Learning the Visual-Oculomotor Transformation: Effects on Saccade Control and Space Representation. ☆

Marco Antonelli[a], Angel J. Duran[a], Eris Chinellato[b], Angel P. del Pobil[a,c,*]

[a] *Robotic Intelligence Lab, Universitat Jaume I, Spain.*
[b] *School of Computing, University of Leeds, UK.*
[c] *Sungkyunkwan University, Seoul, Korea.*

Abstract

Active eye movements can be exploited to build a visuomotor representation of the surrounding environment. Maintaining and improving such representation requires to update the internal model involved in the generation of eye movements. From this perspective, action and perception are thus tightly coupled and interdependent. In this work, we encoded the internal model for oculomotor control with an adaptive filter inspired by the functionality of the cerebellum. Recurrent loops between a feed-back controller and the internal model allow our system to perform accurate binocular saccades and create an implicit representation of the nearby space. Simulations results show that this recurrent architecture outperforms classical feedback-error-learning in terms of both accuracy and sensitivity to system parameters. The proposed approach was validated implementing the framework on an anthropomorphic robotic head.

Keywords: stereo vision, cerebellum, humanoid robotics, Gaussian process regression, sensorimotor transformation

☆This paper describes research done at the Robotic Intelligence Laboratory. Support for this laboratory is provided in part by Ministerio de Economía y Competitividad (DPI2011-27846), by Generalitat Valenciana (PROMETEOII/2014/028) and by Universitat Jaume I (P1-1B2011-54).

*Corresponding author

Email addresses: antonell@uji.es (Marco Antonelli), abosch@uji.es (Angel J. Duran), e.chinellato@leeds.ac.uk (Eris Chinellato), pobil@uji.es (Angel P. del Pobil)

1. Introduction

Building a robot that operates autonomously in an unstructured environment requires to obtain a robust representation of the surrounding space. Vision provides a great deal of information about the physical characteristics of the environment and it is extensively used by many biological organisms, including humans. However, extracting such an information is generally an ill-posed problem (Bertero et al., 1988) that can be simplified by means of active vision (Aloimonos et al., 1988). Consequently, humans and other primates exploit eye movements systematically during the interaction with the environment (Aytekin and Rucci, 2012) and these strategies can be replicated in robotics to obtain a consistent representation of the scene (Antonelli et al., 2013b, 2014a).

Among the several types of eye movements, saccades play a fundamental role. A saccade is a fast eye movement that shifts the gaze at a target point and can be used to scan the visual space. This movement can be either voluntary or not and can be triggered by either visual or auditory stimuli. Independently of the nature of the saccade, this movement is ballistic (open loop), not affected by visual perception during the motion. Despite this, the movement is very precise because it is driven by an internal model of the visual-oculomotor system (Chen-Harris et al., 2008). Moreover, once the movement is completed, if there is an incongruity between the observed and the expected position of the target, the internal model is adjusted. This behavior is called saccadic adaptation and has been extensively studied in humans. The effects of saccadic adaptation not only influence the motion of the eye, but also the perception of space (Collins et al., 2007; Lappe, 2008).

Thus, while saccades are employed to create a consistent representation of the surrounding space, the interaction with the environment is also used to adjust the internal models involved in the saccade generation. From this perspective, perception and action are tightly coupled in a bidirectional way: perception triggers an action and the outcome of the action changes our perception.

From a robotics point of view, generating a saccade requires to solve an inverse control problem, in which the retinotopic position of the stimulus has to be converted into a shift of the eye position. Several techniques have been proposed to solve this problem. Some of them model the inverse controller directly, others employ a forward model or combine the direct and inverse models (see Nguyen-Tuong and Peters (2011) for a review). In this work we

compare two of these approaches, the*feedback error learning* (Kawato, 1990) and the *recurrent architecture* (Porrill and Dean, 2007). Both approaches were proposed to model the cerebellum which is the part of the brain that contains the internal models and is one of the brain areas involved in the generation of saccadic control.

Even though our system is biologically inspired, we limit the parallelism with biological systems to high level concepts, and we model low-level characteristics according to real-time requirements imposed by robotic applications. Our model takes as inputs the visual target and the current eye position to generate a precise saccade. Both the *feedback error learning* and the *recurrent architecture* employ a fixed controller and an adaptive element. The fixed controller provides a coarse inverse model of our robot's oculomotor system. The adaptive element implicitly learns the parameters of the system to improve the precision of saccades. It acts as an inverse model in the *feedback error learning* and as a forward model in *recurrent architecture* (Porrill and Dean, 2007). The adaptive controller, which represents a high-level model of the cerebellum, is implemented using the incremental sparse spectrum Gaussian process regression (I-SSGPR), a state-of-the-art artificial neural network that guarantees real-time performance, incremental learning and fast convergence (Gijsberts and Metta, 2012).

In our experiments, we tested the two control strategies in the task of generating a precise eye movement, and we show that the *recurrent architecture* is more accurate and less sensitive to the choice of the inverse model with respect to *feedback error learning*. Moreover, we tested how the joint or separate control of the two eyes influences the implicit representation of the space, showing that treating the two cameras together is essential to achieve a coherent representation (Chinellato et al., 2011). The main analysis about the performance of the controller is conducted by means of exhaustive simulation tests. Finally, the proposed approach is validated on a real humanoid torso.

The remainder of this paper is organized as follows. Section 2 provides an overview of the related work in the fields of neuroscience, robotics and neural networks, while section 3 describes the *feedback error learning* and the *recurrent architecture*. The experimental setup and the achieved results using both simulations and the robot are provided in Sections 4 and 5, respectively. We conclude in Section 6 with a discussion about the main results of this study.

2. Background

2.1. Saccade control in the primate brain

A saccade is a fast, ballistic movement that is used to bring a visual stimulus in the center of the field of view (called fovea in primates). Due to the velocity of the movement, the saccade control is generated without the benefit of sensory feedback. Despite this, the movement is accurate because it takes advantage of an internal model of the visual-oculomotor system (Chen-Harris et al., 2008). The internal model is plastic and is updated when the parameters of the oculomotor system change (Lappe, 2008). Indeed, once the movement is completed, if there is an incongruity between the observed and the expected position of the target, the internal model is adjusted. This phenomenon has been studied in psychophysical experiments with humans using the saccadic adaptation paradigm (McLaughlin, 1967). The experiment consists in moving the target during a saccade directed toward it. Due to saccadic suppression (Bremmer et al., 2009), that makes the visual system blind during such movement, the brain does not recognize that the target has been displaced. At the end of the movement, if there is a gazing error this is attributed to an imprecision of the internal model, which is updated to compensate for the error. Indeed, after an adaptation phase, subjects learned to gaze "correctly" at the displaced stimulus. From these findings, we know that the saccadic controller is adjusted after each saccade.

In humans, several brain regions are involved in the generation of a saccade: the superior culliculus, the basal ganglia, cortical areas –such as the posterior parietal cortex and the frontal eye field–, the cerebellum and the saccadic burst generators in the brainstem (Girard and Berthoz, 2005). The superior culliculus is the main structure that provides the target to saccade burst generators. It receives multiple inputs from the cortical areas and interacts with the basal ganglia to select which target to attend. The saccadic burst generators are the closest circuits to the execution of the movement. They are triggered by the target provided by the superior culliculus and are influenced by adaptive modulation from the cerebellum.

In this work we focus our attention on the cerebellum, which is the area of the brain that stores the internal models of the motor apparatus (Wolpert et al., 1998). The cerebellum has a regular structure in which Mossy fibers carry the input signals that are processed by a population of granular cells. The outputs of these cells are carried by parallel fibers which are connected to the Purkinje cells that project to the output of the cerebellum. From

a computational point of view, the cerebellum performs a linear regression (Purkinje cells) of non-linear basis functions (granular cells) (Porrill et al., 2013), so it can be modelled with a one-layer neural network. The synaptic weights of Purkinje cells are modified by a long term depression mechanism which is driven by the teaching signals conveyed by the climbing fibers proceeding from the inferior olive. This mechanism is usually modeled using a correlation learning rule.

Following these findings, in our model we generate saccadic eye movements by using an adaptive controller that stores the inverse model of the oculomotor system in the case of the *feedback error learning.* and the forward model in the case of the *recurrent architecture* (see section 2.2).

2.2. Inverse controller in robotics

Producing an accurate saccade requires 1) to convert the visual position of the target into a shift of the eye position and 2) to generate an eye movement to get the desired eye position. In our work the eye movement is generated using a PID controller (closed loop with respect to the eye position), so we focus on the transformation that links the visual position of the stimulus into a target position of the eye (open loop with respect to vision). Learning this transformation requires to learn the inverse kinematic model of the robot head, so we have to cope with the problem of the lack of a teaching signal. Indeed, after a saccade is performed, the adaptive controller would use the motor error to correct the performed movement, but only the visual error is observable. Hence, we need again the inverse model of the system to convert the visual error into a motor error.

Several strategies have been proposed to solve the inverse control problem. An early proposed strategy is the direct inverse modeling (Kuperstein, 1988). In its original formulation, the direct inverse modeling consists in performing random movements and then learning the inverse association between the motor command and its perceptual outcome. This technique was employed in the learning of saccade control in several works together with some ad hoc strategies to reduce the exploration process (Schenck and Möller, 2006; Chao et al., 2010; Antonelli et al., 2013a). The main drawbacks of this approach are that it does not cope with redundant systems, and it is not goal-directed. Even if redundancy is not a problem in our application, we still desire a goal-directed approach, in which the system learns while moving toward the observed visual target.

The drawbacks of the direct inverse modeling were addressed by *feedback error learning* (Kawato, 1990). The *feedback error learning* model consists of two inverse controllers. A fixed feedback controller slowly drives the system toward the target and provides a learning signal to a second adaptive controller. At the beginning the control law is provided entirely by the fixed controller, while, once the system is trained, better performance is obtained due to the effect of the adaptive controller. In robotics, the *feedback error learning* has been used in several inverse control tasks, among which we can find saccade control (Bruske et al., 1997) and smooth pursuit (Shibata et al., 2001).

Instead of directly learning the inverse controller, the same problem can be addressed by first learning the forward model, and then inverting it by means of an exhaustive incremental searching (Forssén, 2007), or by inverting the Jacobian (Sun and Scassellati, 2004). The inverse and forward models can also be used contextually Jordan and Rumelhart (1992); Wolpert and Kawato (1998); Damas et al. (2013). For example, the distal teacher approach, uses the inverse of the Jacobian to convert the sensory error into a teaching signal for inverse control (Jordan and Rumelhart, 1992).

An alternative way to use the forward model is the *recurrent architecture* proposed by Porrill et al. (2004). A fixed inverse model, such as the one used in the *feedback error learning*, is combined with a forward model that provides an additional input to the inverse model. The interaction between the inverse-forward model creates a recurrent loop which terminates when the output converges to a stable value. As for the *feedback error learning* and distal teacher, the adaptive controller is learned incrementally during goal directed movements. However, in this case, the teaching signal is provided directly by the sensory error ontained as output of the plant. Thus, the teaching signal is proximal to the sensory processing. Moreover, it does not require the inversion of the Jacobian, which is not a biologically plausible solution. The *recurrent architecture* was proposed as a computational model of the cerebellum, and it was tested in several simulations (Porrill et al., 2004; Porrill and Dean, 2007; Porrill et al., 2013) and in one robotic setup to learn the vestibulo-oculomotor control (Lenz et al., 2008), but not for saccade.

2.3. Neural networks and visuomotor transformations

Learning the inverse model, as described in the previous section, requires an adaptive controller that can be tuned according to the input-output data. The easiest solution is probably the look-up table, in which the input space is

subdivided into cells, each one containing the value of the learning function. The accuracy of this approach depends on the granularity of the cells and suffers the curse of dimensionality of the input space. This approach has been used in saccade control to map the monocular visual position of the stimulus (2 dimensions) into an eye movement (Marjanovic et al., 1996; Chao et al., 2010).

A more general approach is the use of neural networks. In this section we focus on the neural networks that have only one hidden layer. The reason is twofold. The first one is related to the modelling of the cerebellum, that suggests that one hidden layer (granular cells) provides the basis functions that are linearly combined to compute the output activation (Purkinje cells) (see Section 2.1). The second reason is that one-layer neural networks can be trained using fast converging algorithms (Karayiannis and Venetsanopoulos, 1992; Haykin et al., 2001), which is a desirable property in robotic applications.

In previous works we addressed the reference frame transformation problem by using Gaussian basis function networks (Chinellato et al., 2011; Antonelli et al., 2011; Chinellato et al., 2012). RBF were chosen for their biological plausibility and for their ability to approximate any kind of non-linear functions (Park and Sandberg, 1991). One drawback of this approach is that, due to the gradient descent based learning rule, the network converge slowly. Moreover, they suffer the curse of dimensionality: the number of neural units increases geometrically with the number of input variables.

One way to address the curse of dimensionality is provided by the *growing neural networks*, that increase the number of neurons depending on the error of the approximation. An example of this approach, which has been tested in the learning of saccade control (Bruske et al., 1997), is the dynamic cell structure (Bruske and Sommer, 1995).

The problem of speeding up the convergence rate can be solved using the recursive least square (Karayiannis and Venetsanopoulos, 1992) or the Kalman filter (Haykin et al., 2001). These approaches update the covariance of the weights in order to combine efficiently new and old observations. Using these techniques, in our previous works we learned eye-hand coordination with three degrees of freedom (Antonelli et al., 2012b,a) and we exploited the covariance matrix to choose the most informative visual targets for learning the inverse saccade control (Antonelli et al., 2013c).

The advantages provided by the recursive least square and by the growing neural networks were combined in the local weighted projective regression

(LWPR) (Vijayakumar et al., 2005). These networks have been very successful in several applications in robotics also for generating the eye trajectory necessary to execute a saccade (Shibata et al., 2001), due to their capacity of learning with high degrees of freedom.

However, LWPR are quite difficult to use due to the number of parameters that need to be set. An alternative to the use of growing neural networks is to approximate the Gaussian activation of neurons using sparse features (Rahimi and Recht, 2007). This approach has been used in a recent neural network called incremental sparse spectrum Gaussian process regression (I-SSGPR) (Gijsberts and Metta, 2012). Beside handling high dimensionality input, the I-SSGPR updates the weights using an incremental algorithm that performs the *maximum a posteriori* estimate. This the algorithm has very few parameters, that can be tuned using log marginal likelihood optimization (Gijsberts and Metta, 2012). In this work, we use the I-SSGPR to implement the adaptive controller that is required to generate precise saccades.

3. Model

This section describes the two approaches that we have implemented for the generation of the saccade control, the *feedback error learning*(FEL) and the *recurrent architecture*(RA). The goal of both models is to convert the visual target $\vec{t}$ and the current eye position $\vec{e}$ into an eye shift (saccade) $\Delta\vec{e}$, that would bring the stimulus in the center of the visual field. Both control schemes can be used to drive the movement of each eye independently (monocular encoding) or to move both eyes jointly (binocular encoding). The main difference in the implementation of the two encodings is the dimension of the input and output vectors and the accuracy that can be achieved.

Independently of the encoding, both approaches are composed by a fixed controller $\mathbf{B}$ and a non-linear adaptive controller $C(\cdot)$. In this work we set $\mathbf{B}$ to be a linear inverse model of the oculomotor system $P(\cdot)$. Even if using $\mathbf{B}$ alone is enough to drive the eyes toward the target, the execution of a ballistic movement does not provide a precise saccade. This is due to the non-linearity of the oculomotor system, which is not modelled correctly by the linear controller. In order to improve its performance, we add an adaptive controller $C(\cdot)$. The difference between the two control schemes is how the controllers $\mathbf{B}$ and $C(\cdot)$ are interconnected, and the role of the adaptive controller $C(\cdot)$, which is an inverse model ($C_f(\cdot)$) in the *feedback*

error learning and a forward model ($C_r(\cdot)$) in the *recurrent architecture*. The details of the two approaches are provided in the following sections.

Feedback Error Learning

In the *feedback error learning* the adaptive controller provides an inverse model of the plant (robotic head) which is used to correct the output of $\mathbf{B}$. The input of the controller is the visual target $\vec{t}$ and the current eye position $\vec{e}$ while the output is the saccade command $\Delta\vec{e}$. The eye movement is then sent to the robotic head which moves accordingly. After the movement, the new visual position of the stimulus $\vec{t}'$ is converted into a motor error to adapt the inverse controller $C_f(\cdot)$. This conversion is performed again by the approximated inverse model $\mathbf{B}$ (Fig. 1). Using this approach, the adaptive filter learns to compensate the poor response of the fixed feedback control, and the expected response of the cerebellum is $C_f = P^{-1}(\cdot) - \mathbf{B}$ where $P(\cdot)$ is the model of the oculomotor system (plant) (Porrill and Dean, 2007).

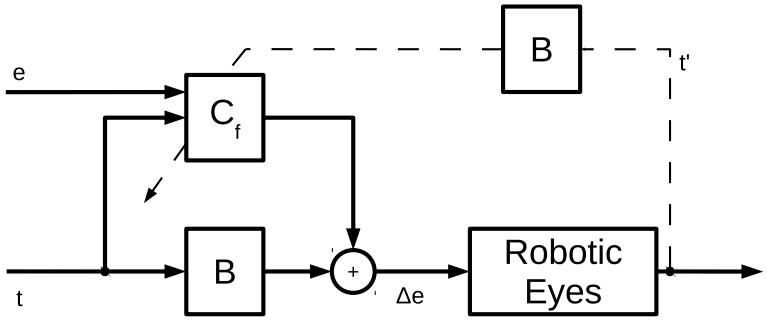


Figure 1: Feedback error learning. The visual position of the stimulus ($\vec{t}$) and the current eye positions ($\vec{e}$) are converted into a motor command ($\Delta\vec{e}$) by summing up the contributions of a fixed element $\mathbf{B}$ and an adaptive element $C_f(\cdot)$. Training the weights of $C_f(\cdot)$ requires a motor error $P^{-1}(t') \approx \mathbf{B}\vec{t}'$ instead of a sensory error $\vec{t}'$.

Recurrent Architecture

In the *recurrent architecture* we have the same linear inverse model ($\mathbf{B}$) that we used in the FEL, but in this case the adaptive controller $C_r(\cdot)$ provides a correction of the input that is sent to the linear controller $\mathbf{B}$. This

correction is fed into the linear controller to obtain a new eye shift. Using the new command, the adaptive controller provides a new correction, thus creating a loop between the fixed and the adaptive elements. This loop terminates when the motor command converges to a stable value. In our experiments we set a threshold on the increment of the motor command, and we also limit the maximum number of iterations. In this configuration, the input of the adaptive controller are the visual target $\vec{t}$, the current gaze position $\vec{e}$ and the motor command $\Delta\vec{e}$. One of the main advantages of using this approach is that the teaching signal is provided by the sensory error $\vec{t}'$, without needing an additional transformation as in the case of *feedback error learning* (Fig. 2). Using this approach, the adaptive controller converge to $C_r(\cdot) \approx \mathbf{B}^{-1} - P(\cdot)$ (Porrill and Dean, 2007).

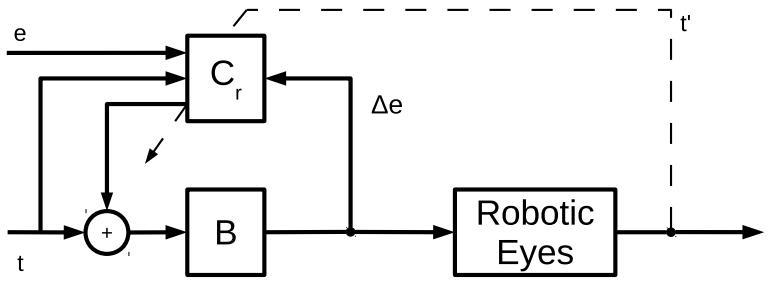


Figure 2: Recurrent architecture. The visual stimulus is sent to the fixed element $\mathbf{B}$ that generates a motor command $\Delta\vec{e}$. This command, together with the visual target $\vec{t}$ and the current eye position $\vec{e}$, provides the input to the adaptive element $C_r(\cdot)$. The output of $C_r(\cdot)$ is then used as a correction to the input to $\mathbf{B}$. In this recurrent architecture, the visual stimulus obtained after the movement of the head is used directly as a teaching signal.

3.1. Monocular and Binocular Encodings

Saccade control can be implemented using either a monocular or a binocular encoding. In literature, both approaches have successfully been used for learning the saccade control, however there are not works that compare them when the goal is the space representation (see Sections 4.2 and 5.3).

In monocular encoding we have an independent controller per each camera. The visual target is provided by the horizontal ($u_{l/r}$) and vertical ($v_{l/r}$)

coordinates of the target in the image (left or right). Eye position is provided by the angular position of the left or right pan ($\theta_{l/r}$) and common tilt (θ_t), and the motor command changes the position of the controlled eye ($\Delta\theta_{l/r}, \Delta\theta_t$). Thus, in the monocular case, the matrices $\mathbf{B}_{l/r}$ that converts the visual target into an eye movement are 2×2 matrices:

$$\begin{bmatrix} \Delta\theta_{l/r} \\ \Delta\theta_t \end{bmatrix} = \mathbf{B}_{l/r} \begin{bmatrix} u_{l/r} \\ v_{l/r} \end{bmatrix} \tag{1}$$

In the case of binocular encoding, we have just one controller that commands both cameras. Among possible alternatives for representing binocular visual information we favor the composition of a cyclopean image representation (c_x, c_y) with a disparity map (d), over the option of having separate left (u_l, v_l) and right (u_r, v_r) representations:

$$\begin{bmatrix} c_y \\ c_x \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_l \\ u_r \\ v_l \\ v_r \end{bmatrix} \tag{2}$$

We represent eye position by means of the tilt (θ_t), version (θ_{vs}) and vergence (θ_{vg}) angles:

$$\begin{bmatrix} \theta_t \\ \theta_{vs} \\ \theta_{vg} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \theta_t \\ \theta_l \\ \theta_r \end{bmatrix} \tag{3}$$

where θ_l and θ_r are angular position of the left and right eye, respectively. In this case $\mathbf{B}$ is a 3×3 matrix:

$$\begin{bmatrix} \Delta\theta_t \\ \Delta\theta_{vs} \\ \Delta\theta_{vg} \end{bmatrix} = \mathbf{B} \begin{bmatrix} c_y \\ c_x \\ d \end{bmatrix} \tag{4}$$

We have implemented four neural networks, two for the binocular encoding (FEL and RA) and two for the monocular encoding (left and right cameras with RA). The forward architecture has six inputs, three for the retinotopic position of the target, three for the current direction of the gaze and three outputs that represent the saccade movement. The binocular *recurrent architecture* has nine inputs, three for the retinotopic position of the target, three for the current direction of the gaze, three for the motor

command sent by the brainstem model, and three outputs that provide an increment of the retinotopic position. The two monocular networks have six input each, two for the position of the stimulus in the image, two for the angular position of the camera, two for motor command, and two outputs corresponding to an increment of the visual position in the left or right image (see Table 1).

Table 1: Signals involved in the monocular and binocular encoding and their symbols.

	Symbol	Monocular	Binocular
Visual target	$\vec{t}$	$\vec{t}_m = (v_{l/r}, v_{l/r})$	$\vec{t}_b = (c_x, c_y, d)$
Eye position	$\vec{e}$	$\vec{e}_m = (\theta_t, \theta_{l/r})$	$\vec{e}_b = (\theta_t, \theta_{vs}, \theta_{vg})$
Eye movement	$\Delta\vec{e}$	$\Delta\vec{e}_m = (\Delta\theta_t, \Delta\theta_{l/r})$	$\vec{e}_b = (\Delta\theta_t, \Delta\theta_{vs}, \Delta\theta_{vg})$
Fixed controller	B	2×2 matrix	3×3 matrix
Adaptive Controller	C_f	$f(\vec{t}_m, \vec{e}_m)$	$f(\vec{t}_b, \vec{e}_b)$
	C_r	$f(\vec{t}_m, \vec{e}_m, \Delta\vec{e}_m)$	$f(\vec{t}_b, \vec{e}_b, \Delta\vec{e}_b)$

4. Simulation Results

4.1. Methodology

In this section we describe the experiments that we performed using a 3D simulation of the kinematics of the our robot (see Section 5.1). In the first experiment we compared the FEL and the RA. In the second experiment we used the RA to study how the choice of the encoding (monocular v.s. binocular) influences the precision of the saccade and the representation of the space.

In both experiments we used the same dataset, composed by 8205 points. The dataset was acquired with the following protocol. We placed 500 virtual targets in the space close to the robot, and we defined 125 ($5 \times 5 \times 5$) eye positions in the gaze-centered frame of reference. For each head position we computed the projection of the 500 stimuli in the left and right images. Then, for each visible target, we stored in the dataset the head position and the visual stimulus[1]. In the experiments, we partitioned the dataset into testing and training sets using the K-Fold cross validation (herein K=5).

[1]We obtained 8205 points instead of 62500 because not all points are visible from each eye position.

Figure 3 shows a scatter on the transverse plane of the initial gazing points (black crosses) and of the target points (red dots). The dataset covered a wide region of the peripersonal space of the robot, as can be seen in Table 2, showing the range and the statistical distribution of the sample points.

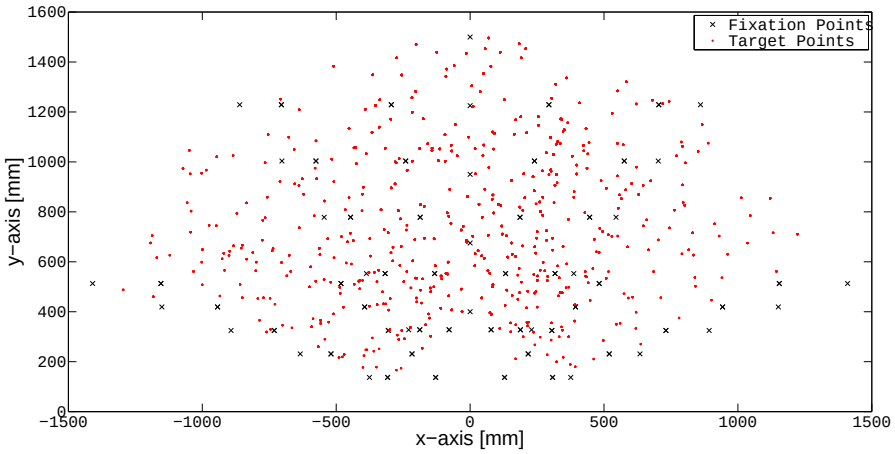


Figure 3: Distribution of the initial fixation points (black crosses) and target points (black dots) used for creating the dataset.

Table 2: Range and distribution of the input data provided by the dataset.

Input	Min. Value	Max. value	Mean	Std. Dev.
c_y [pixels]	-382.6	372.6	-1.4	180.91
c_x [pixels]	-503.4	507.7	7.3	240.8
d [pixels]	-560.4	625.1	-17.1	204.9
θ_t [degrees]	-70.25	70.25	-0.33	23.91
θ_{vs} [degrees]	-70.21	70.22	-0.50	24.62
θ_{vg} [degrees]	3.50	38.19	17.39	8.87

In each experiment we trained the neural networks adopting an on-line strategy that can be replicated on the robot. For each sample of the training, we computed the eye shift ($\Delta\vec{e}$) by multiplying the visual position of the stimulus ($\vec{t}$) by matrix $\mathbf{B}$. The achieved eye shift was then corrected using the adaptive controller $C(\cdot)$. In the case of the *feedback error learning*, the controller $C_f(\cdot)$ was fed with the visual target and the current eye position

($\vec{e}$). In the case of the recurrent loop, the adaptive controller $C_r(\cdot)$ obtained as input also the output of the linear controller ($\Delta\vec{e}$). The recurrent loop terminated when the correcting factor calculated by the controller was lower than 1 pixel, or when it reached 30 iterations. In both architectures we initialized the weights of the adaptive controller to zero, so at the beginning they did not influence the inverse control. Once the eye movement was computed, we simulated the execution of a saccade and then we computed the projection of the target on both cameras. The new visual position of the target ($\vec{t}'$) was used to change the weights of the adaptive controller.

The testing phase followed the same paradigm of the training phase, without adapting the networks. In this case, we also used the configuration of the head after the saccade, to compute the fixation point of the robot in the Cartesian space.

4.2. Feedback error learning v.s. Recurrent architecture

In this experiment we compared the performance of the FEL and the RA using binocular encoding. Both control schemes were composed by fixed and adaptive controllers. The fixed controller provides a fixed linear inverse model of the plant so that: $\Delta\vec{e}_b \approx \mathbf{B}\,\vec{t}_b$. The 3×3 matrix $\mathbf{B}$ was calculated using the least sqaure method on the input-output data of the dataset. The matrix $\mathbf{B}$ is the a priori information that we have about the system. In order to study the sensitivity of the system to this matrix, in the experiments we multiplied the linear controller $\mathbf{B}$ by a scalar gain that was changed systematically between 0.5 and 1.4. We prefer an architecture that is insensitive to the choice of $\mathbf{B}$, as long as it represents a reasonable approximation of the inverse model of the plant.

The adaptive controllers were different for the two architectures, but both were implemented using an I-SSGPR network with 500 random features (see Gijsberts and Metta (2012) for details). In *feedback error learning*, $C_f(\cdot)$ has a six-dimensional input composed by the visual target ($\vec{t}_b$) and the current eye position ($\vec{e}_b$), while the output is an eye movement that should correct the output provided by $\mathbf{B}$ alone. In the *recurrent architecture*, $C_r(\cdot)$ has a nine-dimensional input composed by the target ($\vec{t}_b$), the current eye position ($\vec{e}_b$) and the upcoming eye movement ($\Delta\vec{e}_b$), while the output is the correction of the input which is sent to $\mathbf{B}$.

In the FEL, the linear model is used to convert the visual error into a motor signal (distal error) which is used to train the network (Fig. 1). Thus, changing the gain we are degrading the quality of the teaching signal. This

does not happen in the *recurrent architecture*, in which the visual error provides directly the teaching signal (proximal error) for the adaptive controller (Fig. 2). Therefore, we expect FEL to be more sensitive to the gain with respect to the *recurrent architecture*.

We trained and tested the two architectures using the K-Fold cross validation, with K=5. The error of a saccade was calculated as the Euclidean distance of the stimulus from the center of the image. Figure 4 shows the mean and the standard deviation of the error on the test set for the two architectures, as a function of the gain used by the linear controller.

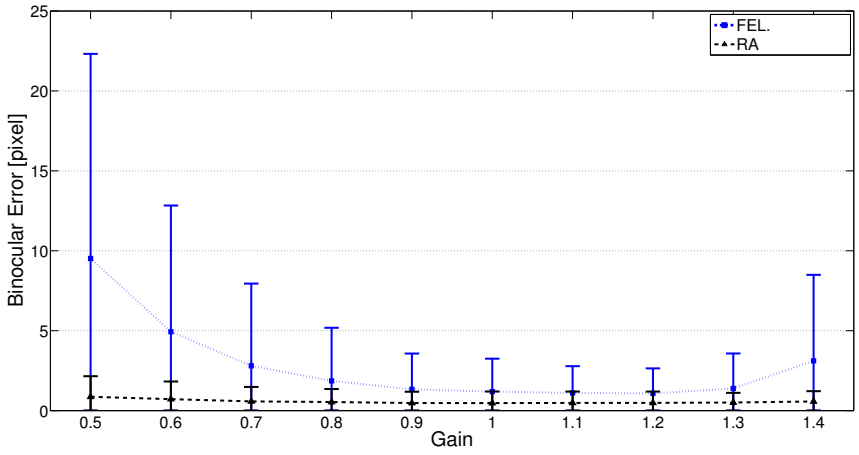


Figure 4: Performance of FEL and RA as a function of the gain used to change the fixed linear controller. Markers represent the mean error of the saccade, bars represent one standard deviation.

Using the *feedback error learning* the best performance was achieved with the gain set to 1.2. In this condition the mean error was 1.07 ± 1.57 pixels. In the worst trial the saccade brought the target at 27.37 pixeles, not very far from the center, considering that the size of the image is 1024×768 pixels. In the *recurrent architecture*, the best performance was achieved with a gain set to 1.0. In this condition the mean error was 0.47 ± 0.72 pixels. In the worst trial the saccade brought the target at 25.15 pixeles from the center. Thus, in the worst case the two architectures provide similar results, but the average behavior of the *recurrent architecture* is twice as good as the *feedback error learning*, even if the input dimension is bigger and the neural networks have the same number of units (500). The difference between the two approaches

is significant when the gain changes. Indeed, for *feedback error learning*, the performance degraded considerably with low and high gains. For example, with a gain of 0.5 the mean error was 9.5 ± 12.80 pixels and the worst trial left the target at a distance of 235.55 pixels from the center. With the same gain, the mean error of the *recurrent architecture* was 0.86 ± 1.29 pixels and the worst trial left the target at a distance of 31.66 pixels from the center (Fig. 4). So, the performance of RA is more invariant to the choice of the gain, and this solution is preferable for saccade control.

A disadvantage of the *recurrent architecture* is the computational time required to compute the eye movement. As shown in Fig. 5, the number of iterations depends on the chosen gain. With the tested gains, the average number of loops is between 3 ± 1 and 9 ± 2, while the maximun number of loops changes from 10 to 24. Thus, the computation time of the *recurrent architecture* depends on the trial, and it is not stationary as in the case of the *feedback error learning*.

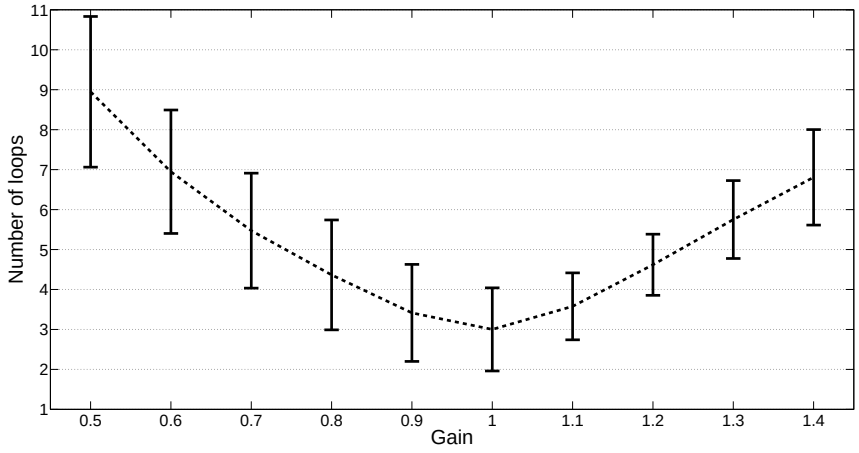


Figure 5: Number of loops (mean and standard deviation) in the *recurrent architecture* as a function of the gain.

4.3. Monocular v.s. Binocular encoding

In this experiment we compared the performance of the monocular and binocular encodings using the RA. In the monocular encoding we have two separate controllers for the left and the right cameras. Each monocular circuit has less input than the binocular one as both the visual target and the eye

position are composed by two values instead of three (see table 1). Similarly to the binocular case, the two 2×2 matrices $\mathbf{B}_l$ and $\mathbf{B}_r$ were calculated using the least square method on the input-output points of the dataset. When the same visual stimulus is observed in both images, the two circuits generates two independent saccades. However, given that our robotic head has only one tilt motor common to both eyes, we choose to generate the tilt command by averaging the output of the two controllers.

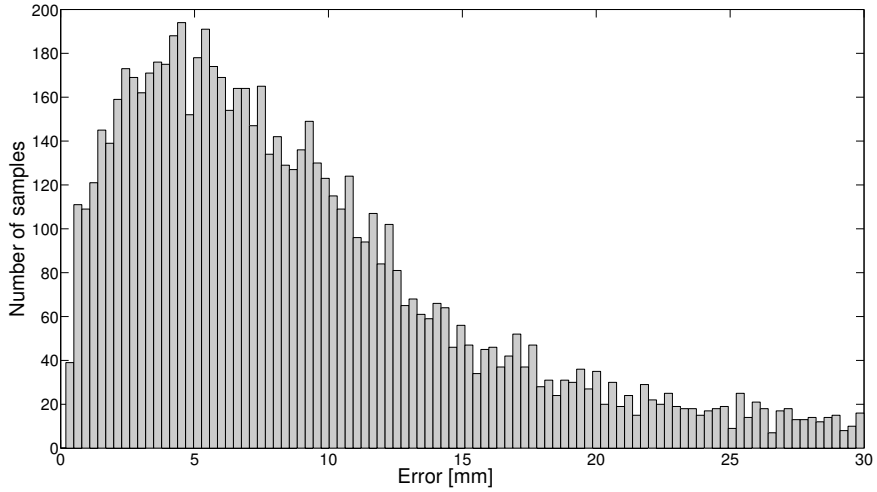
As in the previous experiments, we performed the simulations changing the gain at the input of the linear controller. As expected, we observed that also for the monocular case the error is almost independent from the gain, although, in this case the error is higher (5 ± 5 pixels).

Even if the binocular encoding works better, the monocular approach requires less inputs and can work reasonably well if the targets are placed on a plane. For example, Chao et al. (2010), in their algorithm for visual-oculomotor coordination, considered a target to be in the fovea when it was inside a radius of 20 pixels from the center of the image, that in their case had a resolution of 300 pixels. In our experiments, considering a fovea with a radius of 20 pixels in an image with a resolution of 1024×768 pixels, a saccade brings the target inside the fovea in the 97.9% of the trials for the monocular case and in the 99.99% for the binocular one. Thus, if the goal of the system is to bring the target inside the fovea, the monocular approach works reasonably well and could be preferred for its simplicity.

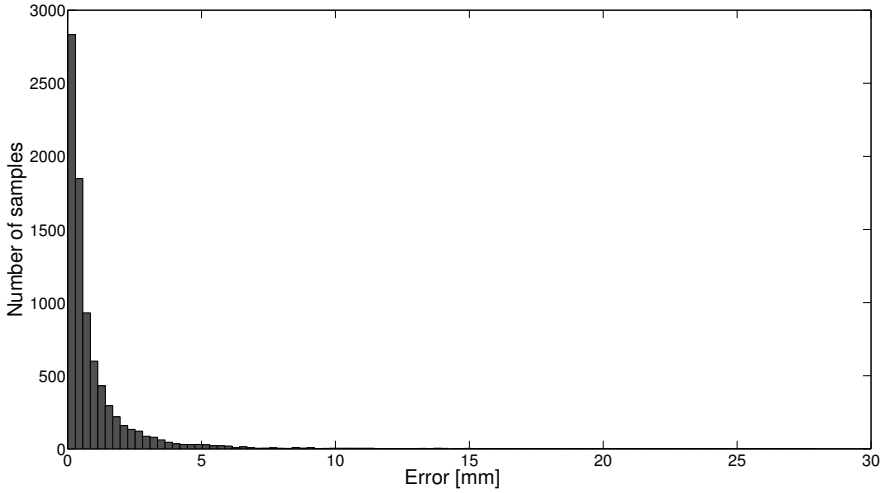
However, when the robot needs to physically interact with the environment, it needs to precisely compute the 3D position of the target. For example, if the task is reaching an extra-foveal target without gazing at it (covert attention), the robot should convert the retinotopic target position into a head/body-centered representation, and then convert this representation into an arm movement. For this reason, we also compute the error of the gazing point in the Cartesian space, as the Euclidean distance between the position of the target and the fixation point. In this experiment, we used the gain set to 1 for both the encodings, which is the one that provides best results. The mean error on the testing sets was 1.04 ± 2.09 mm for the binocular approach and 12.39 ± 15.11 mm for the monocular one. In the worst case the error was 60.13 mm for the binocular approach and 276.24 mm for the monocular one.

Figure 6 shows the histograms of the gazing error in the Cartesian space. For the sake of clarity we discarded errors greater than 30 mm. From the histogram we can appreciate that the typical error is 0.15 mm for the binocular

case and 4.5 mm for the monocular one.



(a) Monocular encoding



(b) Binocular encoding

Figure 6: Histogram of the gazing error for the monocular and binocular approach. The error is computed as Euclidean distance in the Cartesian space between the target point and the fixation point after the saccade.

The worst performance of the monocular approach with respect to the

binocular one is mainly due to the lack of depth cues, which are necessary due to the fact that the center of rotation of the camera does not lie on the optical center of the camera. Figure 7 shows the gazing error as a function of the distance from the target. For the binocular approach, the gazing error tends to increase with the distance of the target, and this is expected because the binocular disparity, as other visual cues, are inversely proportional to the distance. On the other hand, the monocular approach is more precise for the targets that are located around 900 mm of distance, and its accuracy degrades for targets placed at a closer or farther distance. This is because the training of the adaptive controller depends on the statistic of the input data. Given that depth information is lost using the monocular encoding, the system adapts implicitly to perform better in the typical scenario. Indeed, the mean distance of the target in the training points was 924 mm, which corresponds with the results of Fig. 7.

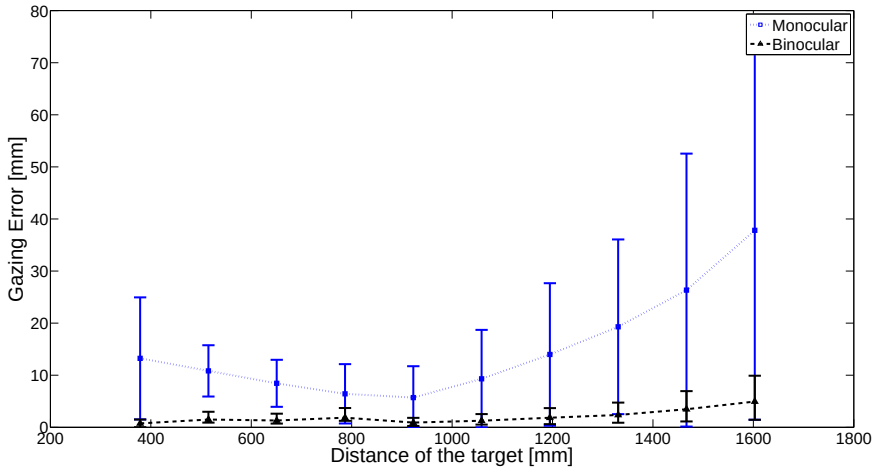


Figure 7: Performance of the monocular and binocular encoding as a function of the target distance. Markers represent the mean error of the saccade, bars represent one standard deviation.

5. Robotic results

5.1. Robot setup

Our robot *Tombatossals* is a humanoid torso endowed with a mechatronic head and two multi-joint arms (Fig. 8). Each arm (*Mitsubishi PA10*) has

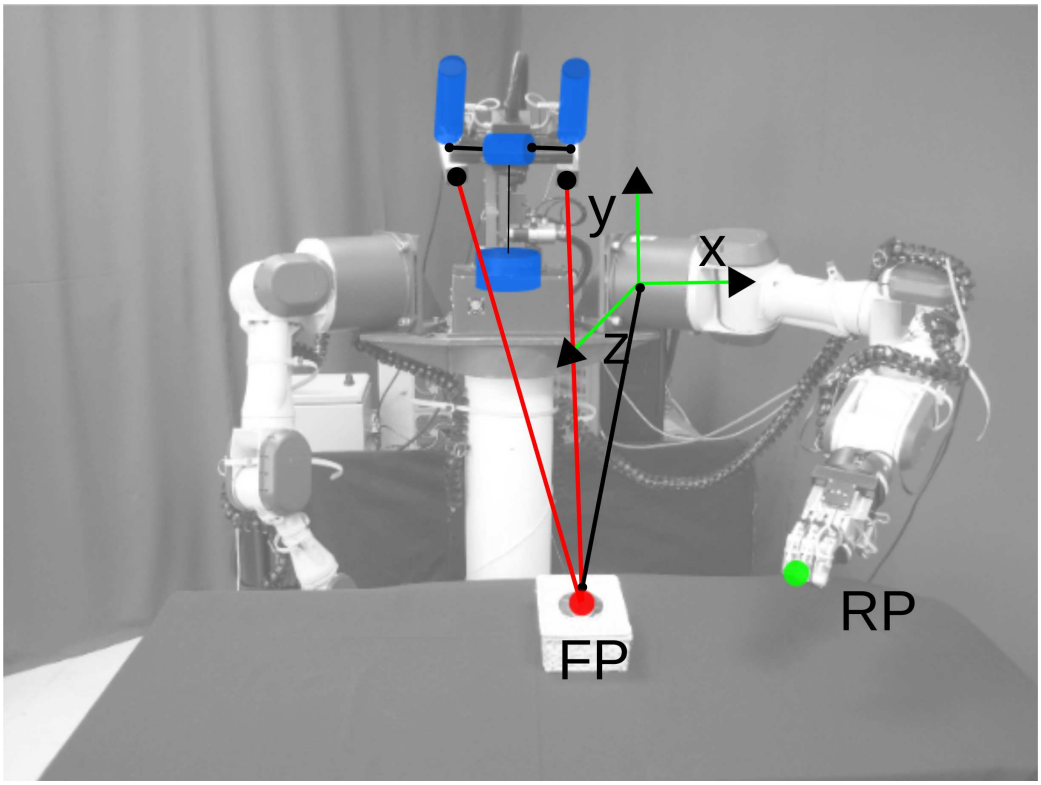


Figure 8: The UJI humanoid torso: Tombatossals. Blue cyclinders represent the four joints of the head. The configuration of the head creates an implicit representation of the fixation target (FP) that can be made explicit trough triangulation (red lines). Reaching a target (RP) requires to convert the gaze direction into a 3D point of the Cartesian space that is centered in the shoulder of the arm.

seven degrees of freedom and is equipped with a three-finger hand (*Schunk SDH2* on the right arm and *Barrett Hand* on the left one). Both the head and the arms are equipped with encoders that provide the motor positions with high precision.

The robotic head (*Robosoft TO40*) mounts two cameras with a resolution of 1024×768 pixels that can acquire color images at 30 Hz (*The Imaging Source, DFK 31AF03-Z2*). The pixel size is 4.65 μm and the focal length was set to 5 mm to obtain a wide field of view. The cameras can actively move by means of a common tilt and two independent pan motors. The baseline between the cameras is about 270 mm. The center of rotation of the motors (tilt, left pan and right pan) do not lie on the optical point of the

Table 3: Denavit-Hartenberg model of the left side of the head. θ_p, θ_t and θ_l are the rotary joints of the common pan, tilt and left verge motors. n_l is a virtual prismatic joint that depends on the focal length of the camera. The model of the right side of the head is obtained by changing the sign of r of the joint θ_t, and by replacing θ_l and n_l with θ_r and n_r respectively.

Joint	θ $[rad]$	r $[m]$	a $[m]$	α $[rad]$	offset
1	θ_p	0.275	0.01	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
2	θ_t	0.135	0.0	$-\frac{\pi}{2}$	π
3	θ_l	0.0	0.0	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
4	0.0	n_l	0	0	0.048

camera lens, so that their rotation produces both rotation and translation of the optical point. Due to this translational component, the visuo-oculomotor transformation depends on the distance from the target. This misalignment between the optical point and the center of rotation depends on the focal length of the camera, and it is present virtually in every robotic system and also in the human eyes (Santini and Rucci, 2007). In the described setup we measured a displacement of about 48 mm.

Table 3 provides the Denavit-Hartenberg parameters of the kinematic model of the robot, that we used in our simulations. The kinematic chain begins at the base of the neck and terminates at the optical point of the left camera. The three degrees of freedom of the left side of the head (common pan, tilt and left verge) are described by the rotatory joints θ_p, θ_t and θ_l. Moreover we added to the model a prismatic joint n_l to simulate the displacement between the center of rotation of the left pan with respect to the optical point of the camera. The same model can be used for the right camera by changing the sign to the parameters r of θ_t, and by replacing θ_l and n_l with θ_r and n_r respectively. The four joints of the robotic head are also shown in Fig. 8. In this work we did not use the common pan of the neck which was set to zero.

5.2. Feedback error learning v.s. Recurrent architecture

After the analysis in simulation, we validated some results on our humanoid torso (Fig. 8). On the robot, we implemented both the FEL and the RA using the binocular encoding. As in the simulation, we used a fixed linear controller $\mathbf{B}$ that was estimated using a linear regression on the dataset composed by visual stimuli and head movements. The visual processing on

the robot was simplified by using as a target a red label placed on the fingertip of the robot left hand. The arm of the robot was used to move precisely the target in the Cartesian space[2] allowing us to reproduce precisely the experimental setup of the simulation.

For both control architectures, we trained the network from scratch using the same paradigm used during the simulation. In this case, we placed the target (fingertip) into 125 positions of the Cartesian space. For each target position the head started the saccadic behavior from 26 different gazing directions. From each position of the head, the robot performed a saccade toward the target and then used the post-saccadic visual error to train the network. In this way we acquired 3250 training points.

In a second phase, we tested the algorithm on other points of the visual space. In this case we initialize the gaze of the robot to 1000 random positions and the task was to perform a ballistic eye movement toward the visual target. After each saccade we recorded the Euclidean distance of the target from the center of the cyclopean visual field. In this case we skipped the training step.

Using a gain set to 1 the average radial error on the testing set was of 4.32 ± 2.34 pixels for the *feedback error learning* and 3.70 ± 2.75 pixels for the *recurrent architecture*. So in this case the performance of the two architectures is comparable. Using a gain of 0.5 we achieved a result of 6.47 ± 4.2 pixels using the *feedback error learning* (performance decrease of a 50%) and 4.83 ± 3.40 pixels using the RA (performance decrease of a 31%). The improvement provided by the *recurrent architecture* with respect to the *feedback error learning* is less evident than it is in the simulation results, probably because we tested the algorithm on a smaller region of the space. However, these results are in accordance with the ones obtained in simulation, and show that the *recurrent architecture* is less sensitive to the choice of $\mathbf{B}$ and it is slightly more accurate.

Table 4 provides an overview of the results reported in the related literature. Even though it is quite difficult to compare the several approaches due to the differences in the robotic setups, the dimension of the working space, and how errors are computed, we can observe that performance achieved by our system is comparable with the state of the art.

[2]The origin of this frame is centered on the shoulder of the robot, as shown in Fig. 8.

Table 4: Performance of the binocular visual-oculomotor transformation in the literature. Resolution of the camera and error in pixels as provided by the authors.

Approach	Resolution	Error [pixel]
Recurrent architecture	1024×768	3.70 ± 2.75
Feedback error learning	1024×768	4.32 ± 2.34
Forssén (2007)	640×480	5.5
Bruske et al. (1997)	512×512	2.5
Schenck and Moller (2004)	1×1	0.06

5.3. Representation of the space

In this experiment we used the *recurrent architecture* to test how the robot is able to represent its peripersonal space. Indeed, the configuration of the head creates an implicit representation of the fixation target that can be made explicit trough triangulation (see red lines in Fig. 8). One way to test this representation is to use the saccade control to obtain the configuration of the head that is required to fixate the target object. This configuration can then be converted into a 3D point of the Cartesian space to analyze the achieved error. In this experiment we perform this test by using the Cartesian space centered in the center of the shoulder, which can be used to reach a target object (Fig. 8).

The robot starts looking straight ahead, and we placed a target at 1000 different positions in a cubic space of $40 \times 40 \times 40 mm^3$ in front of the robot, starting from a distance of 86 cm. For each target we calculated the saccadic movement. Without performing the movement, we used the predicted post-saccadic eye position to compute the Cartesian position of the target. Instead of using triangulation (Hartley and Sturm, 1997), this conversion was performed by a neural network. The input of this network is the position of the head motors, and it provides as output the Cartesian position of the target in the frame of reference centered on the left shoulder (Fig. 8). This network was previously trained, and achieved an average error of 6.0 ± 4.0 mm, where the error was computed as Euclidean distance from the target in the 3D Cartesian space. Thus, we compared the Cartesian position of the target as provided by our prediction with the actual position of the target as provided by the kinematic model of the arm, observing an average error of $25 \pm 17 mm$. The mean and the standard deviation of the error as a function of the target distance is shown in Fig. 9, which shows that the reaching error is quite invariant to the distance, up to about 1100 mm. This

result was achieved just planning the saccade, without executing it, so that it represents the ability of the robot to perform extra-foveal ballistic reaching. Indeed, the achieved accuracy is good enough to allow our robot to reach successfully a target object placed inside the cubic region taken into account in this experiment. The robot is thus able to create and exploit a coherent representation of the nearby space.

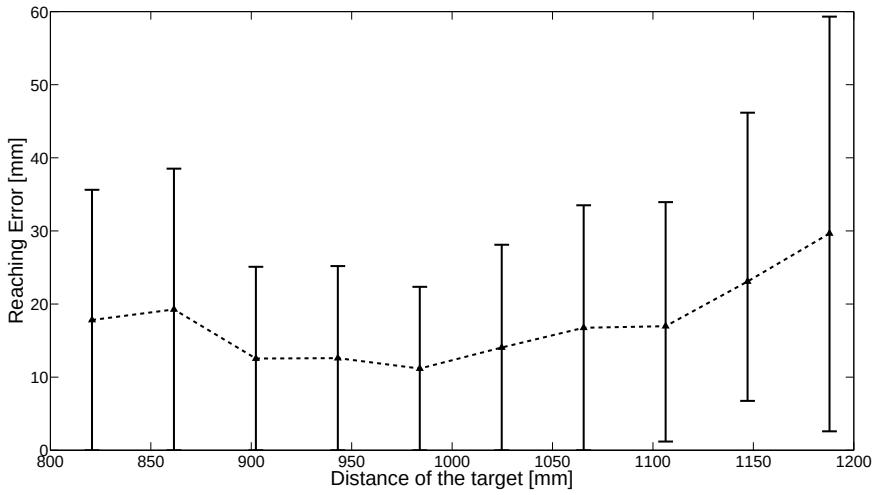


Figure 9: Mean and standard deviation of the reaching error as a function of the target distance.

6. Discussion

This work is part of our research on a sensorimotor framework able to create an implicit representation of the environment and simultaneously learn the sensorimotor associations among vision, the oculomotor and arm-motor systems by means of reaching and gazing (Chinellato et al., 2011; Antonelli et al., 2011, 2013a). The visual representation of space is maintained by a spherical-like coordinate system that is implicitly defined by the gaze direction. This representation is particularly suitable for autonomous learning and has been adopted in several recent works (Schenck et al., 2003; Hoffmann et al., 2005; Chinellato et al., 2011; Jamone et al., 2014).

In this paper we focus on the learning of the visuo-oculomotor transformation. With respect to our previous work, the implemented transformation

takes both visual and proprioceptive cues as input and we use I-SSGPR (Gijsberts and Metta, 2012) instead of standard radial basis function networks. The I-SSGPR approximates the Gaussian activation of neurons using sparse features in order to alleviate the curse of dimensionality. Thus, this neural network can easily handle the input dimensionality required by the visual-oculomotor transformation. On top of that, the algorithm works in real time and its performance is better than other widely used neural networks, such as LWPR (Vijayakumar et al., 2005), as shown by (Gijsberts and Metta, 2012).

In this work we compared two control strategies that were proposed to model the cerebellum. Both models employ a fixed controller that provide a coarse inverse model of the plant, and an adaptive controller that refines the accuracy of the control schema. In the fields of computational neuroscience and bio-inspired robotics, the role of the fixed controller is usually associated to the brainstem, while the adaptive component is associated to the cerebellum (Dean et al., 1994; Bruske et al., 1997). Our results show that the *recurrent architecture* outperforms the *feedback error learning* in terms of accuracy, and it is less sensitive to the choice of the inverse controller. However, it requires more computational time to solve the recurrent loop. The additional amount of time depends on the input data and on the choice of the fixed controller.

Finally, we studied how the choice of the inputs (monocular v.s. binocular encoding) influences the saccade accuracy and representation of the space. In robotics, both encodings have been used to represent such a transformation. For example Marjanovic et al. (1996); Chao et al. (2010); Shibata et al. (2001) employed the monocular encoding, while Schenck and Moller (2004); Forssén (2007) used the binocular one. Only Bruske et al. (1997) compared the two approaches but only in terms of saccade accuracy. In a recent work (Antonelli et al., 2013d) we compared the two encodings also in terms of spatial representation and we showed that the improvement provided by the binocular approach is relevant for the space representation, but not so much for the execution of visually-guided eye movements. However, our analysis focused only on a planar model of the robotic head, whereas here we extended our analysis to the 3D space.

In neuroscience, saccadic eye movements have been studied using both one and the two eyes. However, determining whenever saccades in humans are generated by a monocular or binocular approach is still controversial. More than a century ago, Hering and Helmholtz already debated the neural basis of binocular coordination. Evidence provided by Zhou and King

(1998) suggested that some excitatory burst neurons (EBN) in the paramedian pontine reticular formation (PPRF) encode saccades monocularly, and that binocular coordination was learned in infancy. This conclusion raised many questions about how those putative monocular signals arise and how they are processed downstream. The existence of monocular EBNs requires more substantiation before they can be included in saccadic models (Scudder et al., 2002). In psychophysics, saccades are studied both using monocular and binocular setups but we have not found in the literature a comparison of their accuracies. Some studies report that the presence of a monocular vs binocular distractor has different effects on the saccade accuracy, so showing that binocular information is used when available (Griffiths et al., 2006). Moreover, verifying the actual performance of monocular encoding with the robot allows us to predict the quality of saccades and perception in the case of occlusions, and also evaluate the opportunity of consistently employing a more demanding binocular processing.

We have devised two ways of speeding up the learning process. On the one hand, in the learning of the inverse model, the covariance matrix stored by the neural network can be used to influence the attentive process, thus choosing the target with the greatest uncertainty (Antonelli et al., 2013c). On the other hand, in the learning of the forward model, the network can be trained using the information from multiple visual stimuli at the same time (Forssén, 2007). Given that the *recurrent architecture* employs both the inverse and the forward models, we are now working on extending the model to learn from multiple visual stimuli, and to perform saccades toward most informative targets.

Conclusion

This work is part of our research on a sensorimotor framework that aims at creating an implicit representation of space based on visual and somatosensory cues (Antonelli et al., 2014b). Here we focused on the association between the retinotopic encoding of the target and its head-centered representation. The internal model that describes this association is learned by means of saccadic eye movements which in turn help to create the representation of the surrounding space. On the other hand, accurate saccades are executed thanks to the recurrent interaction between a fixed feedback controller, that emulates the brainstem, and the adaptive internal model, that emulates the cerebellum. The internal model is maintained by an adaptive filter implemented with a real-time version of Gaussian Process Regression.

Experimental results show that this *recurrent architecture* outperforms the classical *feedback error learning* architecture in terms of its sensitivity to system parameters. Finally, we have shown that a coherent representation of the 3D space requires a binocular encoding of the visual-oculomotor transformation. For what concerns this transformation, future work is directed toward the study of how the knowledge of our internal model can influence the attentive process in order to speed-up the learning process.

References

Aloimonos, J., Weiss, I., Bandyopadhyay, A., 1988. Active vision, 333–356.

Antonelli, M., Chinellato, E., del Pobil, A. P., 2011. Implicit mapping of the peripersonal space of a humanoid robot. In: Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on.

Antonelli, M., Chinellato, E., del Pobil, A. P., 2013a. On-Line Learning of the Visuomotor Transformations on a Humanoid Robot. Vol. 193 of Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg, pp. 853–861.

Antonelli, M., del Pobil, A., Rucci, M., 2014a. Bayesian multimodal integration in a robot replicating human head and eye movements. In: IEEE International Conference on Robotics and Automation (ICRA).

Antonelli, M., del Pobil, A. P., Rucci, M., 2013b. Depth estimation during fixational head movements in a humanoid robot. In: Computer Vision Systems. Springer, pp. 264–273.

Antonelli, M., Duran, A., Chinellato, E., del Pobil, A. P., 2013c. Speeding-up the learning of saccade control. In: Lepora, N., Mura, A., Krapp, H., Verschure, P., Prescott, T. (Eds.), Biomimetic and Biohybrid Systems. Vol. 8064 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 12–23.

Antonelli, M., Duran, A. J., Del Pobil, A. P., 2013d. Application of the visuo-oculomotor transformation to ballistic and visually-guided eye movements. In: Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, pp. 1–8.

Antonelli, M., Gibaldi, A., Beuth, F., Duran, A. J., Canessa, A., Chessa, M., Solari, F., del Pobil, A., Hamker, F., Chinellato, E., Sabatini, S., 2014b. A hierarchical system for a distributed representation of the peripersonal space of a humanoid robot. IEEE Trans. Auton. Mental Develop, 1–15.

Antonelli, M., Grzyb, B., Castelló, V., del Pobil, A., 2012a. Augmenting the reachable space in the nao humanoid robot. In: Workshop at the Twenty-Sixth AAAI Conference on Artificial Intelligence.

Antonelli, M., Grzyb, B. J., Castelló, V., del Pobil, A. P., 2012b. Plastic Representation of the Reachable Space for a Humanoid Robot. Vol. 7426 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 167–176.

Aytekin, M., Rucci, M., 2012. Motion parallax from microscopic head movements during visual fixation. Vision Research 70, 7–17.

Bertero, M., Poggio, T. A., Torre, V., 1988. Ill-posed problems in early vision. Proceedings of the IEEE 76 (8), 869–889.

Bremmer, F., Kubischik, M., Hoffmann, K.-P., Krekelberg, B., 2009. Neural dynamics of saccadic suppression. The Journal of neuroscience 29 (40), 12374–12383.

Bruske, J., Hansen, M., Riehn, L., Sommer, G., 1997. Biologically inspired calibration-free adaptive saccade control of a binocular camera-head. Biological Cybernetics 77 (6), 433–446.

Bruske, J., Sommer, G., 1995. Dynamic cell structure learns perfectly topology preserving map. Neural Computation 7 (4), 845–865.

Chao, F., Lee, M., Lee, J., 2010. A developmental algorithm for ocular-motor coordination. Robotics and Autonomous Systems 58 (3), 239–248.

Chen-Harris, H., Joiner, W., Ethier, V., Zee, D., Shadmehr, R., 2008. Adaptive control of saccades via internal feedback. The Journal of Neuroscience 28 (11), 2804.

Chinellato, E., Antonelli, M., del Pobil, A. P., 2012. A pilot study on saccadic adaptation experiments with robots. In: Prescott, T. J., Lepora, N. F., Mura, A., Verschure, P. F. (Eds.), Biomimetic and Biohybrid Systems. Vol. 7375. Springer Berlin Heidelberg, pp. 83–94.

Chinellato, E., Antonelli, M., Grzyb, B. J., del Pobil, A. P., 2011. Implicit sensorimotor mapping of the peripersonal space by gazing and reaching. Autonomous Mental Development, IEEE Transactions on 3, 43–53.

Collins, T., Doré-Mazars, K., Lappe, M., 2007. Motor space structures perceptual space: Evidence from human saccadic adaptation. Brain research 1172, 32–39.

Damas, B., Jamone, L., Santos-Victor, J., Nov 2013. Open and closed-loop task space trajectory control of redundant robots using learned models. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. pp. 163–169.

Dean, P., Mayhew, J. E., Langdon, P., 1994. Learning and maintaining saccadic accuracy: a model of brainstem–cerebellar interactions. Journal of Cognitive Neuroscience 6 (2), 117–138.

Forssén, P., 2007. Learning saccadic gaze control via motion prediciton. Computer and Robot Vision (CRV), 2007. Fourth Canadian Conference on, 44–54.

Gijsberts, A., Metta, G., 2012. Real-time model learning using incremental sparse spectrum gaussian process regression. Neural Networks.

Girard, B., Berthoz, A., 2005. From brainstem to cortex: computational models of saccade generation circuitry. Progress in Neurobiology 77 (4), 215–251.

Griffiths, H., Whittle, J., Buckley, D., 2006. The effect of binocular and monocular distractors on saccades in participants with normal binocular vision. Vision Research 46 (12), 72 – 81.

Hartley, R. I., Sturm, P., 1997. Triangulation. Computer Vision and Image Understanding 68 (2), 146 – 157.

Haykin, S. S., et al., 2001. Kalman filtering and neural networks. Wiley Online Library.

Hoffmann, H., Schenck, W., Möller, R., 2005. Learning visuomotor transformations for gaze-control and grasping. Biological Cybernetics 93 (2), 119–130.

Jamone, L., Brandao, M., Natale, L., Hashimoto, K., Sandini, G., Takanishi, A., 2014. Autonomous online generation of a motor representation of the workspace for intelligent whole-body reaching. Robotics and Autonomous Systems 62 (4), 556 – 567.

Jordan, M., Rumelhart, D., 1992. Forward models: Supervised learning with a distal teacher. Cognitive Science: A Multidisciplinary Journal 16 (3), 307–354.

Karayiannis, N. B., Venetsanopoulos, A. N., Jul 1992. Fast learning algorithms for neural networks. Circuits And Systems-II: analog and digital signal processing, IEEE Transactions on 39 (7), 1–22.

Kawato, M., 1990. Feedback-error-learning neural network for supervised motor learning. Advanced neural computers 6 (3), 365–372.

Kuperstein, M., 1988. Neural model of adaptive hand-eye coordination for single postures. Science 239, 1308–1311.

Lappe, M., Oct 2008. What is adapted in saccadic adaptation? The Journal of Physiology 587 (1), 5–5.

Lenz, A., Balakrishnan, T., Pipe, A. G., Melhuish, C., 2008. An adaptive gaze stabilization controller inspired by the vestibulo-ocular reflex. Bioinspiration & biomimetics 3 (3), 035001.

Marjanovic, M., Scassellati, B., Williamson, M., 1996. Self-taught visually guided pointing for a humanoid robot. From Animals to Animats 4: Proc. Fourth Int l Conf. Simulation of Adaptive Behavior, 35—44.

McLaughlin, S., 1967. Parametric adjustment in saccadic eye movements. Attention, Perception, & Psychophysics 2 (8), 359–362.

Nguyen-Tuong, D., Peters, J., 2011. Model learning for robot control: a survey. Cognitive processing 12 (4), 319–340.

Park, J., Sandberg, I. W., 1991. Universal approximation using radial-basis-function networks. Neural computation 3 (2), 246–257.

Porrill, J., Dean, P., 2007. Recurrent cerebellar loops simplify adaptive control of redundant and nonlinear motor systems. Neural computation 19 (1), 170–193.

Porrill, J., Dean, P., Anderson, S. R., 2013. Adaptive filters and internal models: Multilevel description of cerebellar function. Neural Networks 47, 134–149.

Porrill, J., Dean, P., Stone, J. V., 2004. Recurrent cerebellar architecture solves the motor-error problem. Proceedings of the Royal Society of London-B 271 (1541), 789–796.

Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines. In: Advances in neural information processing systems. pp. 1177–1184.

Santini, F., Rucci, M., 2007. Active estimation of distance in a robotic system that replicates human eye movement. Robotics and Autonomous Systems 55 (2), 107–121.

Schenck, W., Hoffmann, H., Möller, R., sep 2003. Learning internal models for eye-hand coordination in reaching and grasping. Proceedings of the European Cognitive Science Conference, Osnabrück, Germany, 10–13.

Schenck, W., Moller, R., 2004. Staged learning of saccadic eye movements with a robot camera head. Progress in Neural Processing 15, 82–94.

Schenck, W., Möller, R., 2006. Learning strategies for saccade control. Künstliche Intelligenz (3/06), 19–22.

Scudder, C. a., Kaneko, C. S., Fuchs, A. F., Mar. 2002. The brainstem burst generator for saccadic eye movements: a modern synthesis. Experimental brain research 142 (4), 439–62.

Shibata, T., Vijayakumar, S., Conradt, J., Schaal, S., Apr. 2001. Biomimetic oculomotor control. Adapt. Behav. 9 (3-4), 189–207.

Sun, G., Scassellati, B., 2004. Reaching through learned forward model. Humanoid Robots, 2004 4th IEEE/RAS International Conference on DOI - UR - 1, 93– 112 Vol. 1.

Vijayakumar, S., D'souza, A., Schaal, S., 2005. Incremental online learning in high dimensions. Neural Computation 17 (12), 2602–2634.

Wolpert, D. M., Kawato, M., 1998. Multiple paired forward and inverse models for motor control. Neural Networks 11 (7), 1317–1329.

Wolpert, D. M., Miall, R. C., Kawato, M., 1998. Internal models in the cerebellum. Trends in cognitive sciences 2 (9), 338–347.

Zhou, W., King, W. M., 1998. Premotor commands encode monocular eye movements. Nature 393 (6686), 692–695.