Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics

Learning Eye-Hand Coordinate Transformation in a Simulated Humanoid Robot

Master's thesis

2019
Bc. Martin Kellner

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Učenie transformácie súradníc ruka-oko v simulátore humanoidného robota

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2019

Bc. Martin Kellner

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Bc. Martin Kellner
Study programme: Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study: Applied Informatics
Type of Thesis: Diploma Thesis
Language of Thesis: English
Secondary language: Slovak

Title: Learning Eye-Hand Coordinate Transformation in a Simulated Humanoid Robot

Annotation: Sensory processing in the brain entails coordinate transformations (the change of reference frame for a stimulus, e.g. retinotopic to body-centered, effected through knowledge about an intervening variable, e.g. gaze position). This ability is crucial in cognitive robotics in order to endow robots with an ability to operate autonomously in the 3D space.

Aim: 1. Study the cognitive neuroscience literature related to the frames of reference (coordinate systems).
2. Implement and train an artificial neural network model on simulated data relating visual and proprioceptive information, performing a coordinate transformation in hand-eye related task, using the iCub robotic simulator.
3. Evaluate and analyze the behavior of the trained model.

Literature: Švec M., Farkaš I. (2014). Calculation of object position in various reference frames with a robotic simulator. In Proceedings of the 36th Annual Conference of the Cognitive Science Society, Quebec, Canada.
Tikhanoff V., Fitzpatrick P., Nori F., Natale L., Metta G., & Cangelosi A. (2008). The iCub humanoid robot simulator. Advanced Robotics, 1(1), 22-26.
Malinovská, K., Malinovský, Ľ., Farkaš, I. (2018). Towards more biologically plausible error-driven learning for artificial neural networks. In International Conference on Artificial Neural Networks (ICANN), pp. 1-4, LNCS, Springer.

Supervisor: prof. Ing. Igor Farkaš, Dr.
Department: FMFI.KAI - Department of Applied Informatics
Head of department: prof. Ing. Igor Farkaš, Dr.

Assigned: 16.04.2018

Approved: 20.09.2018 prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

Student

Supervisor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Kellner
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Learning Eye-Hand Coordinate Transformation in a Simulated Humanoid Robot
Učenie transformácie súradníc ruka-oko v simulátore humanoidného robota

Anotácia: Spracovanie senzorických podnetov v mozgu zahŕňa transformáciu súradníc (referenčných rámcov, napr. z retinotopického na telo-centrický, ovplyvnenej modulačnou premennou, napr. pozíciou očí). Táto schopnosť je kľúčová v kognitívnej robotike, ak chceme vybaviť robota schopnosťou operovať autonómne v 3D priestore.

Cieľ: 1. Naštudujte si problematiku z kognitívnej neurovedy o referenčných rámcoch (súradnicových systémoch).
2. Implementujte a natrénujte model umelej neurónovej siete, ktorá sa naučí vzťah medzi vizuálnou a proprioceptívnou informáciu, vykonávajúc prepočet súradníc v úlohe týkajúcej sa ruky a očí, s využitím simulovaného robota iCub.
3. Vyhodnoťte a analyzujte správanie natrénovaného modelu.

Literatúra: Švec M., Farkaš I. (2014). Calculation of object position in various reference frames with a robotic simulator. In Proceedings of the 36th Annual Conference of the Cognitive Science Society, Quebec, Canada.
Tikhanoff V., Fitzpatrick P., Nori F., Natale L., Metta G., & Cangelosi A. (2008). The iCub humanoid robot simulator. Advanced Robotics, 1(1), 22-26.
Malinovská, K., Malinovský, Ľ., Farkaš, I. (2018). Towards more biologically plausible error-driven learning for artificial neural networks. In International Conference on Artificial Neural Networks (ICANN), pp. 1-4, LNCS, Springer.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 16.04.2018

Dátum schválenia: 20.09.2018 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

DECLARATION: I hereby declare that this thesis is my own work and that all the sources I have used or quoted have been indicated and acknowledged as complete references.

. . . . . .

ACKNOWLEDGEMENT: First, I would like to thank my supervisor, prof. Ing. Igor Farkaš, Dr., for continuous support and helpful advice. I am also grateful to the members of my family and all my friends, thanks for all what you are doing for me. I would like also to thank God for giving me strength, ability and patience to persevere and complete this thesis satisfactorily.

Abstract

If we want to manipulate with an object we have to know the position of an object in the space. The initial information about the position is usually provided by retinal projections. One of the many questions, that neuroscientists are concerned with, is how the brain encodes particular information at the level of neuronal population. For this purpose, neuroscientists used the term named *reference frame*. In this thesis, we are concerned with learning one of essential tasks that processes sensory signals, which is named *coordinate transformation*. We experimented with a novel biologically plausible training algorithm that allows us to train a bidirectional models. We trained three different models on datasets gained by using the simulator of the iCub humanoid robot. The first model learns one-to-one coordinate transformation between object position represented proprioceptively by arm position, with an object held in the hand, and the head position, with centered eye position, focusing on the object. In this model, all joint positions, as well as retinal object positions, were encoded by real numbers. Thanks to the model, the robot is capable of looking at its palm and taking into account the fact that the model is bidirectional; the robot can also move its palm into its visual field. The second version of the model was tested in two variants, differing in the representation of joint positions, in both the proprioceptive system (arm position) and the eye position. Both variants implemented the same, more complex transformation, namely the coordinate transformation between object position represented by arm position (with an object in the palm) and the retinal coordinate frame, modulated by eye position and vergence. Hence, in this case, the robot does not have to look at the object, so its image can hit any position on the retinas, and hence the object is perceived peripherally. The second variant of the more complex model used the technique called the *pupulation coding vector* for representing of the stimuli. All models learned the transformation, whereas, the last model reached less accuracy. Despite that, some interesting similarities with other works were observed concerining the *receptive fields* of hidden neurons.

Keywords: coordinate transformation, reference frame, artificial neural network, receptive field

Abstrakt

Na manipuláciu s objektmi potrebujeme vedieť, kde sa dané objekty nachádzajú v priestore, a práve takáto informácia je zväčša poskytovaná naším zrakom. Jednou z mnohých otázok, ktorými sa neurovedci zaoberajú je, ako sú jednotlivé informácie v mozgu kódované na úrovni populácie neurónov. Pre účel skúmania tohto problému sa v neurovede využíva pojem *referenčného rámca*. Pre porovnanie pozície ruky a objektu musí mozog vykonať buď transformáciu z jedného referenčného rámca do druhého, alebo sú obe informácie pretransformované do takej reprezentácie, kde môžu byť porovnané. Tento problém sa označuje aj *koordinácia súradníc* oko-ruka. V našej práci sa venujeme práve učeniu takýchto transformácií pomocou *umelých neurónových sieti*. Na trénovanie umelých modelov sme využili nový, biologicky inšpirovaný algoritmus, ktorý umožňuje trénovať obojsmerné siete. Pre účely testovania a získania realistických dát sme použili simulátor humanoidného robota iCub. Výsledkom našich experimentov boli tri modely. Prvý model transformuje pozíciu o polohe ruky a bodu, na ktorý je upretý zrak. Robot je vďaka tejto sieti schopný pozrieť sa na svoju ruku tak, že natáča otáča hlavou bez využitia rotácie očí. Keďže je však tento model obojsmerný, robot je taktiež schopný posunúť ruku približne do stredu svojho rôzneho poľa. Ďalšie dva modely pozostávali z transformácie bodu v priestore, ktorý je reprezentovaný smerom natočenia očí a pozície daného bodu na sietniciach do takej konfigurácie kĺbov ruky, ktorá umiestni centrum dlane na približne rovnaký bod v priestore. Dve natrénované siete sa od seba líšili iba v spôsobe kódovania jednotlivých stimulov. Prvá z nich po natrénovaní umožňuje posun dlane na pozíciu videného objektu, a uprieť pohľad takým spôsobom, že je robot schopný pozrieť sa na ruku tak, že obraz centra dlane dopadne vždy približne na rovnakú pozíciu na sietniciach. Tretí model využíval techniku populačného kódovania signálov, ale nedosiahol tak presné transformácie ako prvý model. No ukázalo sa, že *receptívne polia* skrytých neurónov dosahovali podobné vlastnosti v porovnaní s inými prácami zaoberajúcimi sa transformáciou súradníc.

Kľúčové slová: transformácia súradníc, referenčný rámec, neurónové siete, receptívne polia

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial neural network
API	Application programming interface
BAL	Bidirectional Activation-based Learning algorithm
BP	Backpropagation algorithm
CR	Cartesian controller
DoF	Degree of freedom
GF	Gain field
MLP	Multi-layer percepton
ODE	Open Dynamics Engine
RCF	Receptive field
RF	Reference frame
PMd	Porsal premotor cortex
UBAL	Universal Bidirectional Activation-based Learning
YARP	Yet Another Robot Platform

Introduction

In this thesis, we are dealing with an important operation that is performed by the brain in order to allow us to interact with the surrounding space. This process, known as *coordinate transformation*, is responsible for the transformation of information encoded in different systems at the level of the brain's neuronal populations. Particularly, we focus on a specific transformation called *eye-hand coordinate transformation*. The eye-hand coordination is performed every time when we want to grab a seen object. In this case, the brain has to compare the retinal signals carrying the information about the object's position with the current position of the hand and generate the right activations of the hand's muscles.

We review theoretical bases and significant findings related to this transformation. Chapter 1 is focused on what the brain must take into account to carry out the transformation, what is hidden under the terms *gain modulation* and *reference frame*, and the chapter also reviews one of the most significant works in this topic. The coordinate transformation is very often examined by the training of *artificial neural networks*, and that is our approach as well. We use a biologically-inspired training algorithm to train bidirectional neural networks performing the task of the eye-hand coordinate transformation. All our models are trained and tested on data collected via the robotic platform named *iCub*. Artificial neural networks, the used algorithm, iCub and all other used methods are described in Chapter 2. Then, in Chapter 3, we provide detailed descriptions of our experiments where we evaluate all models and we compare our results with other findings. The trained models provide inspiration for how to develop eye-hand coordination within a robot system and their hidden neurons are used for pointing out similarities with the properties of neurons in the brain. Short Chapter 4 concludes the thesis.

Chapter 1

Theoretical focus and related works

1.1 Reference frames

In order to gain a better understanding of the term *reference frame* or *frame of reference*, consider the following situation. On a moving train there is a passenger who lets a book fall down and another person watching the situation is standing on the ground. From the perspective of the passenger the book is falling straight down, but the observer (the man standing on the ground) sees the book dropping down along a curved path because of the movement of the train. This situation has shown that the description of a physical phenomenon depends on the position from where the case is observed. Using reference frames (RF) on the previously illustrated example, we can say that the book is falling straight down to the train's RF but it is dropping along a curved path in the earth's RF. It can be said about the state of motion of the passenger that it is stationary in the train's RF but moving in the earth's RF (see Fig. 1) (Soechting and Flanders, 1992).

RFs are very often used by physicists and engineers but this term has also been adopted by neuroscientists. In the mathematical sense, the RF is similar to a co-ordinate system that is defined by a set of axes and an origin. The origin can be anywhere in space and the orientation of axes can be chosen arbitrarily because RF is only characterized by the state of motion relative to an object. Therefore, the train's RF that has been mentioned in the illustrated example has the same state of motion as the train, and any point in the RF can be defined by its position along each of the coordinate axes. Alternatively, the location of a point can be defined

by way of vectors, where a vector has two properties: magnitude and direction; the magnitude is the length of a line segment between the origin and the point, and its direction is from the origin to the point (Soechting and Flanders, 1992; Kumar and Barve, 2002).

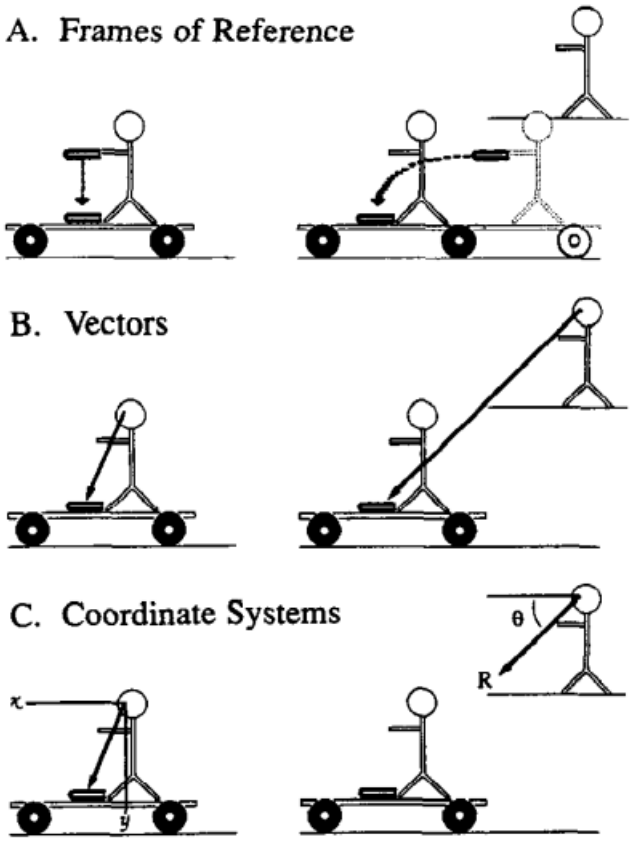


Figure 1: Schematic illustration of the spatial representations of objects in frames of reference (A), vectorially (B) and by coordinate systems (C). On the left, the RF moves with the passenger; on the right, the observer's RF is fixed to the earth. Adopted from Soechting and Flanders (1992).

In order to switch the RF into a term in neuroscience, we have to make some changes against the mathematical definition because neural systems do not report to the position of an object as a vector or as coordinates. Instead, neurons encoding visual space each reports to a restricted area of space known as *receptive field* or *response field* (RCF), thus, the firing activity of the neurons is changed depending on stimuli coming from their RCFs (Batista, 2002). If we want to identify in which RF, for example, a neuron encodes spatial information about the location of an object, then the firing activity of the neurons should stay the same as long as the image of the object falls on the same locus on the retina and stays constant. Once

that happened we can say the location is encoded in a *retinocentric* RF (Soechting and Flanders, 1992).

Different brain regions encode spatial locations in different RFs. In neuroscience RFs are mostly divided into two main groups: *ego-centric* and *allocentric* RFs. In an egocentric RF, the location of the object refers to the observer. At a neural level responses of the neural population are attached to a reference point (Commiteri et al., 2004). For instance, an eye-centred RF moves together with the eyes. Spatial locations are mostly encoded in egocentric RFs, especially space coding neurons in the parieto-frontal cortex are associated with RFs centred to the eye, the head or the hand (Colby, 1998). They have found neurons that encode space in RFs centred to parts of the body, in the monkey's posterior parietal cortex and in connected regions of the premotor cortex (Cohen and Andersen, 2002; Colby, 1998). Information encoded in an allocentric RF is described with respect to other objects, for example, the book is on the table. Coding of space in allocentric RFs is not so well-studied as egocentric RFs, but it has been shown (Marshall et al., 2000; Galati et al., 2000) that in the posterior parietal cortex, in the dorsal premotor cortex (PMd) and in early visual areas neurons report during object-based spatial judgement. Also it has been proven that during the solving of a complex task, such as landmark knowledge, orientation in large-scale space and navigation, some operations refer to both types of RFs, allocentric and egocentric, which are difficult to untangle (Commiteri et al., 2004).

1.2 Sensorimotor transformations

If humans, aminals or insects want to carry out basic operations to interact with the world they use their bodies and senses. To manipulate with an object, moving in space or doing other daily activities, the brain must continuously update the internal representation of the world, deal with restrictions that may occur while performing an operation and change body configuration to achieve the goal of the action. For instance, during the reaching of an object the human brain must take into account the location of the target and the current position of the particular hand to generate the right movement for the hand. One non-neuroscientist might

say that the operations as above mentioned are not very complicated because we do not need to put a lot of effort into getting the goal but when we look closely at the neural level the task is more complicated as it would seem. The information about the location of an object can be produced by stimuli from different modalities and each of these modalities provides neural information that is encoded in a different RF. Therefore, neuroscientists focus their research to clarify the following questions: How is the information represented in a RF at the neural level? How is the information from different modalities combined to encode a correct representation of space? How is the information in a particular RF transformed to another one? Coordinate transformation in neuroscience is a term used for the transformation from one RF to another. In the next section we closely discuss the eye-hand coordinate transformation that is crucial for many human daily activities.

1.2.1 Spatial Transformations for Eye–Hand Coordination

The eye-hand coordination is reviewed by Crawford et al. (2004) and Blohm et al. (2009) in very detail. The authors have brought assumptions about using gaze-centred representations during the process of coordinate transformation that is needed to perform such tasks as reaching or grabbing an object. In this chapter we closely discuss their conclusions.

In order to reach an object, the brain needs information about the location of the object being reached and the position of this object has to be encoded in relation to the body part performing the operation. Arms of primates are anchored to shoulders and the spatial information about an object comes primarily from projections on retinas, therefore, the brain must perform a coordinate transformation between at least two representations. First of them is the representation specifying the object's location relative to retinas and the second representation within witch the activations of arm muscles must be determined to reach the target. In this section we discuss how the brain deals with the transformation of visual stimuli into hand motion commands and what processes are included as grabbing, reaching or manipulation with objects.

A robot system controlling its arm to reach or to grab an object is usually simpler than the human control system. Such robotic system is ordinarily reduced only by

using visual feedback. Basically, driving the hand to a point that is seen by the visual system is carried out only by comparing the current position of the hand with the target position. This solution may be sufficient because the speed of visual feedback of such robotic system is not so dramatically limited as the brain. The limitation of the robotic visual feedback is caused only by processing the time of the processor and the speed of the electrical flow, whilst the speed of neural conduction and processing time in a real primate brain is not adequate to perform a fast hand movement. Thus using just visual feedback would probably cause that the hand would be out of the field of vision before a new visual feedback would arrive to accurately update the movement. Consequently, the brain of primates cannot use only visual feedback driving the hand's movements and it must find another way to carry out correct and fast movements. The answer is the usage of internal models of the physical system representing the world based on initial conditions and eye-hand coordinate transformation performing in the feedforward way. Nevertheless, we cannot say that visual feedback does not contribute to the guiding hand movements because visual feedback actually helps to achieve the best performance of grabbing a goal and is also essential for dealing with unexpected events or danger that might occur during the performance of a particular movement.

We still do not know all the processes that are included in the brain to produce a correct transformation from eyes to hands and how these processes are exactly carried out at the neural level. Regardless, there are several conclusions made by neuroscientists that we discuss in the following two conceptual steps.

Early visual representation for arm movements

For operations as coordinate transformation we usually describe internal models in the brain by using the concept of RF (1.1). The final RF for eye-arm coordination is associated with points in the upper arm and shoulder and not with a hand-centred representation as we could mistakenly think. The first stimuli are controlled in a retinal RF and the first problem occurs here: *gaze shifting*. Every time when the eyes (or/and head) shift(s) with the gaze, the visual relationship between the sensory apparatus and the external world changes (Hallett and Lightstone, 1976). One solution, how the brain could deal with this problem, is to wait until the gaze

shifting finishes and then update its visual information (O'Regan and Noe, 2001) but that might cause that the original target of interest would be moved into a less-sensitive peripheral retina or even out of the visual field. It is good to realise that this solution would produce also long lags in processing the time and redundant visual computations. Therefore, the necessary representations must be stored for future action either in an eye-independent form, or in such form that is internally updated if the gaze is shifted, regardless of the fact that they were caused by movements of eyes or the head (Duhamel et al., 1992).

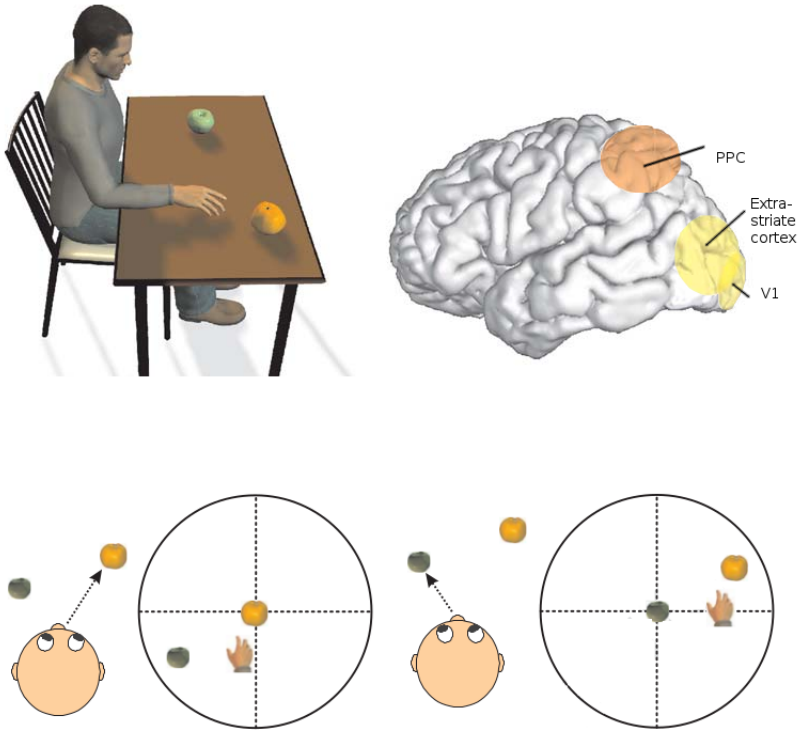


Figure 2: Gaze-centered encoding of reach space. (a) Drawing depicting the egocentric visual directions of the hand, an orange, and an apple, as shown by the gray arrows. (b) Side view of a human brain showing areas (highlighted in yellow and orange) that encode reach space in gaze-centered coordinates: V1, striate cortex; PPC, posterior parietal cortex. (c) The consequence of an eye movement on the gaze-centered representation of the visual field. The head/eye diagrams depict current gaze position; the circles represent the gaze-centered representation of this visual scene (dotted lines represent visual horizontal and vertical axes and intersect at the fovea). If the person in the upper diagram looks at the orange, the hand and apple are represented in the left visual field (upper circle). In contrast, if the person fixates the apple (lower diagram), the orange and hand are now represented in the right visual field (lower circle). If the orange and the hand were no longer visible when the eye movement occurred, the brain would need to remap their position by taking the intervening eye movement into account. Adopted from Blohm et al. (2009).

The next raising question is: How the system in the human brain, which appears to be used for early planning of pointing or reaching, stores early motor representations during eye movements? In order to answer the question, Henriques et al. (1998) performed the following experiment. Human subjects that were carrying out the experiment, were asked to point toward a location using one hand in total darkness. At the beginning of the experiment, the target location that was pointed to, was centred to subject's fovea. Then subjects shifted the gaze by moving their eyes and then pointed toward the remembered location.

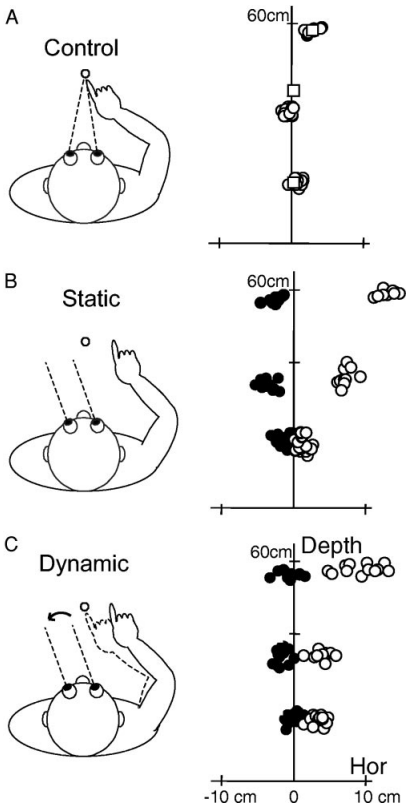


Figure 3: Gaze-centered pointing performance in humans for targets in near and far space. Left column: 3 tasks, where subjects either (A) look directly toward the target before pointing (control task) or (B) view the target peripherally before pointing (Static Condition) or (C) foveate the target, then shift their gaze and then pointing (Dynamic Condition). Right Column: final fin- gertip positions (circles) in the horizontal plane of one subject these conditions. Squares represent the actual target locations of the two reaching targets and the fingertip location for pointing toward the continuously illuminated pointing target. In static and dynamic tasks, open circles indicate 20 leftward eye fixation; solid circles represent data for 20 rightward eye fixation. Targets were located at 2 m, 42 cm, and 15 cm. Adopted from Crawford et al. (2004).

Authors of the experiment tried to find out whether the responses of such a pointing task are affected by an intervening eye displacement. They compared them to the pointing to the remembered foveal targets or pointing to the retinally peripheral targets. If subjects were pointing toward the location using an updated gaze-centred representation, pointing behaviour would echo pointing to peripheral targets. Whereas pointing by using a nonretinal representation would cause no effect. As shown using Fig. 3, this experiment has cleared up the question about the early motor representation and it has supported the idea of using an updated gaze-centred RF. Other studies have also come up with findings where during point-

ing to auditory and proprioceptive targets the gaze-centred updating has been also recorded (Pouget et al., 2002). Evidence of coding the spatial information of an object in a gaze-centred RF can be also found in both the early visual pathway (i.e., retina, lateral geniculate nucleus, striate cortex) and also later in the visual pathway (e.g., extrastriate and parietal cortex).

The gaze-centred representation naturally moves depending on the gaze because the retinal projection of the project depends on the gaze. Therefore, the positions of the target being reached can be defined by the target's direction (horizontal and vertical angular eccentricity) relative to *fovae* and by the distance between the goal and the eyes. Early visual areas carry out the computation of target direction as seen by a virtual eye which we can imagine to be placed between the right and the left eye, whereas the required distance is determined by taking into account monocular information (accommodation, relative object size, shading, perspective, etc.) and binocular information (retinal disparity and convergence). Later, during the early movement planning, the signals that contain this information seem to be merged into a single gaze-centred representation of the space in the posterior parietal cortex.

As we have mentioned above, the gaze-centred representation must be updated to maintain a stable representation of the world. Therefore, the brain must update this representation all the time the eyes are moving. This process, which performs during the rotation of eyes, is known as *updating* or *remapping* and it also occurs during the movements of the head and the body. An illustrating example is in Fig.1(c). The observer originally looks at the orange, and the apple is placed left-down relatively to the orange, thus the orange corresponds to the centre of the gaze-centred representation, and the apple is located on the lower-left quadrant of the visual field. A fast orientation movement approaching the apple causes that the objects are remapped by the same rotation as the movement but in the opposite direction. That means the apple is mapped at the centre and the orange is located on the upper-right quadrant. It has also been shown that the updating can be achieved even if the vision is removed, thus only by using information about the eye movements.

Developing the reach plan

In order to correctly generate the reachable plan, the information about the initial position of the hand must be included. Buneo et al. (2002) suggested that the comparison between initial hand location and gaze-centred representation of the visual target is done earlier and in a gaze-centred RF, and even if the hand is out of the visual field. That implies that the proprioceptive signals carrying the information about hand location must be transformed into gaze-centred coordinates. These kinds of signals were found in the process of a gaze-centred transformation in parietal area 5, but these findings regarding the earlier comparison do not claim that the next RF transformation is not required, or needed to reach the target.

The next important problem, the brain has to deal with in order to accurately generate and perform a reaching plan, is how the ego-centre is translocated during the head rotation. This problem has to be taken into account because the rotation of the head causes eye's translation with respect to the shoulder. Ignoration of eye translational aspect would cause erroneous reach pattern at noncentral head position.

A summary of conceptual and physiological models of visually guided reaching movement is described in Fig. 4. The first stage corresponds to the fact that the 3D representations of target direction are stored and maintained in a RF. The second stage shows an illustration of the transformation according to the above-mentioned Buneo et al. (2002)'s schema to compute the hand displacement in a RF coordinate. There are other RF transformations from gaze coordinates to shoulder representations that are not included in the figure but they are necessary to reflect the eye-head-shoulder system. Nevertheless, such a model may be useful to tell us which signals are required and used for such a transformation but it is obvious that all such kinds of explicit intermediate representations are not used by the brain directly, and they also do not tell us how and where the mentioned signals are coded.

To design a complete 3D reachable plan, the brain must know the amplitude of the desired movement and the direction. It has been shown that in the dorsal premotor cortex the direction and the amplitude are encoded together inside the same neurons, but there is also an evidence that the direction and the amplitude might also be encoded independently. One explanation of this contradiction might

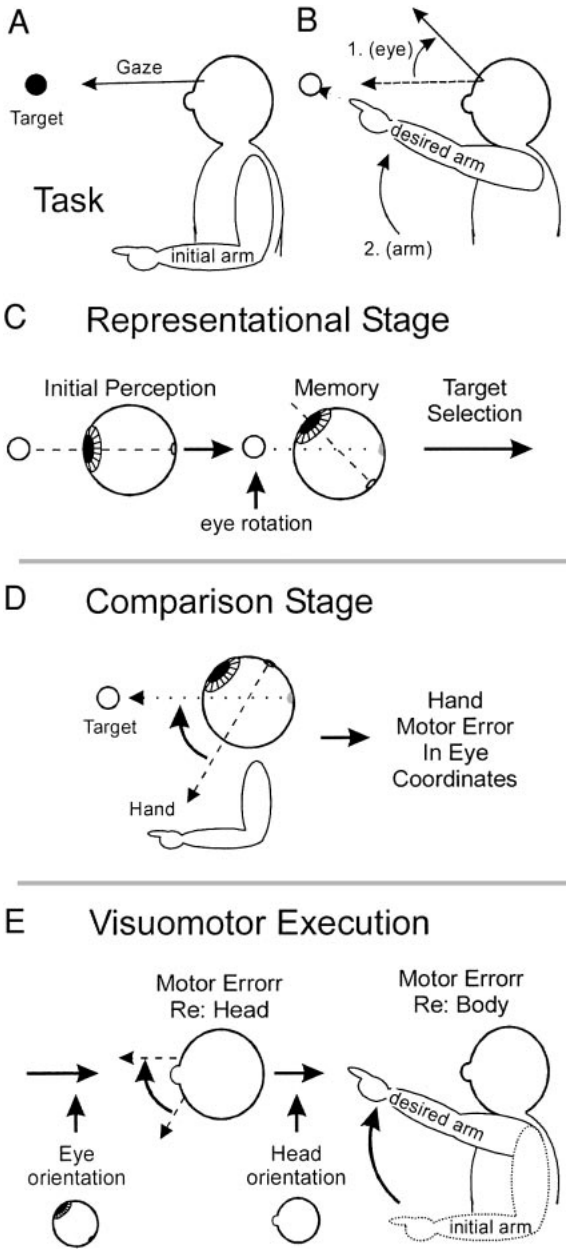


Figure 4: Conceptual scheme for spatial transformations in eye–hand coordination. To illustrate the model, consider the following "task": a subject looks at a briefly flashed target (F) with the arm at resting position (A). Then (B) the subject makes 1) an upward eye movement, followed by 2) a reaching or pointing movement toward the remembered target location (E). We hypothesize that the brain uses the following stages to do this. C: an early representational stage. Target location is stored in eye coordinates, such that this representation (E) must be counterrotated (updated) when the eye rotates. D: comparison stage. Updated target representation (E) is compared with an eye-centered representation of current hand location to generate "hand motor error" in eye coordinates (Buneo et al. 2002). E: visuomotor execution stage. "Hand motor error" signal is rotated by eye orientation and head orientation (or perhaps by gaze orientation) to put it into a body coordinate system appropriate for calculating the detailed inverse kinematics and dynamics of the movement. This last stage would also have to include internal models of the geometry. Adopted from Crawford et al. (2004).

be that, at the neural level, amplitude and direction are encoded together but are used by different mechanisms to generate muscle activation. What must be taken to account is the rotation of the head and body and also values of correct forces for specific muscles. Amplitude and direction are properties of vectors, therefore, we can reformulate the problem of designing a reachable plan into the terminology of vectors as that is often done.

The conclusion of the problem of developing a reachable plan is that the brain must design a hand movement vector, which depends on the difference between the hand location and the location of the target. The areas in the brain that are responsible for calculating the vector, have to contain encoded spatial information

of the hand and the position of the goal. In addition, the spatial information must be represented in the same RF. To determine the movement vector, the particular areas might use information from different sources, and also perform multisensory integration in order to get the most likely estimation of both the hand and the target location. One of the approaches that help us to examine how the population performs the task of coordinate transformation is using artificial neural networks (ANN). During studying of neural encoding of a cognitive function, we very often focus on changing response magnitude of neurons known as *gain field* (GF) what we are discussing in the next section.

1.3 Gain fields

The term *gain field* comes from Andersen and Mountcastle (1983). The authors tested the visual RCF for a specific neuron at different eye positions. The RCF of a neuron or a neural population can be described as a specific region in sensory space of which stimuli can produce an influence on the activity of the single neuron or on the neural population. For instance, activations of neurons encoding the gaze-centered representation of a target's location should be constant while gaze angles stay constant, contrariwise, firing activities of the neurons should change with every motion of the eyes. During the mentioned experiment the frequency of the neural action potential was changing in a manner of multiplication the frequency by a gaze angle that is scaled by a constant. Both of the shape and the locations of the visual RCF were unchanged, only the RCF was scaled by some *gain* factor. This phenomenon was the first time characterized in neurons inside LIP and visual area 7a. In another study, Zipser and Andersen (1988) examined the task of coordinate transformation from visual target position on retinas and signals of gaze position into the position represented by the target location in a space-fixed RF by an ANN. The detailed analysis of their neural model showed that the network was able to develop visual RCFs modulated by the position of the eyes in a very close way compared to the modulation in parietal cortex. Fig. 5 illustrates the results of their study.

At the neural population level, GFs modulate the responses of every particular

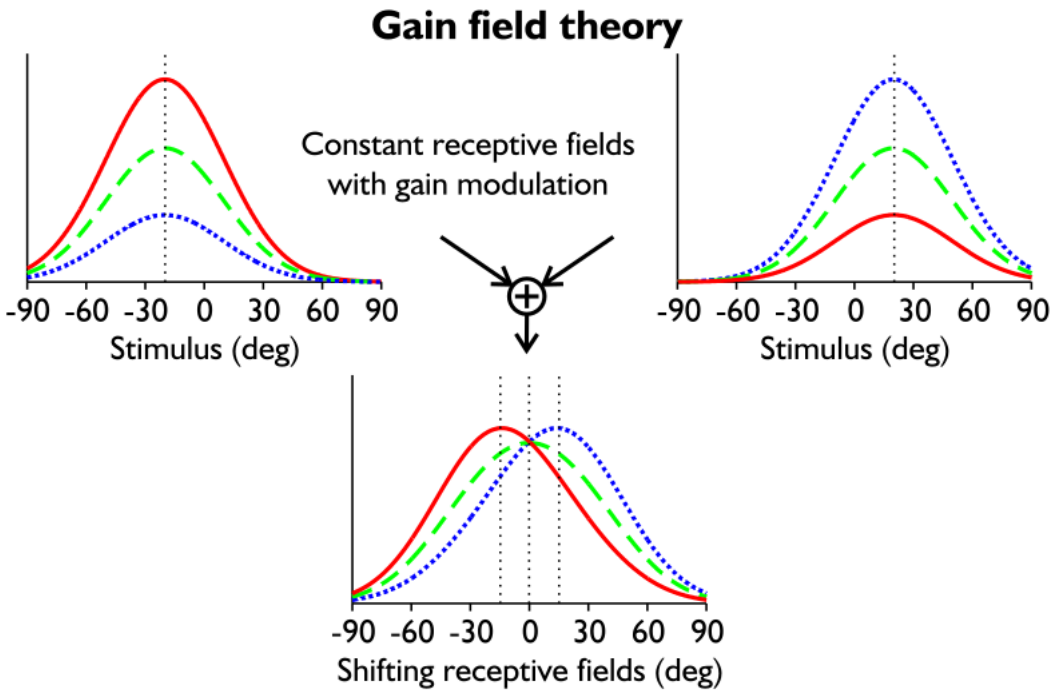


Figure 5: Working principle of gain fields, based on Zipser and Andersen (1988). The upper part of the panel shows the hypothetical receptive fields of two neurons that are gain modulated (e.g., by eye or hand position) in opposite ways without shifting. For example, the three lines in each graph could represent visual receptive fields mapped relative to gaze at a leftward eye position (red solid line), a central eye position (green dashed line), and a rightward eye position (blue dotted line). Here, eye position modulates the strength of response of two neurons, but does not cause them to shift. However, summation of these two gain-modulated neural responses results in shifting receptive fields in the output, e.g., eye position (or in other cases hand position) has shifted the receptive field. Adopted from Blohm and Crawford (2009).

neuron in a whole population responding to the same RCFs, and therefore, the total output of the population can be increased or decreased that allows to regulate the total strength of the RCF. In the next steps, the brain can integrate such an output in various ways and use them in next computations. For instance, something similar to the output of the above mentioned ANN could be hypothetically compared with information about the current hand's location to produce the hand movement to reach a target. It has been observed that the effect of gain modulation is also produced by other types of signals, and GFs were found in many other brain areas (Blohm and Crawford, 2009).

1.3.1 Links between Gain Modulation and Coordinate Transformations

In order to demonstrate the role of gain modulation with the performance of the task of coordinate transformation in the brain, we will discuss again the results that have been introduced by Zipser and Andersen (1988). As mentioned, they trained a neural network to solve coordinate transformation from the input signals, gaze direction and retinal location to output stimuli in a body-centred RF. The model was trained using one of the major-used training algorithms in machine learning known as *backpropagation* (BP). This algorithm allows the learning of properties of *hidden* units (neurons) in order to map inputs to corresponding outputs with the effort to achieve as low error as possible. All of the output and input representations of stimuli were composed to match the measured response of neurons in the brain, thus the model was able to perform a coordinate transformation in the similar way as it is made by the visual system (see Fig. 6). After a successful training, the authors have shown that hidden units developed gaze GFs similar to those in PPC. This experiment has brought a significant contribution that supports the role of gain modulation within the process of solving a coordinate transformation in the brain (Salinas and Sejnowski, 2001).

Other works also took an effort to study similarities between real and artificial neurons, especially between their responses. Xing and Andersen (2000) trained a more complex neural model to represent two consecutive saccades separated by a delay interval. They used a recurrent neural network that is able to learn a sequential task through their internal *memory*. Input data combined a visual map in retinal coordinates, an auditory map in head-centred coordinates and eye-position units. The model was taught to produce outputs that encode simultaneous movements of both eyes (Xing and Andersen, 2000). After the training, they demonstrated that their network developed GFs and sensory and memory responses matched those of LIP neurons (Salinas and Sejnowski, 2001).

Salinas and Abbott (1995) studied the relationship between the coordinate transformation described in Fig. 6 and gain modulation. Their research was focused on neurons that are gain-modulated by the gaze and drive downstream neurons in another neural population included in the process of generating arm movements. They

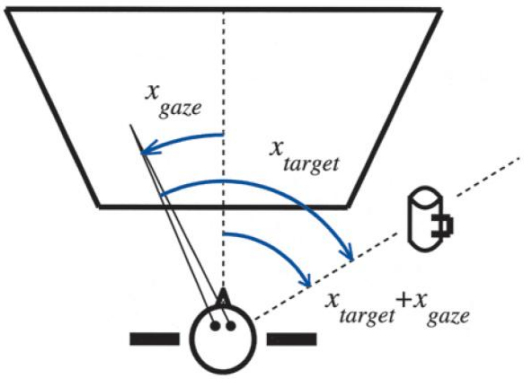


Figure 6: A coordinate transformation performed by the visual system. While reading a newspaper, you want to reach for the mug without shifting your gaze. The location of the mug relative to the body is given by the angle between the two dashed lines. For simplicity, assume that initially the hand is close to the body, at the origin of the coordinate system. The reaching movement should be generated in the direction of the mug regardless of where one is looking, that is, regardless of the gaze angle xgaze. The location of the target in retinal coordinates (i.e., relative to the fixation point) is xtarget, but this varies with gaze. However, the location relative to the body is given by xtarget + xgaze, which does not vary with gaze. Through this addition, a change from retinal, or eye-centered, to body-centered coordinates is performed. Adopted from Salinas and Sejnowski (2001).

considered a neural population that responds to the visual stimulus at the retinal location x_{target} and is modulated by the gaze angle x_{gaze} that constitutes gain modulation. Then all responses r of such a population can be described by the following equation:

$$r = f\left(x_{target} - a\right) g\left(x_{gaze}\right), \tag{1.1}$$

where $g\left(x_{gaze}\right)$ is the function of which result is the GF of particular neuron, and $f\left(x_{target} - a\right)$ represents the response of neurons as a curve with a single pick (see Fig. 7). Then, downstream neurons driven by a population as mentioned must have responses R theoretically formed by the following function of x_{targer} and x_{gaze}:

$$R = F\left(c_1 x_{target} + c_2 x_{gaze}\right), \tag{1.2}$$

where both constants c_1 and c_2 depend on the synaptic weight, and the RCF of a downstream neuron is represented as the peaked function R.

This mathematical formulation is only a simplification of the problem but the authors found conditions under which this can happen and they experimentally confirmed that the downstream neurons encode the sum of $x_{target} + x_{gaze}$, which means the downstream neurons that must encode target in a body-centred RF respond as a

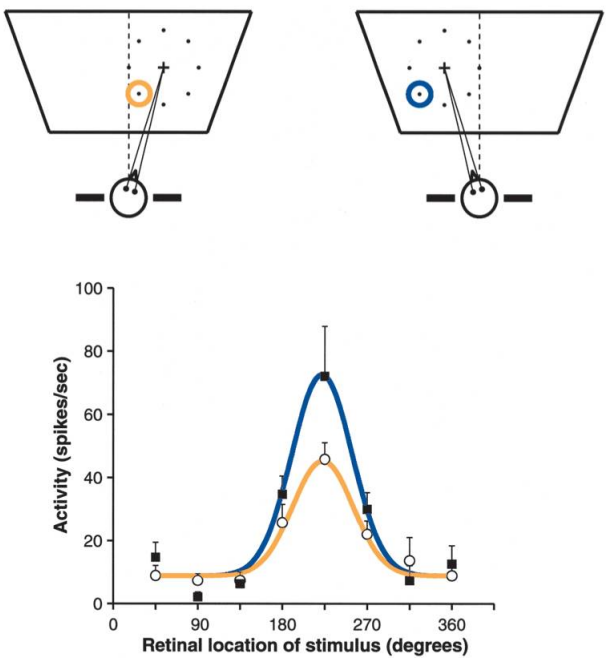


Figure 7: Visual responses that are gain-modulated by gaze angle. The response of a parietal neuron as a function of stimulus location was measured in two conditions, with the head turned to the right or to the left, as indicated in the upper diagrams. In these diagrams, the cross corresponds to the location where gaze was directed, called the fixation point; the eight dots indicate the locations where a visual stimulus was presented, one location at a time; and the colored circles show the position of the recorded neuron's receptive field. This was centered down and to the left of the fixation point. In the diagrams, the rightmost stimulus corresponds to 0 degrees, the topmost one to 90 degrees, and so forth. The dashed line indicates the direction straight ahead. The graph below plots the neural responses in the two conditions, indicated by the corresponding colors. The continuous lines are Gaussian fits to the data points. When the head is turned, the response function changes its amplitude, or gain, but not its preferred location or its shape. Adopted from Salinas and Sejnowski (2001).

function of x_{target} that swifts if x_{gaze} changes (Salinas and Abbott, 1995). A similar swift has been demonstrated in several next cortical areas associated with a variety of coordinate transformations (Salinas and Sejnowski, 2001).

1.4 Use of an artificial neural network

Zipser and Andersen (1988) have been the first ones who trained an ANN and they have brought significant findings that ANNs are able to decode spatial transformation in a similar way to the brain.

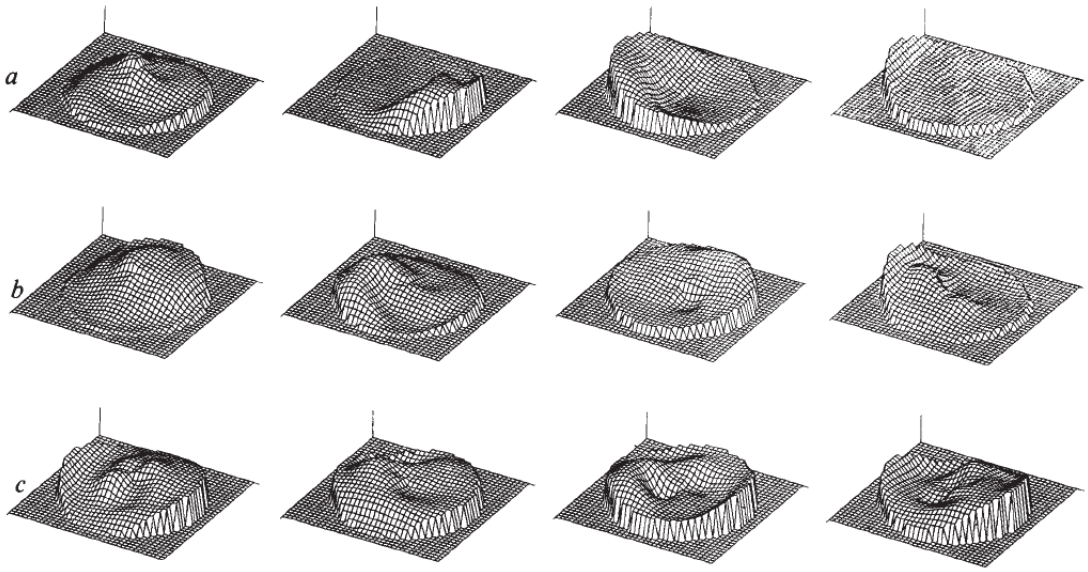
1.4.1 Findings from area 7a

The authors were concerned with the question of how the brain performs such a coordinate transformation that translates sensory inputs into motor commands mainly focused on the brain's area named 7a and neural processes that are performed by this area. This area is most likely used to perform spatial transformations. It also contains cells that receive a convergence of both retinal and eye-position signals as a non-linear interaction, and visual responses seem to be modulated by a function of position of the eyes multiplied by the response of retinal RCFs. It basically means that the visual RCF remains retinotopic but the intensity of the response is modulated by the position of the eyes (gain modulation).

To facilitate a comparison of cells of the ANN with cells of area 7a, they analysed experimental data collected in studies with awake, unanaesthetized monkeys. The experimental data were collected in the way of recording neuronal activities while the monkeys performed various visuospatial tasks. They focused on three major kinds of neurons in 7a area: the cells responding to eye-position only, the visual cells responding to retinal stimuli only and the cells that respond to both of mentioned cell classes. This interaction produces a head-centred representation of visual targets that depend on the position of the eyes.

In order to test the eye-position cells, the following experiment was performed. The animals were fixated on a small point during different eye positions, their heads were fixated and the experiment took place in total darkness. During the experiment, a linear increase was recorded for activities for a range of horizontal or vertical eyes positions but some cells coded information of the position of the eyes in a more complex way. For the testing of RCFs of visual cells, there was used a flashing spot stimulus placed at different locations of the visual fields while the animal was fixated on the target. These cells had usually large RCFs that were uniformly distributed over the visual field for the population of neurons. These activities respond to a single peak function.

The neurons that were specially tuned, showed a convergence of eye position and retinal position. That means the visual responses of the neurons alter as a function of the eye position. That was observed by collecting data under the condition in which the visual stimulus appears at the peak locations in the retinal RCF with an



.

Figure 8: The receptive fields of spacially tuned neurons from area 7a, arranged in rows with the eccentricity of the field maxim increasing to the right, and in columns with the complexity of the fields increasing downwards. Receptive fields were sampled at 17 radially spaced points, with one sample taken at the centre of the field, and four samples taken on each of four circles of radius 10, 20, 30 and 40 degrees. All the fields in row a have single peak. Those in row b have a single peak but some complexities in the field. The fields in row c are the most complex with multiple peaks. The data have been normalized so that the highest peak in each field is the same height. Adopted from Zipser and Andersen (1988).

animal fixating at nine different eye positions. The visual RCFs of these neurons are shown in Fig. 8.

Fig. 9 refers to the spatial GFs of 9 neurons in the area 7a, where the diameter of the black inner circle represents the visually evoked GFs that are calculated as the difference between the background activity recorded before the stimulus onset and the total activity during the stimulus. The outer circle's diameter corresponds to the total activity. Most of the spatial GFs were planar, thus, the diameter of the inner circle is proportional to the outer diameter representing the total response. By the comparison of the black inner circles and the white annuli, which corresponds to the eye position contribution, there were shown three types of GFs. The activities of the first type (Fig. 9b, e, f) change in a parallel fashion. The activities of the second type (Fig. 9a, c, d) change with eye position and the background activity remained constant. For the last type of neurons, the background and the evoked activities changed in different directions (Fig. 9g, h, i).

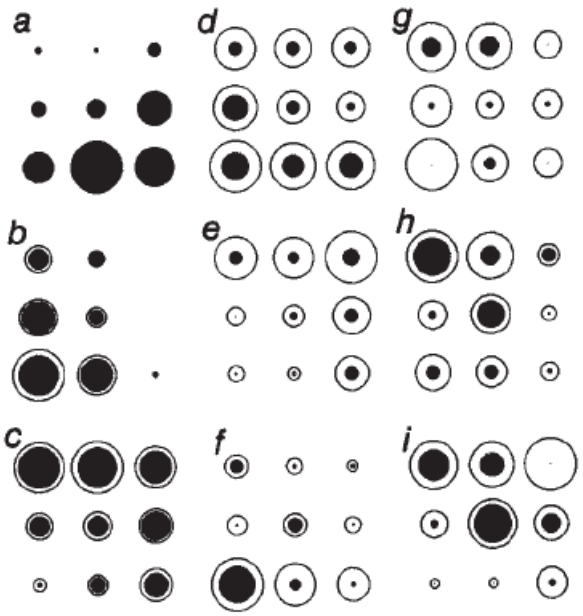


Figure 9: The spatial gain field of 9 neurons (a-i) from area 7a. Adopted from Zipser and Andersen (1988).

1.4.2 Model and results

They used a three-layer neural network. The network was trained to learn the mapping of visual targets to head-centred coordinates. The input data had two parts, an array of units that represent the visual stimulus and representation of eye position via another set of units. Both of the retinal and eye position inputs were formed by using characteristics of the cells in PPC that response to visual stimuli and to eye stimuli, respectively. For the output layer of the network there were used two representations. The neurons of the first format had gaussian RCFs coding location in a head-centred RF. The second representation was a monotonic format in which the activity of each neuron is a linear function of stimuli in head-centred representation. These two formats were chosen because they represent the most common types of coding formats discovered in brain cells. The hidden layer was trained using BP.

BP (Rumelhart et al., 1986) is used to train a feedforward neural network, information that is provided by the input x is propagated up to the hidden units at each layer to produce output $\hat{y}$, this part is called the forward propagation. BP is an algorithm that finds a minimum of a cost function of mapping x to y completed by updating the strength of synapses between neurons. Therefore, during the backward

flow through the network the algorithm computes the gradient of the cost function. Here, it is used the *chain rule* that lets computing derivatives of a function (cost function) that is formed by the composition of other functions with derivates that are known. Consider the following: $x \in \mathbf{R}^m, y \in \mathbf{R}^n$, the function g maps values from $\mathbf{R}^m$ to $\mathbf{R}^n$, and f from $\mathbf{R}^n$ to $\mathbf{R}$, and if $y = g(x)$ and $z = f(y)$, then the partial derivates of the z with respect to variable x_i is:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_i} \frac{\partial y_j}{\partial x_i}. \tag{1.3}$$

The result of the cost function E that expresses an error between the output of the network and the expected output is calculated via the composition of activation functions of neurons, which is a continuous and differentiable function of weights w_1, w_2, ..., w_l in the network. Therefore, the minimalization of E is calculated by using an iterative process of gradient descent that consists in calculating the gradient:

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_l} \right), \tag{1.4}$$

and using the *chain* rule, each weight is updated as:

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \text{ for } i = 1, ..., l, \tag{1.5}$$

where γ donates a variable that defines the speed of learning (Goodfellow et al., 2016; Rojas, 1996).

The model was trained using 1,000 samples and after successful training, the authors compared the experimental and trained RCFs. This comparison has shown that the trained model generates single-peak as well as some moderately complex RCFs, thus, the RCFs of the model match those that were observed experimentally. The characteristics of generated respective fields of the trained network were obtained by holding the eye-position input to the network constant and simulating visual stimulation at the same retinal positions as used in the experiment on area 7a. The GFs were illustrated by the graphics method described above, and the comparision with the observed GFs in the brain was also shown similarities in this aspect.

By this comparison, the authors have shown that their model trained through BP gives similar results like experimentally observed ones during performing the task of the coordinate transformation.

Chapter 2

Methods

2.1 Artifical neural networks

The human brain can be described as a highly complex, nonlinear and parallel information-processing system capable to organize its information-processing units know as neurons to solve plenty of such cognitive tasks as, for example, coordinate transformation or multisensory integration. Investigations of the brain have brought the inspiration that was at the beginning of the history of *artificial neural networks*. In general, an ANN is a machine that is usually implemented by electronic components or is simulated in a software and is designed to perform a particular task or function of interest. The neural networks are characteristic by strong interconnectivity among neurons through those basic computations that are performed (Haykin, 2009). Sometimes a neural network is defined as an adaptive machine (Haykin, 2009, p. 2):

> *A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
>
> 1. *Knowledge is acquired by the network from its environment through a learning process.*
> 2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

As already mentioned above, neuron is a basic computation or information-processing unit of neural networks. In order to achieve a better understanding of neural networks, it may be helpful to elucidate the model of an artificial neuron. For

this purpose, we introduce Fig. 10 that illustrates the model of a single neuron. Via the Figure, we can observe three basic elements of a single artificial neuron. The first of them, the connecting links between every input and a specific neuron, are also known as *synapses* and are characterized by *strength* or *weight*. For example, the input x_1 is connected to the neuron k through the synapse characterized by the weight labelled as w_{k1} which determines the strength of the synapse. It basically means that the input x_1 is multiplied by the weight w_{k1}.

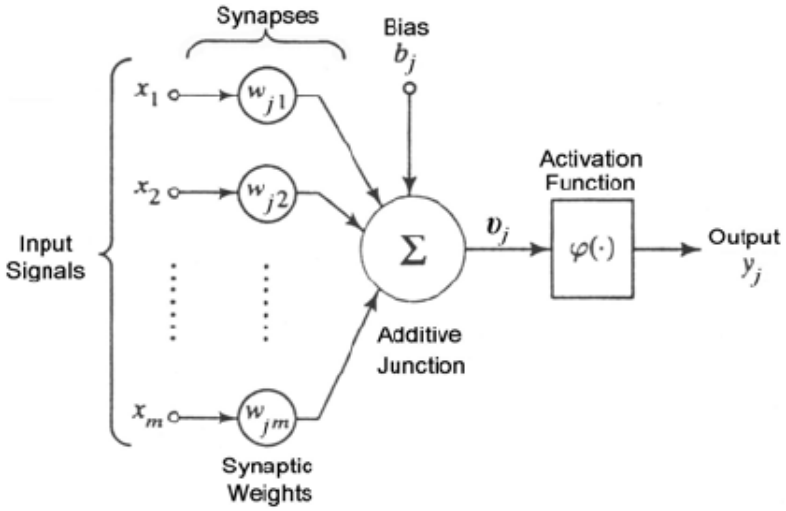


Figure 10: Nonlinear model of a neuron, labeled k. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1], or, alternatively, [-1,1]. Adopted from Haykin (2009).

The second element of a neuron is the summing junction that basically sums each one of input signals weighted by the particular synapse strength. In mathematical terms this operation may be described by the following equation:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{2.1}$$

where u_k is the result of summing operation for a single neuron. This result is also named as *input net*. The last element is an *activation function* that forms the output of a neuron applying an elected function $\varphi(.)$ on the input net. In Fig. 10, we can observe the element labelled as b_j which expresses external *bias* that is responsible for increasing or decreasing the input net of the activation function (Haykin, 2009).

Finally, we can express the output of the neuron k as the following equation:

$$y_j = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_j\right). \tag{2.2}$$

There are a lot of different architectures of neural networks, where each architecture has its own unique properties and is designed for specific tasks, or better said, each architecture is more suitable for some kinds of tasks and less capable to perform other kinds of tasks. For an illustrative purpose, we introduce the model named the *multi-layer percepton* (MLP). The MLP is a *feed-forward* network composed of an input layer, hidden layers and an output layer as shown in Fig.11.

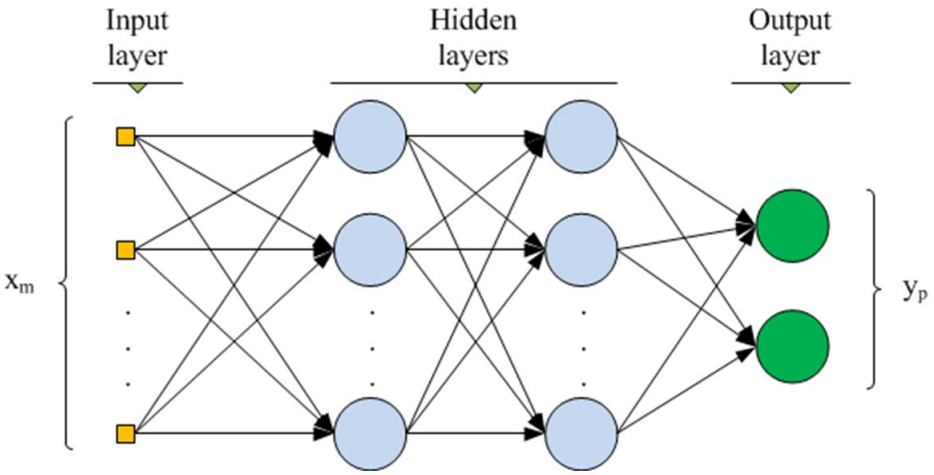


Figure 11: Typical architecture of an MLP with two hidden layers. Adopted from Cruz et al. (2013).

This term groups neural models within which the input signals are transmitted in one direction. Thus, there are not any loops, so the output of a particular neuron does not affect the neuron itself. Feed-forward networks neither contain any *backward* connections between two neurons. The kind of NNs that allows such connections are called *feed-back networks*. The input layer of MLP is not composed of such neurons as we described above but its job is just to transmit the input signals to the first hidden layer. Hidden layers are layers that do not have a direct connection to the environment (Marius et al., 2009).

Weights on all synapses between every two adjacent layers are set by a procedure named *learning*. During the training of a feed-forward neural network, a learning algorithm drives hidden layers to acquire the best possible input-output mapping.

In more mathematical terms, we can label the best-approximating function of real estimated outputs as $f^*(x)$ and let $f(x)$ be the function that is driven by a learning algorithm to match $f^*(x)$. The function $f(x)$ in our case in Fig. 11 composes together three different functions to the form:

$$f(\mathbf{x}) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\mathbf{x})\right)\right), \tag{2.3}$$

where $f^{(1)}$ is the first hidden layer, $f^{(2)}$ is the second hidden layer, and $f^{(3)}$ is the output layer. Then, the learning algorithm is responsible to drive each of the functions $f^{(1)}$ to $f^{(3)}$ to acquire an approximation of $f^*(x)$. ANNs can be used to learn some tasks of regression or classification, where for classification, $y = f^*(x)$ maps an input x to a category y. For a regression problem, an input x is mapped to the particular numeric target y.

2.1.1 UBAL

In this section, we introduce a biologically-inspired algorithm named the *Universal Bidirectional Activation-based Learning* (UBAL). UBAL is one of the answers to biologically-more-plausible alternatives to BP that is based on the backward propagation of errors and its biological plausibility is contentious (Malinovská et al., 2018).

In order to provide a more biologically-plausible algorithm, Oreilly (1996) introduced the model named *Generalized Recirculation* (GeneRec) that has been designed to avoid the computation of error derivates. This model consists of three layers with full connectivity between the layers and its learning process is based on the local differences of activations. The activations of the model flow in two directions through the same weight matrix. The flow of activation from input to output is named the *minus* phrase. In the second phrase, the *plus* phrase, the desired target is *clamped* on the output layer and activations flow back from the output to the hidden layer. The learning rule is given by:

$$\Delta w_{pq} = \lambda a_p^- \left(a_q^+ - a_q^-\right), \tag{2.4}$$

where λ is the learning rate, a_p^- denotes the presynaptic activation, a_q^- denotes the

postsynaptic activation of a unit in the minus phrase, and a_p^+ denotes the presynaptic activation from the plus phase (Farkaš and Rebrová, 2013).

Based on GeneRec, Farkaš and Rebrová (2013) proposed an algorithm named the *Bidirectional Activation-based Learning algorithm*. The difference between GeneRec and BAL is that BAL does not use the same weight matrix for the minus and plus phrase, but uses different weight matrices for both directions. However, BAL failed to reach 100% convergence on basic tasks such as the canonical 4-2-4 encoder. Therefore, the Malinovská et al. (2018) have brought an improvement of BAL named *UBAL*.

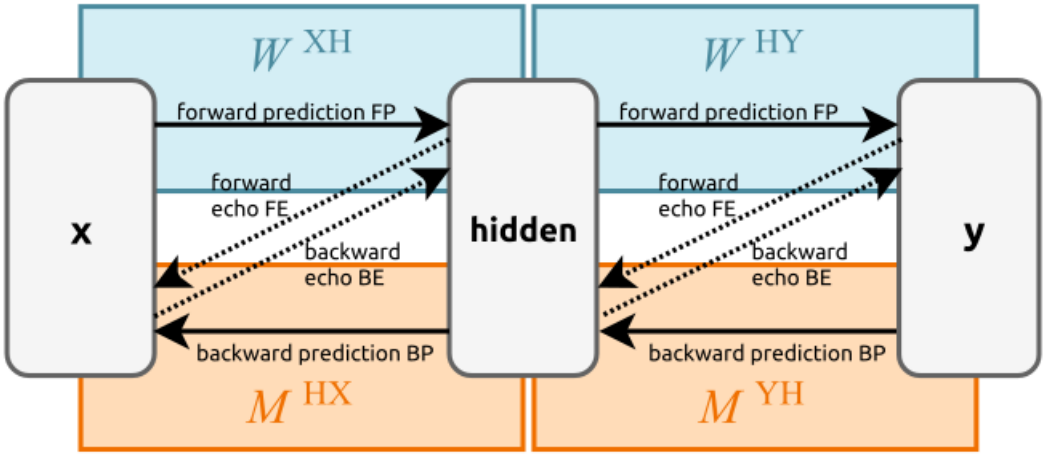


Figure 12: Activation propagation in a network with input-output layers x and y and one hidden layer. Adopted from Malinovská et al. (2018).

The layer's architecture and flows of activations are illustrated in Fig. 12. UBAL uses two separate matrices W and M for each direction of activation flow. In contrast with BAL, both flows contain two phases: *prediction* and *echo*. In the forward

Forward prediction	FP	$q_j^{FP} = \sigma\left(\sum_i w_{ij} p_i^{FP} + b_j\right)$
Forward echo	FE	$p_i^{FE} = \sigma\left(\sum_j m_{ji} q_j^{FP} + d_i\right)$
Backward prediction	BP	$p_i^{BP} = \sigma\left(\sum_j m_{ji} q_j^{BP} + d_i\right)$
Backward echo	BE	$q_j^{BE} = \sigma\left(\sum_i w_{ij} p_i^{BP} + b_j\right)$

Table 1: Propagation of activations between two layers p and q.

prediction phase, input signals are propagated to the layer p and subsequently, the activations are propagated to the layer q. The backward prediction phase is carried out in reverse. Echo activation phases consist of echoing previous outputs q_{FP} and

p_{BP} back to p and q via the matrices M and W. Table 1 depicts these phases in mathematical terms.

The learning rules updating weights of matrices W and M are the following:

$$\Delta w_{ij} = \lambda t_i^B \left(t_j^F - e_j^F \right), \tag{2.5}$$

$$\Delta m_{ij} = \lambda t_j^F \left(t_i^B - e_i^B \right), \tag{2.6}$$

where λ is the learning rate and the other terms are described in Table 2.

Forward target	t_j^F	$\beta_q^F q_j^{FP} + \left(1 - \beta_q^F\right) q_j^{BP}$
Forward estimate	e_j^F	$\gamma_q^F q_j^{FP} + \left(1 - \gamma_q^F\right) q_j^{BE}$
Backward target	t_i^B	$\beta_p^B p_i^{BP} + \left(1 - \beta_p^B\right) p_i^{FP}$
Backward estimate	e_i^B	$\gamma_p^B p_i^{BP} + \left(1 - \gamma_p^B\right) p_i^{FE}$

Table 2: Definition of elements used in learning rules

Every ANN has attributes that are set before the training procedure. These attributes are called *hyperparameters* and their best configuration, for a particular learning problem, is searched experimentally. Hyperparameters of UBAL are described in Table 3.

Learning rate	λ
Hidden layer size	h
Initial weights: mean ν and variance σ^2	$N(\nu, \sigma)$
Forward clamping strength	β^F
Backward clamping strength	β^B
Forward estimate strength	γ^F
Backward estimate strength	γ^B

Table 3: UBAL hyperparameters

UBAL was tested with experiments that consisted in learning the 4-2-4 encoder. As mentioned, BAL failed to learn this task, but UBAL was already able to converge to a solution. At the time of writing this thesis, UBAL was not tested by a very complex learning problem yet. Despite this, there was a suggestion that UBAL could be a more biologically plausible alternative to BP.

2.2 The iCub: Humanoid Robot

Robotic platforms are being used by cognitive scientists with great popularity. One of these platforms is the iCub, a humanoid robot replicating the size of a three-and-a-half-year-old child. The platform was developed within a collaborative project *RobotCub* funded by the Europan Commission. The height of the robot was proposed to correspond to the size of a three-and-a-half-year-old child, which is approximately 100cm high. The overall weight of the robot is around 22kg. The iCub provides 53 degrees of freedom (DoF) that allow the robot's moving ability and the ability to manipulate with objects (Metta et al., 2019). Fig. 13 shows the iCub robot and its proportions.

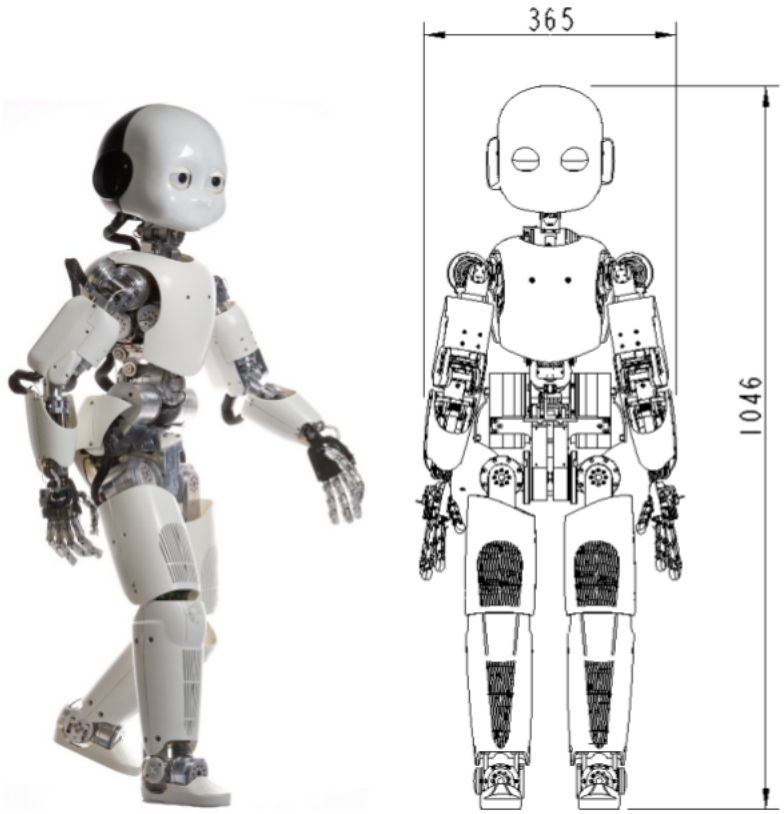


Figure 13: The iCub. (a) The figure shows a photograph of the iCub robot. (b) The overall robot dimensions. Adopted from Parmiggiani et al. (2012).

Such a real humanoid robot like the iCub can be too expensive for some science departments. The cost of one iCub is about 200,000$. Therefore, a lot of researchers use a robotic simulator instead. A robotic system provides a biased model of the real environment, although not so fully complex as the real environment, that might be

very useful, for example, in order to test algorithms performing cognitive functions or to collect suitable data for neural modelling via ANNs (Tikhanoff et al., 2012). The simulator of the robot iCub is distributed as Open-Source system following the GPL licenses and it is supported by Windows, Linux, and also MacOS (Parmiggiani et al., 2012).

The iCub simulator is based on the widely used physics engine called the *Open Dynamics Engine* (ODE). The ODE is used for simulating rigid bodies and for computing the physical interaction with objects. In order to render the scene, the combination of OpenGL and SDL is used, that enables to render the scene in such an easy way that is sufficient to carry out computationally-efficient simulation experiments. Another tool the iCub simulator is based on is an open-source middleware called Yet Another Robot Platform (YARP). The YARP is very suitable for real-time applications interfacing with complex and changing hardware. From the view of the device's API or network, the real robot and the simulator have the same interface and are interchangeable from the user perspective (Tikhanoff et al., 2012). The architecture of the simulator is illustrated in Fig 14.

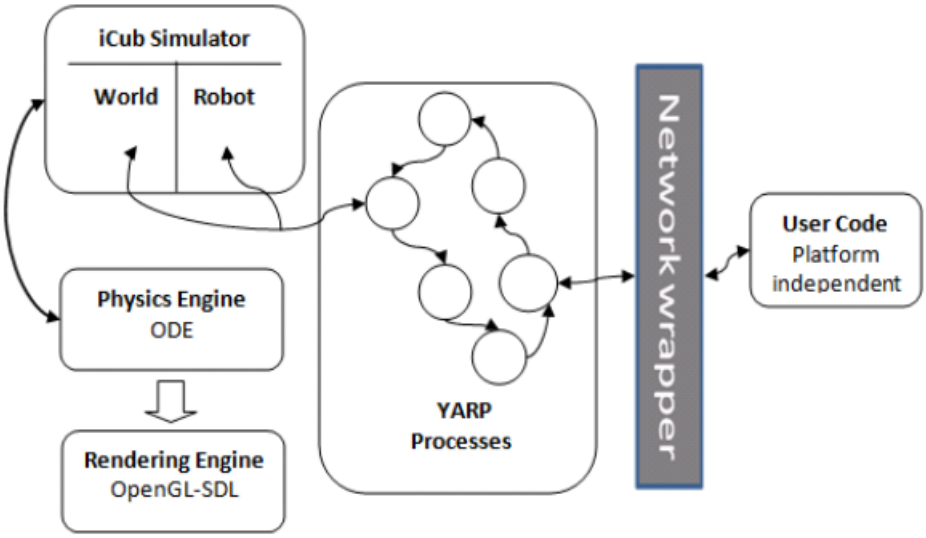


Figure 14: This figure shows the architecture of the simulator with YARP support. The User code can send and receive information to both the simulated robot itself (motors/sensors/cameras) and the world (manipulate the world). Network wrappers allow device remotization. The Network Wrapper exports the YARP interface so that it can be accessed remotely by another machine. Adopted from Tikhanoff et al. (2008).

2.2.1 The Cartesian Controller

Pattacini et al. (2010) developed a cartesian controller (CR) that is capable of dealing with issues like a large number of DoF, computing of trajectories effectively and producing smooth movements, designed such a kind of controller around the iCub robot.

Reaching the cartesian position of the target by the CR is performed in two modules:

- the Solver module including *a nonlinear optimization technique*,
- the Controller module consisting of *a biologically inspired kinematic controller*.

The first module allows computing the arm joints configuration that reaches the desired position. The second stage, the biologically inspired kinematic controller, is responsible for performing a human-like trajectory of the end-effector by computing the velocities of motors (Pattacini et al., 2010).

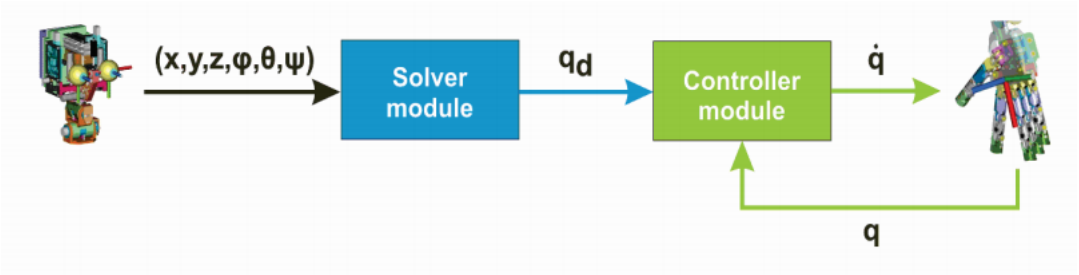


Figure 15: Diagram of proposed Cartesian controller. Adopted from Pattacini et al. (2010).

To introduce an example of working with the CR that is used in our experiments, we introduce the method `askForPosition(...)`. This method returns the joints configuration that achieves the desired position without performing any movements. For instance, if we want to know joints configuration for the position that is determined by $x = -0.25$, $y = 0.0$ and $z = 0.1$, then, we just run the following code:

```
Vector xd(3);
xd[0]=-0.25;
xd[1]=0.0;
xd[2]=0.1;
Vector xdhat, odhat, qdhat;
icart->askForPosition(xd,xdhat,odhat,qdhat);
```

where `xd` is the vector determining the desired position, the parameters `xdhat`, `odhat` and `qdhat` are filled up inside the implementation of the method. `Xdhat` and `odhat` represent the results of the optimization process for the cartesian position and orientation, respectively. The parameter `qdhat` contains the joints configuration composed of three joints for the robot's torso and seven joints for the arm.

2.2.2 The Gaze Controller

A robotic system that is supplied with moving cameras has to be able to focus on the object of interest in order to perform such operations like the exploration of the surrounding environment, manipulation with seen objects or tracing an object. The performance of mentioned tasks includes movements of the neck and both eyes that must be performed as effectively and smoothly as possible. The iCub robot's head has six DoFs and that creates such a complex kinematics that achieves the complexity of the human ocular system.

Roncone et al. (2016) designed a gaze control architecture that is implemented around the iCub's technologies. Their approach of how to solve the controlling of a robot's gaze consists in the explicit control of the 3D point that is defined by both eyes of a generic binocular system. This *fixation point* can be defined as a virtual end-effector that is the result of the kinematics involving the eyes and the neck (see Fig. 16).

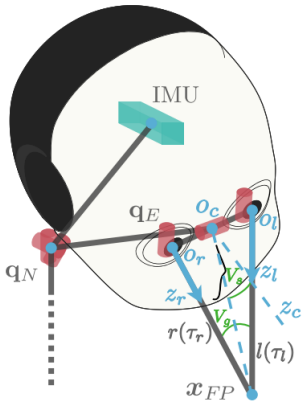


Figure 16: Kinematics of the head system for the iCub humanoid robot. It is composed of a 3-DoFs neck and a 3-DoFs binocular system, for a total of 6-DoFs (depicted in red). Each of these joints is responsible for the motion of the fixation point. The Inertial Measurement Unit (IMU) is the green rectangle, whose motion is not affected by the eyes' movements. Adopted from Roncone et al. (2016).

We used the gaze controller in order to find out where the robot is looking at. For this purpose, there is the method textttgetFixationPoint(...) that returns the current fixation point. The following example corresponds to the call of the method in C++:

```
Vector x;
igaze->getFixationPoint(x);
```

After a successful carry-out, the vector x contains the fixation point corresponding to the current configuration of the neck and the eyes.

2.3 Neuronal Population Vector

The *neuronal population vector* has been introduced by Mahan and Georgopoulos (2014) who investigated the brain mechanisms coding the direction of the arm in 2D space. Their approach consisted of recording activities of cells in the monkeys' motor cortex while the monkeys were performing various arm movements in different directions. Particular neurons of the neural population had the orderly variation of neural activities corresponding to the directional tuned rate of firing action potentials shaped in the form that the highest firing was reached by a particular movement (named as the *preffered direction* of a cell). The activity was being decreased progressively as movements differed more and more from the preferred direction. The function describing the behaviour of a particular cell's activity against its preferred direction is known as the directional *tuning curve*. The example of a tuning curve is illustrated in Fig. 17 (Mahan and Georgopoulos, 2014).

We introduce an example of the neuronal population vector where the activity tuning function of every single neuron is the *gaussian* function, and thus, the activity of a particular neuron fires as the following equation:

$$f(x) = a \exp^{-\frac{(x-b)^2}{2c^2}}, \tag{2.7}$$

where the parameter a determines the height of the curve's peak, b determines where the centre of the curve's peak is placed at, and the parameter c controls the width of the curve.

Consider a population of neurons that is composed of five neurons. The preferred directions of neurons are uniformly distributed on the interval $[-30, 30]$, and the parameters of their Gaussian tuning curve are $a = 1$, $c = 7$ and values of b for each one of the five neurons correspond to $[-30, -15, 0, 15, 30]$. Then, for instance, the value $x = 12°$ encoded by activities of the population's neurons corresponds to

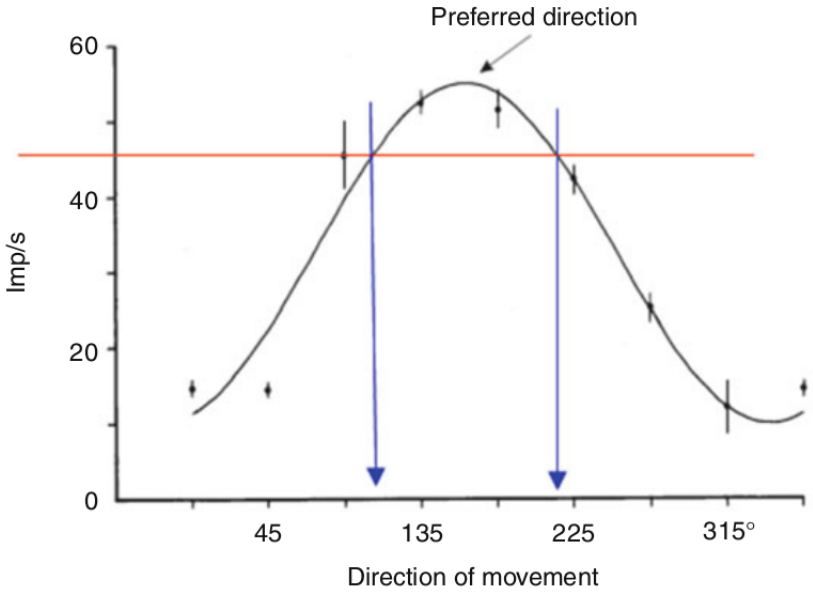


Figure 17: A fitted cosine curve with the preferred direction at the peak of the fitted curve. The ambiguity of the curve with respect to other movement directions is indicated by the red horizontal line at a given discharge rate: the line crosses the curve at two symmetric points corresponding to two different movement directions (blue arrows). This ambiguity points to the need for a unique, unambiguous coding of movement direction by a neuronal population. Adopted from Mahan and Georgopoulos (2014).

values: $\{0, 0, 0.23, 0.912, 0.036\}$, the same example is also illustrated in Fig. 18.

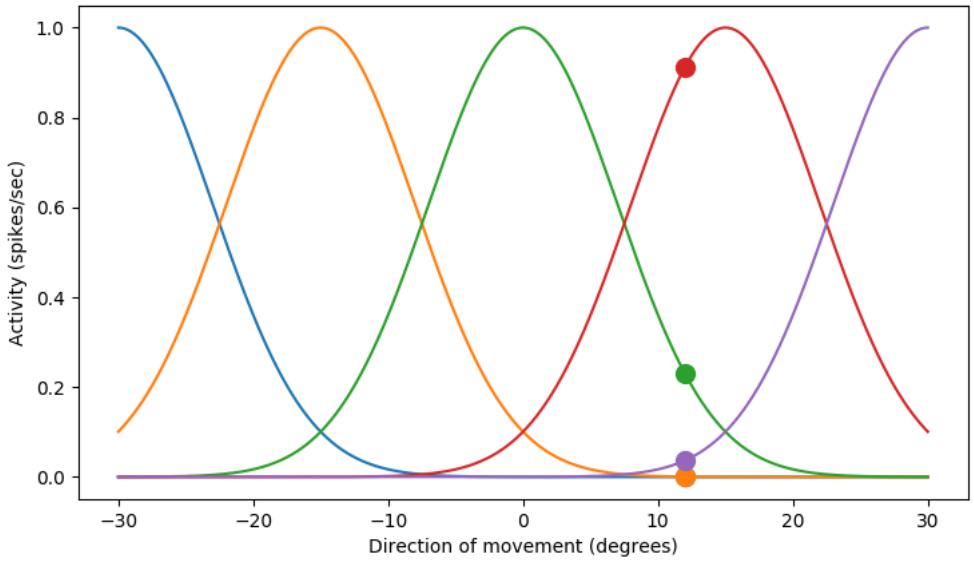


Figure 18: The example of a neuronal population vector with gaussian tuning curves. The population is composed of five neurons with preferred directions uniformly distributed over the interval $< -30, 30 >$. This Figure illustrates the encoding of the value 12° that corresponds to the following activities of the neurons: $\{0, 0, 0.23, 0.912, 0.036\}$.

Chapter 3

Experiments

We carried out three experiments of learning coordinate transformations via UBAL in order to demonstrate the capability of the algorithm to learn such a complex task like a coordinate transformation, and to point out to the properties of neurons of learned models in comparison with another experiment performed by learning a different type of ANN, and with real neurons in the brain. Experiment 1 consisted of the learning of coordinate transformation where one modality is mapped to another one. Particularly, the direction of the gaze is mapped on arm angles that correspond to the point where the gaze is focused on. The aim of this experiment was to test the capability of UBAL model to learn the mapping of more complex data. Experiment 2 and 3 were designed to map the combination of two modalities to the third modality. That allowed us to examine the presence of gain modulation in neurons of hidden

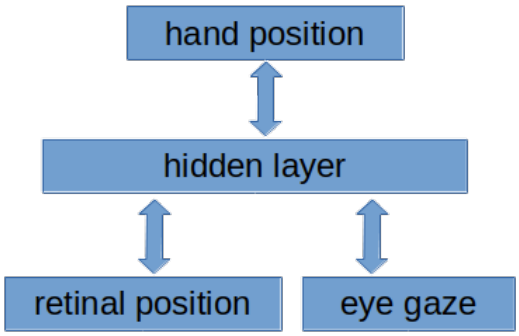


Figure 19: Block schema of more complex models.

layers. The mapping consisted of the gaze direction and the position of an object on retinal images mapped to the angles of the right arm (see Fig. 19). The difference

between these two experiments was only in the encoding of input and output data, whereas the first experiment had data scaled to the same interval, in the second, we used the method called the neuronal population vector that is described in Section 18. We describe each experiments in more detail in the following sections.

3.1 One-to-one coordinate transformation

3.1.1 Data

The model of the first experiments (model A) was trained to learn a one-to-one mapping in the sense of modalities. Data for this model we collected using the iCub simulator (2.2). The fact that communication with the iCub simulator is established through YARP allows us to control the simulator via a program written in C++ or Python. We wrote the implementation of a collecting algorithm that we designed to produce data. The entire code was written in C++ and it generated data in the form composed of seven angles of right arm joints, three angles that control the rotation of the robot's head, and points placed where the robot looked at.

Figure 20: The iCub looking at the palm centre of its right hand. The figure illustrates the data collecting process. For randomly generated values of the robot's head joins, we collected corresponding angles of the right head joints that move the palm to the position where the iCub is looking at. In Experiment 1 we used two joints controlling the head's horizontal and vertical movements. In Experiment 2, two joints moving with eyes are used instead.

Particularly, we chose two joints (denoted as `neck_pitch` and `neck_yaw`) that control the vertical and the horizontal rotation of the head, respectively. These two

joints combined with the third joint (`eye_vergence`) define the fixation point of the gaze, where neck_pitch and neck_yaw define the gaze direction because the rotation of the eyes keeps the same during the whole process, and the vergence defines the distance value from the virtual 'cyclopean' eye that can be imagined as placed in the middle between eyes. In this case, we chose the neck pitch and yaw instead of the joints controlling the rotation of the eyes because it allows us to cover a bigger space of the points which the robot is able to be focused on. These two modalities are supposed to correspond to the same point in the space. Thus, the head joints force the robot to be focused on the same point that represents the position of the palm centre which is defined by the values of seven arm joints.

In order to describe a precise way of how the data was collected, we introduce the most important examples of the code that can help us to get a better understanding of the collecting procedure.

```
...
getHeadController();
getRobotGazeInteface();
getArmController(RIGHT);
getDataFile(path);
setRandomVergenceAngle();
srand(NULL);

const double maxError = 0.03; // 3cm error limit
double maxAngle = 15;
double minAngle = 3;
int direction, steps;
steps = 6;
tuple<int, int> doneCorrectly = make_tuple(-1, 0);
for (int i=0; i < 10; i++ ) {
  int cntSamples = 0;
  setVergenceAngle(24+i);
  while (cntSamples < 100) {
    direction = (get<0>(doneCorrectly) == -1 ||
          (get<0>(doneCorrectly) > 7 &&
           get<0>(doneCorrectly) < 11)
           ? (rand() % 9): get<0>(doneCorrectly);
```

```
    doneCorrect = randomHeadMotions(
      direction, steps, minAngle, maxAngle, maxError);
    cntSamples += get<1>(doneCorrect);
  }
}
```

After initialization of all the required controllers of robot parts, the main procedure, that controls data collection, increases the vergence angle gradually on the interval $[24, 44]$ that was chosen experimentally in order to not exceed the distance that is the robot able to reach with the arm as well as to avoid collisions between the head and the hand. For each vergence value from the interval, we collect several samples by calling the sub-method named `randomHeadMotions(...)`, the core of which corresponds to the following code example.

```
...
for (int i=0; i<steps; i++) {
  nPitchDiff = randomDoubleValue(minAng, maxAngle)*xDir;
  nYawyDiff = randomDoubleValue(minAng, maxAngle)*yDir;
  headAngles[0] += nPitchDiff; headAngles[2] += nYawyDiff;
  if (!checkHeadAngles(headAngles)) {
    return make_tuple(10, i);
    // Break the cycle if a joint angle is out of range!
  }
  setHeadAnglesAndMove(headAngles);
  getCurrentFixPoint(fixPoint);
  icart->askForPosition(fixPoint, xd, od, jointConf);
  getArmJoints(jointConf);

  for (int j = 0; j < 3; ++j) {
    error[j] = fixPoint[j] - xd[j];
  }

  if ((errChck = checkError(error, maxError)) != -1) {
    return make_tuple(errChck, i);
  } else {
    datafile << vectorDataToString(headAngles)
        << vectorDataToString(jointConf)
        << vectorDataToString(fixPoint)
        << vectorDataToString(error) << '\n';
```

```
    datafile.flush();
  }
}
return make_tuple(-1, steps);
```

This sub-procedure was designed to move with the head in one of the eight directions (left, left and up, up, up and right, and so on...), where every movement in a particular direction consists of a series of steps. For each step of a movement, the procedure randomly moves with the head in a particular direction and then, checks if updated values of the head joints do not exceed the robot limits. If this check is successful, then the process continues with receiving the fixation point of the gaze, values of joints of the right arm (see the example in Section 2.2.1). After that, the difference is evaluated between the fixation point and the point that defines the centre of the palm. If the difference is not over the designed limit (3cm), the values of the seven arm joints, values of the three head joints, the current fixation point and the difference between the fixation point and the centre of the palm are saved as one sample. The returning value contains the number of successfully collected samples and the number that indicates whether it was possible to perform all steps of the particular movement or not.

3.1.2 Training and Validation

We collected 1870 unique samples in total. As described in the previous section we used them to train the UBAL model to perform a coordinate transformation from the head angles stimuli representing a fixation point to the angles of the right arm corresponding to the same point, and also vice versa, from the hand angles to the head angles.

Before training, we scaled each feature in the dataset on the interval $[0, 1]$ using a *min-max scaler*, the transformation of which is given by:

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{3.1}$$

where x denotes the feature value.

The scaled dataset was randomly split to the *training* dataset (1590 - 85%) and *testing* dataset (250 - 15%) and we kept the datasets without any change during the

whole process of searching for best hyperparameters.

In total, we trained 192 models with different configurations of hyperparameters. We tried various values for the learning rate, the size of the hidden layer, forward and backward clamping strengths, and forward backward estimate strengths (see UBAL hyperparameters in Section 3). Each one of the trained models consisted of one hidden layer, initial values of weights were always set to zeros, and as an activation function was used the sigmoid function. The rest of the hyperparameters were tested iteratively. We used the same γ value for the forward and the backward estimate strength, and for the clamping strengths was used pairs of values calculated as $[\beta, 1-\beta]$ for the forward strength, respectively, $[\beta, 1-\beta]$ for the backward strength. The particular values that were included during the process finding the best configuration can be seen in Table 4.

Learning rate	λ	$\{0.05, 0.1, 0.15\}$
Hidden layer size	h	$\{10, 15, 20, 25\}$
Forward/Backward clamping strength	β	$\{0.2, 0.5, 0.7, 0.9\}$
Forward/Backward estimate strength	γ	$\{0.25, 0.5, 0.75, 0.9\}$

Table 4: Sets of values used in the process that seached for the best configuration of hyperparameters.

Each model was trained during 200 epochs, and after each epoch, we carried out testing on the test dataset, and if the sum of errors of forward and backward predictions was less than the best previous result, then the current weights were saved as the best result. To measure errors between outputs predicted by a model and the real output we used the *mean square error* that is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \tag{3.2}$$

where $\hat{Y}$ is the vector of predicted values, Y denotes the vector of the real values, and n is the size of the vectors.

After the training of each model, we evaluated the error on each joint belonging to the head or to the arm. This evaluation was carried out by comparing the predicted dataset and the real dataset, and as the error function we chose the *mean absolute error* (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|, \tag{3.3}$$

where $\hat{Y}$ is the vector of predicted values, Y denotes the vector of the real values, and n is the size of the vectors.

As the best model, we chose the model with the best combination of the MSE during training and the MAE with an emphasis placed more on errors of the arm joints. Hyperparameters for the best model are $\lambda = 0.1$, $h = 20$, $\beta = 0.2$ and $\gamma = 0.9$. The model achieved the best result after 149 epochs.

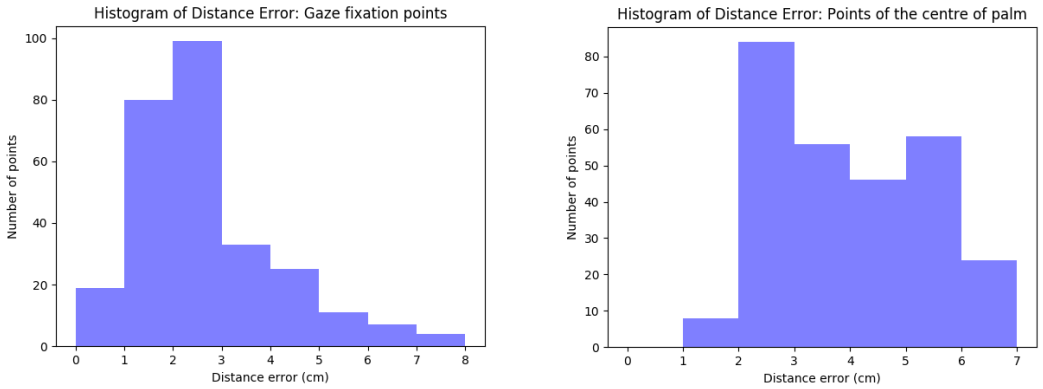


Figure 21: Histograms of errors given by distances between predicted and real values.

To evaluate the accuracy of the best model we achieved fixation points of the gaze and the points of palm centre for predicted head and the arm angles, respectively. These points we compared with corresponding points in the collected dataset in the following way. We calculated the average *Euclidean distance* between the points resulting from predicted values of the arm joints and the points determined by the angles in the collected dataset, and also between the gaze fixation points defined by predicted and real values of the head joints. The average distance between fixation points reached 2.6 cm, and the average distance between the points of palm centres matches up to the value 4,03 cm. Fig. 21 shows histograms of the numbers of points with a specific distance error.

3.2 Eye-hand coordinate transformation

After we successfully trained the first model, we decided to carry out more complex experiments where stimuli coming from two different modalities are mapped to one stimulus, and vice-versa. We gathered new data where the joints controlling the head rotation were replaced by joints maintaining the rotation of the robot's eyes,

thus the iCub's head stayed motionless and the shifting of the gaze was caused by eyes movements. In order to include another modality, retinal images were also included in the data collecting process. The third modality was composed of the right arm joints that correspond to the same point as the fixation of the gaze.

3.2.1 Data

We again use an example of the code to gain a better clarification of the gathering process. The example refers to the core of the collecting process that collects data in the way that we describe in a few following lines. For each value of the vergence on the interval $[17, 41]$ (determined experimentally), the procedure collects the same number of samples. One sample consists of the values of the three eyes joints (eye_tilt - horizontal movement, eye_version - vertical movement, and eye_vergence - the distance between the fixation point and the eyes), the values of the right arm joints, and two retinal images containing the green object (the square) that represents the point of the palm centre. The first step of the process is the generation of a random rotation of the eyes, and subsequently gaining the fixation point of the gaze. The second step acquires values of the angles for the arm joints answering to the gaze point, and the evaluation of the distance error between the point describing the centre of the palm and the gaze fixation point. If the distance between these points is not over the designed limit, the process continues to the path of the dealing with retinal images. For every random configuration of eyes as previously described, the green square is placed on the position of a fixation point (or a palm centre) and four movements of the eyes are proposed in order to capture the green object at different locations on retinas. By following the described algorithm, we received four samples that differ from each other only in different retinal images.

```
int vergence = 17;
...
// For each value from interval <17, 41> collect
// "numberForVergence" of points in the space
while (vergence < 41) {
  if (totalCountVergence > numberForVergerce) {
    totalCountVergence = 0;
```

```
    vergence ++;
    setVergenceAngle(vergence);
  }
  // Generating new random orientation of the robot's eyes
  xDiff = randomDoubleValue(-20, 10);
  yDiff = randomDoubleValue(-30, 30);
  // Setting new orientation of the eyes
  setEyesPosition(xDiff, yDiff, false);
  // Receiving the point where the gaze is focused on
  getCurrentFixPoint(gazeFixation);
  // Receiving angles of hand's joints corresponding
  // to the fixation point
  getInvKinHandAngles(gazeFixation, xd, od, handAngles);
  Vector error(3);
  for (int k=0; k<3; ++k) {
    error[k] = abs(gazeFixation[k] - xd[k]);
  }
  // Check if the difference between the hand point and fixation
  // point is not over the limit!
  bool success = checkErrorGazeHand(gazeFixation, xd, 3.0);
  if (success == true) {
    totalCountVergence ++;
    // Creating an object representing the position of the palm
    // in the space
    runYarpCommand(WorldYaprRpc::deleteAllObjects());
    runYarpCommand(WorldYaprRpc::createBOX(gazeFixation));
    Vector angles, results, headAngles;
    getCurrentAyesAngles(angles);
    // Designing of four head motions
    designChanges(angles, results);
    // Take image for each head conf in results and save data
    for (int r=0; 4 > r; r++) {
      double xEDiff = results[r*2];
      double yEDiff = results[((r*2)+1)];
      setEyesPosition(xEDiff, yEDiff, false);
      takeAndSaveImages(...);
      Vector changedAngles(2);
      getCurrentAyesAngles(changedAngles);
      datafile << vectorDataToString(changedAngles)
```

```
            << vectorDataToString(handAngles) << endl;
        totalCount++;
      }
    }
  }
```

After we collected all data, retinal images were further processed using *OpenCv*. Each image was converted to green-black colours to remove the background, and images received in this way were used to gain a point that determines the position of the green object on retinas. From images, we were extracted the size (the number of green pixels) of the green mass, and the value on axes x and textity defining the point of the object's centre. One example is illustrated in Fig. 22, where the size of the object corresponds to the value $s = 180$ and the centre is located at coordinates $(x = 148, y = 135)$. After collecting up a significant number of samples, we filtered out those samples that did not contain the green object or contained just a small piece of the object. After that, we got 934 samples prepared to be used for training of new models.

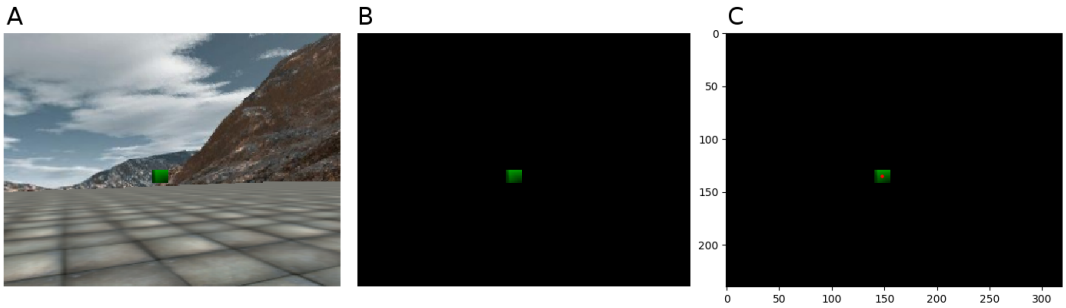


Figure 22: Processing of retinal images using the library OpenCV. (A) The original form of images from one retina. The green square is placed at a point that represents the robot's palm centre instead of the image of the whole palm. (B) The first step of images processing. Only the green pixels are preserved, all others are converted to the black colour. (C) The redpoint corresponds to the centre of green mass. Final information about the palm centre is given as x- and y- coordinates on both axes of the point. In this particular case, coordinates correspond to values (x=148, y=135).

3.2.2 Training and Validation

Model B1

Data for Model B1 were scaled using the min-max scaler (see Eq. 3.1) on the interval <0,1>, and randomly split to training (793 - 85%) and testing dataset (141 - 15%).

The network was composed of three layers, where the input layer contained three neurons for eyes joints (tilt, version, vergence), three neurons coding the position of the object (x, y, and size) of the left retinal image, and the next three neurons for the right image. The output layer was the same as used for the first experiment and contained seven neurons for seven particular joints of the arm. In order to find the best hyperparameters, we first experimentally determined boundaries of each one of the searched hyperparameters, and then, iteratively trained 108 models with different configurations of hyperparameters, the values of which are listed in Table 5. We again used the same value of γ for the forward as well as for the backward estimate strength, and the clamping strengths were determined as the pairs of values calculated as $[\beta, 1-\beta]$ for the forward strength, and $[\beta, 1-\beta]$ for the backward strength. As the activation function, it was used the sigmoid function and the training lasted 200 epochs.

Learning rate	λ	$\{0.08, 0.1, 0.12\}$
Hidden layer size	h	$\{10, 12, 15, 17\}$
Forward/Backward clamping strength	β	$\{0.5, 0.7, 0.9\}$
Forward/Backward estimate strength	γ	$\{0.5, 0.7, 0.8\}$

Table 5: Sets of values used in the process that seached for the best configuration of hyperparameters.

We determined the best model in the same way as in Experiment 1, and its hyperparameters were the following: $\alpha = 0.12$, $h = 15$, $\beta = 0.8$ and $\gamma = 0.9$.

In order to validate Model B1, we collected all predictions of the arm and eyes joints, and these data were set as values of particular joints in the iCub simulator. The predicted values of the arm were used to obtain points of the centre of the robot's palm. Such gained points, we compared with points, defining the centre of the palm, that correspond to samples in the test dataset. We evaluated the average Euclidean distance error between these points and the average distance between the palm centre of collected data. The mean was 2,4cm. The histogram of the error can be seen in Fig. 23.

The mean distance error is smaller than the error in Experiment 1, where we used joints that rotate the head instead of eyes. However, the space of points, where the iCub is able to look at as well as to reach the place with its palm, is bigger than the space of such points reached by rotating the eyes.

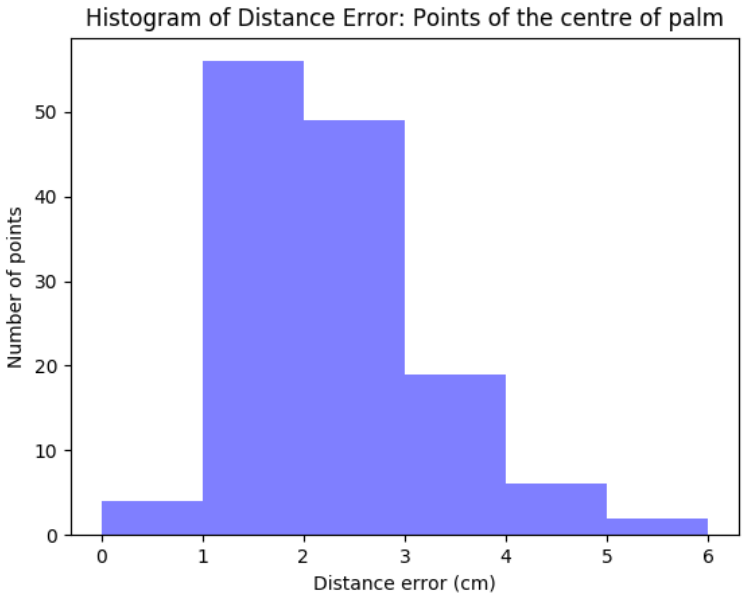


Figure 23: Histogram of errors given by distances between predicted and real values.

For backward predictions, where angles of the arm were mapped to two modalities, we did not compare distances between predicted and collected points because for one sample of the arm joints there are different configurations of the angles of the eyes and retinal positions. Therefore, we rather decided to examine where the iCub looks at using predicted values of eyes joints, and whether the robot using the trained model is able to look at the point that represents a particular palm centre. For each test sample, we set predicted angels of eyes to the iCub, and at the position where the centre of the palm given by a specific sample of the arm joints was placed, we created a green object of the same size as the one used during the procedure of collecting data. Then, we stored both the left and the right retinal image to evaluate whether the iCub looked at the object or not. Such collected images we converted to the 2D binary matrices, where the values of pixels containing the green object (green colour) represented the value 1, and all black pixels were represented by value 0. These matrices of the left, respectivelly the right retinal images, were summed up to two matrices and we used them to create heatmaps showing how the green object was projected on retinas. As we can see in Fig. 24, the iCub was able to look at the object in such a way that the projections of the object fell down near the centre of the field of vision.

We can imagine these results as someone trying to look at the centre of palm

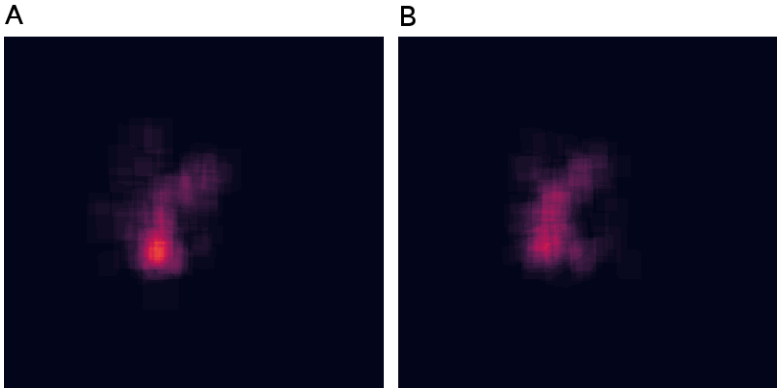


Figure 24: Preferred way of looking at green objects that represent the robot's palm centre. After the training of Model B1, the predicted values of eyes joints together with corresponding original points of the palm centre were used to examine whether the iCub is able to look at the positions or not. It was found out that the robot is able to look at the positions in a preferred way. These images represent positions where the projections of green objects, representing the palm centre, were most often falling on the left and right retina.

with closed eyes and only by using rotations of the eyes. This property of backward predictions probably results from the nature of the training data, where for one sample of the arm joints there are more configurations of eyes joints, therefore, the network was able to learn a preferred way how to look at the centre of the palm.

Model B2

Model B2 was trained in the similar way, but instead of single input and output neurons that encode an angle of the arm's or eyes' position we used population coding for each joint (see Section 2.3) which is a biologically plausible way of information encoding. The network again contained one hidden layer with a sigmoid activation functions. The preferred direction of neurons in a population was shaped by the Gaussians. The width of the Gaussians and the number of neurons in a particular population were selected experimentally to get the best results shown in Table 6.

Coded values	Number of neurons	Parameter c
Retinal positions (x, y)	$(10, 8)$	$(15, 13)$
Three eyes angles	$(4, 5, 4)$	$(7, 8, 7)$
Seven arm angles	$(4, 2, 5, 5, 10, 3, 4)$	$(7, 1.5, 9, 9, 14, 9, 9)$

Table 6: Model B2: Lengths of used population coding vectors.

By using the technique of the neuronal population vector, the first layer contained 13 neurons for the joints of eyes and 36 neurons coding the position on one retina,

thus, 49 neurons in total. The third layer was composed of 33 neurons coding the arm's angles. The hyperparameters were being searched just experimentally and the best result was acquired using $\alpha = 0.1$, $h = 24$, $\beta = 0.7$ and $\gamma = 0.9$ and the model was trained for the duration of 400 epochs.

The distance error between fixation points from the collected dataset and the points resulting from predictions was 4.3cm, thus, bigger than the first version reached. The increased error was probably caused by increased dimensions of data, and therefore, UBAL model with only one hidden layer has a bigger problem to learn more accurate transformation. The increased dimensionality of data was probably also the reason why the network did not learn a preferred way in backward predictions.

3.2.3 Receptive fields

To compare our models B1 and B2 to the model of Zipser and Andersen (1988) and to data experimentally collected during the experiments with monkeys, we examined RCFs. The RCFs of both versions of models, mapping the position of eyes with the object's position on retinas to the values of arm joints, were examined in the same way and this way corresponds to the approach of the mentioned authors. The information about the position of eyes stayed constant during the whole process, and for every possible position of the green object, the activity of a neuron was recorded and scaled using the min-max scaler to gain the height of a peak same for all neurons.

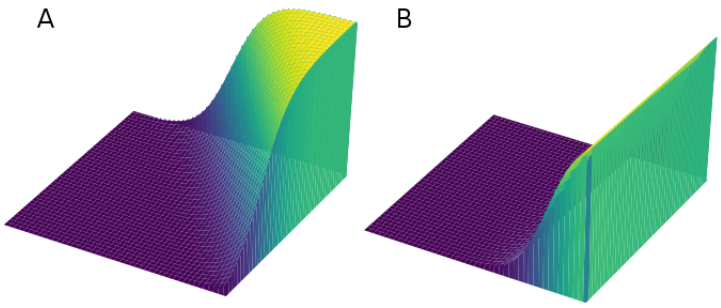


Figure 25: The receptive fields of Model B1. (A) The first kind of receptive fields, the peak of which is placed at the position of maximal values of axes. (B) The second kind of receptive fields that increase along x-axis regardless of y-axis. Surfaces fit activities of neurons while eye stimulus is kept constant, and as retinal stimulus are sampled all combination of values for x- and y-axis.

All neurons of Model B1 had RCFs of two kinds (see Fig. 25). The type had a peak always placed where the horizontal and vertical position reached the highest values, and the type 2 one was increasing from the minimal to maximal value along the horizontal axis regardless of the vertical axis. The reason why the network did not develop more complex RCFs with the peak placed on different position is the fact that the model B1 used just single neurons coding the retinal position.

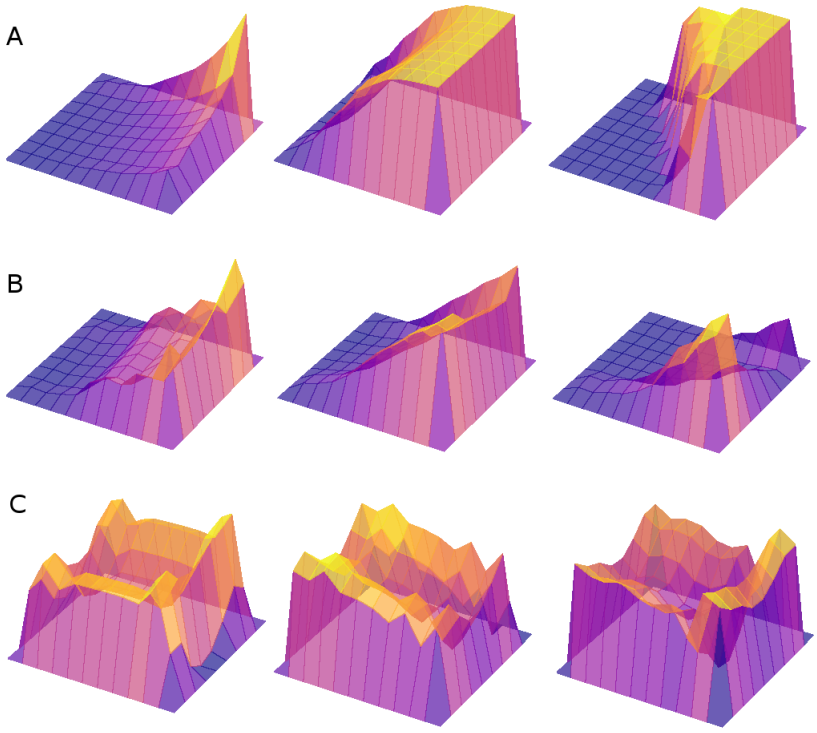


Figure 26: The receptive fields of model B2. (a) The group of receptive fields with single peaks. (b) More complex receptive fields with a single-peak. (c) Very complex receptive fields with multiple peaks. Surfaces fit neuron activities while the eye stimulus is kept constant, and as retinal stimuli are sampled all combination of positions corresponding to peaks of two populations that encode x- and y- positions.

Model B2, using the neuronal population vector coding, showed more common RCFs compared to Zipser and Andersen (1988)'s artificial model. The RCFs, in this case, were plotted as the fitted surface of 80 points that correspond to all combination of the peaks of populations coding the horizontal and vertical position. Fig. 26 illustrates 3 groups of RCF sorted from less complex to the most complex. We found similar groups to those developed by the authors' ANN, and also to those experimentally observed during the experiments with monkeys (see Fig. 8). The RCFs of the first group (a) had single peaks, the second group (b) of RCFs with a single peak but more complex than the first group, and the last group (c)

containing the most complex multi-peak RCFs. Despite the fact that this version was less accurate, these results indicate that a UBAL model is able to develop retinal RCFs that are comparable to those that were developed by the feed-forward NN of mentioned authors as well as to those experimentally observed in monkeys' brains.

3.2.4 Gain modulation

After analysis of RCFs, we took a closer look at the presence of GFs in neurons of both models. The term *gain fields*, as mentioned in Section 1.3, comes from experiments where visual RCFs were tested at different position of eyes, and it was observed that the RCF of a neuron was multiplied by the gaze angle scaled by some constant values. In order to illustrate the GFs of our trained models, we took the graphic method from Zipser and Andersen (1988) that we described in the section 1.4 reviewing their results. However, we made some changes against the method that were inspired by Švec and Farkaš (2014).

In Fig. 27 there are shown the GFs of a hidden neuron in model B2. Each one of the nine subplots corresponds to the gain modulation of the neuron's RCF where the retinal stimulus is placed at nine different positions in the visual field, which are indicated by the green coloured square situated left down. The 20 circles placed in each subplot represent 20 different gaze directions determined by the tilt and version joints controlling the rotation of eyes, whereas the value of the vergence stays constant during the whole process.

The green circle denotes the maximal possible firing activity of a neuron, which is given by the nature of a used activation function. The red circles illustrate the total output response of a neuron, and the blue filled circle represents the contribution of the eye-angle stimulus to the total output.

The GFs of the hidden neuron correspond to the top left RCF in the group a illustrated by Fig. 26. This example shows that the RCF of the neuron is truly modulated as observed in the brain neural population performing the task of coordinate transformation, and also as in hidden neurons of the Zipser and Andersen (1988)'s model. We observed the effect of gain modulation in both models, and this fact confirms that UBAL is suitable for biologically plausible learning of the coordinate transformation.

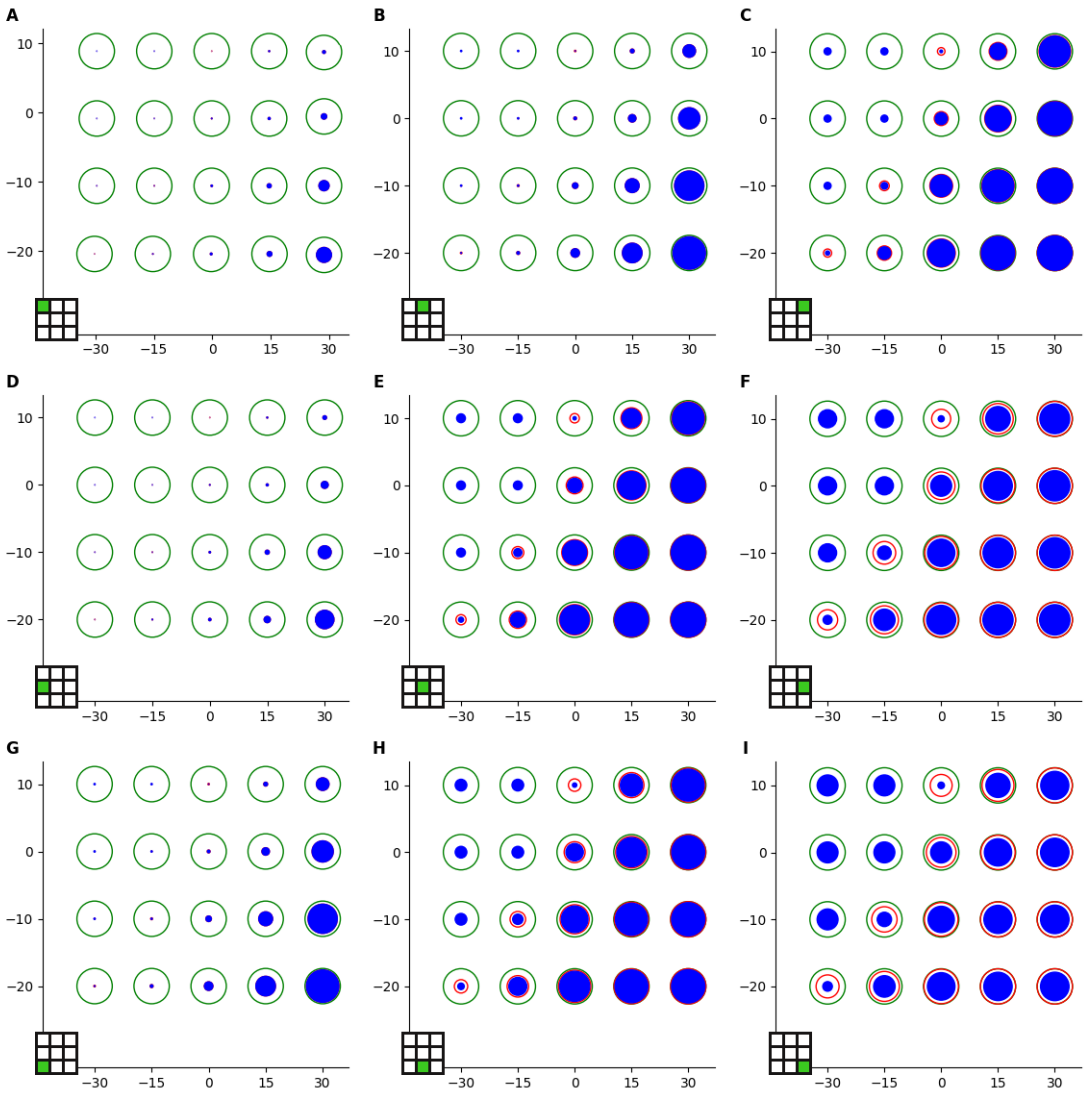


Figure 27: Gain fields of a hidden neuron of Model B2. This image illustrates the effect of gain modulation in one neuron of Model B2. Nine subplots correspond to nine retinal stimuli carrying the information about different positions on retinas. Each subplot shows neuronal activations while the retinal stimulus was constant, and the eye stimulus acquires 20 different combinations of two eyes angles. The green circle represents the total possible value of the neuron's output activity, which matches up to the value 1 because of the used activation function. The red circle represents the total output of the neurons for a particular configuration of eyes and retinal stimuli, and the contribution of the used retinal stimulus to the total output is expressed by the blue filled circle. In the figure, it can be seen that the shape and the placement of the neuron's receptive field stay preserved, but is only scaled in the sense of gain modulation. The graphics method was taken from Zipser and Andersen (1988), and some changes were inspired by Švec and Farkaš (2014).

Chapter 4

Conclusions

Coordinate transformation is always carried out by the brain, if we want to manipulate with an object. Humans use typically their hands to perform such a task as grabbing, reaching or pointing to an object. During all these tasks, we use mainly information from retinas that is initially coded in a gaze (or eye-)-centred reference frame and must be transformed into the activation of muscles (Blohm et al. (2009)). This process is called eye-hand coordinate transformation and is frequently studied by neuroscientists. One of the most popular approaches is the use of *artificial neural networks* to train the task of the eye-hand coordinate transformation, and subsequently to compare the properties of artificial neurons with findings observed in the brain. Over several years there were a couple of studies that used various artificial models to perform a type of coordinate transformation. The first significant model was trained by Zipser and Andersen (1988) who used a feed-forward network trained via the algorithm called backpropagation. Other studies focused on more biologically plausible learning algorithms. Salinas and Abbott (1995) L. used an associative learning rule, and Navarro et al. (2018) trained a self-organising neural network. Our approach consisted in using a novel biologically plausible training algorithm called UBAL that was designed by Malinovská et al. (2018). This algorithm allows us to train a bidirectional model, and we used it to train three models on data that we produced using the simulated robot called iCub. Model A was trained to perform a transformation between two modalities where the head angles of the robot, which define the gaze fixation point by rotation of the head, are transformed into the angles of the right arm's angles that control the hand to reach a point, and vice versa.

The best average euclidean distance between the fixation points resulting from the predicted values of the head's angle and the original fixation points was 2.6cm. The average distance between the position of the centre of the robot's palm reached the value 4cm. The robot driven by the model is able to look at its palm, or reversely, to move its palm to its visual field. The next two experiments used the robot's joints that move with the eyes instead of the head, and we added retinal information about the palm's centre position. These two sensory signals were mapped to the angles of the head, and vice versa. The first of these models (Model B1) was trained on data that was just scaled to the same interval. After training, the robot gained the abilities to move the palm to the gaze fixation point with an average error 2.4cm and to look at the palm in the preferred direction. The last model (Model B2) used population of neurons to code input and output data. However, this coding implied an increase of dimensions and the average distance rose to 4.3cm and the model did not learn a preferred way how to look at the palm. Despite the increased error, we examined the receptive fields of the model and observed that they correspond to the kinds of receptive fields of Zipser and Andersen (1988)'s model as well as to those observed experimentally while recording neural activities in the brain. The similarities between the neurons of both models are worked out by the presence of gain modulation of the receptive fields that was also very similar to the mentioned authors' model.

We can conclude that an artificial network learned via UBAL is suitable to perform the task of a coordinate transformation in both directions and can be used in a robotic platform. All our models used only one hidden layer, but we think that the addition of next hidden layers would increase the precision of transformations. Regarding the comparison between our hidden neurons and the neurons in the brain, or developed by another artificial network, we observed some similarities but there are certainly many possibilities for further experiments.

Bibliography

Andersen, R. and Mountcastle, V. (1983), The Influence of the Angle of Gaze Upon the Excitability of the Light- sensitive neurons of the posterior parietal cortex, *Journal of Neuroscience* **3**(3), 532–548.

Batista, A. (2002), Inner space: Reference frames, *Current Biology* **12**(11), 114–117.

Blohm, G. and Crawford, J. D. (2009), Fields of Gain in the Brain, *Neuron* **64**(5), 598–600.

Blohm, G., Khan, A. and Crawford, J. (2009), Spatial Transformations for Eye–Hand Coordination, *Encyclopedia of Neuroscience* p. 203–211.

Buneo, C., R Jarvis, M., Batista, A. and A Andersen, R. (2002), Direct visuomotor transformations for reaching, *Nature* **416**, 632–6.

Cohen, Y. E. and Andersen, R. A. (2002), A common reference frame for movement plans in the posterior parietal cortex, *Nature Reviews Neuroscience* **3**(7p), 553–562.

Colby, C. (1998), Action-Oriented Spatial Reference Frames in Cortex, *Neuron* **20**(1p), 15–24.

Commiteri, G., Galati, G., Paradis, A., Pizzamiglio, L., Berthoz, A. and Lebihan, D. (2004), Reference Frames for Spatial Cognition: Different Brain Areas are Involved in Viewer-, Object-, and Landmark- Centered Judgments About Object Location., *Journal of Cognitive Neuroscience* **16**, 1517–1535.

Crawford, J., Medendorp, P. and Marotta, J. (2004), Spatial Transformations for Eye-Hand Coordination, *Journal of Neurophysiology* **92**, 10–9.

Cruz, C., Campos, F., Aguiar, P., Bianchi, E. and Geronimo, T. (2013), MLP and ANFIS Applied to the Prediction of Hole Diameters in the Drilling Process, *Artificial Neural Networks - Architectures and Applications* .

Duhamel, Colby, C. and Goldberg, M. (1992), The Updating of The Representation of Visual Space in Parietal Cortex by Intended Eye Movements, *Science* **255**(5040), 90–92.

Farkaš, I. and Rebrová, K. (2013), Bidirectional Activation-based Neural Network Learning Algorithm, *Artificial Neural Networks and Machine Learning – ICANN 2013 Lecture Notes in Computer Science* p. 154–161.

Galati, G., Lobel, E., Vallar, G., Berthoz, A., Pizzamiglio, L. and Bihan, D. (2000), The neural basis of egocentric and allocentric coding of space in humans: a functional magnetic resonance study, *Experimental Brain Research* **133**(2p), 156–164.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press, p. 200–203. `http://www.deeplearningbook.org`.

Hallett, P. and Lightstone, A. (1976), Saccadic Eye Movements Towards Stimuli Triggered by Prior Saccades, *Vision Research* **16**(1), 99–106.

Haykin, S. (2009), *Neural Networks and Learning Machines*, 3 edn, Pearson, pp. 1–39.

Henriques, D. Y. P., Klier, E. M., Smith, M. A., Lowy, D. and Crawford, J. D. (1998), Gaze-Centered Remapping of Remembered Visual Space in an Open-Loop Pointing Task, *The Journal of Neuroscience* **18**(4), 1583–1594.

Kumar, A. and Barve, S. (2002), *How and Why in Basic Mechanics*, Orient BlackSwan Universities Press.

Mahan, M. Y. and Georgopoulos, A. P. (2014), Neuronal Population Vector, *Encyclopedia of Computational Neuroscience* p. 1–7.

Malinovská, K., Malinovský, L. and Farkaš, I. (2018), Towards More Biologically Plausible Error-Driven Learning for Artificial Neural Networks, *Artificial Neural Networks and Machine Learning – ICANN 2018 Lecture Notes in Computer Science* p. 228–231.

Marius, P., E Balas, V., Perescu-Popescu, L. and Mastorakis, N. (2009), Multilayer Perceptron and Neural Networks, *WSEAS Transactions on Circuits and Systems* **8**.

Marshall, J., Fink, G., Shan, N., Weiss, P., Halligan, P., Grosse-Ruyken, M., Ziemons, K., Zilles, K. and H, F. (2000), Line bisection judgments implicate right parietal cortex and cerebellum as assessed by fMRI, *Neurology* **54**(6), 1324–1331.

Metta, G., Sandini, G., Vernon, D., Natale, L. and Nori, F. (2019), iCub: the open humanoid robot designed for learning and developing complex cognitive tasks, *IEEE RSJ International Conference on Intelligent Robots and Systems* .

Navarro, D. M., Mender, B. M. W., Smithson, H. E. and Stringer, S. M. (2018), Self-organising coordinate transformation with peaked and monotonic gain modulation in the primate dorsal visual pathway, *Plos One* **13**(11).

O'Regan, J. and Noe, A. (2001), A Sensorimotor Account of Vision and Visual Consciousness-Authors' Response-Acting Out Our Sensory Experience, *Behavioral and Brain Sciences* **24**(5), 1011.

Oreilly, R. C. (1996), Biologically Plausible Error-Driven Learning Using Local Activation Differences: The Generalized Recirculation Algorithm, *Neural Computation* **8**(5), 895–938.

Parmiggiani, A., Maggiali, M., Natale, L., Nori, F., Schmitz, A., Tsagarakis, N., Santos-Victor, J., Becchi, F., Sandini, G. and Metta, G. (2012), The Design of the iCub Humanoid Robot, *International Journal of Humanoid Robotics* **9**.

Pattacini, U., Nori, F., Natale, L., Metta, G. and Sandini, G. (2010), An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots, *IEEE RSJ International Conference on Intelligent Robots and Systems* .

Pouget, A., Deneve, S. and Duhamel, J.-R. (2002), A Computational Perspective on Multi-sensory Spatial Representations, *Nature reviews. Neuroscience* **3**, 741–7.

Rojas, R. (1996), *Neural Networks*, Springer, p. 155–157.

Roncone, A., Pattacini, U., Metta, G. and Natale, L. (2016), A Cartesian 6-DoF Gaze Controller for Humanoid Robots, *Robotics: Science and Systems XII* .

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), Learning Representations by Back-Propagating Errors, *Nature* **323**(6088), 533–536.

Salinas, E. and Abbott, L. (1995), Transfer of coded information from sensory to motor networks, *Journal of Neuroscience* **15**(10), 6461–6474.

Salinas, E. and Sejnowski, T. J. (2001), Book Review: Gain Modulation in the Central Nervous System: Where Behavior, Neurophysiology, and Computation Meet, *The Neuroscientist* **7**(5), 430–440.

Soechting, J. and Flanders, M. (1992), Moving In Three-Dimensional Space: Frames Of Reference, Vectors, And Coordinate Systems, *Annual Review of Neuroscience* **15**(10), 167–191.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L. and Nori, F. (2008), An Open-Source Simulator for Cognitive Robotics Research: The Prototype of the iCub Humanoid Robot Simulator, *In Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems, PerMIS '08, New York,* pp. 57–61.

Tikhanoff, V., Fitzpatrick, P., Nori, F., Natale, L., Metta, G. and Cangelosi, A. (2012), The iCub Humanoid Robot Simulator, *IROS Workshop on Robot Simulators* .

Xing, J. and Andersen, R. A. (2000), Memory Activity of LIP Neurons for Sequential Eye Movements Simulated With Neural Networks, *Journal of Neurophysiology* **84**(2), 651–665.

Zipser, D. and Andersen, R. A. (1988), A Back-propagation Programmed Network that Simulates Response Properties of a Subset of Posterior Parietal Neurons, *Nature* **331**(6158), 679–684.

Švec, M. and Farkaš, I. (2014), Calculation of object position in various reference frames with a robotic simulator, *In Proceedings of the 36th Annual Conferenceof the Cognitive Science Society, Quebec, Canada* .

Appendix

The appendix contains source codes that we used in experiments. The implementation of the data collecting algorithm was written in C++ and the used IDE was JetBrains Clion Professional 2019.1. Source codes for the training of neural networks, visualisations, and data processing were written in Python using JetBrains Pycharm Professional 2019.2. All source codes were developed under Linux Mint.

List of used Python libraries:

- Pandas
- NumPy
- MatPlotLib
- sklearn
- seaborn
- cv2

List of used C++ libraries:

- YARP
- Eigen3