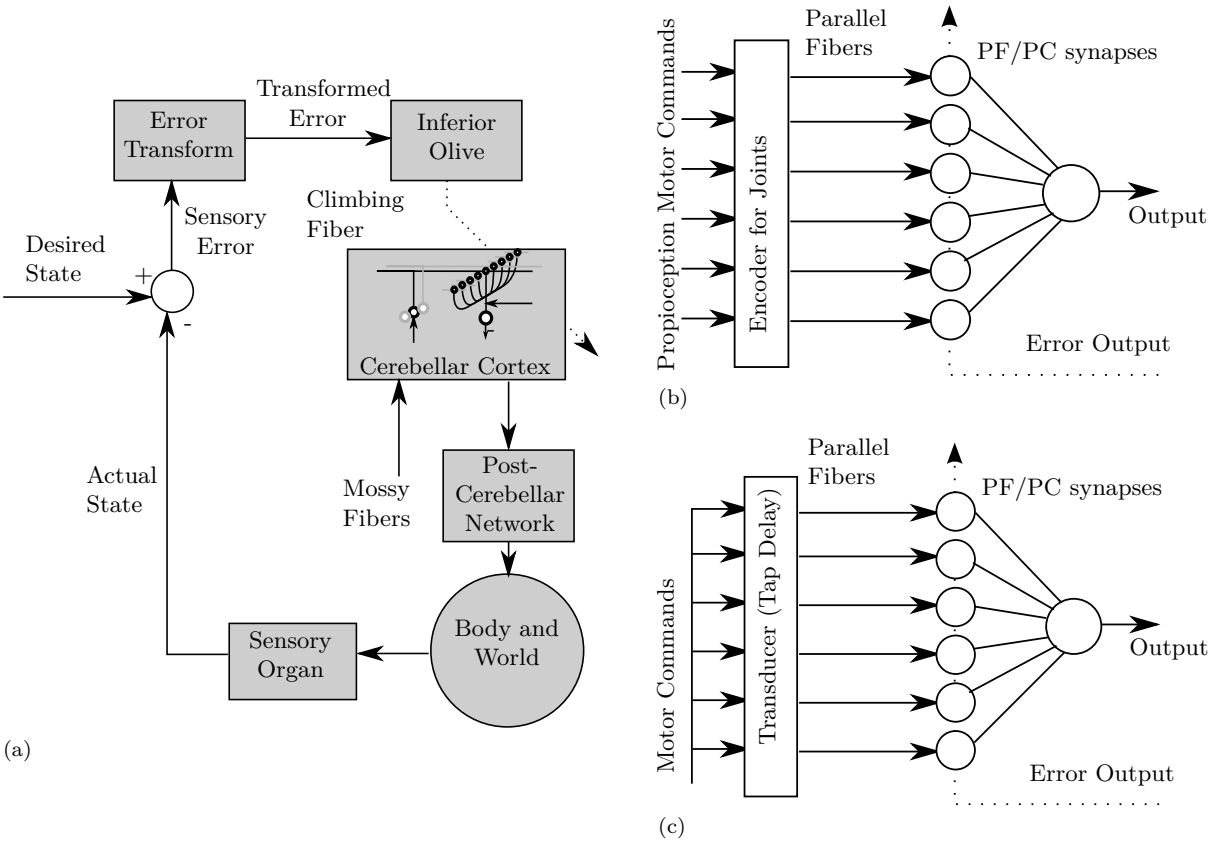


Figure 2.3: (a) The simple representation of the cerebellar learning model, adapted from the paper of Kawato (2009), (b) the CMAC representation of Albus (1975) and (c) the adaptive filter theory according to Fujita (1982). According to Dean and Porrill (2011), the transducer stand for a tapped delay line.

Since the early mentioned time window is necessary for LTD, timing synchronization models from the cerebellum have become more apparent. Many activities require the limbs to be synchronized relative to each other, where locomotion is an important example of this. The gait of walking depends on how the legs are coordinating with each other, and must be adapted to different surfaces in order to sustain stability (Takakusaki and Okumura (2008)). These theories suggest that LTD will occur if there is a synchronization error between these mechanisms, where the CF plays a significant role. A study by (Molinari et al. (2007)) showed that cerebellar patients did have difficulty with a rhythmic task, like finger tapping, apposed to the control group. However, this does not mean that the cerebellum is not necessary the location for this 'time-keeper' functionality, as that some timing information did preserve with cerebellar-damaged patients.

Physiological data shows that the simple spike firing from the PC encodes dynamic features of movement. The internal model theory is therefore a supported functionality of the cerebellum as it fits in the framework of the kinematic representation and supervised learning (Wolpert et al. (1998), Imamizu et al. (2003)). An internal model is a term from field of the optimal control and is used for prediction of a next world state. This internal model could represent the dynamics of the external world or the system/body itself. It is therefore of great importance for the improvement of motor control in terms of execution and planning. This would mean that the CF would carry the difference of what the internal model represents, and what the actual real life interaction is.

In terms of eye movements, the eyes are the most important performance feedback for the supervised learning. Instead of subtracting an actual state with a desired state, the retinal slip detected can be used as motive for the adaption of compensatory eye movements. Graf et al. (1988) had proposed that the climbing fiber carry information about retinal slip to the PC's. However, retinal slip as a performance measure for the eye movements are not necessarily the same. Frens et al. (2001) have done a study with rabbits, where several visual flow stimuli were presented, so that the eyes can not compensate for all of them. They found that the climbing fiber complex spikes do not change for different optical stimuli,

with identical eye movements. They therefore concluded that the CF are encoding the performance of the eye movement itself rather than retinal slip.

As the cerebellar signal processing is time depended, and its climbing fiber error imposes dynamical systems, there have been theories about the GC function as well. Fujita (1982) has developed the adaptive filter theory of the cerebellum, implying that the GC layer acts out as a transducer (Fig. 2.3(b)). This means that the signal entering the cerebellum will be transformed to fit the task's dynamic requirements. Dean and Porrill (2011) has implied that this must be a tapped delay line. A tapped delay line means that the input by the MF will be delayed in several time steps. The cerebellum will therefore not only receive the current version of a signal, but its previous versions as well. With this, the cerebellar model is more able to determine the dynamics of the tasks it encounters.

2.2.2. Cerebellum's influence on motor control

The previous subsection discussed mostly the inputs of the cerebellum, and how it manifests those signals to adapt itself. Here, the output from the PC and the effect on the other brain areas will be explained, in the form of motor control schemes.

Different theories about the cerebellum's function in this scheme exist (see Manto (2012) for a review). It has been said that the cerebellum acts like a comparator between different sensor signals or that it is an on-line error corrector, which sends corrective motor signals to the muscle to smooth out the execution movement. It has also been claimed to be a temporal pattern generator which is used by the rest of the brain as a certain internal clock by Jacobson et al. (2008). However, it has been noticed that these claims are less supported in the recent years. A widely supported hypothesis about the role of the cerebellum nowadays, looking at the large amount of available recent papers is that of an internal model (Wolpert et al. (1998), Imamizu et al. (2003)). An internal model is a term from the optimal control terminology and is used either for prediction of a next world state, a forward model, or the production of motor output given a desired state, an inverse model. This internal model could represent the dynamics of the external world or the system/body itself. It is therefore of great importance for the improvement of motor control in terms of execution and planning.

There are researchers who support the claim of the cerebellum being an inverse model, where it generates motor commands with a desired state as input (Taig et al. (2012), Shidara et al. (1993)). This feed forward motor command would be added upon an already generated feed backward motor command. This can be seen in the study done by Kawato and Gomi (1992), as he had developed a cerebellar feedback-error learning model, illustrated Fig. 2.4(a). The cerebellum is presumed to receive a desired state through it GC and gives a feed forward motor command as output. The inverse model is adapted by the feedback controllers' motor command by the CF, therefore updating its model to different situations.

However, recordings of the output of the cerebellum do not show signals similar to motor commands (Horne and Butler (1995)). Miall and Wolpert (1996), Lisberger (2009) and many more, therefore support the theory of the cerebellum being a forward model instead of an inverse model. A example of a forward model representation, is the state-predicting feedback controller (SPFC) from the paper by Frens and Donchin (2009) (Fig. 2.4(b)). This principle is based on the optimal control theory of neuro-control of human movement by Todorov (2004). Here the cerebellar model receives the current motor command, in order to make a prediction about the state a step later. The prediction can be directly used by the feedback controller (like with the (smith) predictor architecture of Miall et al. (1993)), but in the SPFC model a trade-off is made between the reliability of the current sensed state and the predicted one, like in a Kalman filter. This state estimation is then used by the feedback controller within the scheme to create a motor command for the muscles.

These were several suggestions and theories, based on clinical research or in vivo cerebellum recordings of the PC. In the up-following section, some implementations of cerebellar models for the control of robotic systems will be explained.

2.2.3. Previous Cerebellum Inspired Robotic Implementations

Many robotics researchers have noticed the amount of research done on the cerebellum, and have tried to implement its functionality within a robotic system. These cerebellum-inspired models have been used for a variety of tasks. For instance, Eskiizmirliler et al. (2002) have implemented a model of the cerebellum for the movement control of a single joint robot arm by simulating the structure of its connections. McKinstry et al. (2006) created a cerebellar model which was detailed up to the neuron

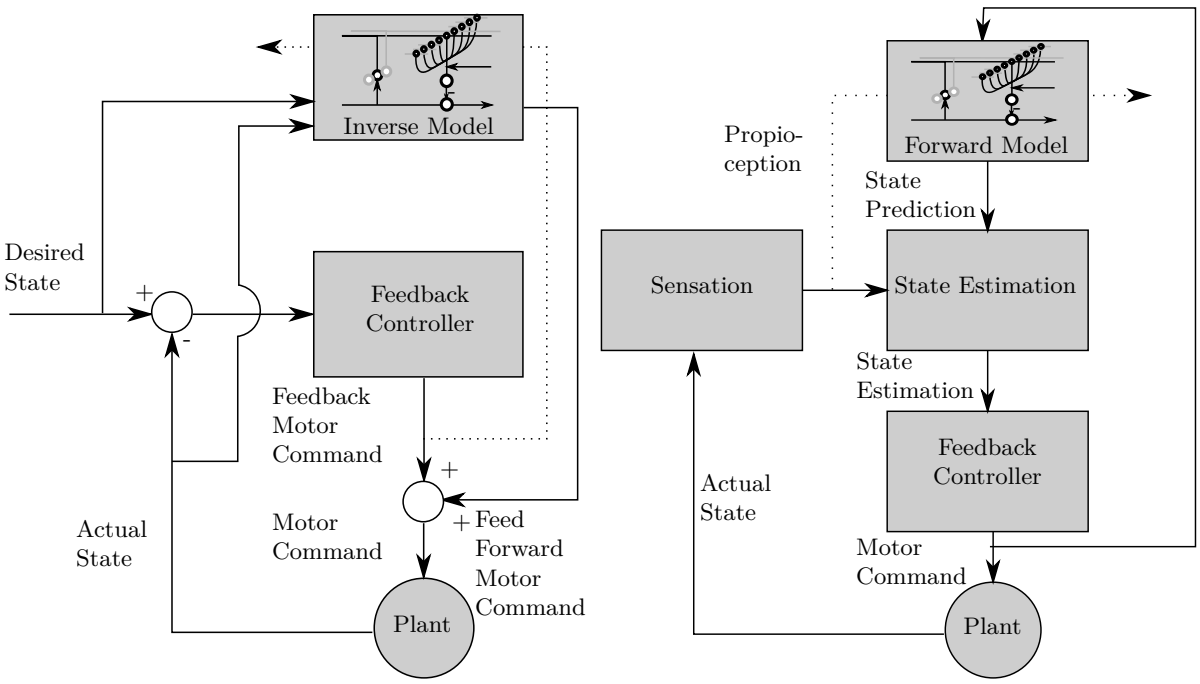


Figure 2.4: (a) The scheme of the cerebellar feedback-error learning model, adapted from the paper of Kawato and Gomi (1992) and (b) the state predicting feedback controller (SPFC) on the right, modified from the paper of Frens and Donchin (2009).

(a) (b) (c)

Figure 2.5: Robotic setups used for the implementation of cerebellum-inspired control are (a) One joint arm robot (Eskiizmirliler et al. (2002)), (b) Segway robot McKinstry et al. (2006) and (c) robotic eyes (Lenz et al. (2009)).

and synapse level and implemented this in a Segway-like robot. With visual cues, it was able to predict the collisions with obstacles and acted accordingly.

Lenz et al. (2009) used an actuated robot eye to test out the VOR with a simulated cerebellum. Carrillo et al. (2008) and Luque et al. (2011) have done this as well by implementing an adaptive cerebellar spiking model into a brain-based control of a 2D actuated arm. This model was able to adapt to different kinematics where corrective actions were required. A similar spiking cerebellar model has been used on a virtual robot to learn a certain behavior, giving it only two wheels, two photo-receptors and a tendency to go towards the light. Finally, Nassour et al. (2013) has used inspiration from the cerebellum for a brain based control algorithm, where it acts as a comparator of sensory output (see Fig. 2.5 for the implementations).

If we want to implement a cerebellar inspired model within a robotic setup for this master thesis, it is wise to look at already existing implementations. These models have already been modified to fit the hardware of their setups, therefore modifications needed will be kept to the minimum. In the preamble of the next section, a trade off is made between these, to be able to choose one as an inspiration for our implementation in a robot-head ourselves.

2.3. Implementation of Adaptive Control with a Cerebellar Model

In the previous section computational models of the cerebellum have been explained. Since the objective of this thesis is to implement the cerebellums' functionality into an active binocular vision system, it is important to see which of these systems have been implemented into robotics before. To be able to choose a cerebellar learning model for implementations, there are some criteria to consider:

- The model can be used in combination with vision and/or applied for simulating eye-movements.
- The model should be able to be implemented on the available hardware for this research.
- The model should have a good foundation from neuro-scientific research.

From the previous cerebellar-inspired robotic implementations, the research of Eskiizmirliler et al. (2002), McKinstry et al. (2006), Lenz et al. (2009) and Iwadate et al. (2014) have been using vision in combination with their cerebellar model. Only the model from the research of Lenz et al. (2009) has been used for simulating eye movements. Carrillo et al. (2008) and Luque et al. (2011) used a spiking cerebellar model to actuate their robot, however I feared that this will be too much of a strain for the moderate strength of the computer used for this research. It is also a matter of complexity, since the same computer will have to run the controller for the system, optical flow detection and the neural network at the same time.

Finally, the model chosen should link with the cerebellar learning model as its role in the control scheme of the robot. In terms of the structure of the model, all the implementations have looked to some degree to the physiology of the cerebellum. However, in terms of their role in the control scheme, the models of Eskiizmirliler et al. (2002), McKinstry et al. (2006) and Iwadate et al. (2014) have implemented their cerebellar-model as it were the only control module within their robot. It can be seen in Fig. 2.4, that it has been established that the cerebellum has a sidekick role next to the main feedback controller within the human body. Luque et al. (2011) and Lenz et al. (2009), have incorporated this principle in the control loop. Nassour et al. (2013) did not use the cerebellar model for direct control as well, however, he uses it just as a non adaptive comparator between signals, which is no longer a much supported role of the cerebellum, looking at the recent available neuroscientific papers (see section 2.2.2).

There is one model which clearly fits each criterium, and that is the cerebellar model of the research of Lenz et al. (2009), namely the adaptive filter. It has been used on active camera system as well, the model's complexity is suitable for implementation into the current available hardware. On top of that, it has a good link with the neuroscientific research on cerebellar learning models, as they were inspired from the work of Fujita (1982). Also, Dean and Porrill (2011) have made a good effort to connect their work with clinical research.

The theory of the adaptive filter will be explained within this section, where we will go into the details and a variation is proposed to pull it more towards the physiology of the cerebellum. Afterwards some implications for the remainder of the master thesis will be formulated, which will explain the necessary steps to implement this cerebellar model into the active binocular vision system.

2.3.1. The Adaptive Filter

It was Fujita (1982), who first simulated the cerebellum as an adaptive filter based on this research. Many versions have emerged from this principle, from which the most famous work comes from Dean and Porrill (2011). The vision setup that Lenz et al. (2009) used, was a monocular robotic eye with actuated by four McKibbens' pneumatic artificial muscles (PAM), where two were used for a pan rotation and two for tilt. They used a custom designed micro controller board for the adaptive filter, which allowed a separate control of each of the Degrees of Freedom (DOF).

The cerebellar model developed by Dean & Porrill can simply be explained as a two layered neural network (Fig. 2.6(a)). The input $u(t)$, which is the motor command to the actuator, comes in and is transported by the parallel fibers ($p_i(t)$) to the PF/PC synapses. According to the model, the parallel fibers consist as a vector of:

$$\mathbf{p}(t) = [u(t-1), ..., u(t-N)]^T \tag{2.1}$$

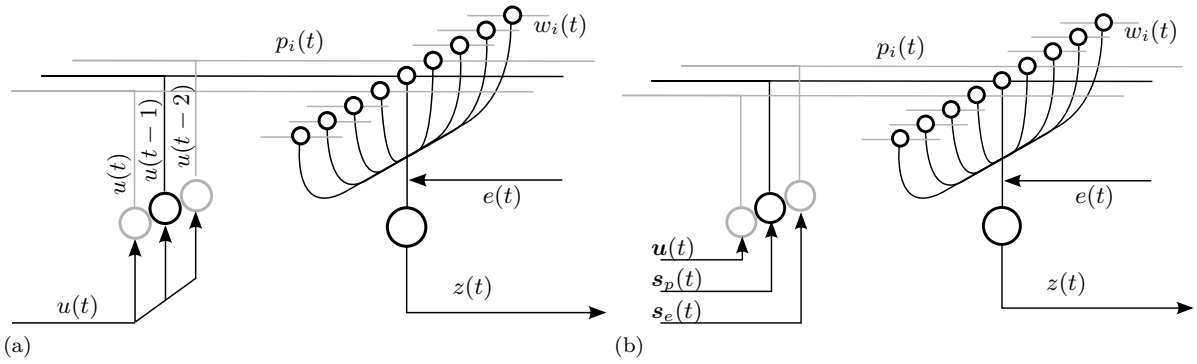


Figure 2.6: (a) The scheme used in the research of Lenz et al. (2009) and (b) the variation on adaptive filter for the compensatory eye movements. s_p and s_e stand for the sensory input of the proprioception and exteroception. u for motor input, e for the error signal which drives the learning in the cerebellum, and z is the output of the cerebellum.

Which means each that both Dean and Porrill (2011) are seeing the granule cell as a tapped delay line, so that old motor inputs can be used for the calculation within the model. With their implementation, they were able to realize a sample frequency of 100 Hz and a tap delay size of N=200. The output signal $z(t)$ is the combination of $p_i(t)$ with the synaptic weight $w_i(t)$:

$$z(t) = p_1(t) \cdot w_1(t) + p_2(t) \cdot w_2(t) \dots p_N(t) \cdot w_N(t) \tag{2.2}$$
$$= u(t-1) \cdot w_1(t) + u(t-2) \cdot w_2(t) \dots u(t-N) \cdot w_N(t) \tag{2.3}$$

Connected to the Purkinje cell dentrices, is the climbing fiber carrying an error signal ($e(t)$). This error signal stands for an entity that needs to be corrected for. In the case of eyes, this indicates retinal slip. The learning rule is formulated as follows:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \gamma E[\mathbf{p}(t)e(t)] \tag{2.4}$$

where $\mathbf{w}$ is a vector of all the weights within the adaptive filter, γ is the learning rate and $E[\mathbf{p}(t)e(t)]$ is the learning rate and $E[\mathbf{p}(t)e(t)]$ stands for the relation between $\mathbf{p}(t)$ and $e(t)$.

A concern with the model of Dean and Porrill (2011) is the notion of the time delay tap. Also, the large number of the tapped delay line seems quite extensive, as one only needs a handful to approximate a dynamic system. Within the neuroscience field, there is little evidence that the cerebellum has a temporary/recurrent mechanism that is able to do this.As a conclusion of section 2.1.2, one of the principles of the cerebellum was its feed forward kind of way of processing. It has recurrent connections to the rest of the brain, but not with itself. Moreover, the fact that the cerebellum exclusively receives motor commands, and only that, is something that neurophysiology data tell otherwise. The reason why the Purkinje cells are so large, is to enable them to reach as many data signals from the human body as possible: state of the joints, the muscles, vestibular organ and so on. This connection of all the signals within body and brain is an asset that will be focused on in my new variation of the adaptive filter. In this thesis, this model will be called the information filter.

In Fig. 2.6(b), this alternative model is represented. As can be seen, instead of the cerebellar model only receiving the motor command that goes to the robot, it will receive all (state) signals of the robot. To keep a distinction, the signals will be denoted as exteroception (signals from outside the robot), and proprioception (signals from inside the robot). Exteroception stands for the input received from the cameras and inertia sensor and proprioception stands for the current states of the actuators and joints, as well of other information resulting from internal processes.

2.3.2. Implications

Both the adaptive and information filter will be implemented into an active binocular vision system. There are some implications to that choice, as the hardware that Lenz et al. (2009) used for their implementations, has many differences with our setup. First of all, all the processing of the cerebellar models and all the higher control functionality, will be done from the computer itself in our research,

since a custom made control chip board is not available. This will result in some modification to the original model, to be able to run it directly from our computer. The next chapter will go more into detail about equipment available for our research and therefore to make a comparison with Lenz et al. (2009)'s equipment.

To be able to implement the adaptive and information filter, there are two subjects that need to be looked into. The cerebellar model is clearly built by means of artificial neural networks, the computational variant of the connections within the brain. Chapter 4 is dedicated to its theory within the supervised learning paradigm, and how to it can be used to build the cerebellar learning models. As the neural network will learn for movement detected from its cameras, we have to look into optical flow detection as well. Chapter 5 treats algorithms for feature detection and optical flow generation and tests these algorithms on our vision setup.

After the building blocks of the cerebellar model are known, we will look how the adaptive and information filter can be integrated into an already existing control architecture of a robotic head. Finally, in order to validate them, several experiments are conducted which ignite optical flow in the image stream from the cameras, which enables the learning of the two. Different disturbances will be performed, were the models will be compared with each other and with a case without their influences. Based on their performances, we can conclude if the objectives given in the introduction are fulfilled.

3

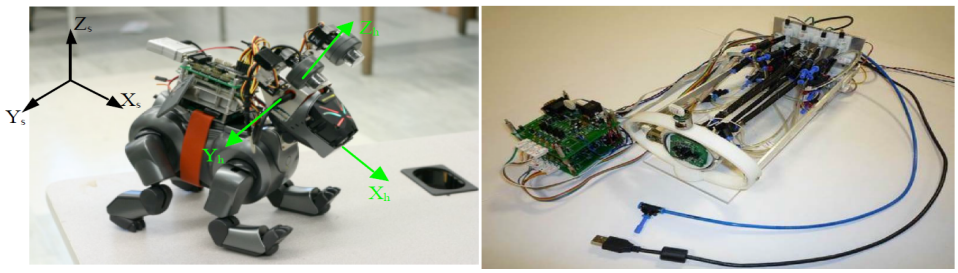
An Active Binocular Vision Setup

An active binocular vision setup is available for this research, with its corresponding control architecture. It was initially used by PhD student Xin Wang for tracking an object and to create a depth map based on convergence of the eyes (Wang et al. (2013)). This chapter explains the equipment and software used for the implementation of the adaptive and information filter. First, some other work on active vision stabilization are given, where comparable robotic platforms systems have been used. Our robot-head is capable of recreating several human eye movements. The second section explains the details of the vision setup and the existing software. Afterwards, a comparison is made of this setup, and the setup used by Lenz et al. (2009)'s work.

3.1. Previous Work on Active Vision Stabilization

The need to actively stabilize a mobile robot's vision has been shared by other parties, who have attempted to mimic eye reflexes for vision guided robots. Kaushik et al. (2007) for instance, had used a quadruple AIBO robot with a mounted active binocular vision system and an inertia measurement unit (IMU) to stabilize the cameras (Fig. 3.1(a)). The IMU's output was coupled with the actuators controlling the orientation of the eyes, compensating for the accelerations and disturbances generated by the robot's locomotion in real-time. The study only shows the compensatory eye movements by the actuators ,however does not show any quantized measurements of the camera's image stabilization. It difficult to establish if their algorithm is truly successful or not.

Ryu et al. (2009) had developed a feature tracker system, which can stabilize the image based on IMU information of the robot's system. The difference with the research of Kaushik et al. (2007), is that their algorithm stabilize the images not by compensatory eye movements, but digitally. However, they are using the actual images as a performance measure of their technique, by comparing the optical flow in the image stream to detect improved stabilization.



(a) AIBO (b) Pneumatically actuated robot

Figure 3.1: The AIBO robot with an mounted active binocular vision system and an IMU, taken from the paper of Kaushik et al. (2007). The camera with pneumatic artificial muscles for implementing a simple VOR, taken from the paper of Lenz et al. (2009)

.

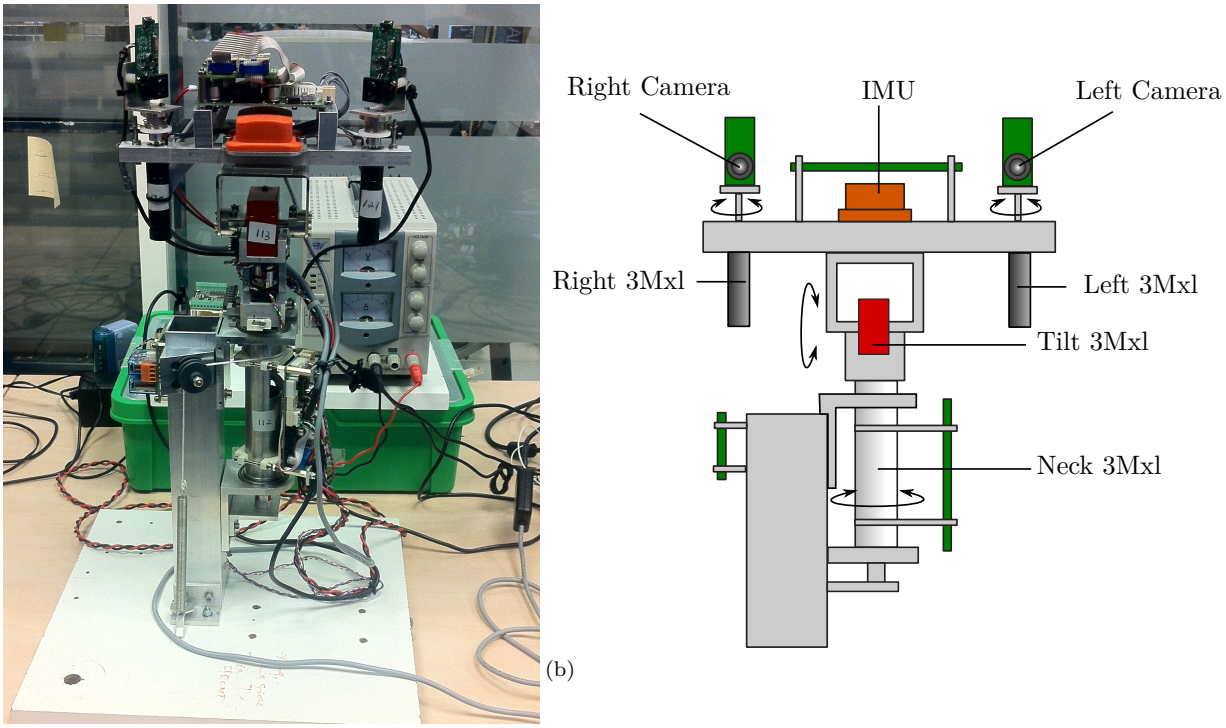


Figure 3.2: (a) A photo of the active binocular vision setup used our research and (b) a representation of the same system, with hardware descriptions.

To implement the opto-kinetic reflex (OKR), Labutov et al. (2008) have done some studies where they combine both the OKR and vestibulo ocular reflex (VOR) into a binocular system. It calculates the signal which drives the OKR based on the movement of the object it is tracking. It combines it with the signal coming from the IMU with sensor integration to generate compensatory eye movement for external disturbances with different frequencies. The study concludes that the VOR and OKR together creates a more robust stabilization than the OKR alone, since the OKR has a slower response due to the processing and VOR has less delay. However, these statements where only backed-up by movement data and not by image stabilization measurements.

As said earlier, the cerebellum is said to be an important regulator of the VOR and OKR. It is also well researched within the neuroscience community since it is the one task that the cerebellum has almost direct influence on. Therefore, Lenz et al. (2009) have implemented a cerebellar inspired model to create compensatory eye movements for a robotic monocular eye system (Fig. 3.1(b)). The camera's are actuated by McKibben's artificial muscles, to make the system's response similar to human eye movements. However, these pneumatic actuators are difficult to implement into a compact system when compared to electrical motors and on-off systems.

3.2. Hardware

This section goes into detail the important elements of the available active binocular robot, which are the motors, the cameras and inertia sensor, which are their description, technical aspects and necessary calibration processes.

3.2.1. Actuators

The robot head vision system contains four motors, where two motors are used to rotate the two camera's, and the other two are for the pan and tilt rotation of the lower side of the robot head. The type of the motor is called 3Mxl, where its modules and firmware have been developed in the Delft BioRobotics lab. They have a similar protocol as the Dynamixel actuators, however its uses Maxon motors instead (*Maxon Motor ag*). Not only the actuation of the motor, but the PID control of the

Actuators	Encoder Resolution (CPR)	Gearbox Ratio (GR)	Dead Time (DT) [s]	K_p	K_i	K_d
Left Eye	512	1:19	0.03	0.2	0.4	0.5
Right Eye	512	1:19	0.04	0.2	0.2	0.5
Tilt	32	1:4.4*60	0.1006	0.5	0.05	0.0
Pan	32	1:231	0.1144	1.5	0.1	0.0

Table 3.1: Technical aspects of the Maxon motors and the PID gains used by the 3Mxl modules used in the binocular setup.

position, speed and torque is done by the 3Mxl modules located on the robot head.

In table 3.1, some technical aspects of the 3Mxl motors are shown. The encoder resolution (ER) stands for the number of counts per revolution of the motor's shaft. The dead time (DT) stands for the time that the motor does not respond after a reference signal is given to follow. The gearbox ratio (GR) stands for the ratio of the velocity of the input gear to the velocity of the output gear. Looking at the two rotating camera motors, they contain the most resolution of the four motors and the smallest dead time, however their GR is low. The pan and tilt motors do have a larger GR, therefore capable of handling more force. However, their DT is higher and encoder resolution lower, making them less useful for fast and delicate tasks as the camera motors.

The PID control performed by the firmware of the 3Mxl modules has to be tuned with the values The DT, ER and GR can say much about the systems' response. The lower PID tuning's values can be found in table 3.1, and the theory behind it in Appendix A.1.

3.2.2. Cameras

Two cameras have been mounted on top of the setup, connected to the two rotating motors. These provide the required image stream to the software, as vision drives this robot's control. The two cameras consist of two simple camera modules, which have a max resolution of 640 x 480 pixels, however now downsized to 320 x 240 for faster processing. The modules can reach a speed of 30 frames per second [fps].

The camera must be calibrated at least ones in its lifespan. This is done tagging multiple pictures containing a checkers board in different positions and orientations, with a corner detection utility (Fig. 3.3). Each image should have a different rotation and translation matrices in M_{ext} for each picture. However, the extrinsic matrix should be the same. By calibrating, the system is told how to interpret the image's coordinates to a 3D model and what its relative coordinate system is.

Where u and v are the image coordinates and X, Y, Z the 3D coordinates of the checker board in the real world, with q as rotation. M_{int} is the intrinsic matrix and M_{ext}, the extrinsic matrix:

$$M_{int} = \begin{pmatrix} fk_u & \rho & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

$$M_{ext} = \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \tag{3.2}$$

where f is the camera's focal length, k_u and k_u stands for the pixel size in distance and u_0 and v_0 stands for principle point (middle of image). r_1, r_2 and r_3 stand the columns of the rotation matrix R and T for the translation matrix.

An effect that also needs to be taken into account for calibrating the cameras is distortion. This is a phenomenon which is created by the lens of the camera. As a result, it causes straight lines in the scene to not appear straight on an image. The two types of distortion to be calibrated are the radial and tangential. Radial distortion can be corrected by the following formula:

$$u_{new} = u_{old}(1 + c_1 r^2 + c_2 r^4 + c_3 r^6) \tag{3.3}$$

$$v_{new} = v_{old}(1 + c_1 r^2 + c_2 r^4 + c_3 r^6) \tag{3.4}$$

where c_1, c_2 and c_3 stands for the radial distortion coefficients. u_{old} and v_{old} stand for the old pixel positions and u_{new} and v_{new} for the new position after transformation. The tangential distortion in the

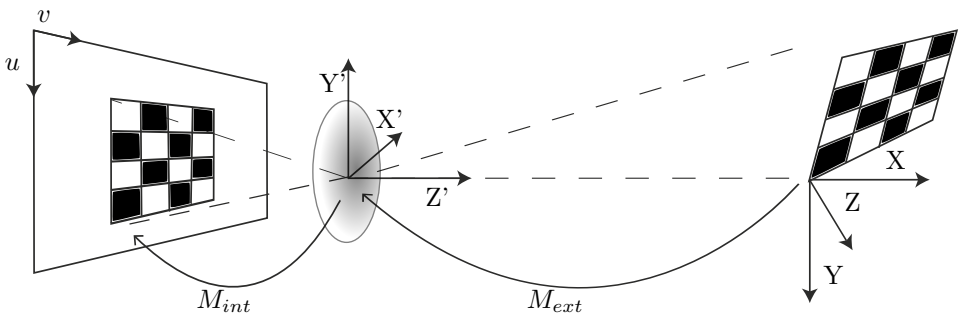


Figure 3.3: Schematic drawing to illustrate object transformation from real world to origin and to image plane. M_{ext} stands for the extrinsic matrix and M_{int} for the intrinsic matrix. X, Y, Z stand for the 3D coordinates of the object, and X', Y' and Z' stand for the same coordinates in respect to the lens and image alignment. u and v are the image coordinates.

	Accelerometer	Rate Gyro	Magneto Meter
Full Scale	50 [m/s2]	300 [deg/s]	5 [a.u.]
Bandwidth [hz]	30	50	10
Default sample frequency [hz]	100	100	100
Default Baudrate [bps]	115200	115200	115200
Noise	0.013 [m/s2]	0.007 [rad/s]	0.001 [a.u.]

Table 3.2: Technical specifications of IMU sensor of the type MTi, from motion technology

image is corrected as follows:

$$u_{new} = u_{old} + (2p_1 u_{old} v_{old} + p_2(r^2 + 2v_{old}^2)) \tag{3.5}$$

$$v_{new} = v_{old} + (p_1(r^2 + 2v_{old}^2) + 2p_2 u_{old} v_{old}) \tag{3.6}$$

where p_1 and p_2 ar the tangential distortion coefficients. These coefficients can be stored in matrix $D = (c_1, c_2, p_1, p_2, c_3)^T$.

The calibration is done by the OpenCV package for camera calibration. The results of the process is the following for the left camera and right camera respectively :

$$M_{int}^{left} = \begin{pmatrix} 269.1600 & 0.0 & 162.2064 \\ 0.0 & 265.5488 & 115.7278 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \quad M_{int}^{right} = \begin{pmatrix} 273.8821 & 0.0, & 160.6790 \\ 0.0000 & 269.8726 & 117.0881 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \tag{3.7}$$

$$D^{left} = \begin{pmatrix} 0.0713 \\ -0.1771 \\ -0.0010 \\ 0.0004 \\ 0.0 \end{pmatrix} \quad D^{right} = \begin{pmatrix} 0.0572 \\ -0.1494 \\ -0.0018 \\ 0.0003 \\ 0.0 \end{pmatrix} \tag{3.8}$$

The calibration tool does offer the option to do a stereo calibration of the two cameras, therefore also providing a rectification matrix. However, the robot will not be used to determine the 3D position of the object its following for this research. A rectification matrix was unnecessary, therefore made it possible to calibrate both cameras separately. This resulted in better calibration approximation for both cameras.

3.2.3. Inertia Measurement Unit

The system has an inertia measurement unit (IMU) to measure its 3D rotational velocity and orientation, linear acceleration and gravitational forces (Fig. 3.4). This sensor is used for various of fields of research in robotics, aerospace, autonomous vehicles, bio mechanics, and motion capture. In the binocular vision setup, it is mainly used to align the camera motors to counter the rotation of the neck motor. The specifications can be found in table 3.2

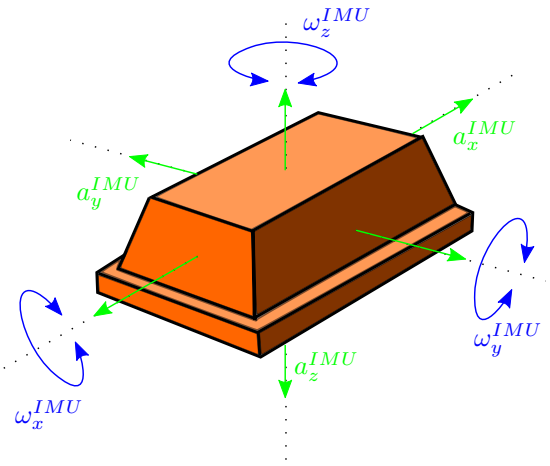


Figure 3.4: Schematic drawing of the coordinates of the Xsens IMU, where ω_x^{IMU}, ω_y^{IMU} and ω_z^{IMU} stand for the rate of the gyroscopes and a_x^{IMU}, a_y^{IMU} and a_z^{IMU} stand for the value of the accelerometers in x,y and z direction.

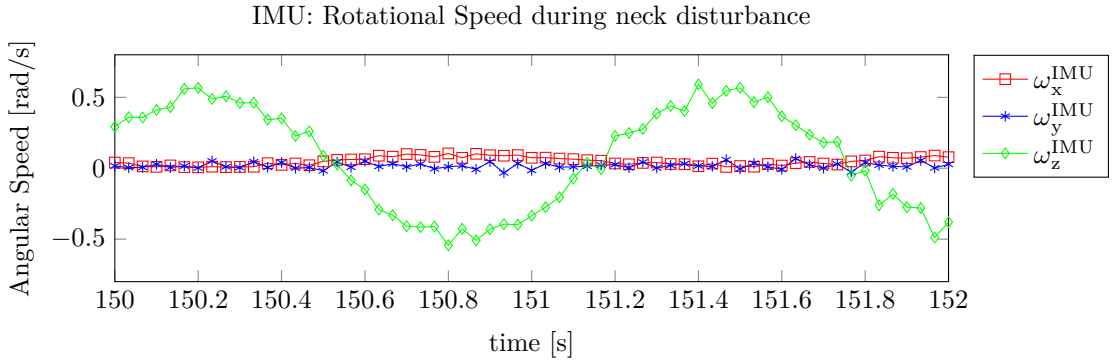


Figure 3.5: The angular velocity, sensed by the Xsens IMU on the robot-head during a neck motor perturbation.

The sensors within the IMU, which are the accelerometers, gyroscopes and magnetometers, are calibrated by means of a linear physical sensor model. The theory behind this model as well as the obtained parameters are found in Appendix A.2. A plot of its response during a rotational neck disturbance can be found in Fig. 3.5.

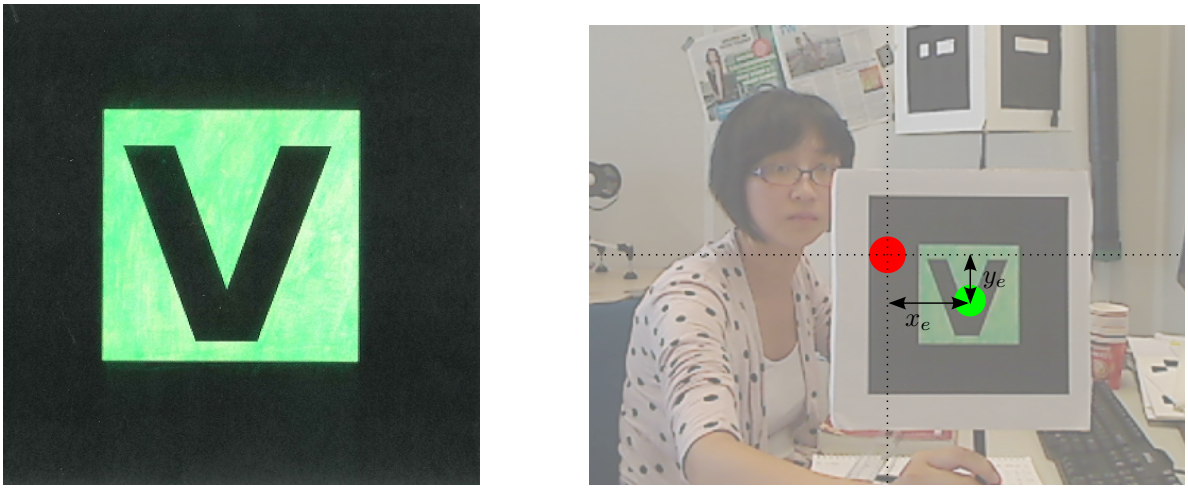
3.3. Software

Many aspects of the active binocular setup's processing are done by the computer it is connected to, which means it is not fully embedded. This section explains the software and control that has been provided as a base for this master thesis research. It consists of a tracker, which is able to detect an augmented reality marker in 3D space in the image streams that the cameras provide. A higher PID control module is used to be able to follow this marker and keep it in the center of its view. These modules are combined by means of an open source operating system for robotics, which all are explained in the following section.

3.3.1. Marker Tracker

To be able to follow a marker within the image stream that the cameras provide, an existing vision library for augmented reality (AR) purposes is used called ARToolKit. The software uses a previously specified marker (fig: 3.6), from which it can calculate its 3D position from the camera and its orientation. With this, a 3D virtual object can be projected upon the marker in the image, creating a connection of that virtual robot with the real world.

In the binocular setup, the 2D position of the center of the marker on the image must be provided. This is done by first using the camera (extrinsic) matrix to get the relative 3D position of the center axis of the lens and camera sensor. Then the 2D position of the marker can be calculated by the following



(a) (b)

Figure 3.6: Marker used for tracking task and the detected off center pixel distance

formula:

$$u_e = \frac{X'}{Z'} - u_c \tag{3.9}$$

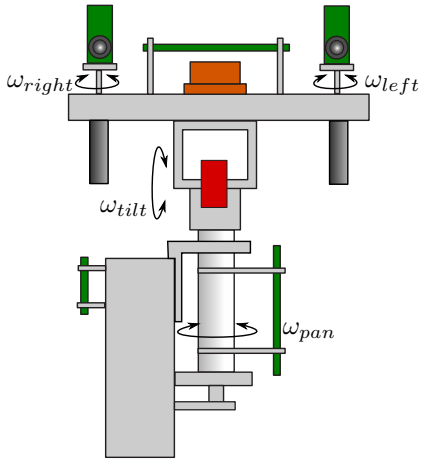
$$v_e = \frac{Y'}{Z'} - v_c \tag{3.10}$$

where u_e and v_e are the horizontal and vertical coordinates of the marker (off-center pixel distance), u_c and v_c are the coordinates of center of the image, X', Y' and Z' are the 3D coordinates of the marker after transformation by the extrinsic matrix (see sec. 3.2.2). The coordinates of the marker are used for the higher control module, which are explained in the next section.

3.3.2. Pursuit Control

From the tracker, the off-center distance of the marker within the image can be detected. This distance is used as error signal for the PID controllers, which role is to minimize this error (Fig. 3.7 a.). There are two PID controllers for each camera actuator, which respond to the horizontal element of the off center distance ($u_{e,left}$ and $u_{e,right}$). They give out a goal speed to the motor ($\omega^*_{e,left}$ and $\omega^*_{e,left}$), which response accordingly. The difference of the position of the left and right camera drives the PID controller of the neck. This is to ensure that the angle of the cameras are similar and the robot head is faced to whatever it is looking at. The tilt motor is driven the same way as the camera motors, only now with the vertical element of the off center distance (v_e). The control schemes of the pursuit can be found in Fig. 3.7(b).

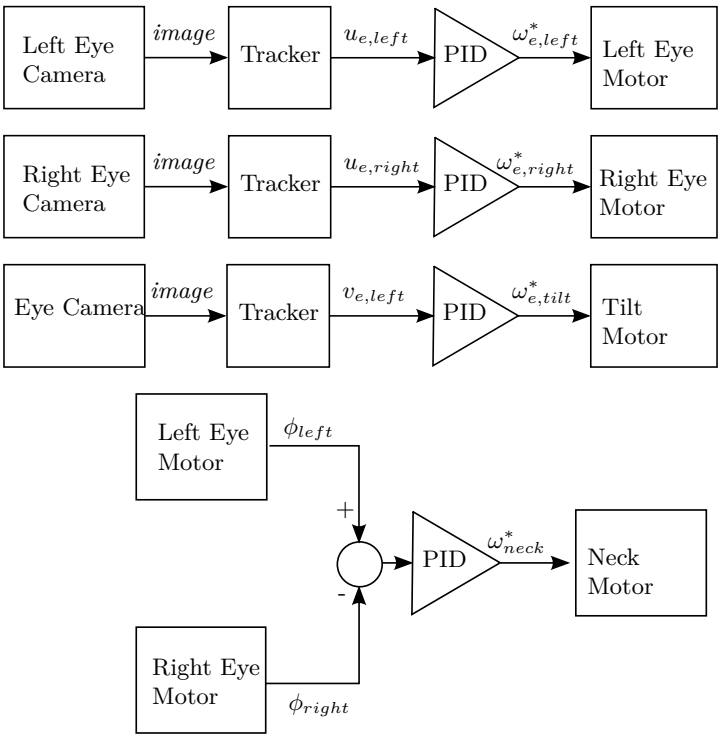
The PID values of the tracker had to be re-tuned, due to the difference of processing speed per computer. The higher PID controllers have been tuned by means of the Ziegler and Nichols (1942) method (see table. 3.3a). This is done by a top down approach: first the camera motors are tuned, then the tilt and finally the pan motor. The proportional gains (K_p) of the actuator is increased until it reaches the ultimate gain (K_u). This can be recognized if the motor keeps on oscillating without damping. From this oscillation, the period (P_u) is measured as well. From these two values, the proportional (K_p), differential (K_d) and integral (K_i) can be calculated depending on the control type (table. 3.3b). After implementing these control types, it has been chosen to drive the left, right camera and the tilt motors with a PI controller and the pan's motor with a P controller.



(a)

Actuators	PI K_p	K_i
Left Cam	0.0151	0.0254
Right Cam	0.0151	0.0254
Tilt	0.0137	0.0208
Neck	0.2700	0.0

(b)



(c)

Figure 3.7: (a) The robotic setup with the direction of the goal speed and (b) the control schemes of the robot responding to the tracking system.

Control Type	K_p	K_i	K_d
P	$0.50K_u$		
PI	$0.45\ K_u$	$1.2\ \frac{K_p}{P_u}$	
PID	$0.6K_u$	$2\frac{K_p}{P_u}$	$\frac{K_p P_u}{8}$

(a)

	K_u	P_u [sec]	K_p	K_i
left eye	0.0335	0.71	0.0150	0.0254
right eye	0.0335	0.71	0.015	0.0254
Tilt	0.0305	0.79	0.0137	0.0208
Pan	0.6	5	0.27	0.0

(b)

Table 3.3: (a)Ziegler and Nichols (1942) method's formulas and (b) the resulting ultimate gain (K_u), it's oscillation period (P_u) and the resulting proportional and integral gain (K_p and K_i) from the robotic setup.

3.3.3. Robot Operating System

The Robot Operating System (ROS), is an open source environment designed for robotic platforms (Quigley et al. (2009)). It contains a modular design, and is distributed by many people all over the world. Its contains a large community of fellow robotics engineers, keen on helping and sharing knowledge with each other. The core components of ROS consist of the packages and the communication between them. Each element of the robot can be contained by a software module, the package. This enables the user to easily add and remove elements of the control of the robot. The packages can send messages (topics) to the ROS environment or subscribe to a messages stream.

ROS can be seen as the glue which binds all the functionalities together within the software of the active binocular vision setup. In Fig. 3.8, the package and topic structure of the binocular setup can be seen. Here the robot sends out its current state of its actuators to the higher control, which on his turn uses the current 2D position of the marker. This position is retrieved by the tracker module, which uses the image stream provided by the cameras. The IMU also contributes to the pursuit control module by sharing its knowledge about the current angular velocity and linear acceleration.

Another benefit of ROS, is the ability to easily connect those modules to a simulation of the robot-

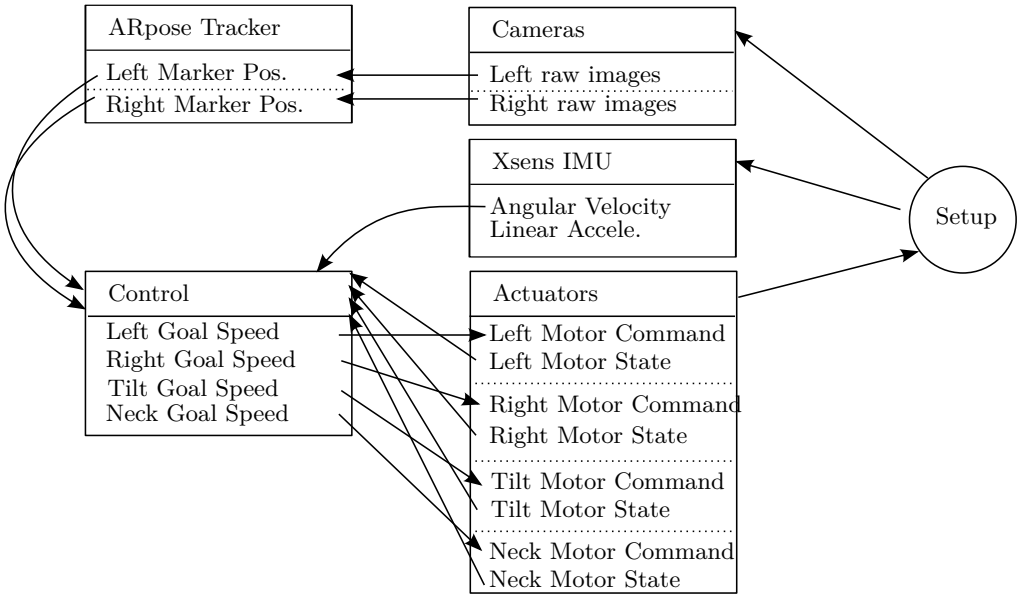


Figure 3.8: Structure of the ROS packages of the binocular eye setup.

head. With a simulation, different concepts of the cerebellar model can be tested without doing harm to the actual hardware. Testing ideas before implementing them in real life improves the safety of the experiments and provides an extra step within the gap between theory and implementation. See Appendix B for a detailed description of the simulation.

4

Simulating Cerebellum with Artificial Neural Networks

When one looks at the structure of the cerebellum and figures out a way to simulate it computationally, similarities are found with artificial neural networks. This chapter explains the theory of these artificial neurons and their connections, as they are used to simulating the cerebellar model within our active binocular vision setup. By implementing a supervised learning algorithm, the setup is able to predict motion within its image stream and respond to it. First, the context of machine learning is explained, as many varieties of learning exist, and to explain why a neural network would be suitable with the cerebellar model. Then several structures of the technique are explained, as well as the adaption of the connections within the network. Finally, the structure of a cerebellum inspired model in our active binocular vision system is presented.

4.1. Types of Machine Learning

One of the most important aspects which sets humans apart from robots, is their ability to adapt en learn. Every time one is introduced to a new environment or a new tool, the brain makes its connections in its neurons and adjusts their synaptic weights, to learn how to deal with these new encounters. Since robots are starting to emerge in our everyday lives, they need to have this ability as well, in order for them to really participate in both cognitive and motor tasks. The field that is focused on this aspect is called Machine Learning, and consists of all kinds of learning techniques involved.

Learning in general can be divided in three divisions: supervised, reinforcement and unsupervised (Russell and Norvig (2010)). In supervised learning, a 'teacher' guides the 'student' the entire way, explaining every step to achieve its goal. In reinforcement learning, the student gets a reward from the teacher, after he has correctly accomplished a task. With unsupervised learning, there is simply no teacher available and no feedback is provided, so the student must establish for himself if he did something correctly.

As for applications within machine learning, supervised learning is mostly applied for function approximation, from which the direct input and output are given as a training example. Although this type of learning is quite rapid, the time for preparation of the training data is quite extensive.

Reinforcement learning is done when the exact input and output are not known, but only the task that the system needs to achieve (Barto (1998)). An application made within the same Delft BioRobotics Lab is LEO, a robot which is able to learn himself how to walk, by giving it a reward if it achieves to move forward in any kind of way (Schuitema et al. (2010)). The downfall of reinforcement learning is that it takes longer to reach a possible solution than with supervised learning.

Unsupervised learning is by far the most difficult type of learning and is still found to be unsuitable for many tasks still. Clustering is an application where this type of learning is mostly used for, where it detects certain groups within data without any prior knowledge, therefore only relaying on the statistical properties of the input's data.

Within the human brain, Doya (2000) has established some types of learning methods for different brain areas. He claims that the cerebellar cortex executes unsupervised learning, which is able to

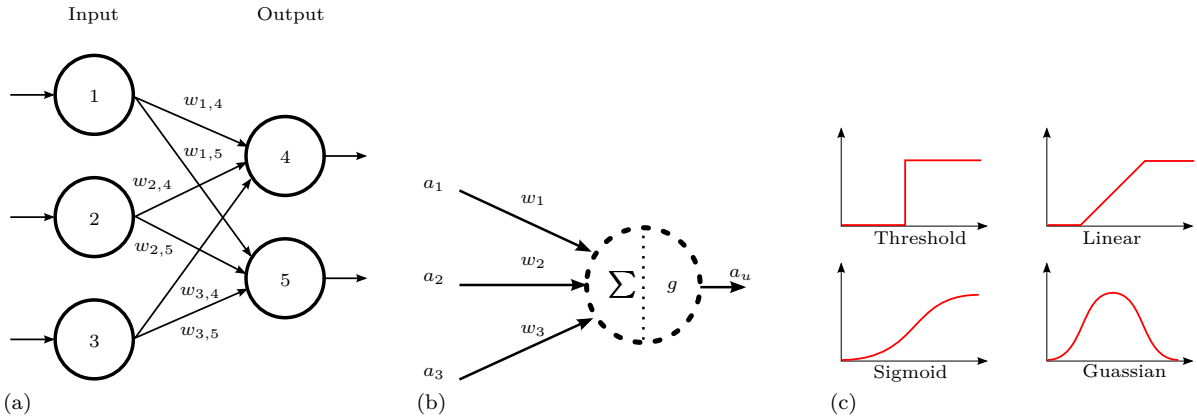


Figure 4.1: (a) A single layered network where the input and output layer are connected to each other by its nodes, with weighted synapses ($\boldsymbol{w}_{i,j}$). (b) is an enlarged neuron with summation of weighted inputs (a_i) and activation to output (a_u and (c) types of activation-function.

recognize a task and chose the appropriate architecture for it. The basal ganglia practices reinforcement learning, since that brain area is responsible for the fabrication of dopamine (the reward function for the human body). The cerebellum learns supervised, as it contains the internal models of the body and environment and has a big task within the fine-tuning of processes. Since this research focuses on implementing the functionality of the cerebellum, supervised learning is going to be further investigated. This is done by the most applied supervised learning technique, namely artificial neural networks (ANN).

4.2. Theory of Artificial Neural Networks

Artificial Neural Networks (ANN) are the computational variant of neuron's structure within our brains. They are used for various functionalities in the field of computer vision, speech recognition and more. Within system control engineering, they are also becoming more common, as they can be trained to predict the next possible consequence of the controllers actions within the loop.

4.2.1. Types of common Artificial Neural Network Models

Artificial neural networks are developed inspired by the connections within the neurons in the brain. It consists of several nodes which are connected to each other by weighted synapses (Russell and Norvig (2010)). The network has several layers and at least has one input and one output layer, from which an example can be seen in Fig. 4.1(a). The input that a neuron receives is the weighted combination of all the signals. On top of that, each node within the hidden and output layers contains an activation function, which means that a certain node only 'fires' an output if some threshold is received. The formula for that goes as followed, given the node in Fig. 4.1(b) as reference:

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right) \tag{4.1}$$

$$a_u = g(in) = g(w_1 a_1 + w_2 a_2 + w_3 a_3) \tag{4.2}$$

where a_i and a_j stand for the output signal of the ith and jth node, $w_{i,j}$ for the weight applied to the output signal that goes from the ith node to the jth node. There are several activation functions available, where the four visualized in Fig. 4.1(c) are the most used ones.

Many types of neural network exist, varying in size, connections or processed signals. A network with only one input and output layer is called a single layered perceptron (Fig. 4.2(a)). This type of ANN is mostly used as a classifier for linear functions and it is suitable for on-line learning due to fast processing. However it was not able to recognize many types of patterns.

The multilayer perceptron (MLP) is the extension of the single layered perceptron, and is able to learn non linear functions due to its added hidden layer (Fig. 4.2(b)). These added hidden layers have proven to be of much use within the field of computer vision, as certain aspects of the scene can be identified, analyzed and finally classified to a specific category. Within the layers of the MLP, the

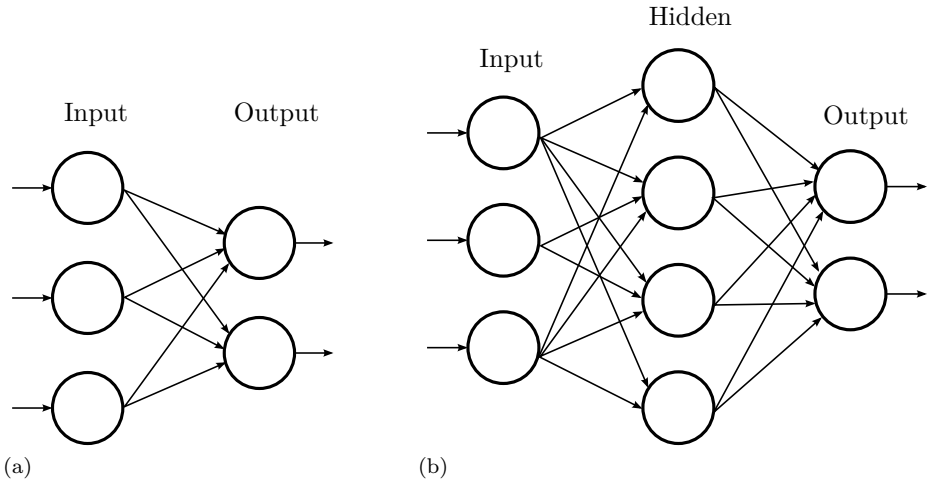


Figure 4.2: A representation of (a) a perceptron and (b) a multi layered perceptron.

hidden layer would extract the specific features (interesting objects) out of an image of a scene, from which the scene can be classified in the output layer. If the number of hidden layers are increased, the scene's features can be decomposed even more (Ciresan et al. (2012)). Such a multi-hidden layer MLP is also known as a Deep Neural Network (DNN).

There are several other ANN structures to consider. A recurrent network is an MLP, where the output of the hidden layers are fed back to the same neuron in the next time step (Fig. 4.3(a)). This enables the network to have short term memory, and makes it suitable to interpret written sentences for instances. The more recurrent connections, the longer those sentences can be. Another type of ANN is the time delay neural network (TDNN), which receives not only one input, but also the past versions of that signal (as can be seen in Fig. 4.3(b)). The TDNN is able to recognize an input sequence, for instance speech recognition, or is able to reproduce one as a time series prediction. As last, the radial basis function network (RBFN) is explained, which can be seen Fig. 4.3(c). This network has just two layers, from which the hidden layer contains a radial basis (Gaussian) function as activation function. Also, only the weights from the hidden layer to the output layer can be tuned and the hidden layer's radial basis functions are tuned in its parameters. RBFN's are used for various applications, varying from classifiers to time series prediction.

To push even closer to the actual simulation of neurons within the brain, spiking neural networks (SPNN) extends that limit even further. The difference with MLP, is that their neurons do not fire at each cycle of the execution, but only when they have reached enough signals themselves, as their 'membranes' electrical charge has reached a certain threshold (Vreeken (2002)). Their signals are more similar in how the human brain processes information, which are short electrical bursts instead of a continuous stream. The principle of the SPNN can be used for almost all the information processing tasks that ordinary ANN are applied for. For now they have mostly been implemented to simulate part of the central nervous system of small insects and animals for computational neuroscience. Although their potential for engineering purposes are great, they have not been applied due to excess of computational power needed to simulate the synapses, therefore have only been applied on a small scale. Recently more and more research has been done on the uses of SPNN, for simulating the cerebellum for motor control(see section 2.2.3) or robot control in general (Batllori et al. (2011) and Krichmar (2013)).

4.2.2. Back Propagation

The most used supervised learning algorithm for ANN, is error back propagation (BP). What BP does is to calculate the errors at the output of the neurons, adjust their weights and propagate the error back into the network to adjust its hidden layers weights. This is usually done for a particular training set of data, containing input and expected output of the network. BP is a first order gradient method and is applied by the following chain rule which calculates the influence of each weight (w_{ij}) to the error of the network (e):

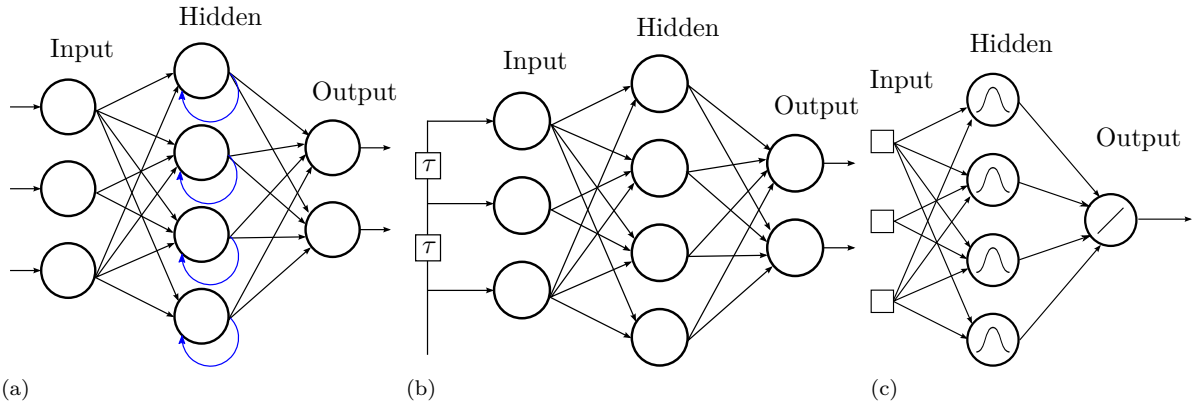


Figure 4.3: A representation of (a) a recurrent neural network, (b) a time delay neural network and (c) a radial basis function network.

$$\frac{\delta e}{\delta w_{ij}} = \frac{\delta e}{\delta r}\frac{\delta r}{\delta a_u}\frac{\delta a_u}{\delta w_{ij}} \tag{4.3}$$

where r is the weighted sum of the neurons inputs, without going through the activation function ($a_u = g(r)$). If each partial derivative is known and the above chain rule can be solved, that value can be used in for a gradient decent rule to update the weights:

$$w_{ij}(t+1) = w_{ij}(t) - \beta\frac{\delta e}{\delta w_{ij}} \tag{4.4}$$

where β stands for the learning rate. This can be done for several epochs, which stands for the amount of times that the train data set is used for the learning. However, choosing the learning rate is of great importance and should be done with care. If it is set too low, the network takes too long time to converge to a certain value. If it is set too high, the learning process becomes be unstable and it might even learn the wrong solution.

To deal with this problem, adaptive learning rate methods have been proposed for learning of an ANN, from which resilient back propagation (RPROP) is the most used one. RPROP is developed by Riedmiller and Braun (1993) and is a local adaption technique, where it is only aware of the local gradient information of the weight itself. This is apposed to having global knowledge about the entire network.

The adaption rule for RPROP goes as follows:

$$\Delta w_{ij}^*(t) = \begin{cases} \beta^+ \cdot \Delta w_{ij}^*(t-1) & \text{if } \frac{\delta e}{\delta w_{ij}}(t-1)\cdot\frac{\delta e}{\delta w_{ij}}(t) > 0 \\ \beta^- \cdot \Delta w_{ij}^*(t-1) & \text{if } \frac{\delta e}{\delta w_{ij}}(t-1)\cdot\frac{\delta e}{\delta w_{ij}}(t) < 0 \\ \Delta w_{ij}^*(t-1) & \text{else} \end{cases} \tag{4.5}$$

$$\text{where } 0 < \beta^- < 1 < \beta^+ \tag{4.6}$$

Where the value of the weight update (Δw_{ij}^*), is depended on sign change of the partial derivative of the weight. This is an indication that the last update performed on that neuron was too heavy and that the updated value needs to be decreased back with a factor β^-. On the other hand, the update value is increased with β^+ if the local minimum has not been reached yet. Subsequently, the update

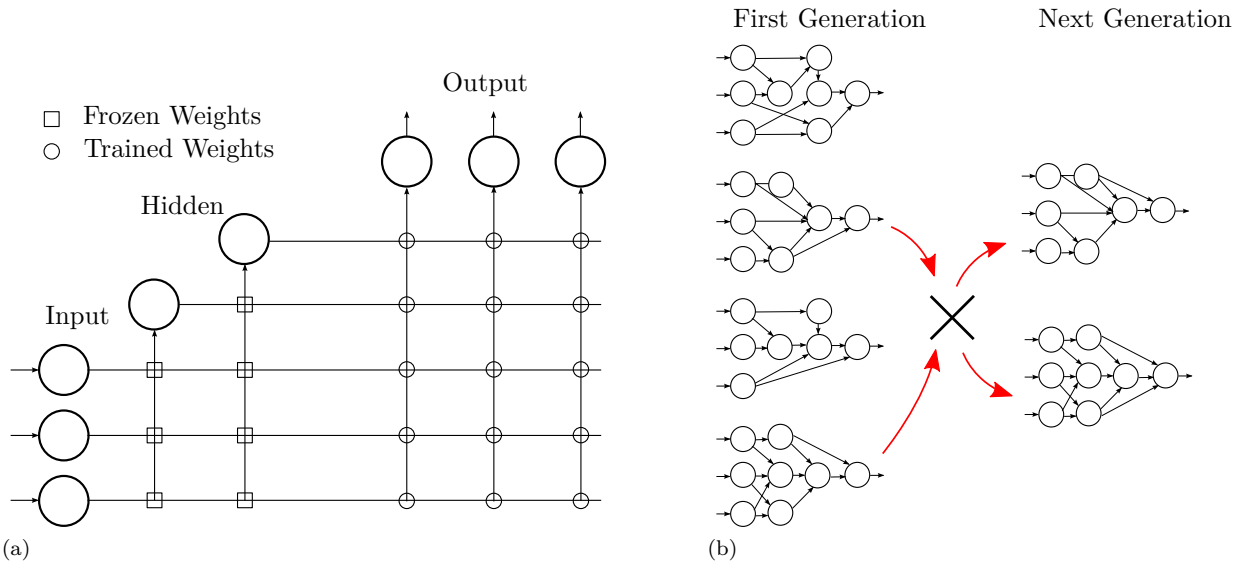


Figure 4.4: (a) A cascade architecture for the Cascade Correlation training adapted from Fahlman and Lebiere (1989) and (b) the evolving topographies of Neuro-evolution

value is added to the current value of the neurons weight in the following manner:

$$\Delta w_{ij}(t) = \begin{cases} -\Delta w^*_{ij}(t) & \text{if } \frac{\delta e}{\delta w_{ij}}(t) > 0 \\ +\Delta w^*_{ij}(t-1) & \text{if } \frac{\delta e}{\delta w_{ij}}(t) < 0 \\ 0 & \text{else} \end{cases} \tag{4.7}$$

$$\Delta w_{ij}(t) = w_{ij}(t) + \Delta w_{ij}(t) \tag{4.8}$$

With RPROP, no initial parameter of the learning rate has to be chosen beforehand, making it more robust than ordinary BP. It is therefore more able to be applied for ANN used in dynamic systems, where an adaptive learning rate is necessary.

4.2.3. Alternative Training Methods

Although back propagation methods are the most commonly used training algorithm for ANNs, the learning abilities can be improved even further. Arguments can be given that are they are un-biological, since there is no evidence that the brain executes error back propagation. Subsequently, there are usually no exact input and wanted output training sets available from which the neurons can learn. Therefore, some alternative ways to train ANN are explained here. Mazzoni et al. (1991) had proposed a method for ANNs to learn by reinforcement learning, as it is based on the update rule proposed by Barto (1998). Every neuron within the network receives a reward signal which is depended upon how close the overall output is to the desired value. This learning rule is formulated as follows:

$$\Delta w_{ij} = \rho r(a_i - p_i)a_j + \lambda\rho(1-r)(1-a_i-p_i)a_j \tag{4.9}$$

$$r = 1 - \left(\frac{1}{K}\sum_{k=1}^{K}|a^*_k - a_k|\right)^{1/n} \tag{4.10}$$

where ρ and λ are constants, p_i is the possibility of the neuron's firing, K is the amount of output neurons and r is the reward signal. This learning rule therefore does not need to back-propagate the exact error back into the network, however, it still needs an exact desired output of the total network to be able to learn.

Another method to train a neural network is by building up its topography from scratch. This is called cascade correlation and developed by Fahlman and Lebiere (1989). The principle is that the

training begins with a minimal network, and adds one new hidden neuron at the time, therefore creating multiple layers. First the minimal network, which is fully connected to each neuron, is trained with a conventional learning method, until the error of the output and the desired is minimized. Afterwards, candidate units are created, which are connected with all the input and hidden layer neurons but not with the output layer. All the links to the candidate unit are trained, until one candidate unit has reached the best with the error of the entire net. This particular unit is chosen to be added to the ANN, and the weights leading up to it are frozen in value. This process is repeated until the overall error of the network has improved enough, and the final network gets a structure similar to Fig. 4.4 (a).

Cascade Correlation modifies the actual structure of the ANN, however there is a method which takes this aspect a step further: Neuro-evolution (Stanley and Miikkulainen (2002)). Evolutionary algorithms are used here, to produce a generation of different ANN structures and connections, train them and submit them to a fitness function (Fig. 4.4 (b)). This fitness function is based upon the performance of the neural network within its task. This does not necessary has to be its direct output, but its effect on the task is sufficient. The structures which perform the best on the fitness function, are selected from the pool and their aspects are used to create new offspring for the next generation. This generation is also trained and goes trough the same process, until an optimal structure of the neural network can be found. This method is based on an overall fitness function, which applies that these neural networks can be more widely used than the conventional supervised learning algorithms, which need an exact input and output pair in order to function. Therefore, evolutionary ANN have been applied within artificial life and evolutionarily robotic applications (Hülse et al. (2004)).

4.3. Cerebellar-inspired Model as an ANN

Now the theory behind artificial neural networks have been explained, the implementation of the adaptive filter from Dean and Porrill (2011) and its variation, the information filter, is described in this section. From their description and the current knowledge about ANNs, the cerebellar inspired model is built by means of the Fast Artificial Neural Network library (FANN). This thesis's objective is to this model run (close to) real time with the robot, so the 'fast' attribute is important to have. First the building block of the model are explained, from which afterwards some considerations are pointed out, which needs to be addressed in the later chapters.

4.3.1. Choices and Considerations

There are some choices and considerations to discuss regarding the implementation of the adaptive and information filter. First of all, although the model is described within the paper of Lenz et al. (2009) as a two layered network, which is basically a perceptron, from theory it is known that it is not be able to handle non-linear problems. Later with the experiments, this needs to be addressed too for testing this principle. It suspected that it is not be able to handle a linear disturbance, which is a non-linear transformation to the motors.

Also, this the reason for the use of the FANN library for the development of the models. The formulas given in the paper of the adaptive filter are not extensive in terms of a ANN, therefore, it seems quite straightforward to implement them. However, there was no source code was made available and Lenz et al. (2009) were using a custom made chip board for processing. Therefore, it seemed more wise to use a library which has been developed and optimized for fast computation, also considering the time frame of this research. Choosing the learning method to be RPROP, however, is a decision based on how the adaptive filter is described in the literature. When the different networks of the adaptive and information filter are compared with each other, it is more fair to compare them containing a similar structure. With cascade or evolutionary training, these topographies would change and differ dramatically from each other which is not useful for the comparison.

Another is the indirect learning problem, discussed in section 4.2.3. It has been referred to as the distal supervised learning problem by Jordan and Rumelhart (1992). It stands for that the learner, does not know its desired output to be trained on, until it has been exposed to the outside world and its effect is known. This issue complicates the supervised learning as the exact input and output training set can not be determined straight away This has been de reason why I looked into alternative training algorithms to circumvent this issue. However, all of them still need an exact training set which had to be determined by its user. For both the cerebellar models, the training set will therefore be handled in the same conventional way, but the distal learning problem will be noted for discussion.

Subsequently, the type of motor input can also be open for debate. Now, the goal speeds originated from the higher PID controllers are used, but the actual voltages given by the lower PID controllers in the 3Mxl modules can be considered as well. Using the goal speeds from the higher control puts the cerebellar model on the same level as them, as they both use information from the cameras and the filters' output is added to the pursuit signal. These signals can also be separated, which can increase the performance if the adaptive or information filter receives motor input without its influences. To give this argument some back-up, the experiments dedicate some small devotion to this as well.

It would seem to be benefit to stay true to the biology as close as possible. However, using a spiking neural network for simulation is not done even though its closer to the actual computation in the brain. SPNN are computationally heavy on a conventional computer. Since three ANN are running and learning at the same time, together with all the functionalities within the binocular robot, it would be unfortunate if the computer's processor on which it runs, is not able to handle excess of tasks. The second generation ANN type is used to simulate the structure of the cerebellum for this research.

4.3.2. Building The Model

In the adaptive filter theory (Lenz et al. (2009) and Dean and Porrill (2011)), the motor input was used as an input for the network with tap delays. This indicates that they have used a TDNN type of network. This indicates that the neural network receives input in form of a time series, so it knows the history of the motor commands given to the system. Its output signal is given to the control of the robot, from which the visual detected motion of the image (e_{motion}) is the signal which drives the learning within the model. See section 2.3.1 for the formulas from the paper of the adaptive filter.

In Fig. 4.5, the schematic representation of the to-be-implemented adaptive filter is seen. The motor inputs ($\boldsymbol{u}$) which is given to the robot by the controller, from the current time-step (k) and six time-steps back. This is due to a tapped delay line, from which its developers believe it resides in the granule cell layer of the physical cerebellum. $\boldsymbol{u}$ is a vector which contains the values of both the camera motor (u_{left}, u_{right}) and the tilt motor input (u_{tilt}). The adaptive network contains only two layers, from which the input layer contains Gaussian activation functions and the single output layer consists linear activation-functions. The inputs are given to an ANN, where the weighted combination of them is the output of the cerebellum. The training of the weights $\boldsymbol{w}_1..\boldsymbol{w}_6$ is done by RPROP. Although from the research from Lenz et al. (2009) is not clear what learning algorithm they used, they did say that it was depended on the sign of the error. Since RPROP is driven by the sign of the error's gradient, it seems to come close to their idea of learning algorithm of the cerebellar model.

To prepare the training data with the exact input and desired output it is important to consider standardizing the data within that set (Sola and Sevilla (1997)). The data's values should stay within a range of [-1, 1], in order for the ANN to be able to process it. The data would have been normalized apposed to each other and used for training, and rescale the output back to its original order of magnitude. In practice it is shown to be a benefit to the ANN's learning and convergence speed and the FANN library provides this functionality as well. However, since the input and output data already consist within a range of [-1 1] (see Appendix C), this made little influence for the learning speed. In theory, any scale difference within the data would be account for by the weights of the neural network. Since it does not make a significant difference in learning for our cerebellar models, the training data is not normalized. Instead, a short check is conducted, to monitor if the training data have stayed within the boundaries.

The information filter is the variation of the adaptive filter made, but pulled closer to recent neurological research. There are some conflicting aspects with the adaptive filter theory and the research about the cerebellum from a neurological point of view. First af all, the cerebellum contains all the information from the brain and the body's senses not only the motor input to the muscles. On top of that, the cerebellum is not indicated to have recurrent connections (see section 2.1.2). This is the reason for the information filter to only receive information of the motor input, the state sensors, the IMU and vision from just one time step k. In table 4.1 shows the exact input of the information filter, from what type of signal, which direction and corresponding symbol. In Fig. 4.6 as well that the rest of the information filter's structure is the same as the adaptive filter's structure.

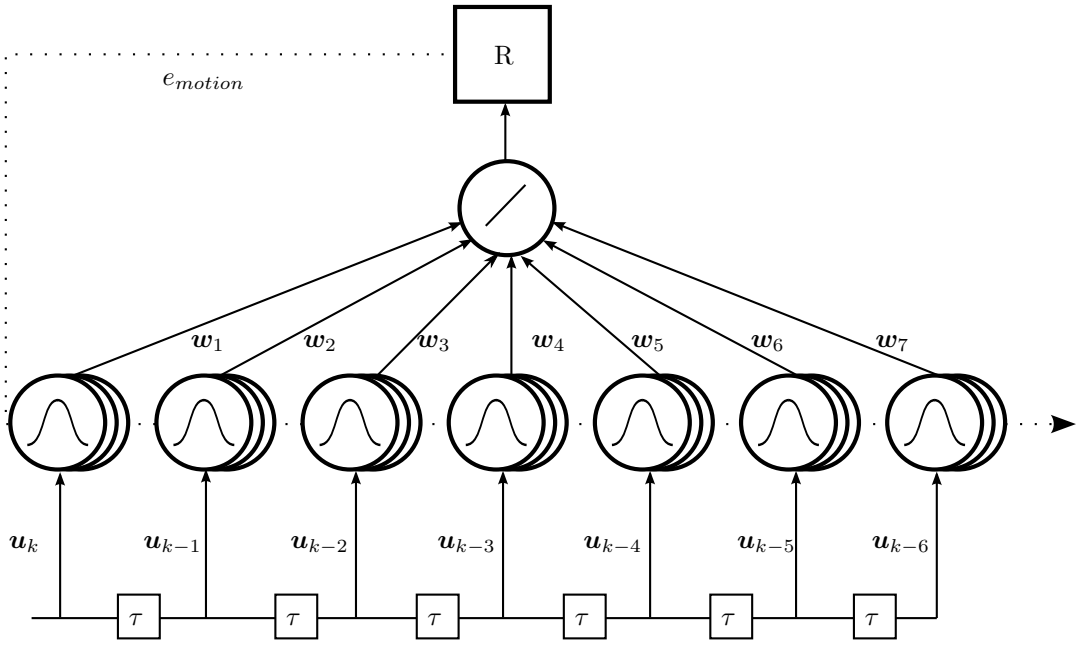


Figure 4.5: The implementation of the adaptive filter model of the cerebellum for the active binocular vision setup (R). The network receives the three motor inputs to the setup's actuators ($\boldsymbol{u} = (u_{left}, u_{right}, u_{tilt})^t$), for the current time step (k) and the six before that ($k - N$), so 21 in total. After computation with the 21 tunable weights ($\boldsymbol{w}_n = (w_{n,left} w_{n,right} w_{n,tilt})^t$) is given to the control of the robot, from which the visual detected motion of the image (e_{motion}) is the signal which drives the learning within the model.

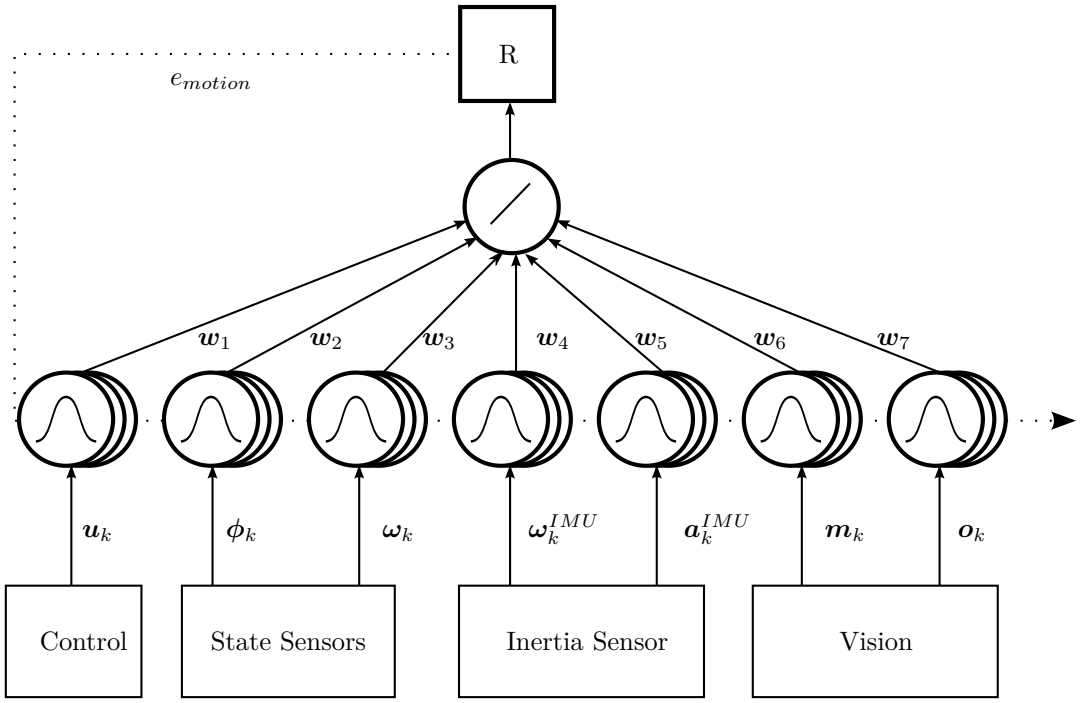


Figure 4.6: The implementation of the information filter model of the cerebellum for the active binocular vision setup (R). The network receives signals from the controller , the state sensors , the IMU and vision as the motor input $\boldsymbol{u}$, position ϕ and velocity $\boldsymbol{\omega}$ of actuators, rotational velocity $\boldsymbol{\omega}^{IMU}$ and linear acceleration $\boldsymbol{a}^{IMU}$, and position of the marker $\boldsymbol{m}$ and detected motion in the image $\boldsymbol{o}$ After computation with the 21 tunable weights ($\boldsymbol{w}_n = (w_{n,left} w_{n,right} w_{n,tilt})^t$) is given to the control of the robot, from which the visual detected motion of the image ($\boldsymbol{e}_{motion}$) is the signal which drives the learning within the model.

Sensors	value	direction	symbol
Control	Motor input	Left	u_{left}
		Right	u_{right}
		Tilt	u_{tilt}
State	Position	Left	ϕ_{left}
		Right	ϕ_{right}
		Tilt	ϕ_{tilt}
	Velocity	Left	ω_{left}
		Right	ω_{right}
		Tilt	ω_{tilt}
IMU	Rotational Velocity	X	ω_x^{IMU}
		Y	ω_y^{IMU}
		Z	ω_z^{IMU}
	Linear Acceleration	X	a_x^{IMU}
		Y	a_y^{IMU}
		Z	a_z^{IMU}
Vision	Marker Position	u (left)	$m_{u,left}$
		u (left)	$m_{u,right}$
		v (both)	$m_{v,both}$
	Optical Flow	u (left)	$o_{u,left}$
		u (left)	$o_{u,right}$
		v (both)	$o_{v,both}$

Table 4.1: The input signals of the information filter displayed in Fig. 4.6, with the sensor they originate from, their type and direction and the symbol used.

5

Visual Detection of Ego-Motion

In the previous chapter, the structure of the cerebellar model as an artificial neural network was explained. The learning signal which adapts the weights of that model, is based on the detection of ego-motion within the image stream from the cameras. Self-motion stands for the movement that the robot conducts on itself, not the movement of other objects within its field of view. In the physiology of the human brain, the cerebellum does the same uses retinal slip as a learning signal to adapt the VOR.

To detect motion within an image stream, one needs to compare the change of pixel intensity from two sequential frames. This can either be done pixel by pixel, or by matching several interesting points which each other. Therefore, some basic algorithms for feature detection are explained, from which the movement, also called optical flow, can be derived from one frame to the next. There are two main aspects to hold into account for feature recognition and optical flow, which are repeatability and speed. Repeatability stands for the ability to detect the same feature in a scene with different configurations, where speed stands for the computational effort to detect them. Although in an ideal world, one would like to have a good performance of both, however in reality a trade off must be made for both of these items. By investigating the theory behind ego-motion detection, it will becomes more clear of what that trade off should be for the implementation with our active binocular vision system.

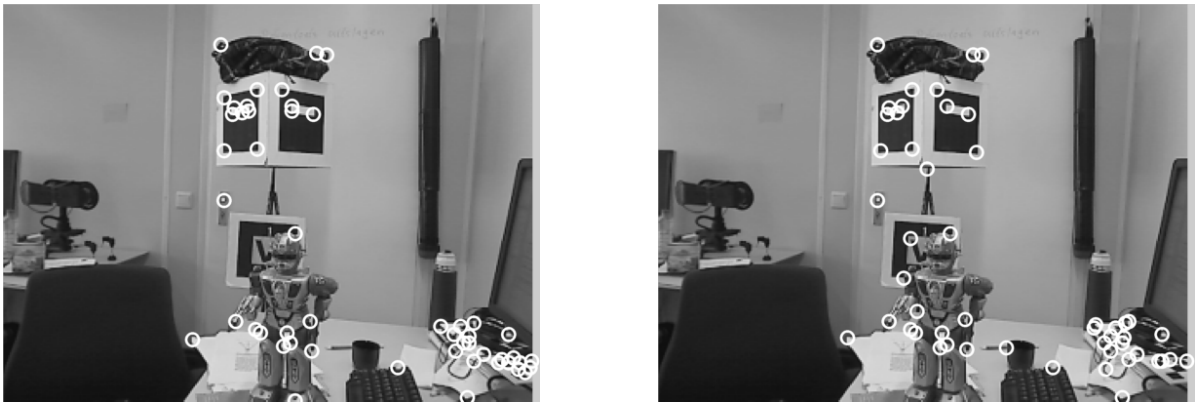
5.1. Interesting Features Recognition

When a human looks at a natural scene for the first time, one does not look at the scene in whole, but makes saccades from one points to another. These points are called features, and their characteristics in terms of texture, gradient and color is what draws one's eye towards them. Combining these features together within the brain, enables the viewer e.g. to recognize of where he/she is standing or whatever he/she is looking at.

For the recognition of features, there are several steps to undertake. First is the detection of an interesting point or area, where an algorithm looks at certain aspects and details of the image. Afterward, that feature needs to be correctly described, related to its orientation, its neighboring pixels or other characteristics. This description increases the ability to find a feature back on another image of the same scene, so that they can be matched. Such a matching algorithm can help in object recognition, creating a depth map or determining the motion of the camera.

5.1.1. Feature Detection

Within a natural image, there are several aspects that one's eyes are easily drawn too. An edge is recognizable as a strong cutoff of contrast in one dimension and a corner has this same characteristic in two dimensions. Therefore, corners can be localized as a single precise point in the scene. These large difference in contrast and color within an image are called a gradient. The Harris corner detection algorithm, developed by Harris and Stephens (1988), actually uses these aspects. The method is applied to the image gradients as follows:



(a) Harris (b) Shi-Tomasi

Figure 5.1: (a) The Harris corner detection and (b) the Shi-Tomasi corner detection from the same scene.

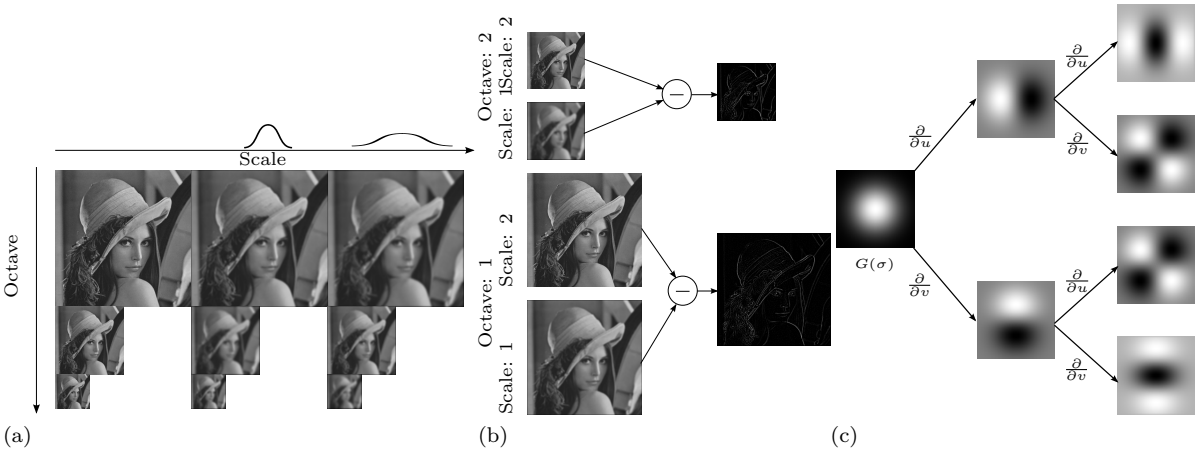


Figure 5.2: (a) The scale invariant representation, and the difference in Gaussians (DoG)

$$S(\Delta u, \Delta v) = \sum_{\Delta u \Delta v} \Delta u^2 I_u^2 + 2\Delta u \Delta v I_u I_v + \Delta v^2 I_v^2 \tag{5.1}$$

$$= [\Delta u\, \Delta v] M \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} \tag{5.2}$$

$$\text{with } M = W(\Delta u, \Delta v) \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \tag{5.3}$$

where S is the degree of variation of the intensity within a certain window (W) and I_u and I_v are the image gradients in the horizontal and vertical direction. M is called the Harris matrix and by its characteristics, corners can be found in an image. In the Harris corner detection the strength of the corner can be found by the determinant and trace of this matrix, but Shi and Tomasi (1994) have taken a simpler approach by determining if the smallest eigenvalue stays above a certain criterion:

$$\min(\lambda_1, \lambda_2) > \lambda \tag{5.4}$$

Where λ_1 and λ_2 are the eigenvalues of M, and λ is a predefined threshold. For tracking corners from frame to frame, the Shi-Tomasi approach is said a more stable way of corner detection. The results of both the Harris and Shi-Tomasi corner detection can be found in Fig. 5.1.

Naturally there are other items than corners which can capture the eye. These areas of interest, also called blobs, can also be detected within the image, for which Lowe (2004) uses the difference of Gaussians (DoG). In Fig. 5.2(a), the scale invariant representation can be seen. Here the image is constantly downsized in its size (octave), and at each level, the images is convolved by a Gaussian filter with a standard deviation of σ (scale). At each level of the scale, σ increases with the multiple of $\sqrt{2}$. The DoG is calculated by taking the difference of 2 images, both neighboring each other by their octave and from the same scale (see Fig 5.2(b)). In formulas, this process goes as follows:

$$L(u, v, \sigma) = G(u, v, \sigma) * I(u, v) \tag{5.5}$$
$$DoG(u, v, \sigma) = L(u, v, k\sigma) - L(u, v, \sigma) \tag{5.6}$$

Where $L(u, v, \sigma)$ stand for the Gaussian convolved image, with the σ the standard deviation of the Gaussian function $G(u, v, \sigma)$ and a constant multiplicative factor k. After the DoG's of each scale and octave are calculated, the local extrema detection can be used. All the DoG's upon the scale are compared with each other at each sample pixel locally per neighborhood at the current location and the adjacent scale. If the value is the largest or smallest at the local comparison, that location is marked as a candidate key point.

Another method to find interesting areas is the fast Hessian detector, which uses the second derivative of the image's gradient. For any point (u, v), in the image I, the Hessian matrix H used for the detection (Bretzner and Lindeberg (1998)) is described as follows:

$$H(u, v, \sigma) = \begin{bmatrix} L_{uu}(u, v, \sigma) & L_{uv}(u, v, \sigma) \\ L_{uv}(u, v, \sigma) & L_{vv}(u, v, \sigma) \end{bmatrix} \tag{5.7}$$

where $L_{uu}(u, v, \sigma)$, $L_{uv}(u, v, \sigma)$ and $L_{vv}(u, v, \sigma)$ are the convolution of the second order derivative of a Gaussian filter , which are $\frac{\partial^2}{\partial u^2}G(\sigma)$, $\frac{\partial^2}{\partial v^2}G(\sigma)$ and $\frac{\partial^2}{\partial u \partial v}G(\sigma)$ respectively (see Fig. 5.2(c)). Where the determinant of the Hessian matrix become maximal, is where the candidate key points can be found. This enables the method to also detect points with strong texture. The features therefore are more resilient to heavy rotation, however less precise in location oppose to the Harris/Shi-Tomasi corner detection. Therefore, these candidate key points first needs to be correctly described, which is explained in the next subsection.

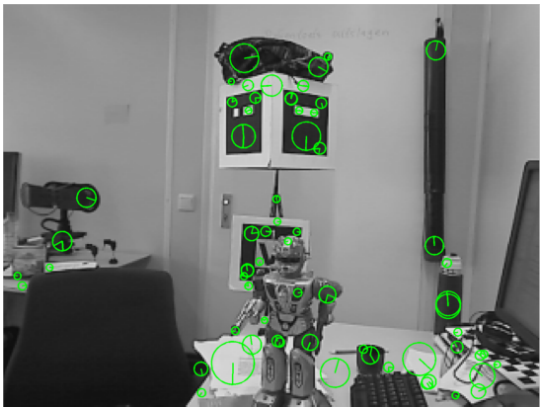
5.1.2. Feature Description and Matching

In order to use specific features for a tracking task like calculating optical flow, interesting points need to be found by a detector in every step of the image stream. The repeatability of the detection is therefore crucial, and this has to be done for different scales, orientations and luminosity. Not just detecting these features is enough, but they need to be selected and described according to these characteristics.

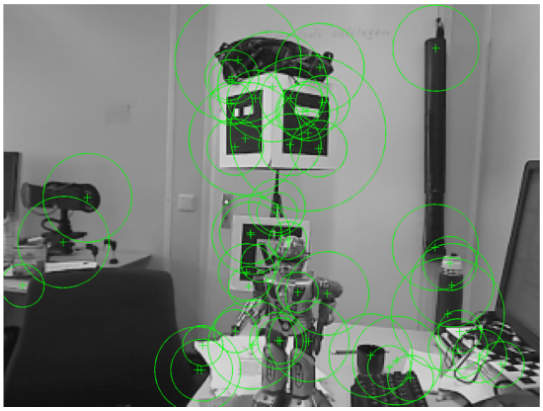
Lowe (2004) has developed a method, which is able to describe features from an image, which can be matched with the same feature from a different scale. This is called the Scale-Invariant Feature Transform (SIFT) and it consists of several steps. Of course, the feature needs to be detected first, which is done by the DoG explained in the last subsection. Afterwards, its characteristics needs to be determined, namely its magnitude and orientation. Depending on what DoG-scale a particular key point is detected, this aspect used to select the Gaussian convolved image to use for calculating its magnitude and orientation:

$$m(u, v) = \sqrt{(L(u+1, v) - L(u-1, v))^2 + L(u, v+1) - L(u, v-1))^2} \tag{5.8}$$
$$\theta(u, v) = \tan^{-1}((L(u+1, v) - L(u-1, v))/(L(u, v+1) - L(u, v-1))) \tag{5.9}$$

where L stand for the Gaussian blurred image for any σ, m for de magnitude and θ for the orientation. This is done for each image sample of the scale-invariant representation, and done so that the 2D coordinate of that descriptor is repeatable for each rotation and scale of the key point (see the result in Fig. 5.3(a)). Although rotation invariance is a great asset to have, it can also be a negative for the preciseness of the descriptor. Making it too invariant might lose its ability to repeatably detect the exact position of that feature every time, especially when one frame is compared to the other and the



(a) SIFT



(b) SURF

Figure 5.3: (a) SIFT descriptors and (b) SURF descriptors

(a) Harris (b) Shi-Tomasi (c) SIFT (d) SURF

Figure 5.4: Comparison of v.l.r. Harris, Shi-Tomasi, SIFT and SURF feature detection with matching one frame to the other .

rotation of the camera is not significantly large. Bay et al. (2006) did remark this in their research, in addition to the needed accelerated computational speed of the detector descriptor combo. They had developed a method called Speeded Up Robust Features (SURF). The first step of the algorithm is to find the features using a Fast-Hessian detector as explained in the last section. In terms of description, the keypoint is similarly described as SIFT features, however it has been slimmed down to some bare basics. Firstly, the circular region around the interesting point is used to describe the orientation. The SURF descriptor is extracted from the squared region aligned to that orientation, which is split up in 4 smaller subregions. From the sample points within each of these subregions, the Haar wavelet responses are calculated and placed within a descriptor vector $\boldsymbol{v}$.

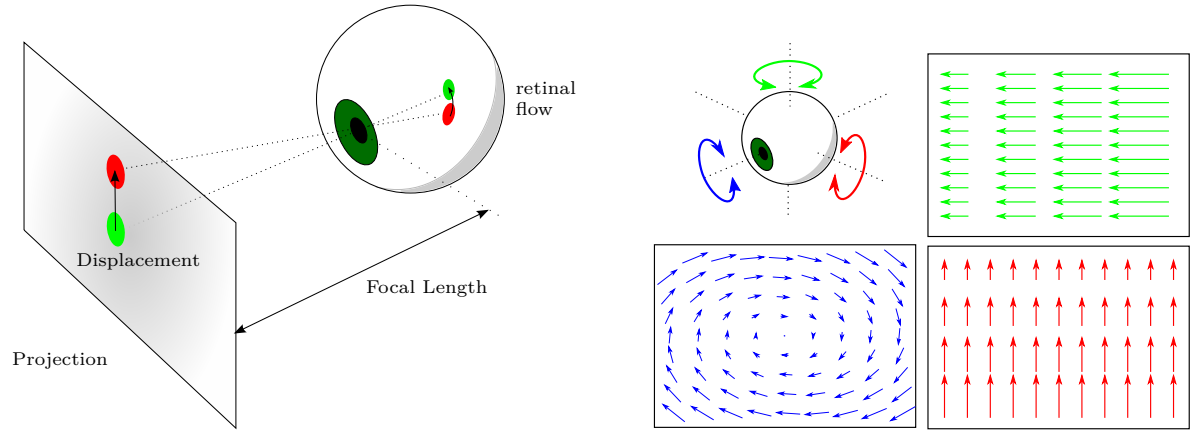
$$\boldsymbol{v} = (\sum d_u, \sum d_v, \sum |d_u|, \sum |d_v|) \tag{5.10}$$

where d_u and d_v stand for the Haar wavelet responses in the horizontal and vertical direction. A haar wavelet is a squared shaped functions, and analysis by these are similar to the Fourier analysis. The lack of continuity of this function is an advantage of analyzing signals with sharp transitions, makes it suitable to describe the interesting features found in a scene (see the result in Fig. 5.3(b)).

The euclidean distance between each feature of two images can be compared with each other and the pair that is the closest to each other is selected as a match. In Fig. 5.4, this principle is done for the feature which are detected and described as above, for two frame images from the active binocular setup own cameras. Different matching techniques are available, which can appeal to object detection, depth map creation and more. However, since the purpose of this chapter is ego-motion detection, the next section goes further onto this by, discussing techniques for optical flow.

5.2. Optical Flow Approximation

To perceive ego-motion within an image stream, one frame must be compared with its version of a previous time step. From this it becomes clear on how much the current image pixels have moved



(a) (b)

Figure 5.5: (a) The representation of optical as retinal slip in the human eye and (b) the type of optical flow depending on the type of ego-motion.

within that time frame. This is called optical flow and its techniques used widely for tracking, motion estimation, navigation systems and visual odometry. The algorithms does not need to be constrained by the 2D plane of the image only, but can also express 3D position and velocity of objects within the scene.

Within the physiology of the human eye, optical flow is detected as retinal slip, which stands for the amount of flow that a detected feature has traveled upon the retina (Fig. 5.5 (a)). Depending on the movement of the body or just the eye ball itself, the motion in the scene have different characteristics (Fig. 5.5(b)). From these, it can become eminent on how the body moves oppose to the outside world.

The mechanism of optical flow can be expressed in the following formulas:

$$I(u, v, t) \;=\; I(u + \Delta u, v + \Delta v, t + dt) \tag{5.11}$$

Where $I(u, v, t)$ stands for the pixel value, u for the horizontal coordinates, v for the vertical coordinates and t for time. It says that the same pixel value from frame at $t = 1$, has moved with dv and dv at frame at $t + 1$. Eq. 5.11 can be rewritten as:

$$I_x \dot{u} + I_y \dot{v} + I_t \;=\; 0 \tag{5.12}$$

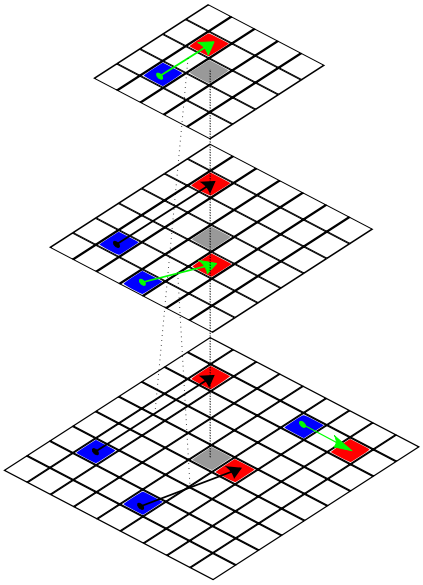
where

$$I_u = \frac{\partial I}{\partial u},\; I_v = \frac{\partial I}{\partial v}$$
$$\dot{u} = \frac{\Delta u}{dt},\; \dot{v} = \frac{\Delta v}{dt}$$

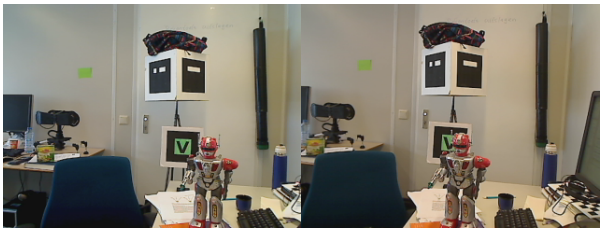
Here, I_u and I_v are image gradient in vertical and horizontal direction and I_t is the gradient among time. $\dot{u}, \dot{v}$ are the optical flow in horizontal and vertical direction as a derivative of time. However, they are the two unknowns in one equation, which can not be solved by simple algebra, therefore can only be approximated. There are two main strategies to detect optical flow in an image stream sequence: dense or sparse optical flow. Dense optical flow is a pixel to pixel algorithm, oppose to sparse which is a feature based optical flow algorithm and only need to process some pixels within the image. The following subsection explains how the optical flow equation can be solved by these two strategies.

5.2.1. Sparse Optical Flow

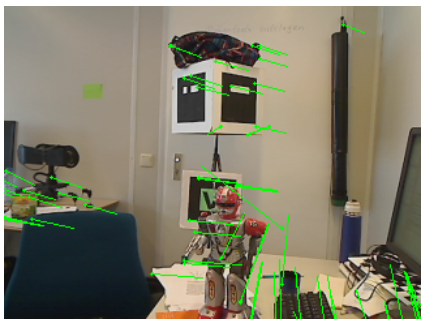
Sparse optical flow stand for the motion detected between matched features, which were discussed the first section of this chapter. Assuming that the change of the position of the camera and/or scene is relatively small, the match of two features are linked to each other and a flow vector can be generated between them. This prevents the need of having to generate the entire image's optical flow. The most used algorithm for sparse optical flow between feature is the Lucas-Kanade (LK) Feature Tracker, developed by Lucas et al. (1981). The LK method assumes that the motion between two frames is as



(a)



(b)



(c)

Figure 5.6: (a)The pyramidal implementation of the Lucas-Kanada optical flow estimation and (c) the result of using the method between features detected by the Shi-Tomasi corner detection, between the two frames represented in (b).

small that it fits within the neighborhood around a feature. It is considering that the image velocity ($\boldsymbol{d} = (\Delta u, \Delta v)$ is found by minimizing the following function:

$$\gamma(\boldsymbol{d}) = \sum_{u-\frac{W_u}{2}}^{u+\frac{W_u}{2}} \sum_{v-\frac{W_v}{2}}^{v+\frac{W_v}{2}} (I_1(u,v) - I_2(u+\Delta u, v+\Delta v) \tag{5.13}$$

The recent, most used version of this algorithm is the pyramidal implementation from the work of Bouguet (2001). Since the LK method is based on small motion within a window, any slightly larger motions can not be detected anymore if it does not fit within the search window. However downscaling the image, as shown in Fig. 5.6 (a), it would be possible to detect motion at each level/octave of the pyramidal representation. At each level the spatial gradient matrix M, similar to the Harris matrix, is calculated:

$$M = W(\Delta u, \Delta v) \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \tag{5.14}$$

which is than used to iteratively solve the following solution:

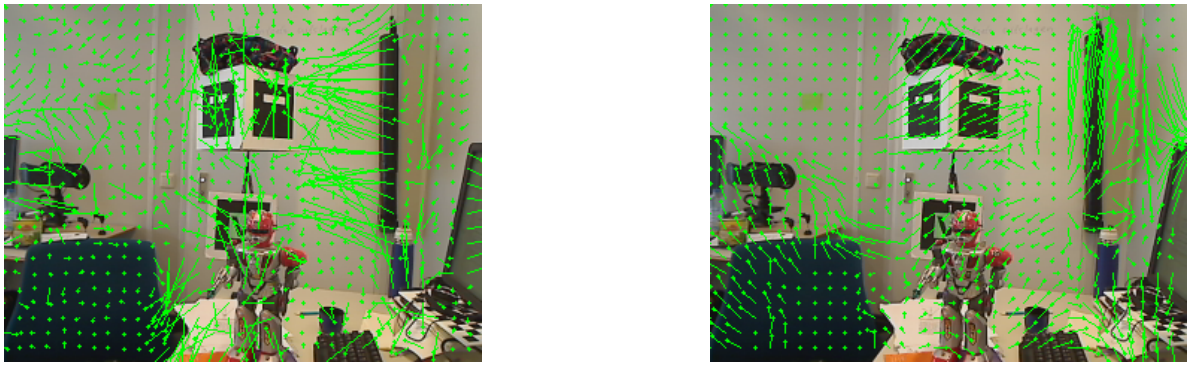
$$\boldsymbol{d} = M^{-1} \cdot \boldsymbol{b} \tag{5.15}$$

where (5.16)

$$\boldsymbol{b} = \begin{bmatrix} \Delta I_k(u,v) I_u \\ \Delta I_k(u,v) I_u \end{bmatrix} \tag{5.17}$$

where $\Delta I_k(u,k)$ is the image difference at iteration k, from the one frame to the other, taken into consideration the guesses of optical flow from the previous level, and previous iteration $k-1$. Thus is done until the optical flow $\boldsymbol{d}$ has reached a certain threshold which is deemed acceptable.

Fig. 5.6 (c), shows the result of the LK method with the features of the Shi-Tomasi corner detection algorithm. Using sparse optical flow increases the ability to segment different moving objects. For visual tracking tasks, sparse optical flow methods is deemed to be more suitable.



(a) (b)

Figure 5.7: Dense optical flow by (a) assuming the smoothness by Horn and Schunck (1981)'s algorithm and (b) Farneback (2003)'s method of using polynomial expansion.

5.2.2. Dense Optical Flow

Another way to detect the optical flow within an image, is not to first calculate the features and selecting them, but to determine the motion directly. Dense optical flow is a direct method which is able to recover motion from every single pixel from the camera, using the intensity variations from frame to frame. There are several assumptions to make for the calculation of optical flow from one frame to the other, and one of them is the brightness constancy assumption. Horn and Schunck (1981) were one of the first to use this as their algorithm, the Horn-Schunck method (HS), deals with the smoothness of the optical flow components. It therefore minimizing the following constraint:

$$\begin{bmatrix} \frac{\partial^2 \Delta u}{\partial u^2} \\ \frac{\partial^2 \Delta v}{\partial v^2} \end{bmatrix} - \frac{1}{\alpha} I_u (I_u \Delta u + I_v \Delta v + I_t) \; = \; \mathbf{0} \tag{5.18}$$

Where α is the smoothness weighing constant, where a higher value leads to a smoother optical flow. Although the HS method holds for most surfaces seen by the camera, features like corners and edges introduces problems to the algorithm as they are no smooth transitions at all. This causes these features to corrupt the optical flow. In Fig. 5.7(a), it can be seen that HS has trouble at the locations with a sharp cut-off of contrast.

One of the most commonly used dense optical flow approximation method that is used in computer vision more recently, is developed by Farneback (2003). The Färneback method (FS) can approximate the neighborhood of each sequencing image frames by a 2nd order polynomial:

$$f_1 = \; (\boldsymbol{x})^T A_1 \boldsymbol{x} + \boldsymbol{b}_1^T + c_1 \tag{5.19}$$

$$f_2 = \; (\boldsymbol{x})^T A_2 \boldsymbol{x} + \boldsymbol{b}_2^T + c_2 \tag{5.20}$$

Where f_1 and f_2 are the signals corresponding to the two frames each, $\boldsymbol{x}$ is a vector of the image coordinates (u, v), and matrices A_1 and A_2, vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ and scalars c_1, c_2 contain the parameters for the polynomial expansion. If a displacement $\boldsymbol{d} = (\Delta u, \Delta v)$ is considered and that $f_1(\boldsymbol{x}) = f_2(\boldsymbol{x} - \boldsymbol{d})$ than the following holds for the translation:

$$\boldsymbol{d} \; = \; -\frac{1}{2} A_1^{-1} (\boldsymbol{b}_2 - \boldsymbol{b}_1) \tag{5.21}$$

However, the above assumptions hold for an ideal situation, where it is expected for the entire global translation to be related to just two signals. Therefore, some approximation are made for practical considerations which leads to the following constraint:

$$A(\boldsymbol{x})\boldsymbol{d}(\boldsymbol{x}) = \boldsymbol{\Delta b}(\boldsymbol{x}) \tag{5.22}$$

where (5.23)

$$A(\boldsymbol{x}) = \frac{1}{2}(A_1(\boldsymbol{x}) + A_2(\boldsymbol{x})) \tag{5.24}$$

$$\Delta \boldsymbol{b}(x) = -\frac{1}{2}(\boldsymbol{b}_2(\boldsymbol{x}) - \boldsymbol{b}_1(\boldsymbol{x})) \tag{5.25}$$

Assuming that the displacement is slowly changing from one frame to the other, this can be integrated over each pixel. For a neighborhood W in the image from coordinates $\boldsymbol{x}$, the following has to be minimized:

$$\sum_{\boldsymbol{\Delta x} \epsilon W} w(\Delta \boldsymbol{x}) ||A(\boldsymbol{x} + \Delta x)\boldsymbol{d}(\boldsymbol{x}) - \Delta b(\boldsymbol{x} + \delta \boldsymbol{x})||^2 \tag{5.26}$$

where $w(\Delta \boldsymbol{x})$ stand for the weight assigned to each point of the neighborhood. In Fig. 5.7(b), the result of the FB optical flow detection can be seen. Here parts with an even contrast, like walls, are not taken into account for the motion detection. However, it does clearly present a better result overall than the HS algorithm. Overall, although dense optical flow gives an overload of flow fields, each small variation can be detected by this method.

5.3. Optical flow detection in the Vision Setup

As said at the beginning of this chapter, selecting an optical flow algorithm comes with a trade-off between repeatability and time. Even within the notion of repeatability, it has to be decided if it is more necessary to have the ability to find a description of feature back a second time or to retrieve the exact location of that interesting point. In order to make a decision on which methods to use for the robotic head's cameras, the algorithms from the previous sections had been evaluated in terms of time and ability to correctly approximate the motion.

At each stage of the feature recognition and the optical flow calculation, the times are registered for each algorithm. This is done with help of the OpenCV library and executed on the same computer were the active binocular robot is controlled from. Using the same video footage from the robotic head on each method, the amount of time to calculate the optical flow is given in table 5.1. The Shi-Tomasi, SIFT and SURF features are used for the LK optical flow approximation. The HS and FB methods however, already use the entire image directly so they do not lose time on feature detection. From these measurements, it has become apparent that, from the algorithms tested, the Shi-Tomasi & LK combo and the FB methods are the two fastest algorithms. They were both able to keep their detection algorithm within the control frequency of 30 hz, which is a requirement in order to make the experiments real time.

All the above mentioned methods have been compared with the markers' speed, which is used for the pursuit control, detected in the same images. During a rotational disturbance of the neck of the binocular setup, the overage horizontal optical flow is monitored, from which the results can be seen in Fig. 5.8. The fact that the optical flow values do not have the same magnitude as the marker, is that it is the average taken over an entire scene. The marker is much nearer to the camera than the rest of the scenery, therefore appears to be moving must faster. To be able to determine a fixed number on the correlation between the marker speed and the average horizontal optical flows, the normalized maximum correlation is calculated, which can be found in the last entry of table 5.1. The closer to one, the higher the correlation between them, and the highest score is achieved for the Färnback method.

However, the differences between these scores are very small, so before making a decision on which optical flow algorithm to use, the LK and FB method are both applied to the active binocular vision setup. In Fig. 5.9(a), some samples of the Shi-Tomasi and LK method in action is shown, where in Fig. 5.9(b), the same is done for the FB method. In the first frame of 5.9(a), a lot of distortion is eminent, which results in high peak noise within the average optical flow signal. Also, the marker itself does not contain as many features as the rest of the scene. In the FB method, the optical flow detected is much smoother of nature, and is also takes the entire marker into account since it is dense.

	Extract [s]	Describe [s]	Opt. Flow [s]	time total [s]	Norm Max Corr.
Shi-Tomasi (LK)	0.0040705	0.01729	0.0015829	0.022943	0.97646
SIFT (LK)	0.039299	0.028454	0.00098833	0.068741	0.97842
SURF (LK)	0.033058	0.061528	0.0039663	0.098552	0.97994
Horn-Schunck (HS)	0	0	0.1361	0.1361	0.96704
Farneback (FB)	0	0	0.022517	0.022517	0.98018

Table 5.1: This table displays the time used by the computer to extract and described the features, the optical flow approximation and the total time for the entire sequence. Also, the normalized maximum correlation of the average horizontal optical flow and the marker speed from Fig. 5.8, is given.

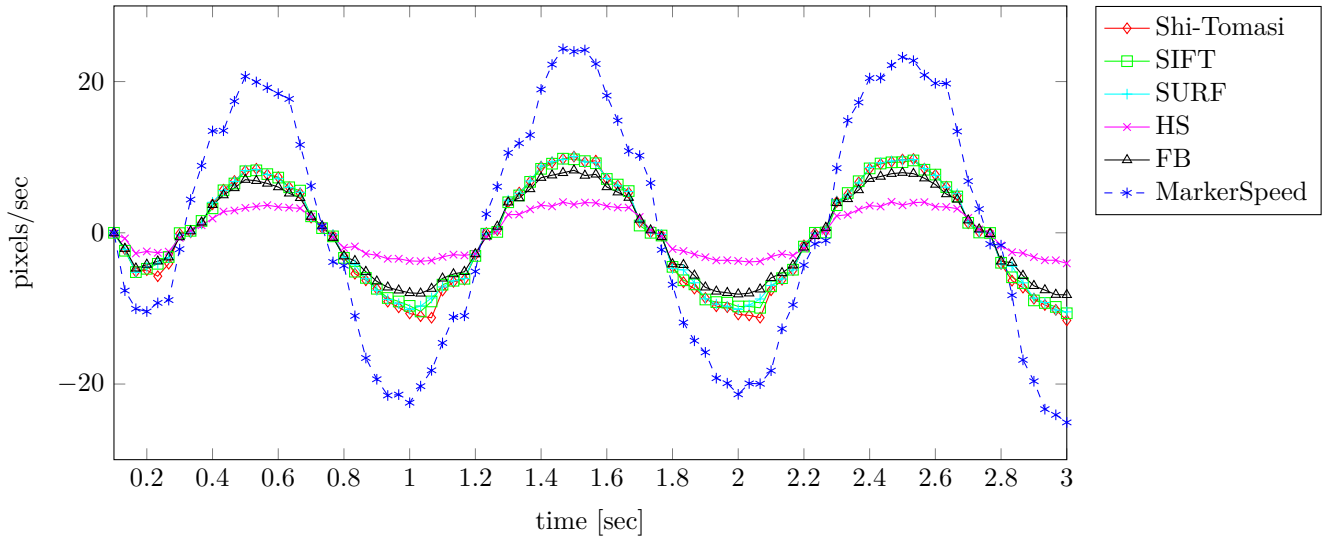
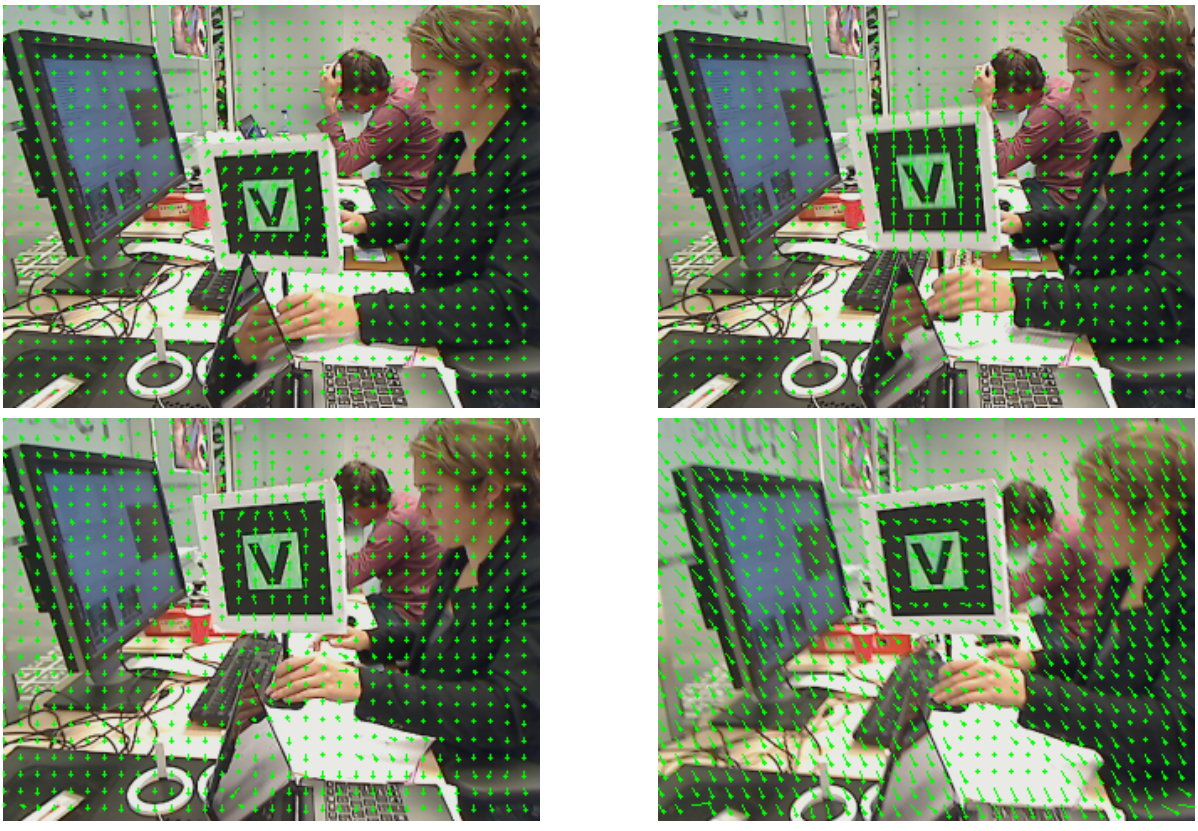


Figure 5.8: This plot gives the average horizontal flow of one set of images, which are made during a rotational disturbance of the neck. This is compared to the marker speed detected in that very same image stream.

In order to make every element of the implementation, hardware and software run at the same time, including the cerebellar model, the optical flow detection could be taking longer than usual. Moreover, since the marker is the object of the pursuit, this is the object to keep stable in its vision. For the final implementation, it becomes necessary to only detect optical flow in the middle, similar to the foveal vision in the human eye. Therefore, a smaller image needs to be processed, which has a bigger effect on the computation speed of the FB than for LK. Considering these aspects of the implementation and the above mentioned advantages of the Färneback method over the others, this is the optical flow approximation used for the active binocular vision system.

(a)



(b)

Figure 5.9: (a) Sparse Optical flow by Lucas et al. (1981) and Bouguet (2001)'s algorithm on the binocular setup. (b) Dense optical flow by Farneback (2003)'s method of using polynomial expansion on the binocular setup.

6

Learning stabilization with the Cerebellar model

The previous chapter explained how to detect motion within an image by optical flow. Before that, the description of the cerebellar-inspired model was given. This chapter explains how to combine these two algorithms together and to test their validity of the adaptive and information filter. The cerebellar models and the optical flow are integrated within an already existing software architecture of an active binocular vision robot. Afterwards, the setup is put up to the test, with a couple of experiments meant for the simulation of a locomoted motion. There are different configurations and situations where both the adaptive and information filter are tested on, conducted in the Delft Biorobotics Lab (DBL). From these, the results from these experiments are given and its observations are discussed in the next chapter.

6.1. Combining Implementations

In chapter 4 explains how the cerebellar model is constructed by means of artificial neural networks. Moreover, chapter 5 shows how to construct a learning signal for that model, by means of the detection of optical flow within the image stream. In Fig. 6.1, a global overview of the current communication between different modules for the experiment is shown, where the cerebellar model and optical flow detection are added. With the pursuit control and marker tracker, these two implementations are fully conducted on the same computer as well. In Appendix D.1, the ROS architecture can be seen, which can be compared with the graph given in section 3.3.3, Fig. 3.8.

To zoom in more specifically on the implementation of the control, the learning module of the cerebellum is placed next to the already existent pursuit and IMU module in the system (Fig. 6.2). The pursuit control follows the marker as seen on its cameras, providing the signal which controls the robot. The IMU detects the rotational speed that the robot head executes, and provides that signal negatively to the robot's neck motor. Finally, the cerebellar model predicts the optical flow with the image stream from the cameras, which is basically the residue movement of both the IMU and pursuit control working together. The combined control signal is as follows:

$$\boldsymbol{\omega}^*_{pursuit} - \boldsymbol{\omega}^{IMU}_{z} - \boldsymbol{\omega}^*_{Cereb} \;=\; \boldsymbol{\omega}^*_{robot} \tag{6.1}$$

where $\boldsymbol{\omega}^*_{pursuit}$ is a vector of goal speeds for pursuit the tracker (see Fig. 3.7 of chapter 3). $\boldsymbol{\omega}^{IMU}_{z}$ is a vector of detected rotation by the IMU, $\boldsymbol{\omega}^*_{Cereb}$ the goal speed to counter the predicted optical flow and $\boldsymbol{\omega}^*_{robot}$ is a vector with the resulting goal speeds for all the motors, to be processed by the 3Mxl modules on the robot for speed control. This combination's purpose is to create a robust method to stabilize the vision of the robotic head, by reducing the average optical flow.

There are a few practical considerations to consider regarding this implementation. Firstly, it is measured that the robotic setup has a dead time of about 100 ms, before it responds on a command from the computer. Given that the optical flow detection takes less than 33 ms, it means that the cerebellar models must detect the optical flow 4 steps in advance, which is about 132 ms in total.

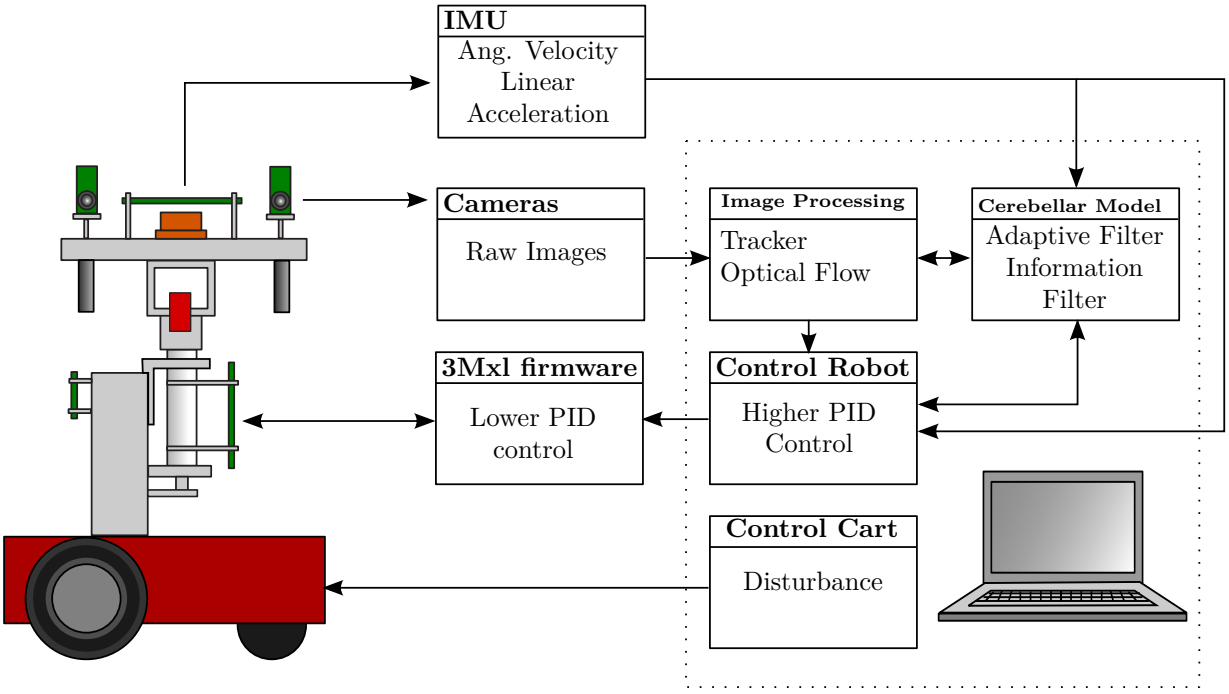


Figure 6.1: The hardware and software of the active binocular setup, with the communication between the higher and lower PID control, the image processing for the tracker and optical flow and the cerebellar model.

Moreover, it was noticed that it was more effective to provide the adaptive filter with the robot's goal speed minus the cerebellar influence ($\omega^*_{robot} - \omega^*_{cereb}$). This was mostly necessary if the robot was learning off-line instead of on-line.

6.2. Experiments

This section gives specific details on the conducted experiments testing the active binocular vision setup and the cerebellar-inspired model, with the accompanied results. As the robot-head must stabilize its vision based on the ego-motion detected by its cameras, disturbances must be excited to simulate the sensation of walking or moving. This is noticeable for the robot through its IMU as well. Moreover, there are different configurations of the information and adaptive filter with various environmental situations, where the setup is experimented on. The stability of the system is verified by amount of reduction of the average optical flow and the motion blur.

There are two types of disturbances, which are going to be administered to the robot, namely a rotational and translational disturbance. In section 1.1.2, a distinction was made between rotational and translational VOR. Although their mechanisms are not the same, the same neurological pathways are used and they are both adapted by the cerebellum. Here with the binocular setup, the rotational disturbance is executed by the neck motor of the robot, therefore decoupling the control of the neck from the pursuit marker system. The linear disturbance is done by a cart where the robot is mounted upon, which moves sideways in a linear fashion.

The step frequency of the human walking cycle is about 1.8 to 2 hz (Pachi and Ji (2005)), which can vary depending on the velocity of his/her pace. The disturbance to simulate the ego-motion for the robot therefore approximately the same frequency. The upward movement can be seen sequences of skips/hops, from which each impact of one of the foots restarts that cycle. Of-course, the exact movement of locomotion is difficult to simulate without mounting the vision system on an actual walking robot. Also, most of the disturbance which can be conducted is mostly rotational and sideways. Therefore, the disturbances are given in a form of a sine, executed on approximately halve of the step frequency of the human walking cycle. Here it is hoped, that the same speeds as locomotion can be achieved.

This section displays all the results for both the adaptive as information filter. First the plots and values for the rotational disturbance are displayed, in terms of reducing the optical flow and motion

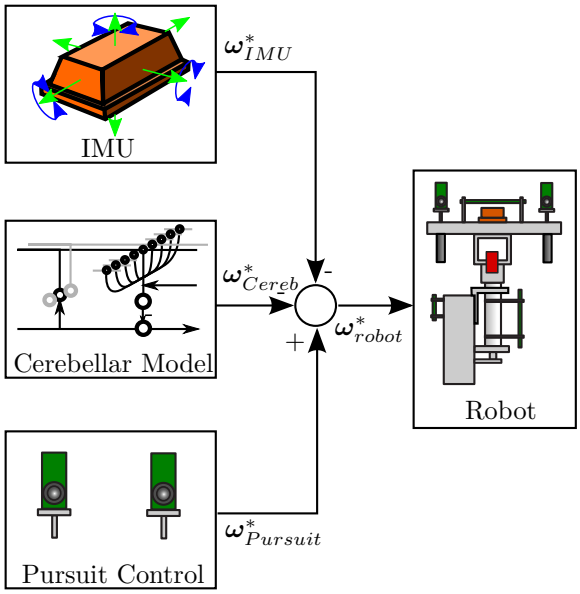


Figure 6.2: A representation of the three types of control combined together for the final goal speed for the robot.

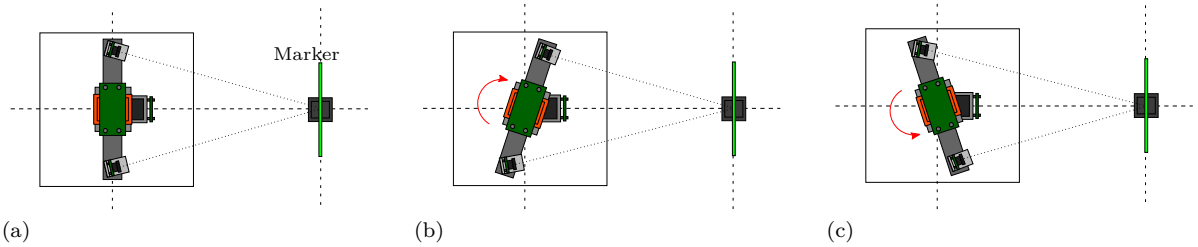


Figure 6.3: The procedure of the rotational neck disturbance, with the marker located at a distance of 650 mm from the center of the robotic setup.

blur within the image stream. The same items are evaluated for the sideways disturbance with a cart. In order to come even closer to a walking-like motion, both of the disturbances initiated at the same time, from which the reduction of the optical flow is noticeable in both horizontal and vertical direction. Finally, an experiment is shown where both the cerebellar-inspired models are learning on-line. Sequence examples of the first two experiments can be found in Appendix D.7 and the learned neuron weights of the first layer of the cerebellar models in Appendix D.8.

6.2.1. Rotational Neck Disturbance

For the first experiment, the ego-motion was initiated by the neck of the robotic setup, where several frequencies of rotational disturbances were given (Fig. 6.3). This started at a low frequency of 0.5 hz and ended at 1.5 hz, with an amplitude of 0.1 rad, for which each level lasted 5 seconds (150 data points). In Fig. 6.4, one can see that the first data was collected for a sequence of 60 seconds (which are 60*30 hz=1800 data points). After collecting data from the first sequence, the cerebellar model has time to train and adapt its weights based on this set.

After the training was finished, which took between 5.5 and 6.0 seconds, the neural network is put to the test for another 60 seconds, which means that it only receives its required input, but its outputs not given to the control of the setup and only being monitored. This gives the possibility of a check for the ability of the model to predict the future optical flow of the system. In table 6.1(a), the mean squared error (MSE) of the prediction is shown for both the adaptive as information filter. In this experiment, the adaptive filter was better able to approximate the future optical flow than the information filter.

In the *execute* mode, the output of either the adaptive or information filter were given to the robotic head, which shows an effect in its performance. In table 6.1(b), both the change in optical flow and

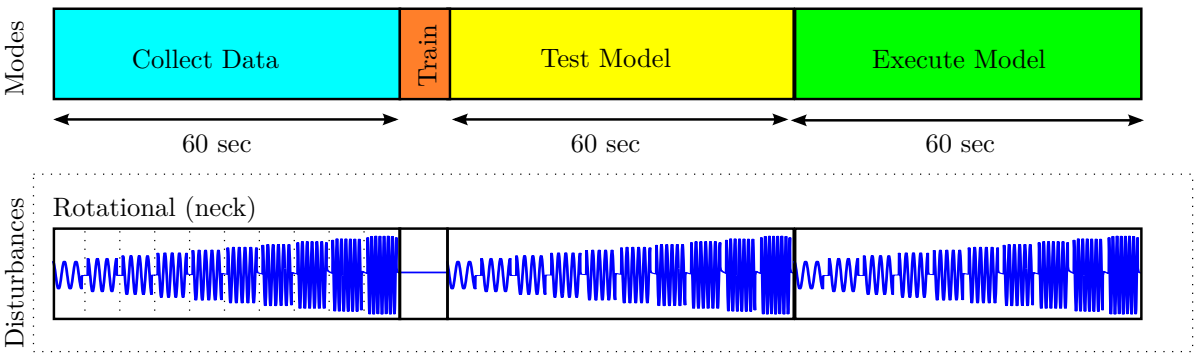


Figure 6.4: The experiments modes of the rotational neck disturbance, with a frequency which increases from 0.5 hz to 1.4 hz.

	MSE prediction
Adaptive Filter	0.019737
Information Filter	0.029885

(a)

	Opt. Flow	**vLAP**
Adaptive Filter	0.13188	1723.0168
Information Filter	0.14812	1631.4649
Pursuit Only	0.19965	1384.9925
Pursuit + Opt. Flow	0.16593	1551.2069

(b)

Table 6.1: (a) The MSE error of the cerebellar-inspired model's output and the optical flow 4 steps ahead and (b) the average optical flow and the variance of Laplacian (VLAP) from both the adaptive and information filter, the pursuit control only and the pursuit with the optical flow. The average is taken from both cameras for this values, for a rotational neck disturbance.

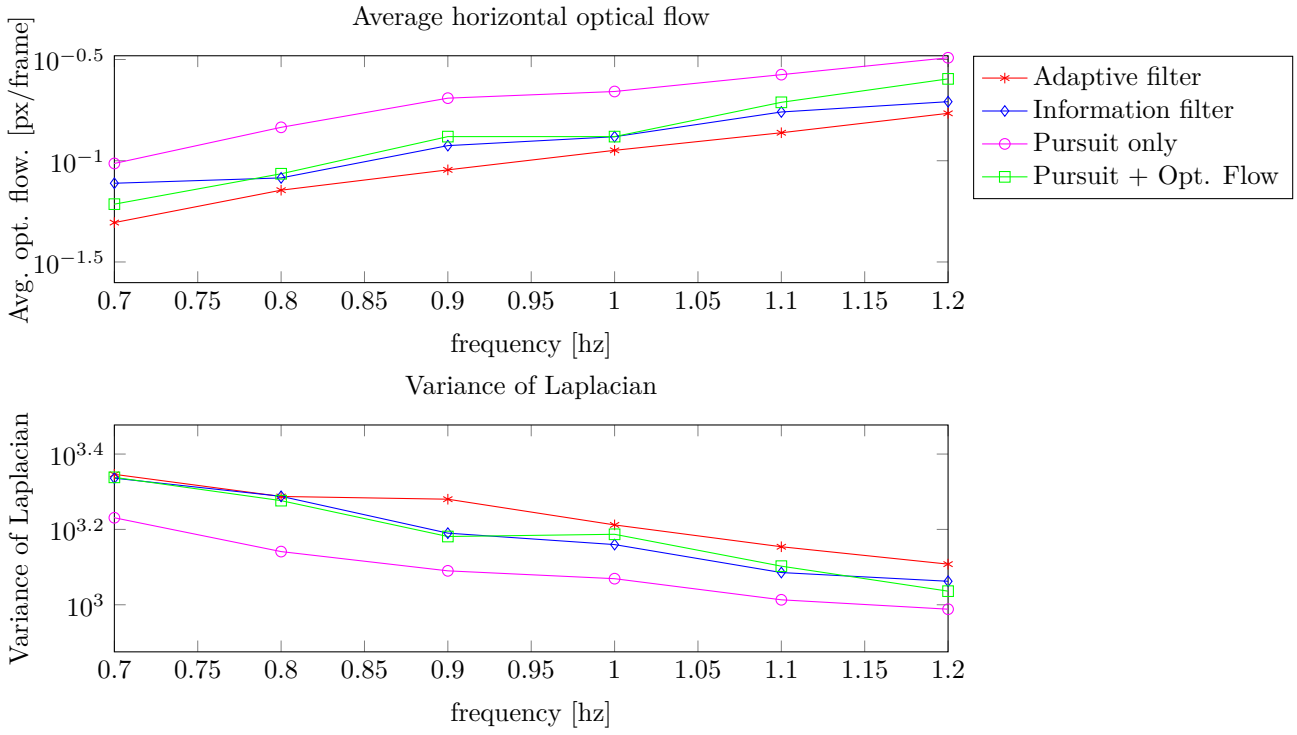


Figure 6.5: The average optical flow per frequency, with and without the adaptive filter and information filter, given that the disturbance is rotational.

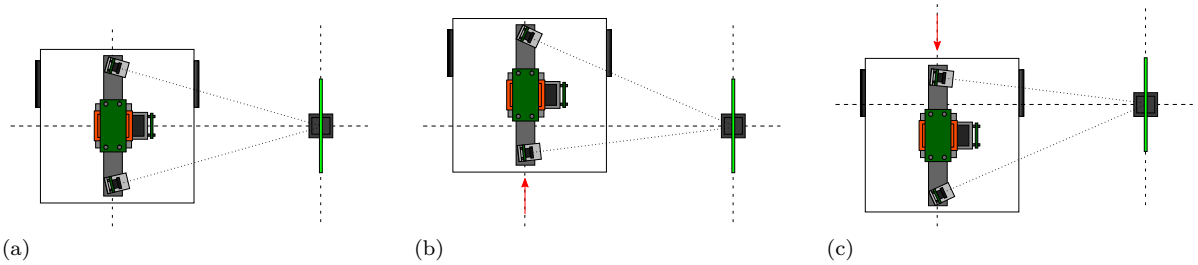


Figure 6.6: The procedure of the linear disturbance, executed by the cart.

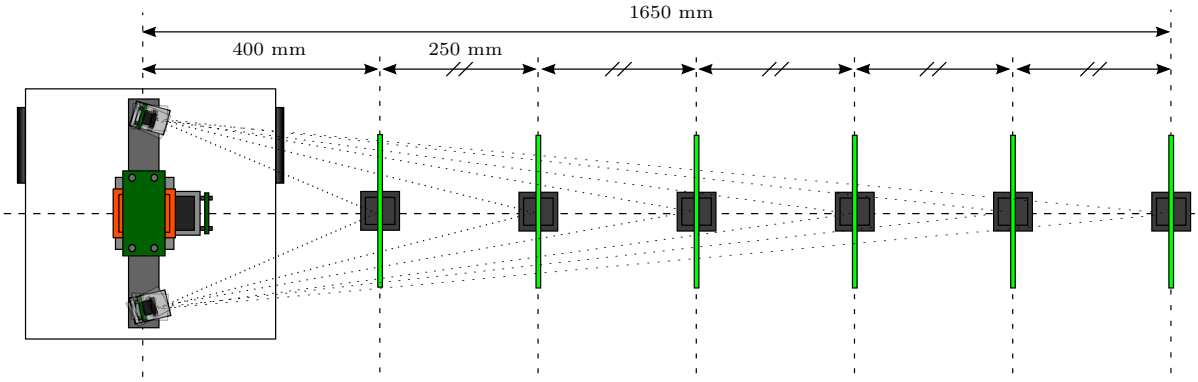


Figure 6.7: The locations of the marker during the linear cart disturbance given in Fig. 6.6

motion blur are given for both filters. The optical flow is the horizontal average optical flow taken for both cameras, using the Färneback method (FB) from Farneback (2003) and motion blur detection, using the variance of the Laplacian (VLAP) from the paper of Pech-Pacheco et al. (2000). The latter is a method which is able to gives a measure on the amount of focus within the image. The higher the value, the 'sharper' the measured image and visa versa. Here, VLAP is a measure of motion blur in the image, however, it is not an absolute measure. This means that it can be used for comparing images from the same scenery with different focus values. Since all the experiments are done within the same configurations and setup, I assume that the compared image streams are similar in content.

In table 6.1(b), the results of adaptive filter and information filter, in terms of average optical flow and VLAP, are compared with the situation with only the pursuit control and using only optical flow. From the entire sequence the average was taken for both values , and it can be seen that both the adaptive and information filter do perform better than their control groups. In Fig. 6.5, their performances are separated per frequency sequence. Overall, it can be seen that as the frequency increases, the optical flow and motion blur are increasing as well. If the pursuit tracker is handling the disturbance alone, it has the worse performance of all three control modes. Using only optical flow and pursuit control does improve the vision stabilization, however the lowest optical flow and motion blur is achieved by the adaptive filter, followed by the information filter.

6.2.2. Linear Disturbance with Cart

The linear disturbance is done by a cart where the robot is mounted upon, which moves sideways in a linear fashion (see Fig. 6.6). The cart does not receive different frequencies as disturbance as in the last experiment, but just one frequency of 1 hz with an amplitude of 0.025 hz. The closer the marker is to the setup, the larger the compensating signal must be to handle for the motion in the image. Also, there is a non-linear connection to the optical flow detected and what the motor must compensate for during the experiments. Therefore, the variation within the experiment for the linear disturbance is based on the position of the marker (see Fig. 6.7). The variation in marker position was handled by an industrial robotic arm from *Universal Robots*. See Appendix D.2 for photos of the experimental setup.

This linear cart disturbance experiment is on many parts similar to the previous one, except for less variation oppose to the amount of frequencies executed by the neck (6 positions opposed to 10

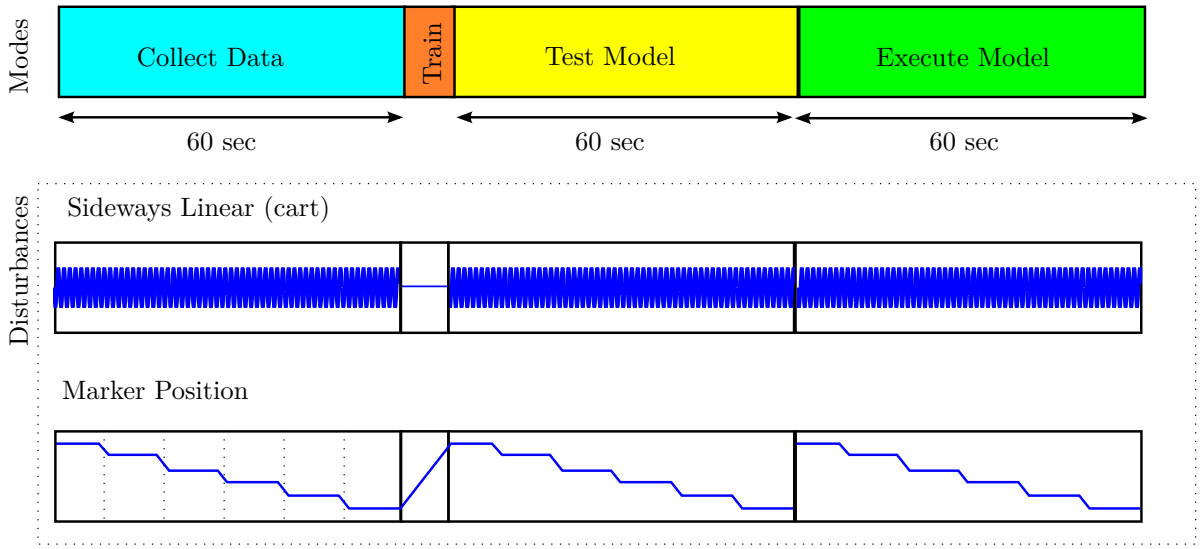


Figure 6.8: The experiments modes of the rotational neck disturbance, with the marking moving from 6 different positions.

	MSE prediction
Adaptive Filter	0.0056259
Information Filter	0.0087144

(a)

	Opt. Flow	**vLAP**
Adaptive Filter	0.093479	2073.1622
Information Filter	0.088954	2043.8414
Pursuit Only	0.16991	1383.3535
Pursuit + Opt. Flow	0.097778	1977.4322

(b)

Table 6.2: (a) The MSE error of the cerebellar-inspired model's output and the optical flow 4 steps ahead and (b) the average optical flow and the variance of Laplacian (VLAP) from both the adaptive and information filter, the pursuit control only and the pursuit with the optical flow. The average is taken from both cameras for these values, for a translational cart disturbance.

frequencies). The markers are held on each position for 8 seconds (8 x 30 hz = 240 data points per marker) and is moved to another within 2 seconds. However, each collect, test and execute mode still lasts 60 seconds (see Fig. 6.8).

In table 6.2(a), the MSE of the prediction performance of both the adaptive and information filter are seen for the test sequence. Here it can be seen that the information filter had more trouble of exactly predicting the future optical flow than the adaptive filter. Also, both MSE are higher for the linear disturbance compared to the rotational disturbance in table 6.1.

In table 6.2(b), the results from the *execute* mode is displayed, where again the adaptive and the information filter performs better than the pursuit control only or the pursuit with optical flow. Interesting to see, is that in terms of optical flow, the information filter is better able to stabilize the vision system, as it is able to reduce the optical flow overall than the adaptive filter. However, in terms of motion blur, it performance slightly lower that the adaptive filter.

Plot are been made on these performances, where the sequences of various marker locations has been separated as well, which can be found in Fig. 6.9. In terms of optical flow reduction, it can be seen that it is difficult to see any real difference between the optical flow only and the cerebellar control, however still a clear difference comparing it to the pursuit only control. Looking at the VLAP, the motion blur over the entire image is approximately the same no matter where the marker stands, which is as expected.

6.2.3. Two-Dimensional Disturbance

With the previous experiments, only the two camera actuators have been used. They are only able to make pan rotations, which makes the stabilization problem quite one dimensional. In order to further validate the cerebellar models, the tilt motor can be exited as well, which means that the detected

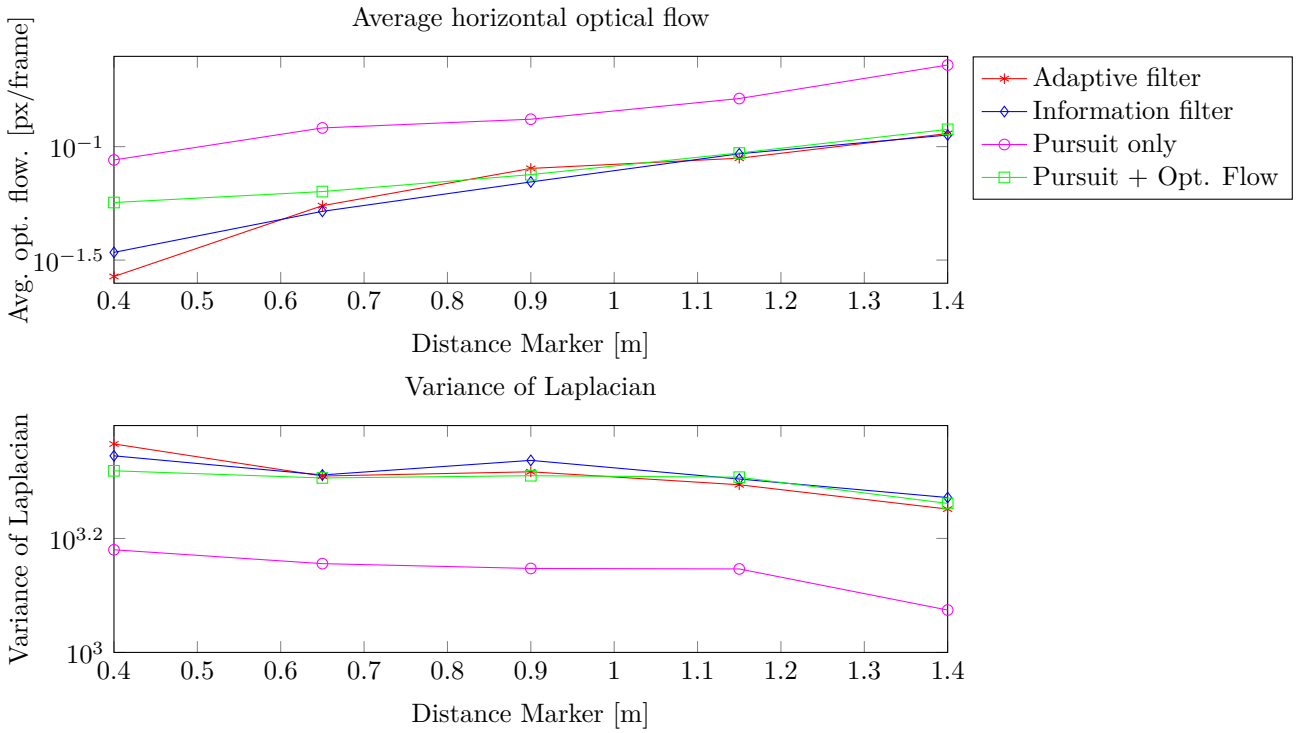


Figure 6.9: The average optical flow and the average variance of the Laplacian per frequency, with and without the adaptive filter and information filter, given that the disturbance is rotational.

ego-motion also has to have a vertical element. In Fig. 6.10, a simple solution is presented, where the cart is put on a slope and moved forth and back. In the mean time, the neck disturbance is initiated as well. Both the cart and neck motors have a frequency of 1 hz, and the neck has an amplitude of 0.75 rad and the cart still has an amplitude of 0.025 meters. This results in type of ego-motion more closely to locomotion. This causes optical flow to also occur in the vertical direction, giving the cerebellar-inspired models a chance to work with the tilt motor as well.

Similar to the rotational and sideways disturbance, this experiment is also conducted with three experiment modes as seen in Fig. 6.11. Since there is no variation in disturbance here, the sequence time per mode is lowered to 30 seconds. First the experiment is performed with only the cerebellar models initialized in the tilt motor (and not running the pan disturbance in the neck). The results of optical flow and VLAP is seen in table 6.3(a), and it can be seen that again that the adaptive filter is better in reducing both values. The information filter is the runner-up, since it is still better stabilizing the vision system that the pursuit tracker alone.

In table 6.3(b), both the cerebellar models and the neck motor are initiated, so the optical flow can be compensated in 2 dimensions in the image. The horizontal optical flow is definitely higher than the vertical optical flow, indicating that the ramp has to be even more steep in order to enlarge that value. Unfortunately, this is where the limit of the experimental equipment lies, however, the image feed from the cameras do show a very similar disturbance which can be related to actual walking. Here the information filter is performing slightly better in compensating the horizontal element of the optical flow but react poorly for the vertical optical flow, even worse than the pursuit only control mode. However, this would indicate that this experiment has to be redone as a second opinion. The adaptive filter is able to compensate for both the horizontal and vertical optical flow, therefore also scores the highest in reducing the overall motion blur in the image stream.

6.2.4. On-line Learning

Finally, since the previous test are only done with off-line learning, the adaptive and information filter are tested on performance with on-line learning as well. Here, both the adaptive and information filter are split in two: one module for data collecting and learning, and one module for the execution of the compensatory signal (Fig. 6.13) The *execute* model makes use of the newly learned weights as soon as the

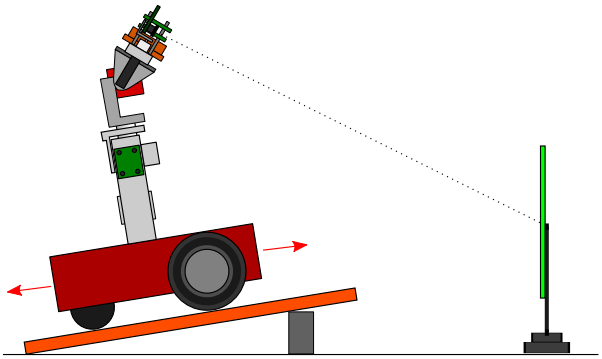


Figure 6.10: The procedure of the 2D vision disturbance, with the cart riding back and forth upon a slope.

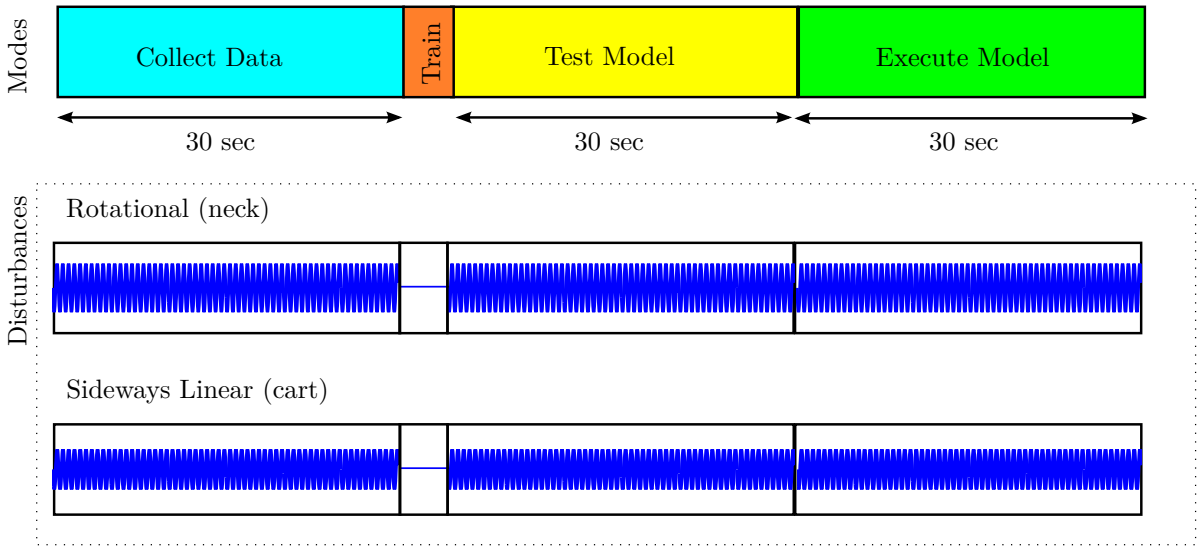


Figure 6.11: The experiments modes of the 2D vision disturbance, with both the cart and the neck giving a disturbance at the same time.

	Opt. Flow (vert)	**vLAP**
Adaptive Filter	0.058708	1637.0309
Information Filter	0.0793	1526.9197
Pursuit Only	0.088916	1399.51

(a)

	Opt. Flow (hor)	**Opt. Flow (vert)**	**vLAP**
Adaptive Filter	0.16027	0.066531	1158.9488
Information Filter	0.14875	0.13261	895.6172
Pursuit Only	0.28843	0.091449	755.2416

(b)

Table 6.3: (a) The MSE error of the cerebellar-inspired model's output and the optical flow 4 steps ahead and (b) the average optical flow and the variance of Laplacian (VLAP) from both the adaptive and information filter, the pursuit control only and the pursuit with the optical flow. The average is taken from both cameras for these values, for a translational cart disturbance and rotational neck disturbance at the same time

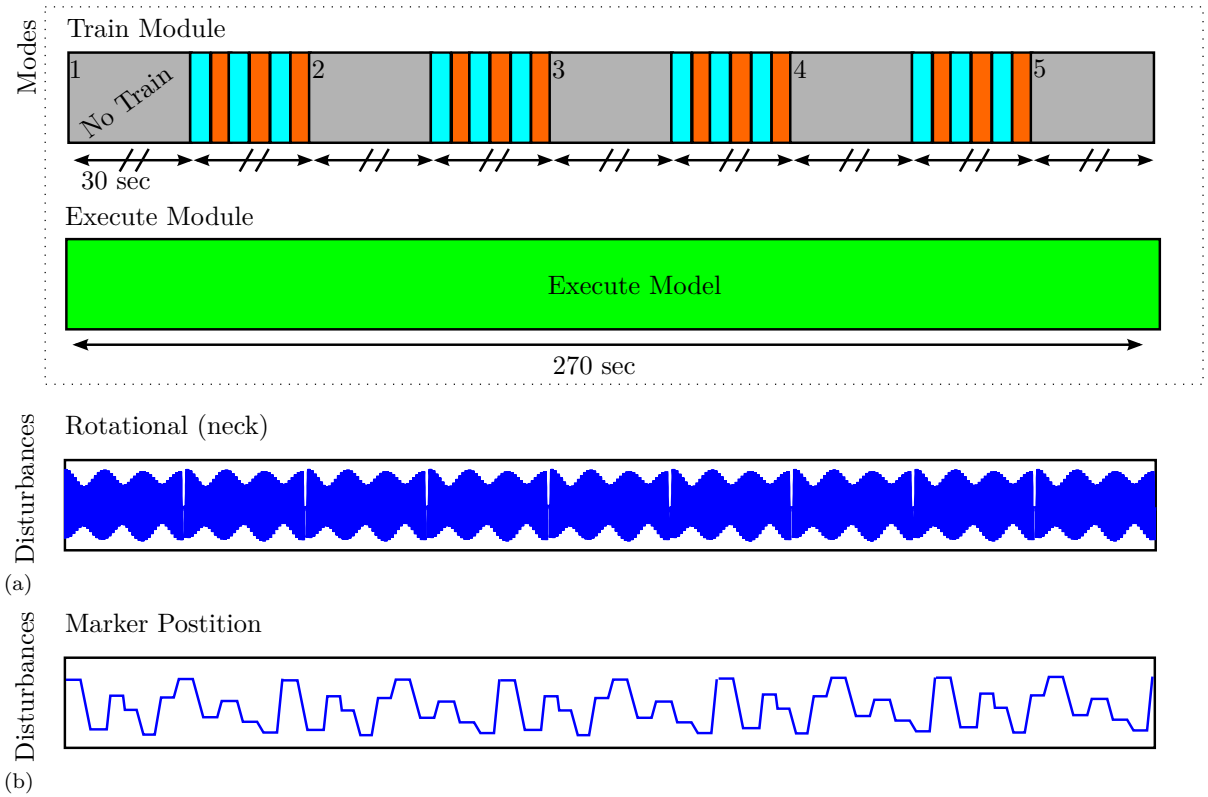


Figure 6.12: Here are experiment modes of the on-line learning experiments, where the training and execute module are taken apart and run parallel from each other. The training module collects and trains data, while the *execute* module uses those adapted weights. The *training* module initiates no-train sequences, where it does not adapt the weights at all, where the learned cerebellar model can be executed without it influence. This training modes are conducted for (a) a rotational neck disturbance and (b) a rotational cart disturbance.

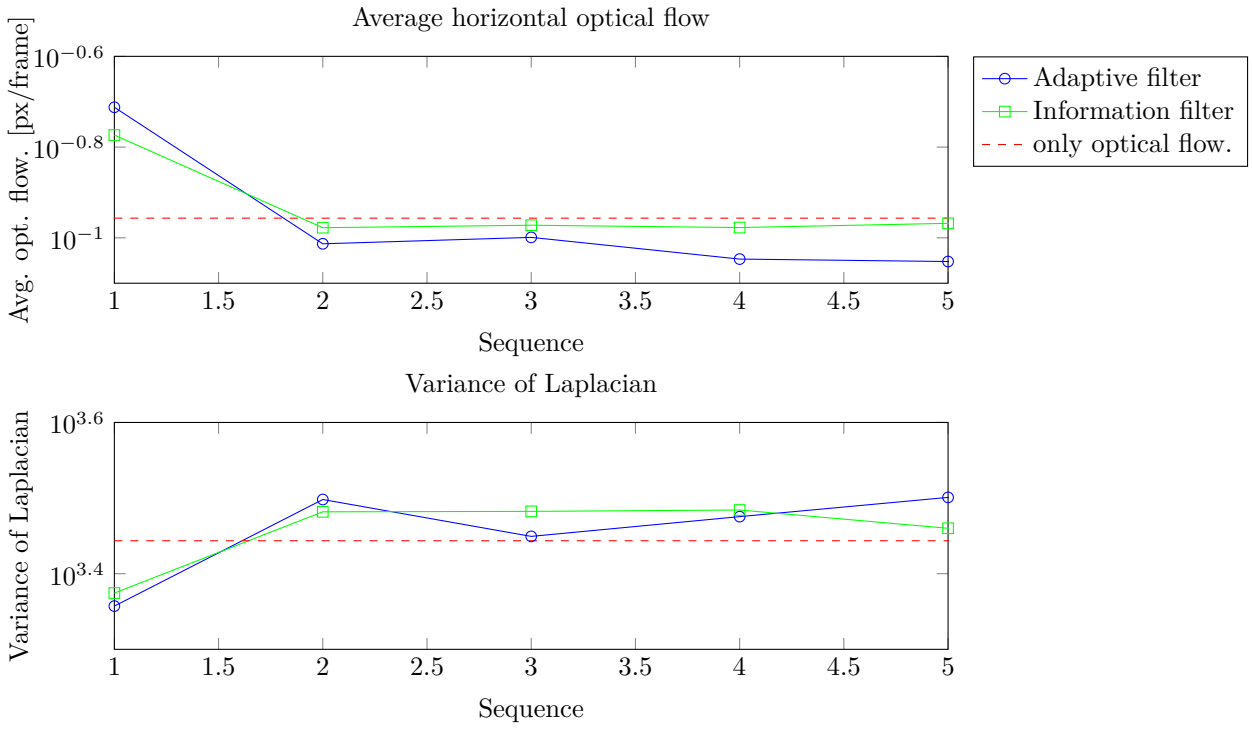


Figure 6.13: The average optical flow and the average variance of the Laplacian per frequency, with and without the adaptive filter and information filter, given that the disturbance comes from the neck. This is done with on-line learning

training module is finished learning a data set. They both run simultaneously and the training module initiates some pauses every 30 seconds. This is done to be able to correctly evaluate the learned model in those no-train periods, where the performance would not be influenced by the constant changing of cerebellar models' weights. This on-line learning experiment is done for both rotational and linear disturbance separately, which are given to the robotic setup with a random variation.

For the random rotational disturbance, which is a multiple of three kinds of sine signals (Fig. 6.12(b)), it was found that it was enough for the adaptive filter to learn with a small data-sets of 31 points. For the information filter, it was enough to learn with just 15 points. However, to compare both filters equally with each other, both data-sets had the same size of 31 data-points. In Fig. 6.13, the average horizontal optical flow and the average VLAP from the training pause sequences is seen in the plot. The on-line learning performance is held against situation where only optical flow and the pursuit control was used, with the same disturbances. In the plot, one can be seen that, that they were able to cross the red dotted line and converged to a better value.

For the linear disturbance, the marker was moved to another random position every 2 seconds. This time, both models were put on a training set size of 151 data points and 151 epochs, as that was the lowest the adaptive filter could handle in terms of on-line learning. In Fig. 6.14, one can see that the adaptive filter is actually creating more optical flow at the second train pause sequence than the pursuit control by itself at the first sequence. The information filter is able to converge faster to an optimal solution, which gives a better performance than the pursuit control and the optical flow alone. The adaptive filter is not able to cross that line until sequence 3. The information filter, however, crosses the comparison line back up again as sequence 4. This same behavior can be monitored for both the average optical flow as for the motion blur plot.

6.2.5. Extra Results

In section 4.3.1, some choices were explained during the built of the cerebellar models. Some discussion was conducted, regarding the input of the adaptive filter, the ability of the models to handle a random disturbance and the amount of layers of the ANN. These variations to the model has been experimented on as well. In appendix D.3, the difference in stabilization performance is shown, if the adaptive filter's inputs was the actual voltage to motors by the 3Mxl modules, instead of the goal speed given to them.

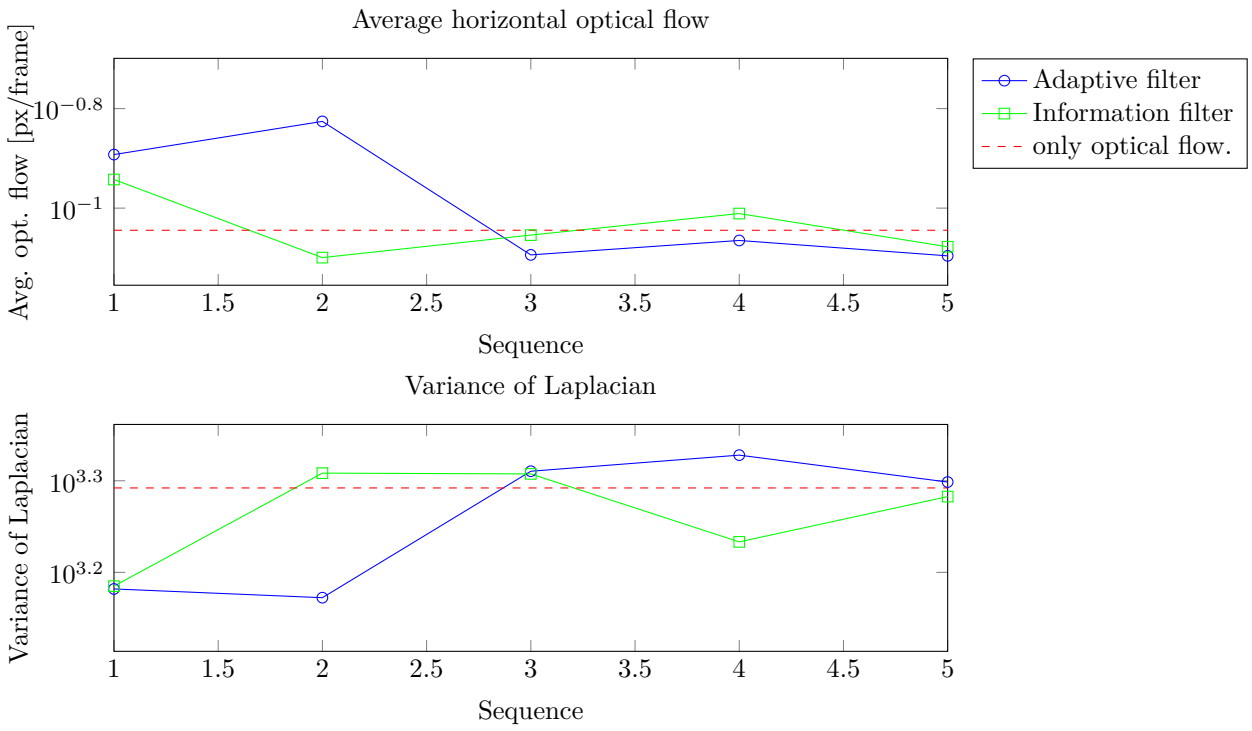


Figure 6.14: The average optical flow and the average variance of the Laplacian per frequency, with and without the adaptive filter and information filter, given that the disturbance comes from the cart. This is done with on-line learning.

In appendix D.4, both adaptive and information filter have been exposed to a random disturbance, to see if they still were able to reduce the optical flow in the image. Appendix D.5 shows a comparison to both the cerebellar models, with two or three layers and to both a rotational and linear perturbation.

To also accommodate the combination of the OKR/VOR mentioned in the introduction, the robotic setup was also tested with a control scheme using only the detected optical flow and or only the detected rotation by the IMU. Also a ratio was set between those two values, determining each influence on the systems control. The results can be found in Appendix D.6 and is done for various frequencies, similar to the rotational neck disturbance experiment.

6.3. Summary Observations Results

In the previous section, the results of the experiments were given, from which the observations are summarized in this section. First of all, with the linear disturbance and off-line learning, it is seen that the both the information filter and adaptive filter are able to better stabilize the vision system, by reducing the optical flow in the image compared with the tracker pursuit with and without direct optical flow compensation. From the two, the adaptive filter exceeded the information filter's performance, and the same trend can also be seen in the MSE of the prediction. Also, with the translational disturbance, the adaptive filter is better in predicting the up following optical flow, however, in terms of optical flow reduction, it performance is slightly worse that the information filter. The same trend could not be spotted with the VLAP, as that the adaptive filter is able to reduce the motion blur more.

With the 2D disturbance, where only one frequency is used, the cerebellar model had been applied to tilt motor alone, and all three motors. Using just the tilt motor, both the information filter and adaptive filter are able to reduce the vertical average optical flow as the motion blur within the image, compared to the pursuit tracker alone. The adaptive filter has again the best result. When the cerebellar-model is used on all three motors, the information filter's effect is able to reduce the horizontal optical flow more than the adaptive filter, however the performance for the tilt motor has dramatically decreased. Therefore, the total motion blur of the implementation of the adaptive filter is a lot less than the other. This indicates for a redemption of the experiment for future recommendations.

Finally, a form of on-line learning was performed, to see how the cerebellar-inspired models would

respond to that, using again both types of disturbance used in the first two experiments. This was possible by separating the training and execution functionalities. The train module collected the data and trained with it, and it was noticeable that for the information filter, fewer data points per training set was needed than with the adaptive filter, in order to enable it to learn. Looking at the rotational disturbance, the adaptive filter converged faster and ended up with a better result than the information filter, but they both were able to cross the red doted comparison line. This indicated that the prediction capability of the cerebellum model does reach a better performance eventually, than if the optical flow was used on the setup's control directly. At the linear disturbance, the faster convergence was again achieved by the information filter. Here both implementations of the filters where able to cross the optical flow-only border at one point. However, it was significantly slower than with the rotational disturbance.

7

Discussion, Conclusion and Future Perspectives

In the previous chapter, the experiments and results of an active-binocular vision system with a cerebellum inspired model were presented. This chapter reflects on those results and more importantly, on the process that lead to them. First, each step of the research is discussed, where some choices and considerations are evaluated. Afterwards, the research objectives given in the introduction are reflected on, to conclude if they have been achieved within this research and in what manner. From these, some recommendations and future perspectives are presented, with ideas to improve the performance and further enhance cerebellar-inspired control for robots.

7.1. Discussion

To briefly summarize the previous chapter in this thesis: First human eye movements and the cerebellum had been investigated. An active binocular vision setup is used to implement two cerebellum-inspired models, to stabilize its vision. The models are built using artificial neural networks, which learns from the visual motion detected by optical flow. To validate the implementation, experiments are conducted to simulate various disturbances. This section reflects on the steps taken within this thesis' research, to determine on how well they were executed in order to obtain the research objectives.

7.1.1. Cerebellar Learning Models

The choice to implement the adaptive filter (from Dean and Porrill (2011)'s work) and its variation, the information filter, into our robotic head setup, is explained in this subsection. For inspiration on how the human eye compensates itself for external disturbance, I looked into the physiology of the cerebellum. In neuroscientific research, this brain area has been thoroughly investigated by studying its influence on compensatory eye movements. It is interesting to find out that it did not only have an effect on motor control within the human body, but with cognitive tasks as well. Within the cerebellum's anatomy, it is discovered that the cerebellum has a large amount of connections with the rest of the central nervous system, which underpins the notion that it might have an important role in the brain. Its structure is very distinguishable from other brain areas, which enables massive parallel forward processing, therefore fast computation.

What does the physiology of the cerebellum mean for its learning capabilities? Several studies, presented in section 2.2.1 indicate that the climbing fiber has a big role in a supervised learning process. The signal it carries, can either be linked to a prediction error, retinal slip or a synchronization error. However, some researchers have discussed that the climbing fiber does not initiate any synaptic weight adaption at all. Electro-physiology studies are still deciphering, what the climbing fiber is *not* encoding to the cerebellum, instead of what it *could* be. Since it is still unsure what type of information drives the cerebellar learning process, I chose to go with the theory that the final implemented model came with. With the adaptive filter, the teaching signal was presumed to be the retinal slip.

From neuroscientific research, various functionalities have been given to the cerebellum in the scheme of human movement control, which have been described in section 2.2.2. Recently many researchers

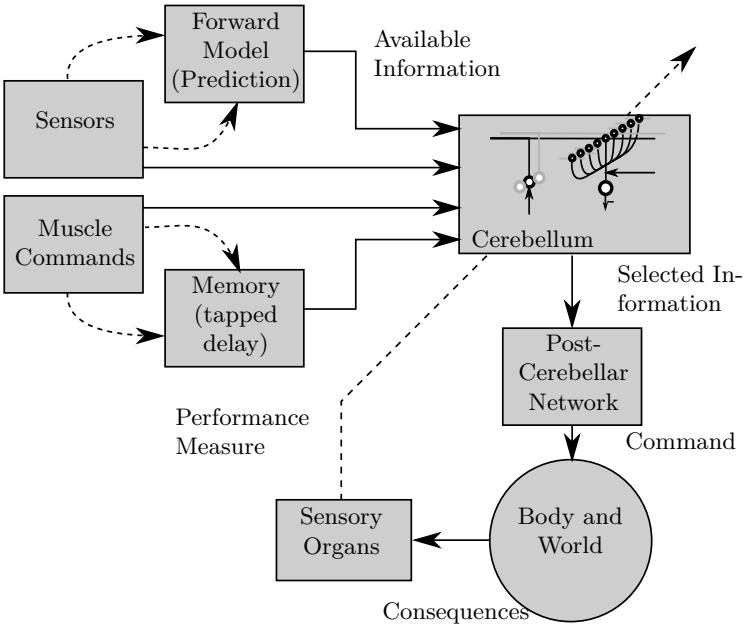


Figure 7.1: My perspective on the cerebellar function. The functionalities of a forward model, tapped delay line, or others, is believed to exist outside of the cerebellum. It will receive the systems output, where it selects the information needed to perform a certain task. With this information, the rest of the brain will create a command signal to the muscles. This creates an interaction with the human body with the outside world, which will result in a positive or negative consequence. This performance measure is fed back to the cerebellum, guiding its synaptic plasticity.

support the internal/forward model role of the cerebellum. However, most of these models assume that it receives only an efferent copy of the muscle commands, in order to predict a sensory outcome for an improved state estimation. The adaptive filter consist of a tapped delay line, where it is assumed that its input are just muscle commands. Looking at the anatomy of the cerebellum, this contradicting as it receives massive amounts of information from the neocortex, vestibular organ, eyes, spinal cord etc. This is one of the conflicts that I came across during my research, which would finally be an important reason why I have proposed the information filter.

However, this proposal is not the end of the discussion of whether the learning models of the cerebellum are close to their biological origins. In section 2.2.1, the synchronization regulator role of the cerebellum was put up for discussion, as Molinari et al. (2007) had found out that cerebellar damaged patients did preserve some timing control in rhythmic based task. They say that the cerebellum does have an influence on it, but does not need to be the one and only position where timing synchronization takes place. To extend this thought to the workings of the adaptive filter, does the adaptive filter's tapped delay line exactly takes place within granule cell layer of the cerebellum? Although in the research of Lenz et al. (2009) its tapped delay goes back for 200 steps, this large amount seems very unnecessary. To determine the dynamics of an action, knowledge about just a few would be adequate, which already will represents values as velocity, acceleration, jerk and other derivatives. This is of course dependable on what kind of movement and task must be determined.

The cerebellum has a bigger role than the functionalities described in section 2.2.2. My current perspective on its role, is that prediction, corrective motor commands, synchronization, storing past sensory signals or motor commands are performed elsewhere in the central nervous system. The cerebellum receives the output of those systems, plus all the somato-sensory signals and muscle commands. From this massive amount of signals, it will be able to select which information is useful to perform a certain task (see Fig. 7.1). This process can be monitored by the climbing fibers input for instance. This is a notion that fits with four principles explained in the conclusion of section 2.1.2, especially in the terms of feed forward processing and modularity. However, with this perspective, it is difficult to implement a cerebellar model, without simulating its connections with the other brain areas as well.

For research conducted for this thesis, I chose to go ahead with implementing the adaptive filter. Although the above mentioned perspectives have started to subvert the use of the model, the main

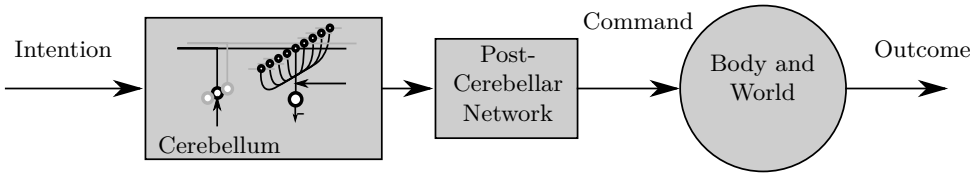


Figure 7.2: A representation of the distal supervised learning problem as Jordan and Rumelhart (1992) describes. With supervised learning, it is necessary to have a training set of input and a direct desired output. However, in the case of the cerebellum, it will not know what its desired output is, after it has been used by the body the external world.

objective is to implement a cerebellar model into a robot, to improve the stability of its vision. Due to the time limitation, reinventing the cerebellar learning model and to make it work on the active binocular setup is unwise. The rest of this section will therefore discuss topics according to the main objective. To accommodate my perspective of the cerebellar function, the information filter had been proposed. This is to shed some light on whether the robotic framework has a need for signal selection. This will be discussed in the subsection on results.

7.1.2. The Active Binocular Vision Setup

For this research, an active binocular vision setup was provided as a platform, to simulate a cerebellar-inspired control for the stabilization of its cameras. As it contained motors for both the cameras, tilt and pan rotations, it is able to recreate many of the human eye movements. However, recreating a movement is not the same as responding at actual eye movements. Giving these motors an impulse will result in a different response than giving a similar impulse to one of the ocular motor muscles. In the study of Lenz et al. (2009), their setup was not driven by electro motors but by McKibben's pneumatic artificial muscles, which are more similar to the way that organic muscles actuate a part of the body[1]The question is, if the current setup had these as actuators, would it have improved the result? It would seem unlikely that it would, as these special actuators are difficult to control as they are by just on-off systems. Moreover, the cerebellum, or the human brain overall, is able to handle all kinds of body types. Every human contains different characteristics for their muscles and joints, which all respond in different ways. If the principle of adaptivity of the human body is incorporated in a robot correctly, it would be able to learn with electro motors as well.

Zooming in more on the current functionalities of the binocular robotic head: many of its features are run from the computer. Only some rudimentary parts of robot are done by embedded modules. However, those 3Mxl modules are capable of more. Next to position and speed control, they can also be used for generating sine motions. If the signal for a goal speed is sent to the robot by a USB cable, some commands can get lost, therefore the movement will not be as smooth as required. Also, communication by a USB cable is a big reason for the delay in sending the goal speeds to the setup. This embedded generation of sine movements could have a big benefit for the initiation of disturbance of the experiments and/or the pursuit control by itself. However, since this knowledge was only available at the end of this thesis' research, it could not have been used to its potential, therefore given as a future recommendation

In the research of Lenz et al. (2009), a custom made chipboard is used to control the adaptive filter and their robot, therefor achieving high control frequency, performance and less delay. As said earlier, one of the fundamentals of cerebellar processing, is the magnitude of the feed-forward parallel processing. The computer used for my research is not able to parallel process the 21 outputs of the cerebellar on its processor, as it only contains 8 cores. This is necessary to truly simulate the cerebellum's processing. This could have not been exploited on in this research's process, but is a pointer to future work.

7.1.3. Simulation of the Cerebellum by Artificial Neural Networks

In this thesis, the cerebellum had been simulated by an artificial neural network, using resilient back-propagation and an exact input-output training set. It can be argued to what degree this happens biologically. It seems unlikely that the human brain has an exact input and output ready for the cerebellum to learn which it propagates back to its synaptic weights. The plasticity could be ignited

[1]However, one McKibben's muscle represents just once muscle cell. Our muscles contains thousands of those cells, where are connected which each other in series and parallel, together sharing the load for contraction.

by an indirect performance measure. Within the human body, there are many links in completing a certain task, however it will not be known if it was performed correctly until the entire task had been complete. This issue is called the distal supervised learning problem (Fig. 7.2) as formulated by Jordan and Rumelhart (1992), which requires some other non-conventional learning architecture.

In section 4.2.3, some alternative training algorithms have been discussed. Comparing it to the cerebellum's physiology: If the climbing fibers encode a reset function, then it would plausible that a connection would be added one by one as cascade training. If the last made connection worsened the performance, the reset function will remove it. If the climbing fiber encodes a fitness function, maybe each Purkinje cell is a blank template, where random connections are being created and evolved if they survive the fitness test. From the remaining Purkinje cells, the other layers will vibrate along with them, therefore creating connections which are similar to theirs. In case the climbing fiber provides a noise signal, it can be that the cerebellum is more involved with reinforcement learning.

Doya (2000) commented that the basal ganglia is learning with a reinforcement reward signal. However, the cerebellum contains all the aspects of a supervised learning system. In section 2.1.1, it was mentioned that those two brain areas might have more connections with each other than initially was assumed (Bostan et al. (2013)). A solution to the earlier mentioned distal supervised learning problem could lie in the cooperation of between those two brain areas. The basal ganglia can determine the information needed, i.e. an input and output training set for a certain task, which is modified by a performance measure as reinforcement signal. The cerebellum will manifest this information by supervised learning, therefore fine-tuning and improving fine motor control.

The cerebellum can also simulated by spiking neural network, also called the third generation of neural network by Maass (1997). Due to computational limitations, this method was not used within my research. However, its computation is closer to what is actually happening in the human brain. This raises the question on how similar the cerebellar-inspired model must be to its original, biological, counter part, in order to grab the essence of its functionality. Although the human body's neurons use spiking signals, it is not necessarily apt for robotics. The ability to extract the functionality of a brain part, able to integrate within a robot, without changing the framework within it operates, would lead to a better understanding of the human brain.

7.1.4. Optical Flow

For this master thesis, a dense optical flow algorithm was used to visually detect the ego-motion of the binocular robot. This signal was then fed back to the cerebellar-inspired model as a learning signal, as it needs to predict the optical flow a couple of steps in advance. In chapter 5, several techniques are given to do this task, and evaluated based on speed and ability to accurately determine what the ego-motion of the algorithm was. I investigated how to achieve this for the 2D image.

However, the resulting motion parallax was not taken into consideration. The rotational and linear disturbance used for the experiments will create different motion characteristics within the image. Objects far away will appear to move faster with rotational disturbance, however with linear disturbance, further objects will actually appear to move less fast than objects nearby. If a more 3D motion detection was used, these motion parallaxes could be identified. For this research, it was sufficient for the algorithm to use the optical flow of the object that the tracker was following.

One of the requirement of the motion detection, was computation speed. Since this optical flow algorithm should drive the real time smooth pursuit, its calculation time should stay under 0.033 seconds, for a control frequency of 30 hz. The computation time calculated within section 5.3, is done without any of the other processes within the implementation running. If the cerebellar-model for each of the motors were initiated, together with the pursuit, the duration was increased three to four times. This is the reason why the optical flow was only calculated by a smaller, middle portion of the image, i.e. on the object on which the eyes focused.

For this research, mostly existing computer vision techniques had been used to calculate optical flow within an image-stream. The main objective of the research required to focus more on the mechanism of cerebellar learning, where the optical flow is the supervising sensing system. Therefore, conventional algorithms to detect optical flow have been used, instead of a developing a new method inspired on the human vision system. For future recommendations, more can be done to study the human processing capabilities for optical flow.

7.1.5. Experiments

In section 1.1.2, some types of compensatory eye movements are explained, where a distinction was made between reflexes of a rotational disturbance, excited by the semicircular canals, and the linear disturbance, excited by the otolith organs. These two types of disturbances has been the inspiration for the conducted experiments. Initiating a rotational disturbance is done by the neck of the robot itself. Since the neck joint is immediately connected to a motor, it was easy to ensure repeatability of the experiment.

For the linear disturbances, the system was put upon a cart and driven back and forth. The difference here, is that using a cart will increase the chance of the cart to slip or be affected by the slope of the floor it is riding on. As a temporary solution for this, a short pause was initiated in between experiment sequences, to have time to put the setup back to its original position. Although it could not be done with precise accuracy, it was enough to prevent a bias in the output of the learned cerebellar model. Eventually, it would have been much easier to have used linear translation actuators instead of a cart.

As the variation within the linear disturbance was given by the position of a marker, initially this was done by hand. Fortunately, an industrial robot was then added to assist placing the marker. Although this definitely increased the repeatability of the experiment, still some arguments can be given on how it is handled. Now the movement of the robotic arm, the vision setup and the execution of the software were manually matched by setting time limits for the systems. A check on the marker position, for the results, was done by registering the distance of the marker as calculated by the cameras. However, it is not ideal in terms of synchronization, especially since ROS does not work real-time and is highly depended on the computer's processor. A better way would have been to connect the robotic arm directly to the same operating system, where the binocular robot and cart also are controlled from as well. It can receive commands on, when to start, when to pause and when to move the marker to a new position from the same program which controls the experiments and collects the data.

Ensuring repeatability of the experiments had been the biggest issues I had to deal with during the experiments. Although this was important to be able to separate the data in theory this should not make much difference for the supervised learning of the cerebellar models. In order to achieve generalization of the models, they must come in contact with as many situations/disturbances as possible. These variations in the experimental setup should actually improve the performance. However, to be able to separate the data, for different frequencies and marker positions, I chose to keep those in-between-experiment variations to a minimum.

7.1.6. Results

The experiments showed that the implementations of the adaptive and information filter were able to stabilize the vision system better, by more reduction of the optical flow and the motion blur, than the pursuit tracker alone. However, when the models were compared to a case where the pursuit control receives the optical flow directly, the difference in performance is not as significant. For the prediction of the optical flow, certain delays are taken into account for the cerebellar models' prediction horizon, which is assumed to be 120 ms in total. However, since ROS is so depended on the computer's processor, the computational limits are reached, increasing the latency of the optical flow detection. The expectation is that both the adaptive and information filter's performance can be increased by extending their prediction horizon.

It was initially thought that the adaptive filter will perform better for the rotational disturbance and that the information filter would do better for linear disturbances. Although the first hypothesis did hold, the second is not as significant. In terms of optical flow, the information filter was able to reduce its value, however the adaptive filter reduced the motion blur more within the image stream. As the optical flow has more access to different representation of the current state, including visual feedback, it seems logical that it would have been more qualified to handle the linear disturbance with the variable marker position. However, the experiments show that this was not the case.

The adaptive filter was able to reduce the optical flow and motion blur at all the experiments. The information filter also achieved this, but not as well. In section 7.1.1, arguments have been given to support the information filter role of the cerebellum. However, it was mentioned as well that prediction and memory systems probably would exist outside of the cerebellum. This information filter is used to predict optical flow, which is contradicting to that principle. It does not necessarily mean that the cerebellum does not exercise the information selection functionality, however it is eminent that this model must be developed further.

7.2. Conclusions

In the introduction of this master thesis, a main research objective had been formulated: *Implementing a stabilization algorithm which handles external disturbances for an active, binocular vision system inspired by the functionality of the cerebellum.* This has been divided in four sub-objectives, which will be evaluated on how they have been achieved in this research. This will be done using the discussion of the previous section.

The first sub objective is about building the cerebellar model, which is able to adapt itself by means of supervised learning with optical flow. Both the adaptive and information filter are built with artificial neural networks. They are both able to learn a prediction of an optical flow signal within the image streams from the cameras, with a given input originating from the robotics head. Although the adaptive filter theory needed some simplifications and adjustments and the help of a neural network library, this objective has been achieved.

The next sub-goal was the implementation of the model within a real life active binocular robot head. To be able to achieve this, a simulation of the robot had been made. Its description can be found in appendix B. This simulation is able to run on the same software as the robotic setup in ROS, which enabled the easy switching between those two. It gave me the possibility to try out the cerebellar models freely, to be even more sure that they were able to learn and work on the real setup. On top of that, developing it provided me with some useful knowledge about the technical specifics and the dynamics of movements. Instead of trying to make the cerebellar models work on the robot straight from the theory, the simulation provided an extra step in between. Therefore, it was possible for me to make the cerebellar models work on a real system, which has much added value for their validation by the experiments.

The third sub-objective was to improve the robot-head's vision by reducing of the optical flow and motion blur detected, with the help of the prediction of optical flow by the cerebellar model. Looking at the results, the implementation the adaptive filter is indeed able to improve the stability of the binocular robot's vision, in terms of reducing the optical flow and motion blur. However, it can be discussed whether the prediction horizon can be made even larger, and if that would have an effect for the performance of the cerebellar model.

The last subgoal covers if the existing cerebellar-model (the adaptive filter) could be compared with the information filter, which is based on more recent neurological research. This adapted filter, is able to still predict optical flow, after changing the input to the neural network. The experiments show that the information does not give a better performance than the adaptive filter. For this particular situation, it appears to be more beneficial to know the history of motor commands, than to know as much information as possible from the current time frame. It was expected to be otherwise, as it was hoped that the information filter was able to select the correct signals needed for the prediction. Moreover, as it is connected to various dynamic signals, I expected that by its nature it would have more state information than just the current time step. This can be reflected on that the information filter either receiving too little state information, or that it needs more history of motor input in order to determine the dynamics of the disturbance, in order to make a prediction. Therefore, a combination of both the adaptive and information filter should have been researched as well.

7.3. Recommendations and Future Perspectives

From the research conducted for this thesis, there are a few recommendations to give to my successor in cerebellar-inspired control for robotics. First of all, one has to look more into the proposed functionality of the cerebellum being an information selector instead of a forward model (see section 2.2.1). A good starting point for this, is to assume the cerebellum is a black box with just input and output. By going into neuroscientific papers, with help of people who have experience with clinical research, one can find which are the established input and output signals to the cerebellum.

The cerebellum's relationship and connections must be investigated with the rest of the brain, to get a better sense of its functionality. To further increase the robustness of the cerebellar-inspired model, one should look into a joint reinforcement learning and supervised learning algorithm, based on the cooperation between the cerebellum and the basal ganglia (as discussed in section 7.1.1). As an example, the basal ganglia will try out different control policies for various of task, therefore providing the training data for the cerebellum to fine tune. This will result in a more robust robot control algorithm.

In terms of hardware, it would be wise to look into various parallel processing architectures, which are able to simulate the forward parallel processing capabilities of the cerebellum. One should look at what other researchers have used for their cerebellum-inspired implementation into a robotic setup, like in section 2.2.3. What should remain the key of cerebellar inspired control, is the determination of what it processes, what it learns and how it should communicate with different systems.

It is important to reflect on the future of the binocular vision robot and cerebellar-inspired model implementations. Not only will it provide a view on more robust and adaptive control methods for robotics, it may become a platform for neuroscientists to express their ideas about the human brain. The ability to transform functionality of a brain area into a robot will underline the theories they have on their neurological counterpart. Therefore, it is recommended to have a more intense collaboration on these topics, where people of both expertises can enhance each-others' knowledge on the processing and functionality of the cerebellum.

A

Hardware

A.1. ThreeMXL PID control Tuning

As calibration, the PID gains must be tuned. The vision system has springs to reduce backlash. To tune it, first a reference speed is given to the system as a step function. The reaction in the motor's position is recorded as well. From these plots (found in the Appendix), certain values are extracted for the calculation of the gains. The initial slope of the response (ΔR) estimates the dead time (DT) , which the time it takes for the system to respond to stimuli. That all with the reference's size (dMV), the gearbox ratio (GR), the encoder's resolution (ER), and a scaling factor (k) the following formula is used to calculate the proportional (K_p) and integral gain (K_i)

$$\Delta P = \frac{4 \cdot ER \cdot \Delta R}{2\pi \cdot GR} \tag{A.1}$$

$$PG = \frac{\Delta P}{dMV} \tag{A.2}$$

$$K_p = \frac{k}{2 * DT \cdot PG} \tag{A.3}$$

$$K_i = \frac{PG}{8 \cdot k} \tag{A.4}$$

The used values and its resulting gains can be seen in table A.1.

	ΔR [rad/s]	ER	GR	DT [s]	dMV [rad]	K_p	K_i
Left Eye	0.32	512	1:19	0.03	0.15	0.1261	0.5256
Right Eye	0.33	512	1:19	0.04	0.15	0.0917	0.2867
Tilt	0.039	32	1:4.4*60	0.1006	0.3	0.7109	0.8833
Pan	0.03	32	1:231	0.1144	0.3	0.9291	1.0155

Table A.1: The table with used values and resulting gains, using a scaling factor of 0.01

A.2. IMU Calibration by Physical model

The sensors within the IMU, which are the accelerometers, gyroscopes and magnetometers, are calibrated by means of a linear physical sensor model. The model is explained as followed:

$$\boldsymbol{s} = K_t^{-1}(\boldsymbol{u} - \boldsymbol{b}_t) \tag{A.5}$$

where $\boldsymbol{s}$ stand for the physical quantity, u for the sampled digital voltages, K_t is the gain matrix and $\boldsymbol{b}_t$ is the bias vector. The gain matrix consist of an alignment matrix $\boldsymbol{A}$ and an gain matrix $\boldsymbol{G}$.

$$A = \begin{pmatrix} a_{1,x} & a_{1,y} & a_{1,z} \\ a_{2,x} & a_{2,y} & a_{2,z} \\ a_{3,x} & a_{3,y} & a_{3,z} \end{pmatrix} \quad G = \begin{pmatrix} G_1 & 0 & 0 \\ 0 & G_2 & 0 \\ 0 & 0 & G_3 \end{pmatrix} \tag{A.6}$$

$$K_t = G \cdot A + \boldsymbol{O} \tag{A.7}$$

where O stands for additional terms related to non linear, temperature modeling offsets and etc. The calibration data for these matrices are as follows:

$$A_{accel} = \begin{pmatrix} 1.00 & -0.01 & 0.00 \\ 0.01 & 1.00 & 0.00 \\ 0.00 & -0.01 & 1.00 \end{pmatrix} \quad G_{accel} = \begin{pmatrix} 549.3 & 0 & 0 \\ 0 & 547.2 & 0 \\ 0 & 0 & 551.1 \end{pmatrix} \tag{A.8}$$

$$A_{gyro} = \begin{pmatrix} 1.00 & 0.01 & 0.00 \\ 0.00 & 1.00 & 0.01 \\ 0.00 & 0.00 & 1.00 \end{pmatrix} \quad G_{gyro} = \begin{pmatrix} 3730 & 0 & 0 \\ 0 & 3816 & 0 \\ 0 & 0 & 3777 \end{pmatrix} \tag{A.9}$$

$$A_{magn} = \begin{pmatrix} 1.00 & 0.00 & -0.04 \\ 0.02 & 1.00 & 0.05 \\ -0.05 & -0.02 & 1.00 \end{pmatrix} \quad G_{magn} = \begin{pmatrix} 7958 & 0 & 0 \\ 0 & 8159 & 0 \\ 0 & 0 & 7941 \end{pmatrix} \tag{A.10}$$

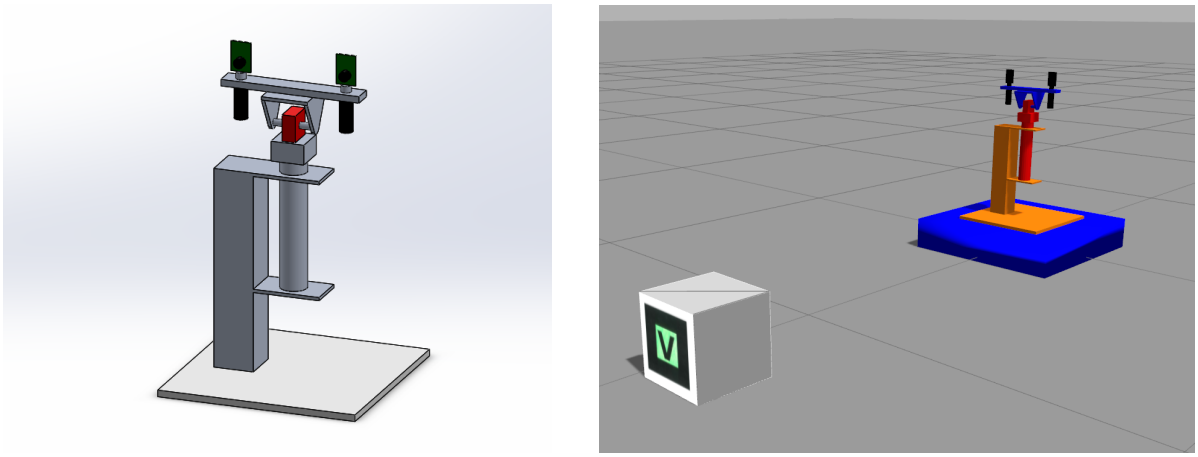
$$\tag{A.11}$$

B

Simulation

Several simulations have been done for this project within Matlab and SimMechanics. However, since the hardware is run by ROS and C++ code, it is more ideal to have a simulation which can work with the same code as well. I therefore decided to build a robot model with cameras in gazebo (Fig. B.1(b)). Gazebo is a robot simulation tool, which enables to easily designed robots and test these within a simulated environment.

I first modeled the robot-head within Solid Works 2014, which can be seen in Fig. B.1(a). This could than be imported into a Gazebo URDF file, which could than be used within the ROS architecture of the robot. The guidelines in the manual of Watson and Strickland had been used to achieve this. Afterwards, have been simulated cameras was placed upon the simulated setup, and a object with the same marker it was programmed to follow. After the PID controllers had been tuned for this occasion, it was able to follow this marker, using the exact same control architecture as the real life setup.



(a) (b)

Figure B.1: (a) The robotic setup modeled in Solid Works and (b) the same model active within Gazebo.

B.1. Experiments on Simulation

To test the cerebellar models, the simulation was used by the prediction of the marker speed, the given the goal speeds and the previous goal speeds (until $k-3$). This has been done for just the left eye, within the simulation

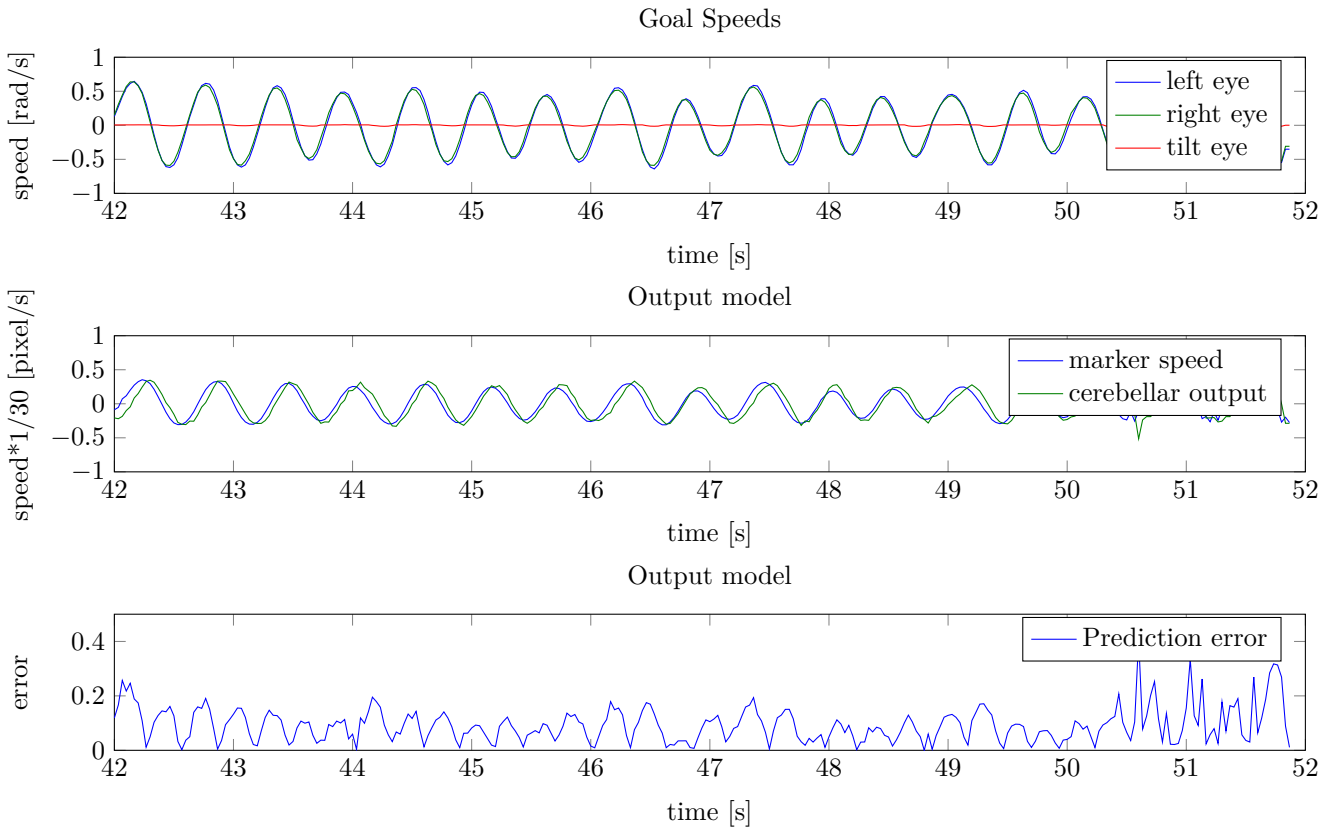


Figure B.2: Here the results can be seen of the adaptive filter's prediction performance in the Gazebo simulation. The goal speeds are seen which were the input of the adaptive filter. It was setup to predict the optical flow a 120 ms in advance, from which the results are placed over each other in the second plot. The third plot shows the prediction error, which has a total mean squared error of 0.0174. This is done for a rotational disturbance of the neck.

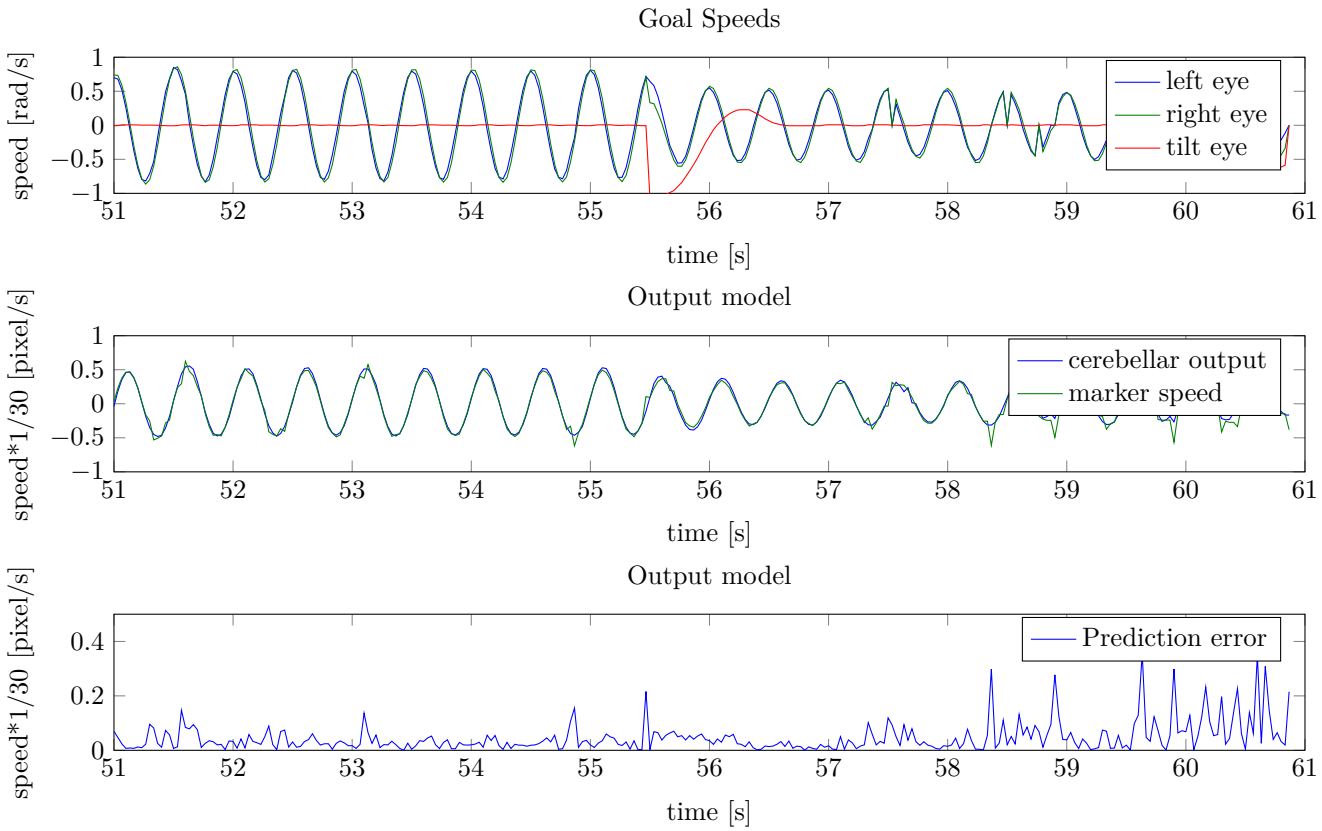
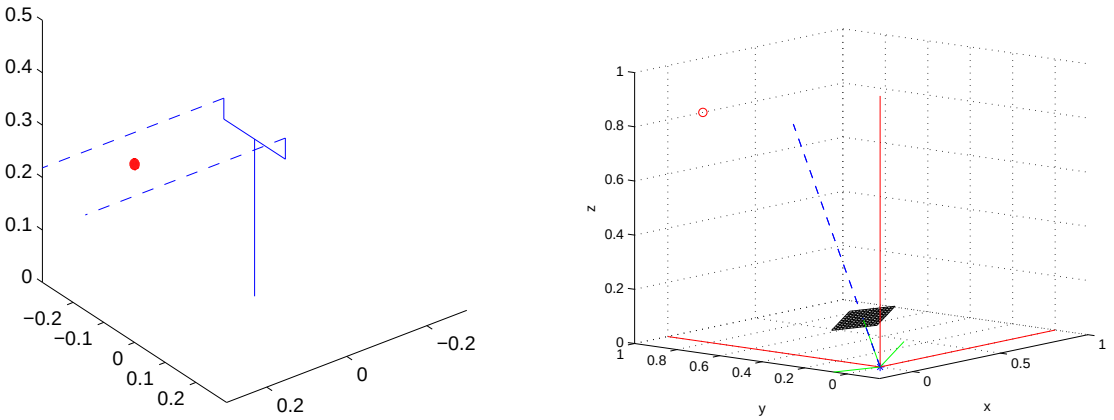


Figure B.3: Here the results can be seen of the adaptive filter's prediction performance in the Gazebo simulation. The goal speeds are seen which were the input of the adaptive filter. It was setup to predict the optical flow 120 ms in advance, from which the results are placed over eachother in the second plot. The third plot shows the prediction error, which has a total mean squared error of 0.0121. This is done for a translational disturbance from right to left of the base

B.2. Active Binocular Setup Dynamics Modeled by Matlab

Before Gazebo was used, the active binocular setup was simulated within Matlab 2014 itself. This is done by using the 3D plot functionality, dynamical equations and the differential equations solver available. The result for the simulation can be found in Fig. B.4(a), for which a camera system which displayed the to be followed object on the cameras, by means of its orientation and translation (Fig. B.4(b)). The differential equations used to simulate the dynamics on the model can be found in the next subsection, which was solved by a variable order method for stiff differential equations called `ODE15s`. The equations also consited of the PID control, to follow the red dot on its camera's

Although, it was chosen not to proceed with this simulation, since it was difficult to implement the cerebellar model, as a fixed time step was not possible to solve these differential equations fast enough. It seemed by then counter productive to solve these problems within this program, and to be able to still fulfill the research objectives. Also, it was difficult to ignite any kind of optical flow with this current camera view. It was therefore chosen to look for other options to model the robotic head. Still, building this simulation modeling its vision system had given me a good experience and sense on the movement and dynamics of the active binocular setup, and stereo vision overall.



(a) (b)

Figure B.4: (a)The robot head modeled within Matlab 2014 and (b) the camera system made for the occasion.

B.3. Dynamic Differential Equations

Here the equations for the dynamical behavior of the hardware are derived. This has been done with the TMT method from van der Linde and Schwab (1998).

$$R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_\beta = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \tag{B.1}$$

$$R_{\gamma_1} = \begin{pmatrix} \cos(\gamma_1) & -\sin(\gamma_1) & 0 \\ \sin(\gamma_1) & \cos(\gamma_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_{\gamma_2} = \begin{pmatrix} \cos(\gamma_2) & -\sin(\gamma_2) & 0 \\ \sin(\gamma_2) & \cos(\gamma_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.2}$$

$$R_1 = \begin{pmatrix} \cos(\alpha)\cos(\beta)\cos(\gamma_1) - \sin(\alpha)\sin(\gamma_1) & -\cos(\gamma_1)\sin(\alpha) - \cos(\alpha)\cos(\beta)\sin(\gamma_1) & \cos(\alpha)\sin(\beta) \\ \cos(\alpha)\sin(\gamma_1) + \cos(\beta)\cos(\gamma_1)\sin(\alpha) & \cos(\alpha)\cos(\gamma_1) - \cos(\beta)\sin(\alpha)\sin(\gamma_1) & \sin(\alpha)\sin(\beta) \\ -\cos(\gamma_1)\sin(\beta) & \sin(\beta)\sin(\gamma_1) & \cos(\beta) \end{pmatrix} \tag{B.3}$$

$$R_2 = \begin{pmatrix} \cos(\alpha)\cos(\beta)\cos(\gamma_2) - \sin(\alpha)\sin(\gamma_2) & -\cos(\gamma_2)\sin(\alpha) - \cos(\alpha)\cos(\beta)\sin(\gamma_2) & \cos(\alpha)\sin(\beta) \\ \cos(\alpha)\sin(\gamma_2) + \cos(\beta)\cos(\gamma_2)\sin(\alpha) & \cos(\alpha)\cos(\gamma_2) - \cos(\beta)\sin(\alpha)\sin(\gamma_2) & \sin(\alpha)\sin(\beta) \\ -\cos(\gamma_2)\sin(\beta) & \sin(\beta)\sin(\gamma_2) & \cos(\beta) \end{pmatrix} \tag{B.4}$$

$$
\begin{aligned}
p'_{neck} &= p_{neck} && \text{(B.5)}\\
p'_{tilt} &= p_{neck} + R_\alpha \cdot p_{cam,left} && \text{(B.6)}\\
p'_{cam,left} &= p'_{tilt} + R_\alpha \cdot R_\beta \cdot p_{cam,left} && \text{(B.7)}\\
p'_{cam,right} &= p'_{tilt} + R_\alpha \cdot R_\beta \cdot p_{cam,right} && \text{(B.8)}\\
p'_{focus,left} &= p'_{cam,left} + R_1 \cdot p_{focus,left} && \text{(B.9)}\\
p'_{focus,right} &= p'_{cam,left} + R_2 \cdot p_{focus,right} && \text{(B.10)}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{q} &= [\alpha, \beta, \gamma_1, \gamma_2]^T && \text{(B.11)}\\
\dot{q} &= [\dot{\alpha}, \dot{\beta}, \dot{\gamma}_1, \dot{\gamma}_2]^T && \text{(B.12)}\\
\boldsymbol{p} &= [p'_{neck}, p'_{tilt}, p'_{cam,left}, p'_{cam,right}]^T && \text{(B.13)}
\end{aligned}
$$

For the motion equations, the TMT method is used. The implementation goes as following.

$$
\mathbf{T_{i,j}} \mathbf{M_{i,j}} \mathbf{T^T_{i,j}} \cdot \ddot{\mathbf{q}}_\mathbf{j} = \mathbf{Q_j} + \mathbf{T^T_{i,j}} \cdot (\mathbf{f_i} - \mathbf{M_{ij}} \mathbf{g_j}) - \mathbf{D_v} \cdot \mathbf{k} - \mathbf{D_d} \cdot \mathbf{c} \qquad \text{(B.14)}
$$

where

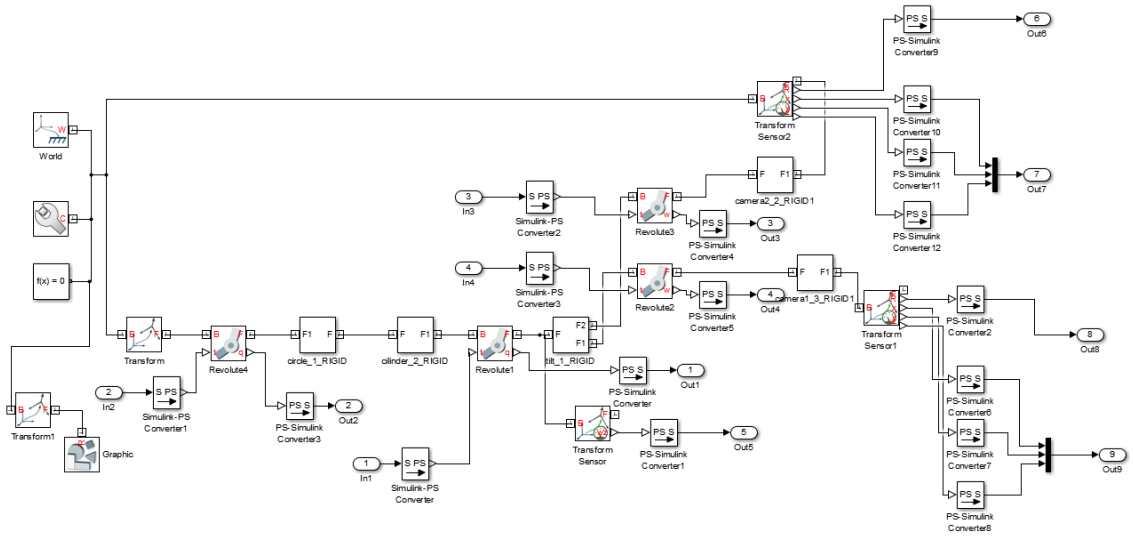
$$
\begin{aligned}
p_i &= T_i(q_j) = p\\
T_{i,j} &= \texttt{jacobian}(T_i, q_j)\\
g_j &= \texttt{jacobian}(T_{ij} \cdot \dot{q}_i, q_j) \cdot \dot{q}_j
\end{aligned} \qquad \text{(B.15)}
$$

and

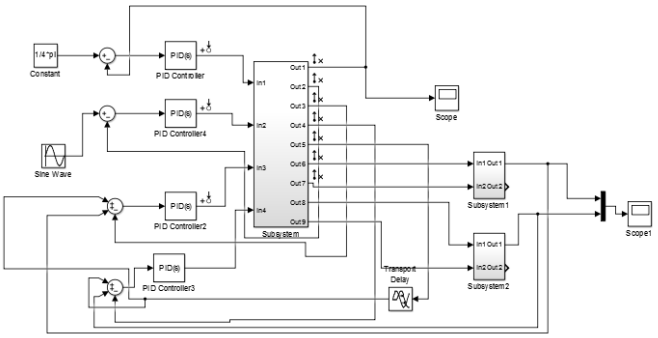
$$
\begin{aligned}
f_i &= g \cdot [0, 0, -m_{neck}, 0, 0, -m_{tilt}, 0, 0, -, m_{cam}, 0, 0, -m_{cam}, 0, 0, 0, 0, 0, 0]^T && \text{(B.16)}\\
Q &= [\tau_{neck}, \tau_{tilt}, \tau_{cam,left}, \tau_{cam,right},]^T && \text{(B.17)}\\
M_{ij} &= \texttt{diag}([m_{neck}, m_{neck}, m_{neck}, m_{tilt}, m_{tilt}, m_{tilt}, m_{cam}, m_{cam}, m_{cam}, m_{cam}, m_{cam}, m_{cam}, \mathbf{0}]) && \text{(B.18)}\\
k &= [k_\alpha, k_\beta, k_{\gamma_1}, k_{\gamma_2}]' && \text{(B.19)}\\
c &= [c_\alpha, c_\beta, c_{\gamma_1}, c_{\gamma_2}]' && \text{(B.20)}\\
D_v &= [\alpha, 0, \gamma_1, \gamma_2]' && \text{(B.21)}\\
D_d &= [0, \dot{\beta}, 0, 0]' && \text{(B.22)}
\end{aligned}
$$

B.4. Active Binocular Setup Dynamics by SimMechanics

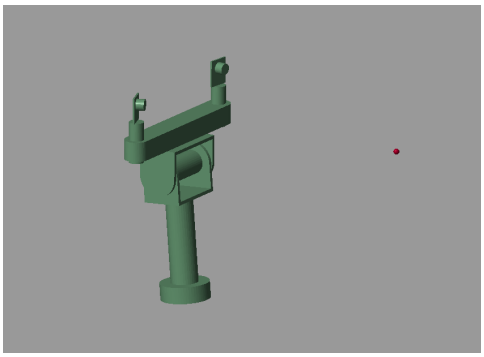
The second model of the active binocular system I made, was with the combination of Simulink and SimMechanics (Fig. B.5). With the latter, a solid works model could be imported, and its joints could be connected to each other with different characteristics. With Simulink, the PID controls and the disturbance could be added to the system. However, due to its limitations of modeling its cameras output, it was also chosen to continue with other solutions to simulate the robot head. Therefore, I ended up with simulating it in Gazebo, as explained in the beginning of this appendix's chapter.



(a)



(b)



(c)

Figure B.5: (a) The control architecture in Simulink in Matlab 2014 from the (b) Solid Works modeled robotic head. (b) This simulation is build with the use of SimMechanics, also part of Matlab.

C

Neural Networks

The cerebellar model is tested on the simulated robotic setup. In fig. C.1, the different signals resulting from the setup can be seen. In Fig. C.2, their prediction capability of the detected marker speed (optical flow was not implemented yet), is shown. This is done for several types of data inputs. The prediction error per various data input, can be seen in Fig. C.3.

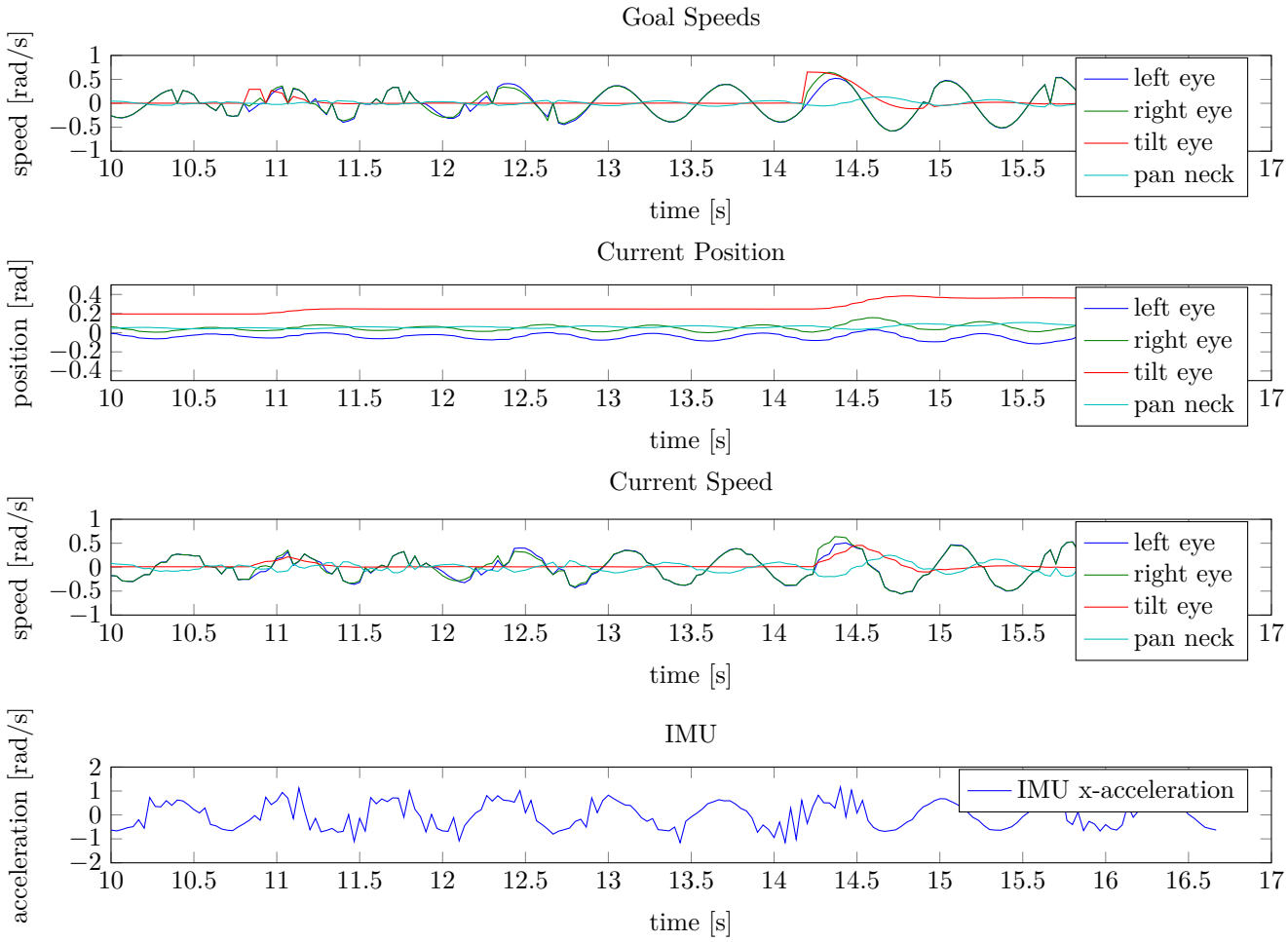


Figure C.1: Different signals resulting from the sensors of the simulated robot-head in gazebo. These are candidate inputs for the cerebellar models.

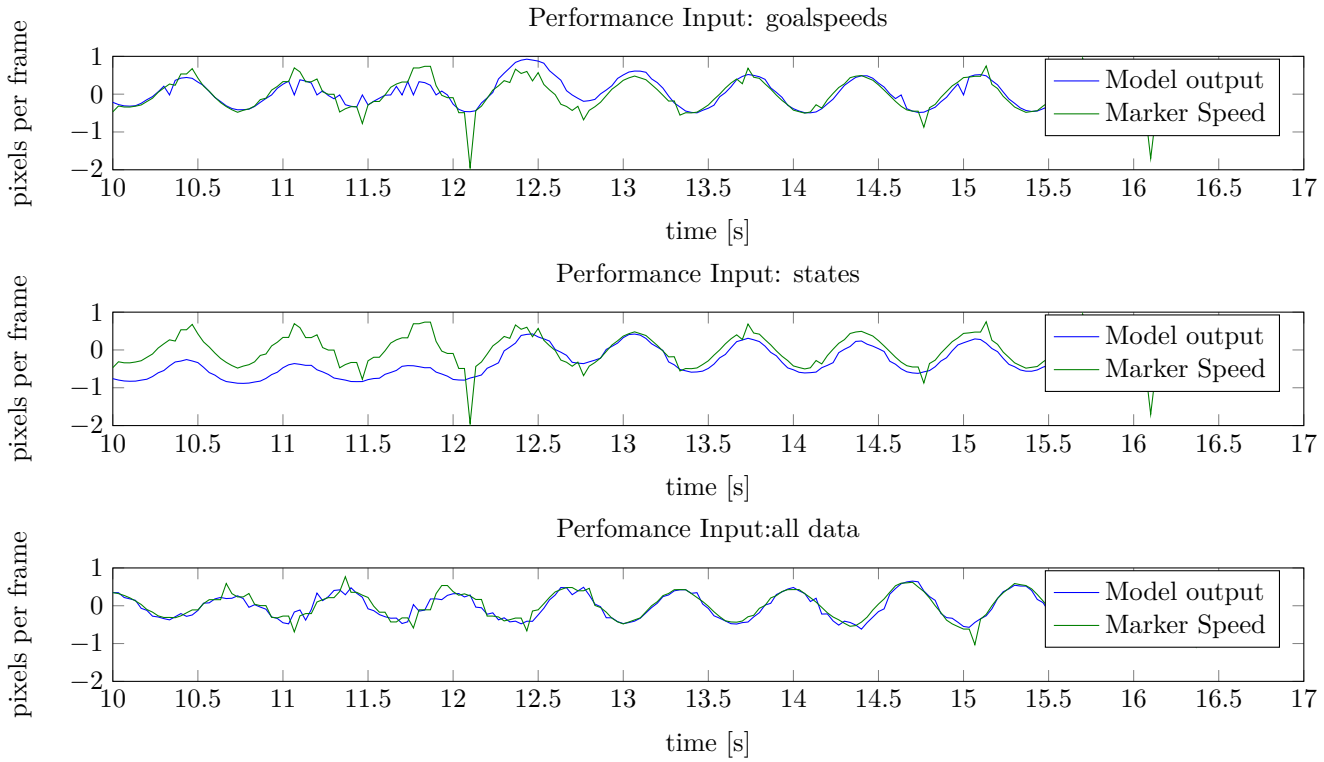


Figure C.2: The cerebellar model received different inputs: goalspeed, states (velocity and speed) and all the data possible. With this, they had to predict the marker speed several steps in advance.

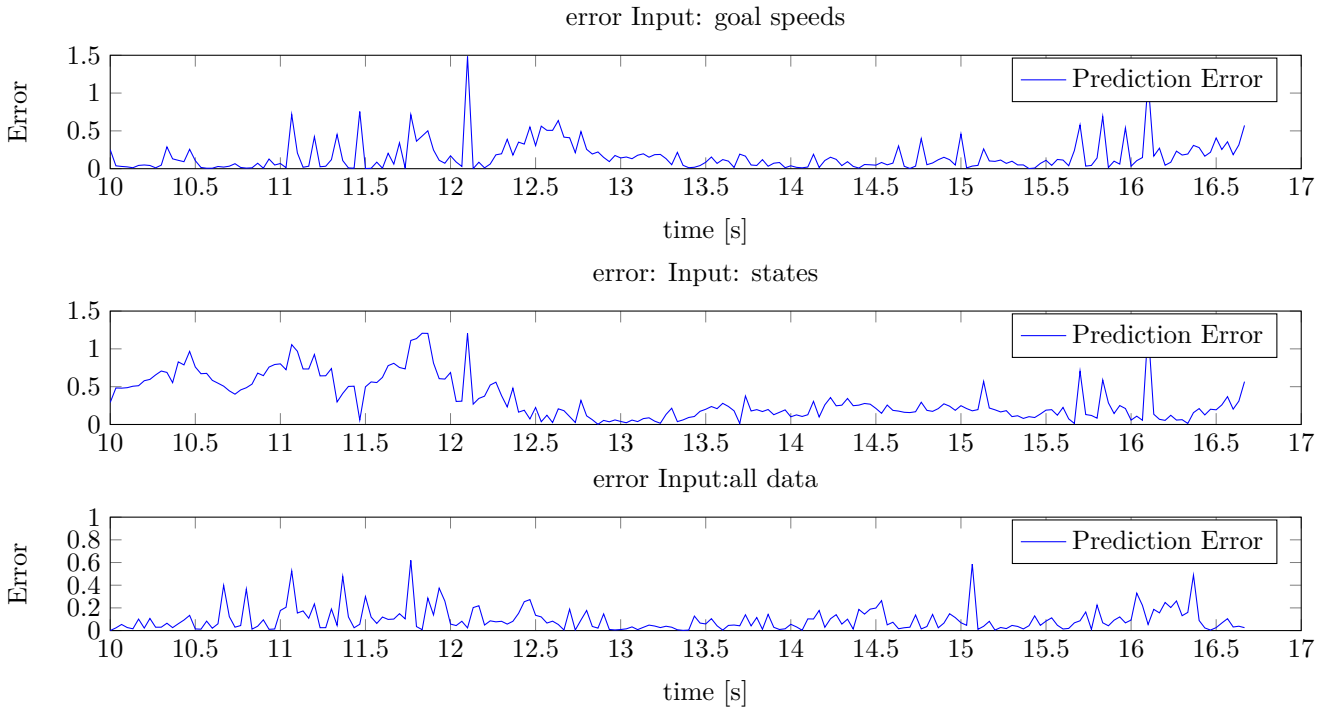


Figure C.3: Here the error of the prediction from Fig. C.2 is displayed. For the entire sequence the MSE is: goalspeeds: 0.0672, states: 0.1983, all: 0.0209

D

Experiments and Results

D.1. ROS graph after Implementation Cerebellar Models

In section 3.3, Fig. 3.8, the simplified ROS architecture is displayed of the existing software architecture, before my implementation. In Fig. D.1, the new ROS architecture is seen, which is based on the `ROSgraph` plot made within ROS itself. However, since that plot was so extensive due to its many connections, an abstraction was made to make it more organized. The three biggest modules added to the architecture was the optical flow, the cerebellar model and the experiment. These consist of separate nodes displayed underneath them. The logging data of Experiments, has the task to store almost all the state data available within the system. Its many connections are not displayed in this graph. The same principle hold for the extensive connections of the information filter.

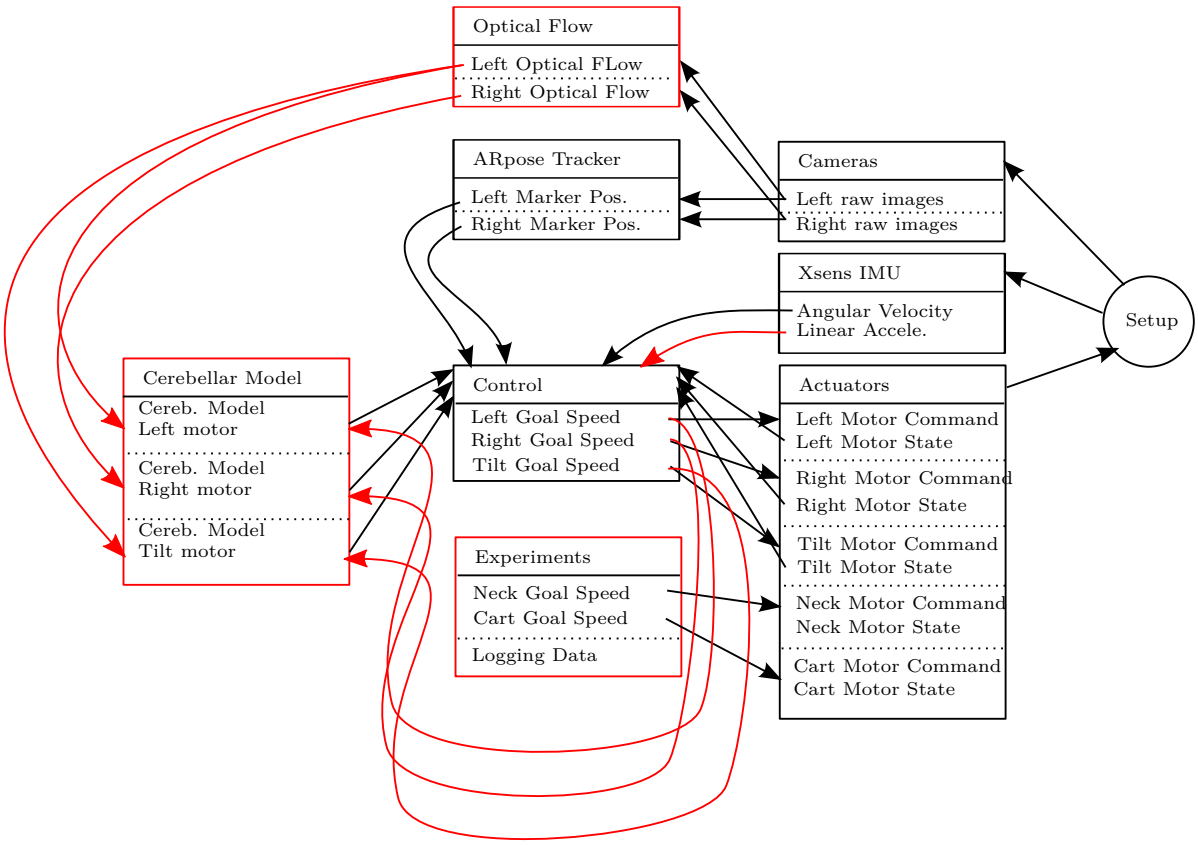
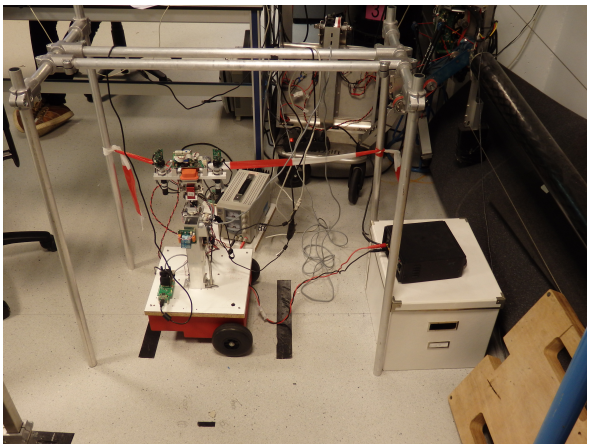


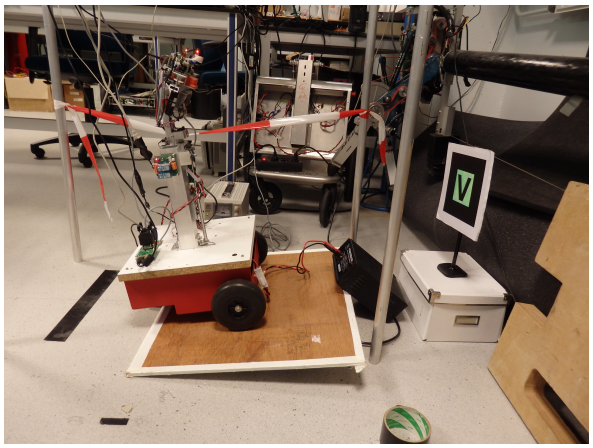
Figure D.1: Structure of the ROS packages of the binocular eye setup, after implementation of the cerebellar models (marked in red).

D.2. Photos Experimental Setup

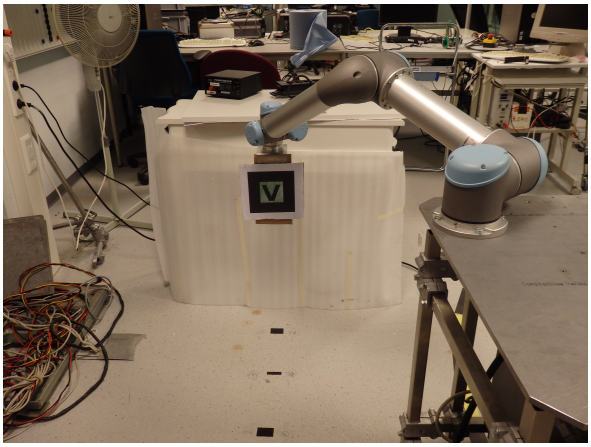
In Fig. D.2 some photos to give a general idea of the experiments performed in section 6.2.



(a)



(b)



(c)

Figure D.2: Photos of (a) the experimental setup of the sideways disturbance with the cart and (b) for a 2D disturbance experiment, where the cart drives forward and backward on a slope. (c) shows an image of the robotic arm (*Universal Robots A/S*) which assisted me during the experiment for the variable marker placement.

D.3. Input Adaptive Filter: Low PID vs High PID

In Fig. D.3, the result is shown for a situation that the input of the adaptive filter would have been the direct voltage of the lower PID controller of 3Mxl consoles. This is compared to the conventional goal speed inputs used in the experiments. The adaptive filter with a higher PID input, was able to reduce the optical flow more (0.1181 px/frame), than when it received an input of the lower PID control (0.1351). This is both opposed to the pursuit tracker only control of the vision setup.

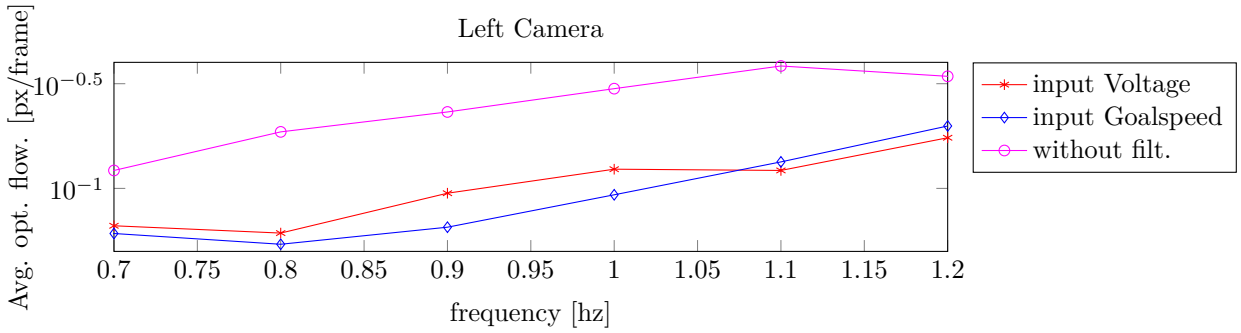


Figure D.3: The average optical flow per frequency for the control with the adaptive filter, with voltage (avg: 0.1351 px/fr) or goal speeds (avg: 0.1181 px/fr) as input, and without the adaptive filter (avg: 0.2336 px/fr).

D.4. Random Disturbance

This section displays the result of both the information and adaptive filter, while handling random disturbances. This was an experiment to check if the cerebellar models were not learning the order of disturbances, but actually the prediction of optical flow. This has been done for a random rotational disturbance, as a summation of multiple sines (Tab. D.1) and for the linear disturbance, with a random marker position (Tab. D.2). Here it can be seen that both the cerebellar models were able to increase the stability of the control better than the pursuit tracker alone.

	MSE prediction
Adaptive Filter	0.13892
Information Filter	n/a

(a)

	Opt. Flow	**vLAP**
Adaptive Filter	0.24604	1033.0322
Information Filter	n/a	n/a
Pursuit Only	0.36415	757.2042

(b)

Table D.1: (a) The MSE error of the cerebellar-inspired model's output and the optical flow 4 time-steps ahead and (b) the average optical flow and the variance of Laplacian (VLAP) from both the adaptive and information filter, the pursuit control only and the pursuit with the optical flow. The average is taken from both cameras for these values, for a random rotational neck disturbance. The information filter's experiment data unfortunately got corrupted, so it's performance can not be compared.

	MSE prediction
Adaptive Filter	0.044817
Information Filter	0.052343

(a)

	Opt. Flow	**vLAP**
Adaptive Filter	0.09755	1917.8424
Information Filter	0.09509	1923.895
Pursuit Only	0.15829	1380.1851

(b)

Table D.2: (a)The MSE error of the cerebellar-inspired model's output and the optical flow 4 time-steps ahead and (b) the average optical flow and the variance of Laplacian (VLAP) from both the adaptive and information filter, the pursuit control only and the pursuit with the optical flow. The average is taken from both cameras for this values, for a linear sideways disturbance, with a random marker position.

D.5. Multiple Network Layers

Here, both cerebellar models are compared, containing 2 or 3 layers in total in their neural network's structure. This is done for both a rotational (Fig. D.4) and translational disturbance (Fig. D.5). It seems that overall, the 2-layered neural network performs better that the 3-layered network, which is different then initially anticipated. Several causes for this is, is that the 3-layered neural network takes longer to generate an input, therefore that delay must be dealt as well. Also, both neural networks have been put on the same amount of epochs and training data sets. Since the 3-layered network had to

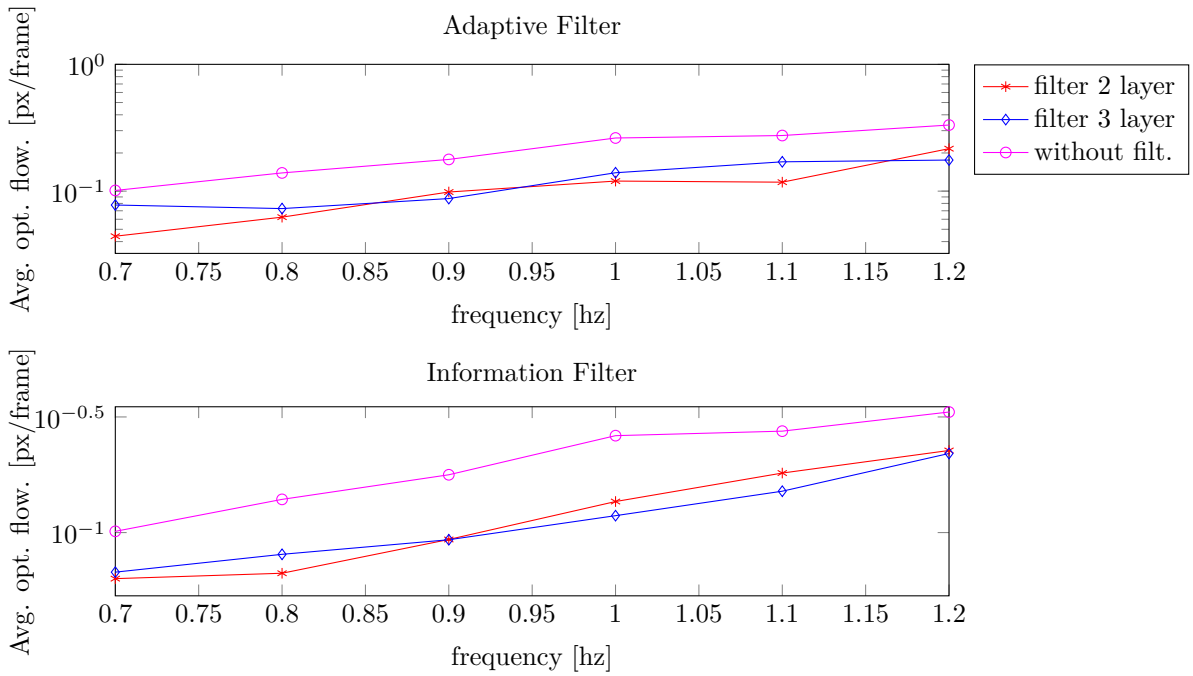


Figure D.4: The results of both the adaptive and information filter for a rotational disturbance of different frequencies. The adaptive filter was compared with 2 (avg: 0.1252 px/fr) and 3 layers (avg: 0.1425 px/fr) . The information filter was compared with 2 (avg: 0.1425 px/fr) and 3 layers (avg: 0.1316 px/fr). This was both compared to the pursuit only tracker (0.2074 px/fr)

learn significantly longer (61 seconds) than the 2-layered one (5.5 seconds), it would need longer time due to its added hidden layer.

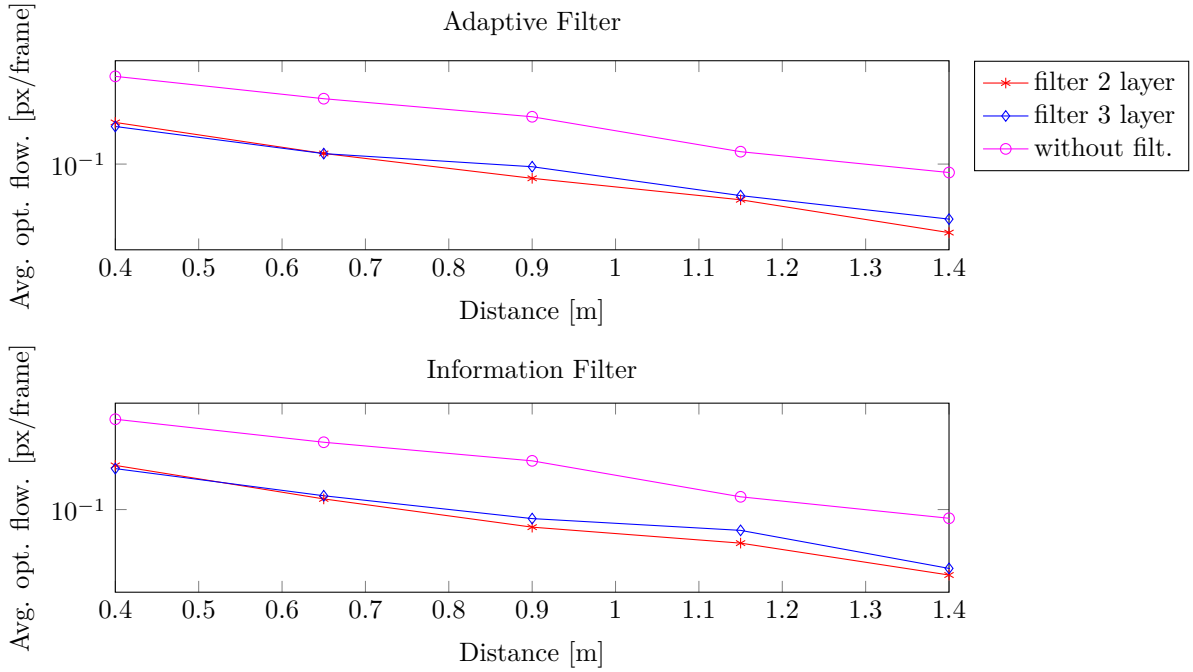


Figure D.5: The results of both the adaptive and information filter for a linear disturbance of different marker positions. The adaptive filter was compared with 2 (avg: 0.0863 px/fr) and 3 layers (avg: 0.0897 px/fr) . The information filter was compared with 2 (avg: 0.0871 px/fr) and 3 layers (avg: 0.0913 px/fr). This was both compared to the pursuit only tracker (0.1567 x/fr)

D.6. IMU vs. Optical Flow Ratio Performance

In this section, a small experiment is conducted, where the active binocular setup left camera's actuator only receives the output of the IMU, or the average horizontal optical flow as input. This is done for several ratios set between the IMU and optical flow, were the result of the average optical flow per frequency sequence can be seen in Fig. D.6. A plot is made, where the data has been normalized per frequency to easily see the difference in performance between the different ratios (Fig. D.7). This was done to compare the active binocular setup's response to the OKR/VOR characteristics of the human body. The OKR (retinal flow compensation) is useful for low frequency disturbances and the VOR (inertial compensation) is suitable for high frequency disturbances. This trend can be seen within an active binocular vision robot as well.

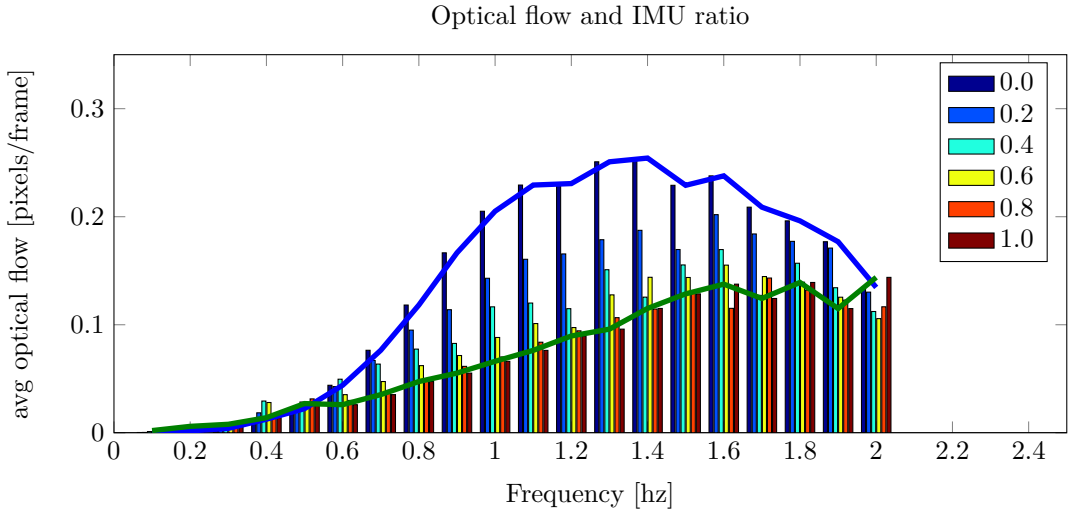


Figure D.6: The average horizontal optical flow per frequency of the left camera of the robot-head, given a control by just the IMU output, the optical flow alone and a ratio in between. This is done for a rotational disturbance with the neck with a frequency from 0.1 to 2.0 hz, with an amplitude of 0.1 rad. The blue line represents the values of the IMU only control, and the green line represents the values of the optical flow only control. The ratio given in the legend stands for the ratio γ (γ*opticalflow=(1-γ)IMU).

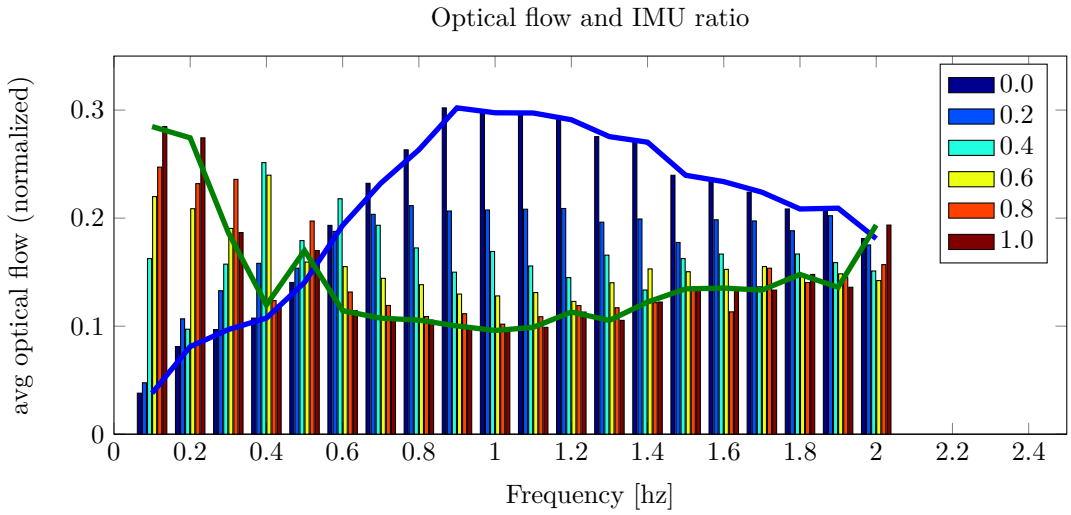


Figure D.7: The same data as in Fig. D.6, but now with normalized data per frequency.

D.7. Experiment Sequence Samples

From the experiments in section 6.2, a few 3 second samples of the *execute* sequence are displayed here, for which the optical flow and the cerebellar models' output are shown for all the motors active. For the

rotational disturbance experiments in section 6.2.1, the sample sequence for the adaptive filter can be found in Fig. D.8 and for the information filter in Fig. D.9. For the rotational disturbance experiments in section 6.2.2, the sample sequence for the adaptive filter can be found in Fig. D.10 and for the information filter in Fig. D.11.

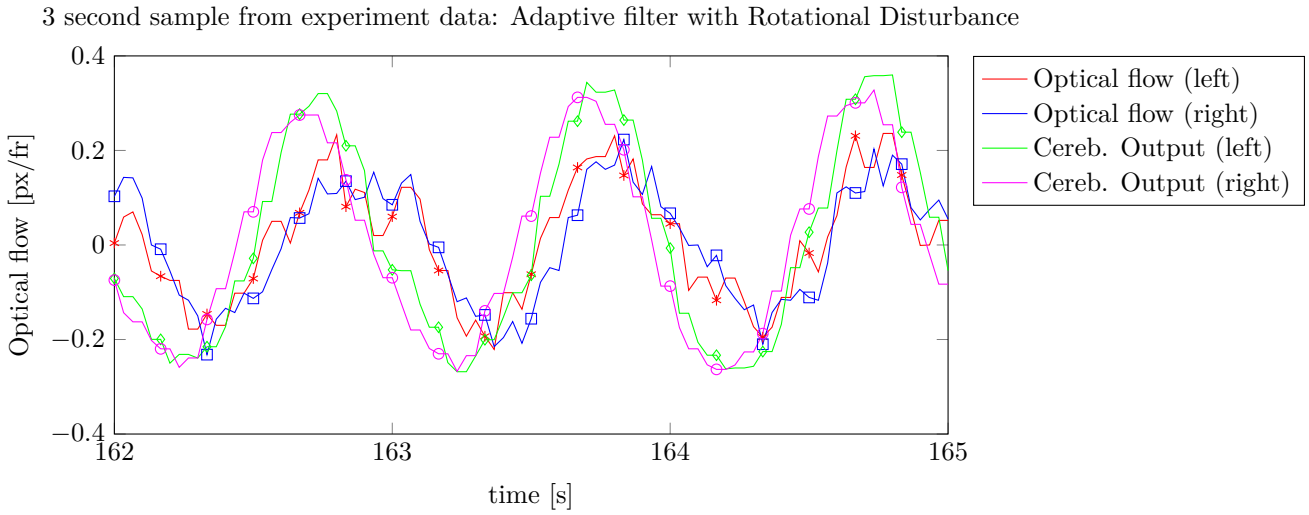


Figure D.8: The 3 second experiment sample of the adaptive filter during a rotational disturbance of the neck. This plot displays the average horizontal optical flow, calculated from both the left and right camera, and the output or the cerebellar models for both cameras.

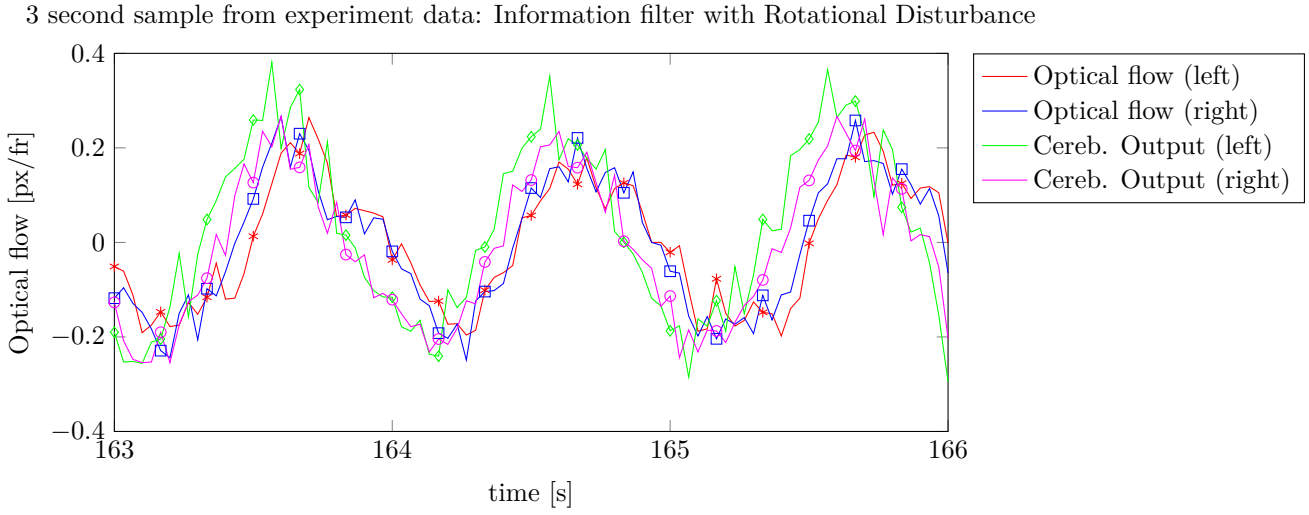


Figure D.9: The 3 second experiment sample of the information filter during a rotational disturbance of the neck. This plot displays the average horizontal optical flow, calculated from both the left and right camera, and the output or the cerebellar models for both cameras.

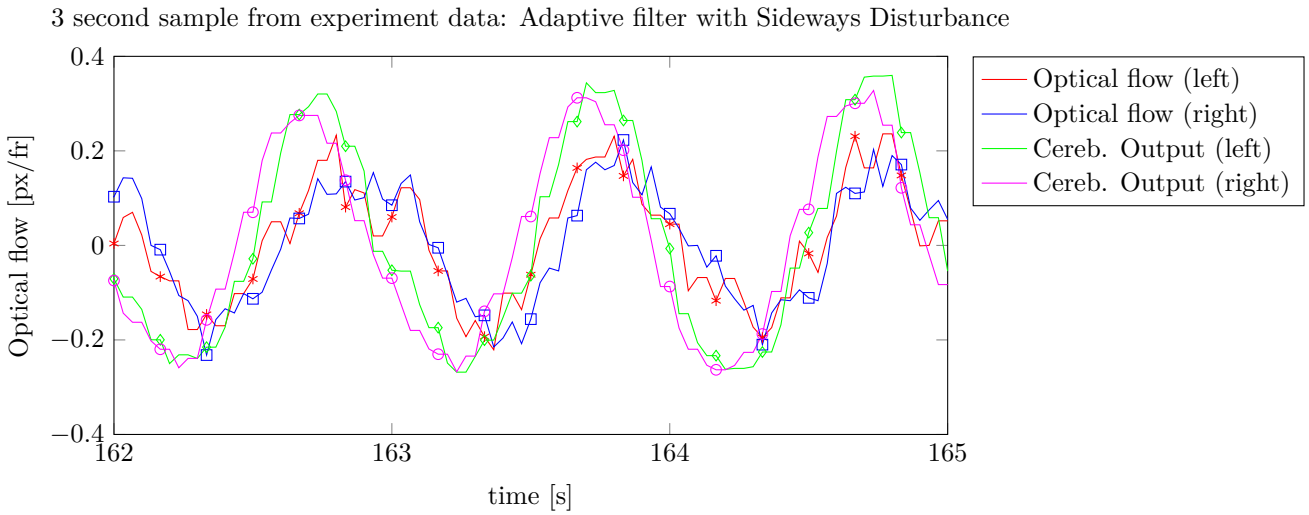


Figure D.10: The 3 second experiment sample of the adaptive filter during a linear disturbance with the cart and a variable marker position. This plot displays the average horizontal optical flow, calculated from both the left and right camera, and the output or the cerebellar models for both cameras.

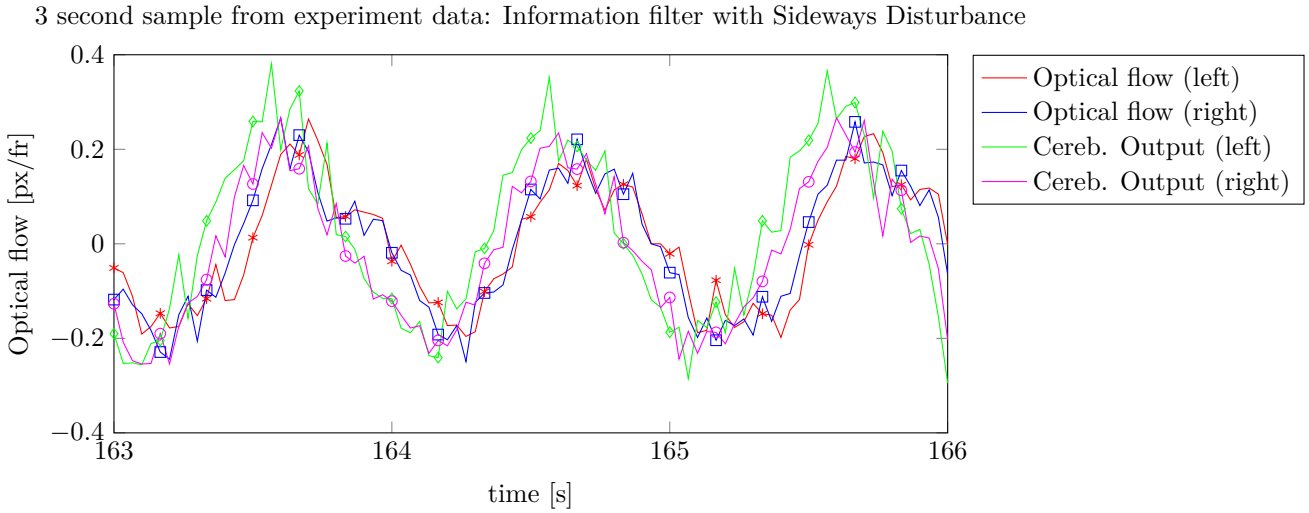


Figure D.11: The 3 second experiment sample of the information filter during a linear disturbance with the cart and a variable marker position. This plot displays the average horizontal optical flow, calculated from both the left and right camera, and the output or the cerebellar models for both cameras.

D.8. Neural Networks Weights

During the training phase of the experiment, both the adaptive and information filter adapt their weights by resilient back propagation on a given training set, with their inputs and the predicted optical flow 120 ms ahead. For the rotational disturbance experiments in section 6.2.1, the trained neural weights in the first layer for the input values for the adaptive filter can be found in Fig. D.12 and for the information filter in Fig. D.13. For the experiments with the sideways disturbance with a variable marker position, discussed in section 6.2.2, the trained neural weights for the input values for the adaptive filter can be found in Fig. D.14 and for the information filter in Fig. D.15.

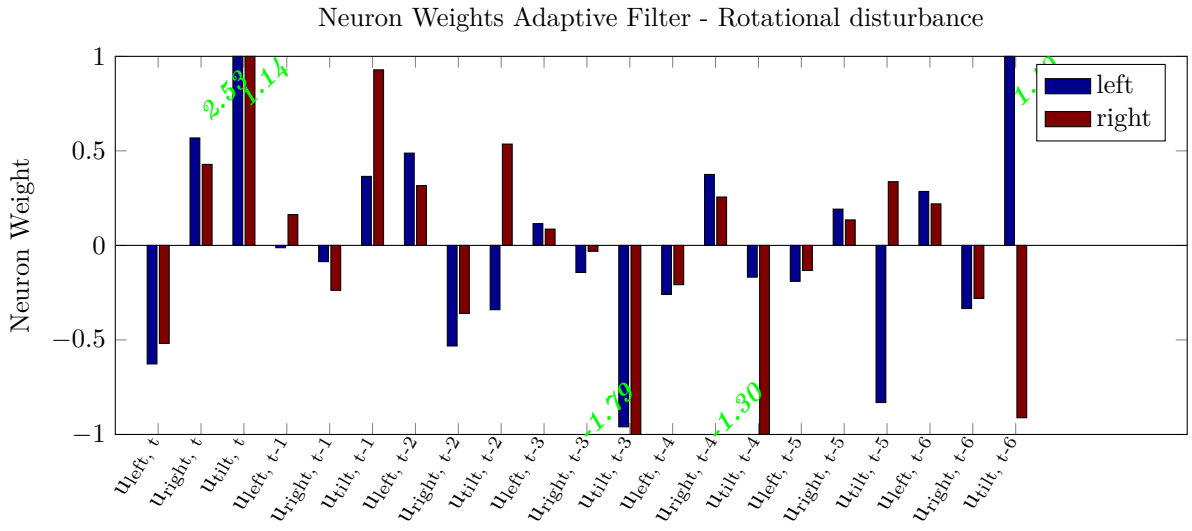


Figure D.12: The trained neural weights of the input value layer from the adaptive filter. $u_{t,left}$, $u_{t,right}$ and $u_{t,left}$ stand for the motor input for the robot for the left camera, right camera and tilt motors. This is equivalent to the goal speed meant for the 3Mxl modules. These weights were obtained during a rotational disturbance with the neck. The neuron values which do not fall within the plot range of [-1,1], have been clipped and their real value have been displayed next to the bars.

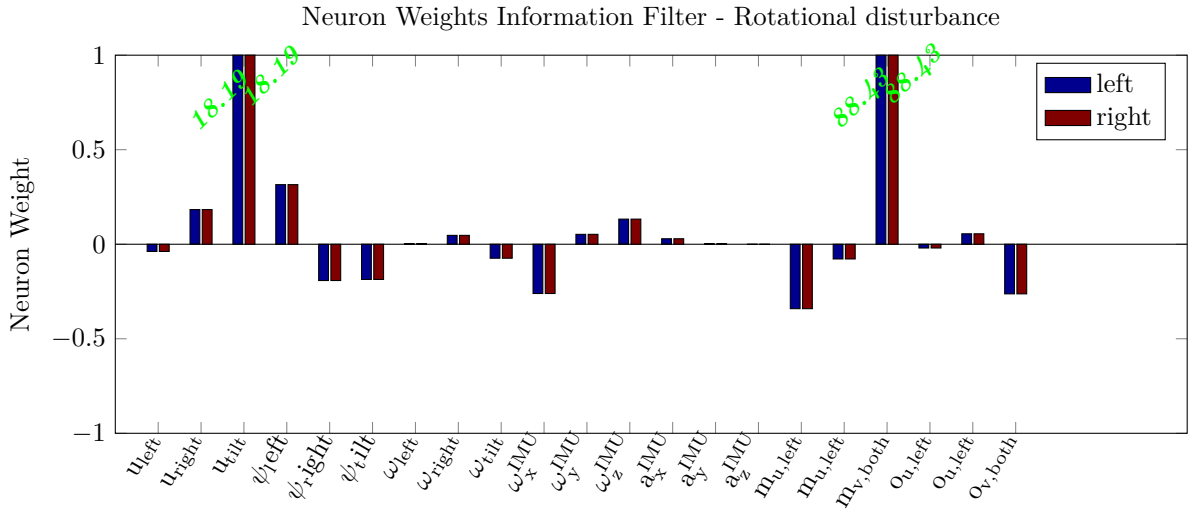


Figure D.13: The trained neural weights of the input value layer from the information filter. The input values can be found in table 4.1 in section 4.3. This is equivalent to the goal speed meant for the 3Mxl modules. These weights were obtained during a rotational disturbance with the neck. The neuron values which do not fall within the plot range of [-1,1], have been clipped and their real value have been displayed next to the bars.

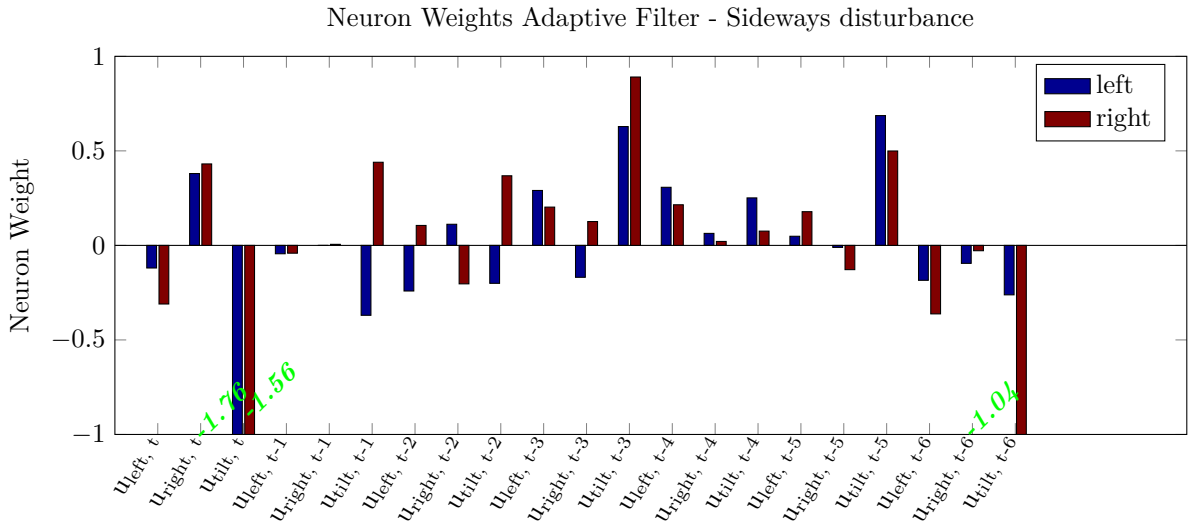


Figure D.14: The trained neural weights of the input value layer from the adaptive filter. $u_{t,left}$, $u_{t,right}$ and $u_{t,left}$ stand for the motor input for the robot for the left camera, right camera and tilt motors. This is equivalent to the goal speed meant for the 3Mxl modules. These weights were obtained during a sideways disturbance with the cart with a variable marker position. The neuron values which do not fall within the plot range of [-1,1], have been clipped and their real value have been displayed next to the bars.

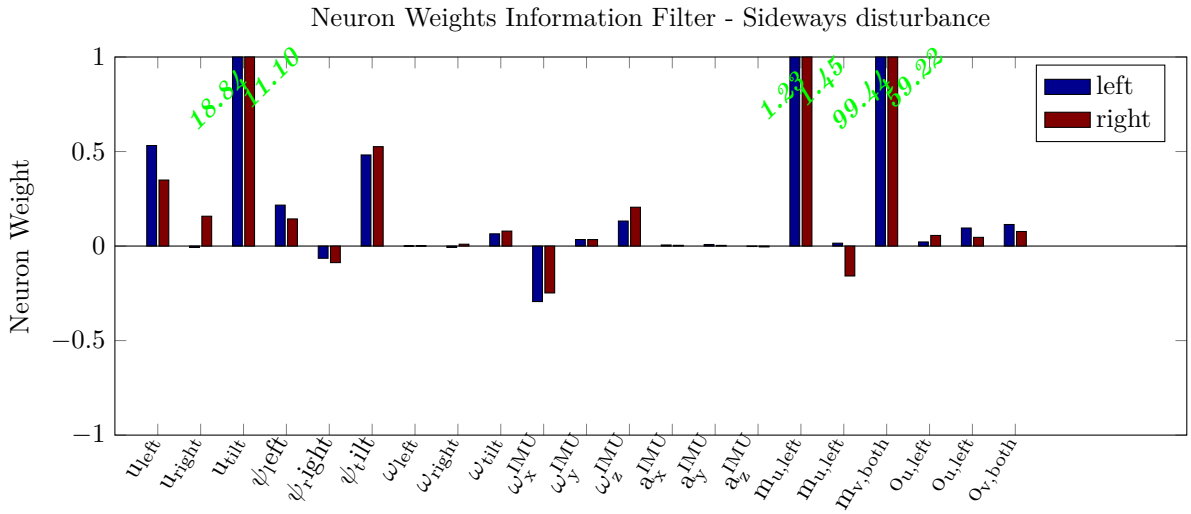


Figure D.15: The trained neural weights of the input value layer from the information filter. The input values can be found in table 4.1 in section 4.3. This is equivalent to the goal speed meant for the 3Mxl modules. These weights were obtained during a sideways disturbance with the cart with a variable marker position. The neuron values which do not fall within the plot range of [-1,1], have been clipped and their real value have been displayed next to the bars.

Bibliography

James S Albus. A theory of cerebellar function. *Mathematical Biosciences*, 10(1):25–61, 1971.

James S Albus. A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227, 1975.

Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

R Batllori, Craig B Laramee, W Land, and J David Schaffer. Evolving spiking neural networks for robot control. *Procedia Computer Science*, 6:329–334, 2011.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

Andreea C Bostan, Richard P Dum, and Peter L Strick. Cerebellar networks with the cerebral cortex and basal ganglia. *Trends in cognitive sciences*, 17(5):241–254, 2013.

Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

Lars Bretzner and Tony Lindeberg. Feature tracking with automatic selection of spatial scales. *Computer Vision and Image Understanding*, 71(3):385–392, 1998.

Richard R Carrillo, Eduardo Ros, Christian Boucheny, and Olivier J Coenen. A real-time spiking cerebellum model for learning robot control. *Biosystems*, 94(1):18–27, 2008.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

E. D'Angelo. Rebuilding cerebellar network computations from cellular neurophysiology. *Front Cell Neurosci*, 4:131, 2010. D'Angelo, Egidio eng Switzerland 2010/11/26 06:00 Front Cell Neurosci. 2010 Nov 4;4:131. doi: 10.3389/fncel.2010.00131.

Paul Dean and John Porrill. Evaluating the adaptive-filter model of the cerebellum. *The Journal of physiology*, 589(14):3459–3470, 2011.

Kenji Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current opinion in neurobiology*, 10(6):732–739, 2000.

John C Eccles et al. The cerebellum as a neuronal machine. 1967.

Selim Eskiizmirliler, N Forestier, Bertrand Tondu, and Christian Darlot. A model of the cerebellar pathways applied to the control of a single-joint robot arm actuated by mckibben artificial muscles. *Biological cybernetics*, 86(5):379–394, 2002.

Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. 1989.

Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

Maarten A Frens and Opher Donchin. Forward models and state estimation in compensatory eye movements. *Frontiers in cellular neuroscience*, 313, 2009.

Maarten Anthonie Frens, Anil Lekhram Mathoera, and Johannes van der Steen. Floccular complex spike response to transparent retinal slip. *Neuron*, 30(3):795–801, 2001.

M Fujita. Adaptive filter model of the cerebellum. *Biological cybernetics*, 45(3):195–206, 1982.

WERNER Graf, JOHN I Simpson, and CHRISTOPHER S Leonard. Spatial organization of visual messages of the rabbit's cerebellar flocculus. ii. complex and simple spike responses of purkinje cells. *J Neurophysiol*, 60(6):2091–2121, 1988.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

Berthold Horn. *Robot vision*. MIT press, 1986.

Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

M. K. Horne and E. G. Butler. The role of the cerebello-thalamo-cortical pathway in skilled movement. *Prog Neurobiol*, 46(2-3):199–213, 1995. Horne, M K Butler, E G eng Review ENGLAND 1995/06/01 Prog Neurobiol. 1995 Jun;46(2-3):199-213.

Martin Hülse, Steffen Wischmann, and Frank Pasemann. Structure and function of evolved neuro-controllers for autonomous robots. *Connection Science*, 16(4):249–266, 2004.

Hiroshi Imamizu, Tomoe Kuroda, Satoru Miyauchi, Toshinori Yoshioka, and Mitsuo Kawato. Modular organization of internal models of tools in the human cerebellum. *Proceedings of the National Academy of Sciences*, 100(9):5461–5466, 2003.

Masao Ito. Cerebellar circuitry as a neuronal machine. *Progress in neurobiology*, 78(3):272–303, 2006.

Kenji Iwadate, Ikuo Suzuki, Michiko Watanabe, Masahito Yamamoto, and Masashi Furukawa. An artificial neural network based on the architecture of the cerebellum for behavior learning. In *Soft Computing in Artificial Intelligence*, pages 143–151. Springer, 2014.

G. A. Jacobson, D. Rokni, and Y. Yarom. A model of the olivo-cerebellar system as a temporal pattern generator. *Trends Neurosci*, 31(12):617–25, 2008. Jacobson, Gilad A Rokni, Dan Yarom, Yosef eng Research Support, Non-U.S. Gov't Review England 2008/10/28 09:00 Trends Neurosci. 2008 Dec;31(12):617-25. doi: 10.1016/j.tins.2008.09.005. Epub 2008 Oct 25.

Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

Ravi Kaushik, Marek Marcinkiewicz, Jizhong Xiao, Simon Parsons, and Theodore Raphan. Implementation of bio-inspired vestibulo-ocular reflex in a quadrupedal robot. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4861–4866. IEEE, 2007.

M Kawato. Cerebellum: Models. *Encyclopedia of Neuroscience*, 2(9):757–767, 2009.

Mitsuo Kawato and Hiroaki Gomi. A computational model of four regions of the cerebellum based on feedback-error learning. *Biological cybernetics*, 68(2):95–103, 1992.

Jeff Krichmar. The cognitive anteater robotics laboratory (carl) at the university of california, irvine. *IEEE Intelligent Informatics Bulletin*, 14(1):1–4, 2013.

Igor Labutov, Ravi Kaushik, Marek Marcinkiewicz, Jizhong Xiao, Simon Parsons, and Theodore Raphan. Combinning linear vestibulo-ocular and opto-kinetic reflex in a humanoid robot. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 132–137. IEEE, 2008.

RJ Leigh and DS Zee. The neurology of eye movements. 2006.

Alexander Lenz, Sean R Anderson, Anthony G Pipe, Chris Melhuish, Paul Dean, and John Porrill. Cerebellar-inspired adaptive control of a robot eye actuated by pneumatic artificial muscles. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(6):1420–1433, 2009.

S. G. Lisberger. Internal models of eye movement in the floccular complex of the monkey cerebellum. *Neuroscience*, 162(3):763–76, 2009. Lisberger, S G eng Howard Hughes Medical Institute/ Research Support, Non-U.S. Gov't Review 2009/04/02 09:00 Neuroscience. 2009 Sep 1;162(3):763-76. doi: 10.1016/j.neuroscience.2009.03.059. Epub 2009 Mar 29.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

Niceto R Luque, Jesús Alberto Garrido, Richard R Carrillo, Silvia Tolu, and Eduardo Ros. Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise. *International journal of neural systems*, 21(05):385–401, 2011.

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

Manto. Consensus paper roles of the cerebellum in motor control Ůthe diversity of ideas on cerebellar involvement in movement. *The Cerebellum*, 11(2):457–487, 2012.

David Marr. A theory of cerebellar cortex. *The Journal of physiology*, 202(2):437–470, 1969.

Pietro Mazzoni, Richard A Andersen, and Michael I Jordan. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437, 1991.

Jeffrey L McKinstry, Gerald M Edelman, and Jeffrey L Krichmar. A cerebellar model for predictive motor control tested in a brain-based device. *Proceedings of the National Academy of Sciences of the United States of America*, 103(9):3387–3392, 2006.

RC Miall and Daniel M Wolpert. Forward models for physiological motor control. *Neural networks*, 9 (8):1265–1279, 1996.

RC Miall, DJ Weir, Daniel M Wolpert, and JF Stein. Is the cerebellum a smith predictor? *Journal of motor behavior*, 25(3):203–216, 1993.

Marco Molinari, Maria G Leggio, and Michael H Thaut. The cerebellum and neural networks for rhythmic sensorimotor synchronization in the human brain. *The Cerebellum*, 6(1):18–23, 2007.

Carmine Mottolese, Nathalie Richard, Sylvain Harquel, Alexandru Szathmari, Angela Sirigu, and Michel Desmurget. Mapping motor representations in the human cerebellum. *Brain*, 136(1):330–342, 2013.

John Nassour, Vincent Hugel, Fethi Ben Ouezdou, and Gordon Cheng. Qualitative adaptive reward learning with success failure maps: applied to humanoid robot walking. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(1):81–93, 2013.

Hideaki Ogasawara, M Kawato, et al. Systems biology perspectives on cerebellar long-term depression. *Neurosignals*, 16(4):300–317, 2008.

Aikaterini Pachi and Tianjian Ji. Frequency and velocity of people walking. *Structural Engineer*, 83 (3), 2005.

José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 314–317. IEEE, 2000.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. ISBN 9780136042594. URL http://books.google.nl/books?id=8jZBksh-bUMC.

Yeon Geol Ryu, Hyun Chul Roh, Si Jong Kim, Kwang Ho An, and Myung Jin Chung. Digital image stabilization for humanoid eyes inspired by human vor system. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 2301–2306. IEEE, 2009.

Erik Schuitema, Martijn Wisse, Thijs Ramakers, and Pieter Jonker. The design of leo: a 2d bipedal walking robot for online autonomous reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3238–3243. IEEE, 2010.

G Schweigart, T Mergner, I Evdokimidis, S Morand, and W Becker. Gaze stabilization by optokinetic reflex (okr) and vestibulo-ocular reflex (vor) during active head rotation in man. *Vision research*, 37 (12):1643–1652, 1997.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

M. Shidara, K. Kawano, H. Gomi, and M. Kawato. Inverse-dynamics model eye movement control by purkinje cells in the cerebellum. *Nature*, 365(6441):50–2, 1993. Shidara, M Kawano, K Gomi, H Kawato, M eng Research Support, Non-U.S. Gov't ENGLAND 1993/09/02 Nature. 1993 Sep 2;365(6441):50-2.

J Sola and J Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *Nuclear Science, IEEE Transactions on*, 44(3):1464–1468, 1997.

Kenneth Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Efrat Taig, Michael Kuper, Nina Theysohn, Dagmar Timmann, and Opher Donchin. Deficient use of visual information in estimating hand position in cerebellar patients. *The Journal of Neuroscience*, 32(46):16274–16284, 2012.

Kaoru Takakusaki and Toshikatsu Okumura. Neurobiological basis of controlling posture and locomotion. *Advanced Robotics*, 22(15):1629–1663, 2008.

Emanuel Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915, 2004.

Richard Q van der Linde and Arend L Schwab. Multibody dynamics b. 1998.

Jilles Vreeken. Spiking neural networks, an introduction. *Institute for Information and Computing Sciences, Utrecht University Technical Report UU-CS-2003-008*, 2002.

Mark F Walker, Jing Tian, Xiaoyan Shan, Rafael J Tamargo, Howard Ying, and David S Zee. The cerebellar nodulus/uvula integrates otolith signals for the translational vestibulo-ocular reflex. *PLoS One*, 5(11):e13981, 2010.

Xin Wang, Joris van de Weem, and Pieter Jonker. An advanced active vision system imitating human eye movements. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6. IEEE, 2013.

TJ Watson and Michael Strickland. *Modeling for Gazebo : A Design Guide for Proper Exporting from Solidworks for Gazebo Simulation*. URL http://blogs.solidworks.com/teacher/wp-content/uploads/sites/3/WPI-Robotics-SolidWorks-to-Gazebo.pdf.

Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

JG Ziegler and NB Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.