Self-calibrating smooth pursuit through active efficient coding

Céline Teulière, S Forestier, L Lonini, Cong Zhang, Y Zhao, B Shi, J Triesch

Self-calibrating smooth pursuit through active efficient coding

C. Teulière[a,c,*], S. Forestier[a,d], L. Lonini[a], C. Zhang[b], Y. Zhao[b], B. Shi[b], J. Triesch[a]

[a] *Frankfurt Institute For Advanced Studies, Goethe University, Frankfurt am Main, Germany*
[b] *Department of Electronic and Computer Engineering, HK University of Science and Technology, Hong Kong, China*
[c] *Clermont Université, Université Blaise Pascal, Institut Pascal, Clermont-Ferrand, France*
[d] *École normale supérieure de Rennes, Bruz, France*

Abstract

This paper presents a model for the autonomous learning of smooth pursuit eye movements based on an efficient coding criterion for active perception. This model accounts for the joint development of visual encoding and eye control. Sparse coding models encode the incoming data at two different spatial resolutions and capture the statistics of the input in spatio-temporal basis functions. A reinforcement learner controls eye velocity so as to maximize a reward signal based on the efficiency of the encoding. We consider the embodiment of the approach in the iCub simulator and real robot.Motion perception and smooth pursuit control are not explicitly expressed as tasks for the robot to achieve but emerge as the result of the system's active attempt to efficiently encode its sensory inputs. Experiments demonstrate that the proposed approach is self-calibrating and robust to strong perturbations of the perception-action link.

Keywords: Autonomous learning, active perception, smooth pursuit, efficient coding, robotics

2010 MSC: 00-01, 99-00

*Corresponding author
Email address: celine.teuliere@univ-bpclermont.fr (C. Teulière)

1. Introduction

Since the development of information theory, the idea of exploiting redundancy of information in signals to encode them in an efficient manner has been widely applied to different scientific areas. Concepts such as sparse coding techniques are now well known tools in mathematics, signal processing and computer science.

In neuroscience, efficient coding was proposed as a principle for the encoding of sensory information in the brain [1, 2, 3]. In particular, one popular expression of the *efficient coding hypothesis* posits that only a few neurons fire at a given time, thus representing sensory inputs with a "sparse code". Several studies proposed to use a sparse coding mechanism on natural stimuli by decomposing them as a linear combination of a small number of basis functions from an over-complete dictionary. Interestingly, it was shown that the basis functions that were learned when optimizing the reconstruction of the input using such a sparse code resemble the receptive fields of sensory neurons in visual, auditory, or olfactory systems [4, 5, 6]. In particular, this efficient coding hypothesis implies that the sensory representation in the brain captures the statistics of the sensory inputs. Those statistics obviously depend on the environment of the agent. Importantly, they are also shaped by its behaviour [7, 8], which can be directed to control the incoming sensory information. However, there is still little work that accounts for the role of behaviour in the development of an efficient sensory coding.

Recently, [9] used the efficient coding of the causal relation between motor actions and sensory feedback as a drive for the co-development of sensory and motor maps. [10] applied the efficient coding principle to active vision and showed that this principle can lead to the joint learning of an efficient depth representation and eye vergence movements. This model was embodied into a robot binocular vision system in [11] and showed strong robustness properties [12]. Even more recently, [13] suggested that this model can be extended to a larger range of action-perception loops, by simulating the emergence of smooth

pursuit behaviour as the result of an efficient encoding criterion. Note that smooth pursuit control in humans appears to improve in the same time period as motion perception which suggests a co-development [14].

In this paper, we extend the approach of [13] and take it to a next step through its embodiment in a robotic system. We believe that the autonomous learning of active perception loops is of great interest in the robotics context in order to develop self-calibrating systems that can flexibly adapt to their environments.

Smooth pursuit abilities and gaze control in general are fundamental not only to humans but also humanoids as they condition lots of basic behaviours such as e.g. reaching objects [15]. Motion perception and tracking have been largely studied in the computer vision and robotics communities. Most approaches for motion perception involve either optic flow computation [16, 17], or some form of object representation along with matching or tracking techniques [18] which sometimes require calibrated sensors [19]. Visual servoing approaches [20] can then be used to close the loop between perception and action. Such methods provide good performances in terms of accuracy. Some specific controllers have also been designed in the context of smooth pursuit [21, 22]. However, the above methods require knowledge of the kinematic link between the camera velocity and the visual change. This requirement implies both the need for a calibration phase and the inability to handle modifications due to, e.g., a mechanical shock to the system. Some attempts have been made to learn this link [23, 24, 25]. However, such techniques still require prior knowledge on a specific goal for the robot to reach. In this work, we consider a different paradigm: motion perception and smooth pursuit control are not explicitly expressed as tasks for the robot to achieve but emerge as the result of the system's active attempt to efficiently encode its sensory inputs. A sparse coding model (perception component) encodes sensory information from pairs of successive frames using basis functions at different spatial resolutions, while a reinforcement learner (action component) generates the camera movement based on the output of the sparse coding model. Importantly, perception and behaviour develop in parallel,

by minimizing the same cost function: the error between the original stimulus and its reconstruction by the sparse coding model. We call this approach *active efficient coding*.

We extend the work of [13] in the following ways: first, we increase the range of motion that the system can perceive by using a multi-scale approach, and demonstrate its benefits. Secondly, we demonstrate that our system can adapt to drastic perturbations in its perception-action link. Finally, we present experiments both in simulation and on a real robot and show that the model performs well in realistic conditions with the presence of noise and distortions.

2. Model Architecture

This section presents the architecture of our model. An overview of the main components is given in 2.1 and their description is detailed in sections 2.2 and 2.3.

2.1. Overview

Our embodiment of the efficient coding principle to the autonomous learning of smooth pursuit relies on the following idea: when one eye is smoothly pursuing an object of interest, the successive images it senses in its foveal region are very similar and can therefore be encoded efficiently when exploiting this redundancy of information. Our model makes use of this basic idea by considering the encoding quality of pairs of successive image patches as the criterion that drives the movements of the eye.

In the following, we will model one eye by a camera, which can rotate in both pan and tilt degrees of freedom. Note that we only consider one eye for convenience but [26] shows that the model can be extended to two eyes. Rotations of the camera around the line of sight as they occur in the primate visual system are left for future work.

Our model consists of two main components (see Figure 1):

- The sensory encoding component receives image patches of two consecutive frames from the camera, and encodes them as a sparse linear combination of basis functions. It is composed of two sparse coding modules dealing with different spatial resolutions.

- The motor control component is based on a reinforcement learning agent that generates velocity commands for the robot camera according to the encoding of the images. The agent receives a reward for the selected action depending on the efficiency of the encoding of the subsequent image patches by the sensory coding component.

The following subsections describe the two components in more detail.

2.2. Sensory encoding

To allow our system to deal with a large range of motion, we consider two different scales or image resolutions and train one sparse coding model for each scale. The structure of the two models is identical, the only difference is the input they receive: fine or coarse scale image patches.

2.2.1. Patch extraction

Input images are acquired from the camera at a resolution of 320×240 pixels. The camera parameters are unknown for the system and the fixation point is simply defined to be at the center of each image. Assuming visual perception is focused in the image center, we only consider the central region of the image, of size 96×96 for the coarse scale model and 72×72 for the fine scale one. This selection prevents the system from considering too much background when performing smooth pursuit. It is somewhat analogous to modelling a foveal region of the eye. Given the focal length of the system, the coarse scale window covers here a visual angle of about 21° while the fine scale window is about 16°. More details can be found in Section 3.1. From this central region, two different sets of patches are obtained:

- For the fine scale, 81 patches of 8×8 pixels are extracted from the smaller central region of 72×72, without any subsampling or overlap.

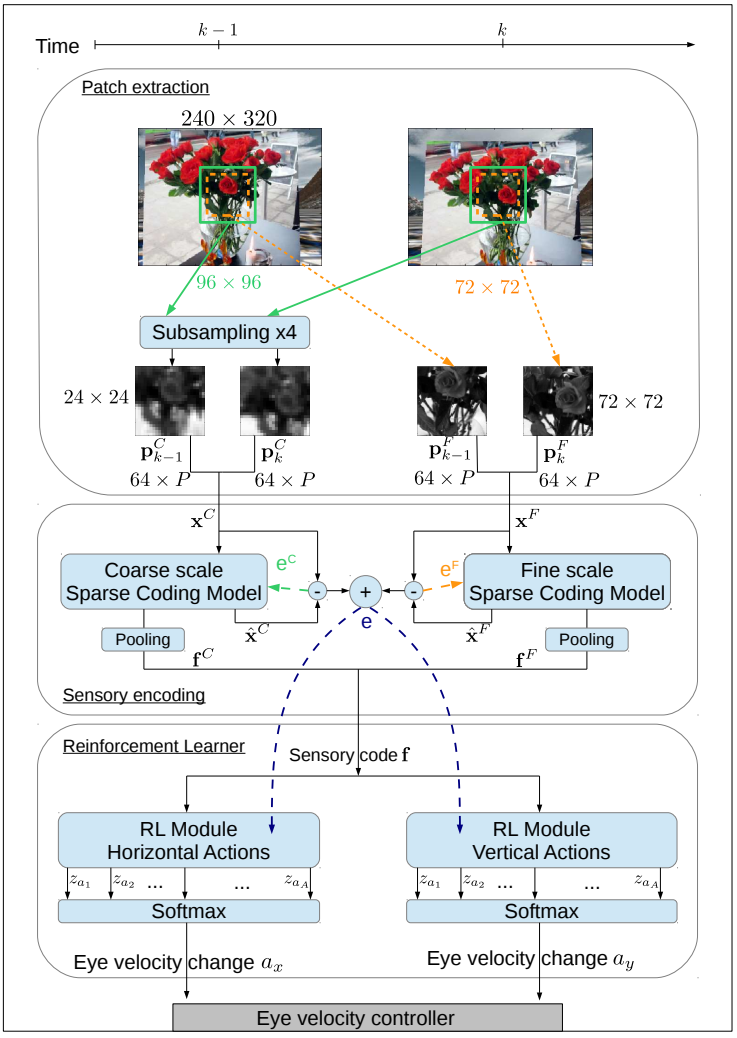


Figure 1: Overview of the model.

- For the coarse scale, the image is subsampled by a factor of 4 using a Gaussian pyramid, and another 81 patches are extracted among all possible overlapping patches of 8×8 pixels. Those patches correspond to patches of size 32×32 in the original image, but are subsampled here to reduce the computational cost.

For each scale s, each patch i of time step k is represented as a column vector $\mathbf{p}^s_{k,i}$. The patches are preprocessed to have zero mean and unit norm. In order to use the redundancy of information between successive images, each patch of time step k is concatenated with the corresponding patch of time step $k-1$ to build spatio-temporal patches :

$$\mathbf{x}^s_{k,i} = \begin{bmatrix} \mathbf{p}^s_{k-1,i} \\ \mathbf{p}^s_{k,i} \end{bmatrix}. \tag{1}$$

At each time step k, each of the two sparse coding models s will receive as input a batch $\mathbf{x}^s_k$ of 81 patches of size 128 ($8 \times 8 \times 2$). As mentioned above, the sparse coding models are identical in structure. In the following we will drop the superscript s and index k, and consider one single sparse coding model receiving a set of patches $\{\mathbf{x}_i\}$ at a given time step.

2.2.2. Sparse coding model

To encode the input patches, we use a sparse coding model, which seeks to best represent the data as a sparse linear combination of basis functions $\mathbf{\Phi}_n$ from a fixed-size dictionary $\mathcal{D} = \{\mathbf{\Phi}_n\}_{n=1:N}$. The dictionary is over-complete, which means that the number of basis functions $N > 128$. We use a value of $N = 288$ in the experiments presented here but we observed that this precise value is not critical to the performances. Formally, each patch $\mathbf{x}_i$ is approximated by:

$$\hat{\mathbf{x}}_i = \sum_{n=1}^{N} \alpha_n^{(i)} \mathbf{\Phi}_n. \tag{2}$$

The sparsity of the encoding is ensured by allowing only 10 coefficients α_n to be non-zero. Starting from a set of random bases the dictionary is updated so that it best represents the input statistics. Learning occurs online using

a two-step procedure. (1) For each patch $\mathbf{x}_i$ the set of coefficients $\{\alpha_n^{(i)}\}_n$ is obtained using the matching pursuit algorithm [27], which tries to find in the dictionary the bases which best explain the input using a greedy search method. (2) The bases are updated through gradient descent to minimize the squared reconstruction error [4] defined by:

$$e = \sum_{i=1}^{P} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2, \tag{3}$$

where $P = 81$ is the number of patches extracted from one image at a particular scale.

Since each patch is composed of the concatenation of information from time steps $k-1$ and k, the learned bases functions capture information about the visual motion between the successive frames.

2.2.3. Pooling

After each patch $\mathbf{x}_i$ has been encoded, the basis function activations $\{\alpha_n^{(i)}\}_n$ are pooled to generate an N-dimensional code $\mathbf{f}$ for the pair of consecutive images, by averaging the squared weighting coefficients over the patches, *i.e.* over different parts of the foveal region:

$$\mathbf{f} = \begin{bmatrix} \frac{1}{P}\sum_{i=1}^{P}\left(\alpha_1^{(i)}\right)^2 \\ \vdots \\ \frac{1}{P}\sum_{i=1}^{P}\left(\alpha_N^{(i)}\right)^2 \end{bmatrix}. \tag{4}$$

$\mathbf{f}$ is an encoding for the image content and motion.

In biological terms, the pooling step roughly corresponds to the operation performed by complex cells, which receive inputs from many simple cells at different locations, but with similar receptive field shapes.

The two N-dimensional codes $\mathbf{f}^C$ and $\mathbf{f}^F$ from the coarse and fine scale models are concatenated into a $2N$-dimensional feature vector which is sent to the reinforcement learning agent that maps it to a velocity change of the eye, as described in the next section.

2.3. Motor control (action)

The motor control component of our model learns the mapping between the visual motion information represented by $\mathbf{f}$ and velocity commands of the eye. Our system is based on a reinforcement learning (RL) framework, where a RL agent seeks to maximize its cumulative reward defined by:

$$R(k) = \sum_{l=0}^{\infty} -\gamma^{-l} \left[e^C(k+l) + e^F(k+l) \right], \tag{5}$$

where e^C (resp. e^F) denotes the reconstruction error for the coarse (resp. fine) scale, and γ is a discounting factor. It is therefore trying to minimize the reconstruction error of the two sparse codes. We use a natural actor-critic algorithm [28], where the policy (the actor) and the value function (the critic) are implemented by neural networks (see Figure 2). Since a separate set of actions controls the pan and tilt rotation of the camera, we use separate policy networks for each axis.

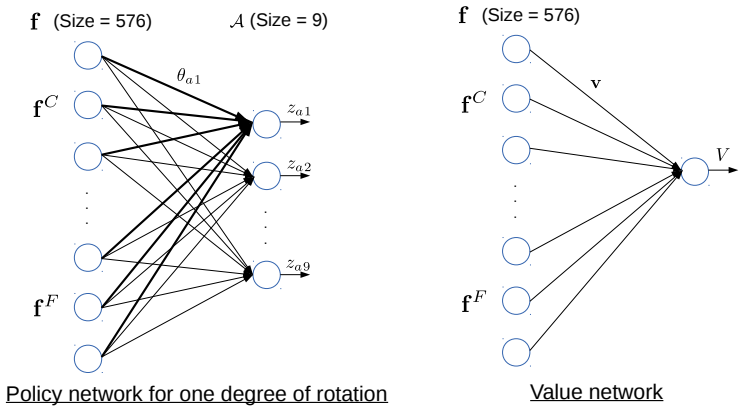


Figure 2: Representation of the policy and value networks. Note that there is one policy network for each rotational degree of freedom of the eye.

The weights of the neural networks are initialized randomly. The critic network receives as input at time k the code $\mathbf{f}(k) = [\mathbf{f}^C(k); \mathbf{f}^F(k)]$, and outputs the value $V(k)$:

$$V(k) = \mathbf{v}^\top(k)\mathbf{f}(k), \tag{6}$$

where $\mathbf{v}(k)$ are the weights of the value network at time k. The policy networks map $\mathbf{f}(k)$ to actions and their output layers contain as many neurons as possible actions. Each action is an increment of the eye velocity, with respect to its current value. We used a finite set $\mathcal{A}_j$ of actions for each rotational degree of freedom j. The actions are spaced on a logarithmic scale to allow coarse and fine changes.

The activation $z_a(k)$ of the output neuron corresponding to the action a at time k is computed as:

$$z_a(k) = \theta_a^\top(k)\mathbf{f}(k), \tag{7}$$

where $\theta_a(k)$ is the vector of weights from $\mathbf{f}(k)$ to the action a.

The probability $\pi_a(k)$ of selecting the action a at time-step k is computed using a softmax non-linearity:

$$\pi_a(k) = \frac{exp\left(\frac{z_a(k)}{T}\right)}{\sum_{l=1}^{A} exp\left(\frac{z_l(k)}{T}\right)}, \tag{8}$$

where A is the number of actions and T is the so-called temperature parameter and controls the amount of exploration *vs.* exploitation of the reinforcement learning agent.

3. Experiments

3.1. Experimental setup

We use the iCub robot head [29] as the experimental platform for the embodiment of our model. This robotic head has a total of six degrees of freedom: three in the neck (pan, tilt and roll rotations), and three in the eyes (independent pan angle for each eye and joint tilt). In our experiments the neck is fixed and only the pan and tilt angle of one eye are controlled.

In order to assess the performance of our method, we used the iCub simulator [30], which provides a controlled environment suitable for extensive training and testing. During the training phase, the robot is placed in a scene where an object is moving at changing velocity. we used a flat textured object moving in

a plane fronto-parallel to the robot, at a distance $d = 1$m (see Figure 3). The translational velocity of the object is converted to the corresponding angular velocity when compared to the eye pan and tilt angular velocities. Accurate smooth pursuit behaviour means that the eye angular velocity matches the object velocity such that the object is stabilized in the image. Both the texture and the object velocity are changed periodically during training. The in-plane velocity of the object $\mathbf{v}_{obj} = [v^H_{obj}; v^V_{obj}]$ is randomly selected using a uniform distribution for the horizontal velocity $v^H_{obj} \in [-25; 25]$ and for the vertical velocity $v^V_{obj} \in [-15; 15]$ (units are degrees per second). The smaller range of vertical velocities was chosen because of the narrower joint limits for vertical eye movements. The velocity vector $\mathbf{v}_{obj}$ is changed every 20 iterations, and its direction is reversed every 10 iterations. One iteration corresponds to 100ms. The texture of the planar object is changed every 200 iterations *i.e.* the same texture is seen with 10 different velocities before being changed. The new texture is randomly selected out of a set of 20 images from the TESTIMAGES project [31]. Examples of such textures are shown in Figure 3. Changing the textures is required so that the sparse coding models receive enough diversity in the input stimuli so that the learned bases can generalize well to new objects.

The images are acquired at a resolution of 320×240 pixels and converted to greyscale. With a focal length equivalent to 257 pixels, 1° of visual angle corresponds to about 4.5 pixels. Therefore, a patch of the coarse (resp. fine) scale covers a visual angle of 7.1° (resp. 1.8°). Although some background can appear depending on the relative position between the object and the eye, most of the time the object covers the entire foveal region.

Our model is implemented in MATLAB and communicates with the robot or the simulator through the YARP middleware [32]. The training time-loop is set to $\Delta t = 100$ms. This time includes the image pre-processing, the sensory encoding and the action selection as presented in Section 2 ($\sim 30ms$), but also a waiting period ($\sim 70ms$) before actually sending the new velocity command. Since this time loop is long with respect to transient periods, the eye velocity can be considered as quasi constant between two image acquisitions. This is a

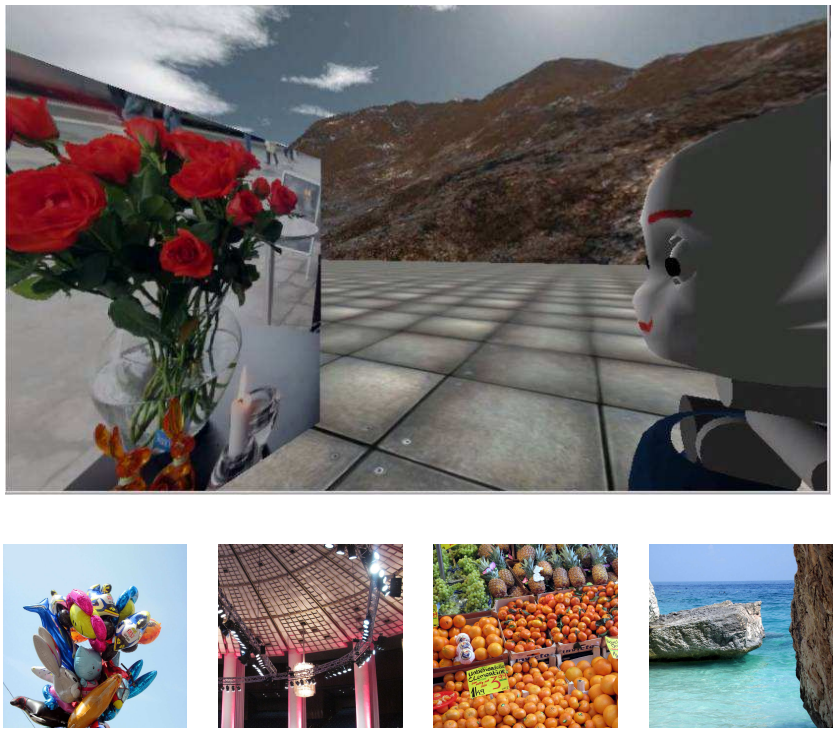


Figure 3: Top: A screenshot of the iCub simulator environment during training. The texture and velocity of the planar object are changed periodically. Bottom: Examples of textures applied on the planar object, extracted from TESTIMAGES dataset [31].

simplifying choice that is linked to our choice of using a time history of only two images to estimate the motion. A longer history window could be considered for example by adding inputs to the policy and value networks (see Figure 2). This would allow the system to take into account more complex dynamics at the cost of an increase in the model complexity. In this work we deliberately chose to simplify the model to assess the performances under easily controlled conditions.

One consequence of the simple dynamics used here is that the action selected at time k will be responsible for the velocity that the eye will have between image $k+1$ and $k+2$ (see Figure 4). Therefore, the reward associated with the action k is received by the agent at time $k+2$.

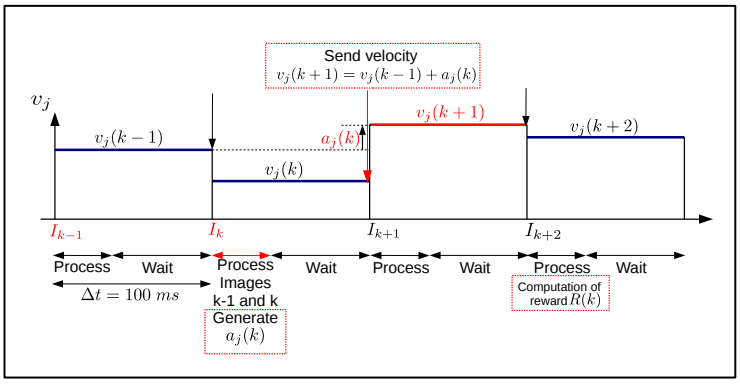


Figure 4: Illustration of the timeline in our setup. The reconstruction error between image patches k+1 and k+2 is used as a reward for the encoding of time step k.

To control the eye motion, we use the velocity controller of the (real or simulated) iCub. The velocity sent to the controller of joint j (pan or tilt) at time k is:

$$v^j(k+1) = v^j(k-1) + a^j(k), \tag{9}$$

where $a^j(k)$ is the selected action for joint j. The delay of two time steps is due to the delay between the selection of an action and its effect on the perception explained above (see Figure 4). We consider the following sets of actions, expressed in $^\circ/s$: $\mathcal{A}_{pan} = \{-32, -16, -8, -4, 0, 4, 8, 16, 32\}$ and $\mathcal{A}_{tilt} =$

$\{-24, -16, -8, -4, 0, 4, 8, 16, 24\}$. The maximal velocity for the pan (resp. tilt) angle is limited to $\pm 30°/s$ (resp. $\pm 20°/s$), and when the eye reaches an absolute angle position of $25°$ (resp. $15°$) it is positioned back to the center. The learning rate for the RL agents is set to 0.4 and the temperature parameter is $T = 1$. We also use a discount factor $\gamma = 0.3$ as defined in equation (5).

The training is performed online. The sparse coding model as well as the RL component are updated at each iteration of the algorithm.

3.2. Performance evaluation

In the following we evaluate the performance of our multi-scale model in autonomous learning of smooth pursuit. In particular, our experiments address the following questions:

- Can the system properly learn to perform smooth pursuit, and how well?
- Can the system autonomously recover from a drastic perturbation after training?
- What is the benefit (if any) of using two different scales?
- Does the learned controller generalize well to a real robot?

Performance is measured by the mean absolute velocity error (MAE) during training and testing. The size of the averaging window for the MAE is set to 1000. Since we use a finite set of actions where the largest action is smaller than the maximal relative speed that can be perceived, more than one step can be necessary for the model to reach the target velocity. Therefore, we measure the error just preceding an object velocity change, to ensure that the eye is given enough iterations to be able to reach the target velocity. This also means that we do not focus on the transitory effects of the control. This is of course a simplification with respect to real smooth pursuit tasks. In this work we chose deliberately to study a model with very simple dynamics (the time history is of two images) to show the feasibility of the approach in terms of learning and adaptation. Moreover, the discrete set of actions we consider in the

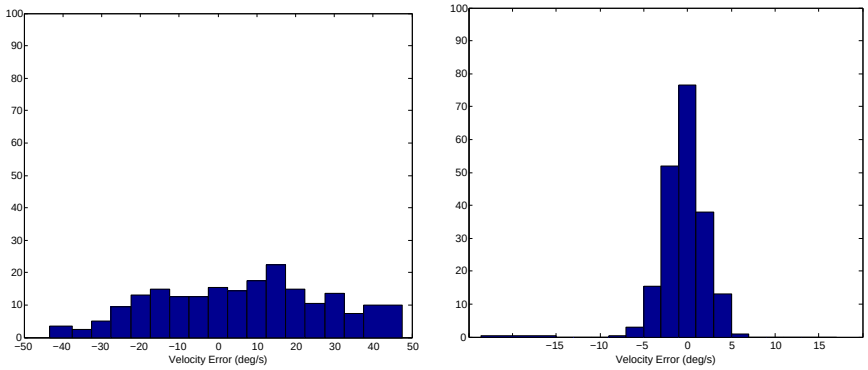
current model implies that the desired velocity might not be exactly reachable which limits achievable accuracy. One possible way to go from a discrete set of actions to a continuous action selection is to use a Gaussian policy instead of the softmax one [13]. Future work will focus on integrating more complex dynamics and improving accuracy.

3.2.1. Model testing

After training the model for 1 million iterations, we tested it in the simulation environment using a new subset of textures from the dataset to assess the smooth pursuit performance. During testing, the basis functions of the sparse coding models, as well as the weights of the neural networks are kept fixed, to evaluate the model at a particular time. We also stop exploration and use the learned policy in a greedy mode: at each iteration we select action a such that $z_a = \max_j\{z_j\}$, that is the best action according to the current policy. This is in contrast with the training phase where there is also exploration as required by the reinforcement learning module. The set of textures is a different subset from the TESTIMAGES dataset [31].

We use a similar procedure as during training, where the velocity and texture of the object are changed every 20 iterations, and the direction of the object velocity is reversed every 10 iterations. Figure 5 shows the histogram of the errors obtained over 200 such sequences on the initial model (Figure 5 (a)) and on the trained model (Figure 5 (b)). The mean absolute error is about $2.6°/s$ (resp. $2.2°/s$) in horizontal (resp. vertical) directions, with a standard deviation of $2.4°/s$ (resp. $1.7°/s$). The slightly smaller errors on the vertical axis are due to the smaller range of velocities and actions used in that direction. Otherwise the system is similar for both axes.

Figure 6 shows two examples extracted from a testing sequence with a constant object velocity period of 20 iterations. The anaglyph images (better seen in color) represent the images at time $k-1$ and k superimposed. When the object is given a new velocity, the relative velocity between the eye and the object leads to a visual shift between the two frames (first two rows). After a few



(a) (b)

Figure 5: Distribution of velocity errors when testing the model (a) at the beginning and (b) after training.

iterations (4 in the first case, 6 in the second one) the eye motion has adapted to the velocity of the object and the $k-1$ and k images are aligned (second row). The velocity is shown in the last row, where the red (resp. blue) dots represent the object (resp. eye) velocity. Only the pan velocity is represented here.

We also evaluated the performances of the model as a function of the object's velocity. Figure 7 shows the mean absolute velocity error in testing, for object velocities from 0 to 40°/s. We observe that the performances start degrading for velocity around 20°/s. With our 100ms time-loop, this velocity corresponds to a visual shift of 2° between two consecutive frames, which is bigger that the visual angle covered by fine scale patches (1.8°).

3.2.2. Benefits of using two scales

To assess what the benefits of using two different scales are, we compare the evolution of the MAE with training of two-scale and single scale models. The results are shown in Figure 8, where the blue and green lines represent the MAE testing performances as a function of the training time for a fine (resp. coarse) scale only model. The MAE of our two-scale model is shown in the red curve.

As can be seen from this figure, the use of the multi-scale approach allows

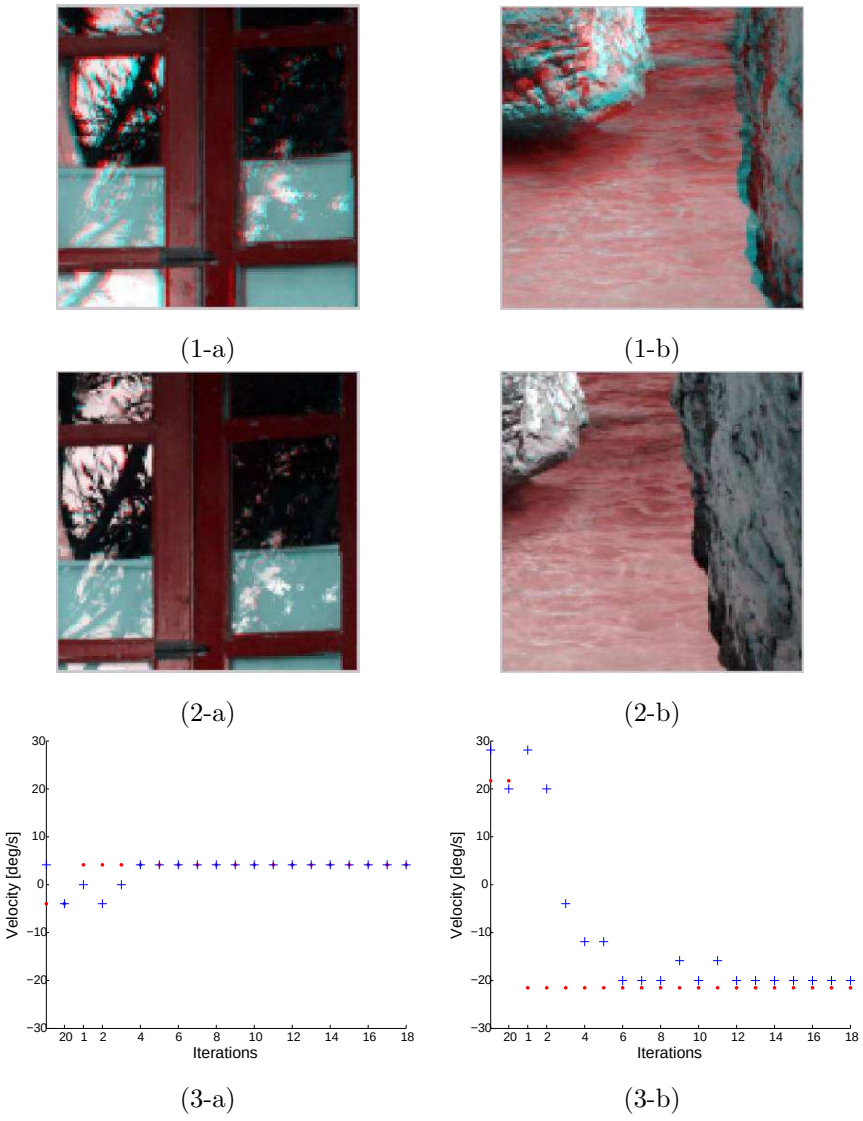


Figure 6: Examples of testing results for two different textures (a) and (b). Each column shows the superposition of the camera view for image k and $k-1$ as an anaglyph image (better seen in color) at the beginning (1) and at the end (2) of one period of constant object velocity. The corresponding velocity error is illustrated in last row, where the red dots (resp. blue crosses) represent the object (resp. eye) velocity. In this test the object velocity was kept constant for 20 iterations. Only the pan velocity is represented. Results for tilt are very similar.

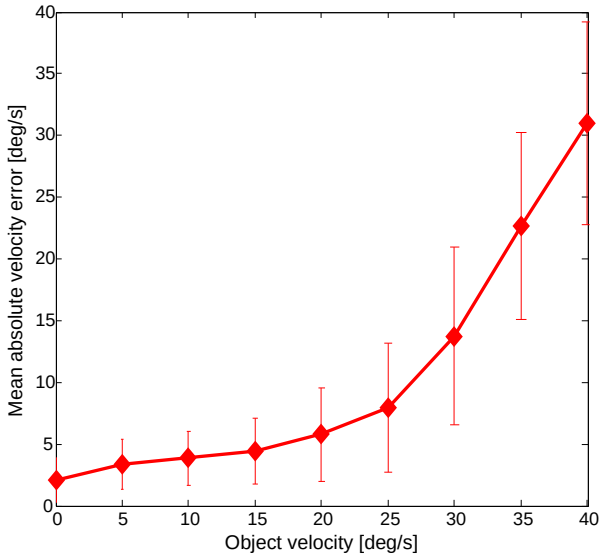


Figure 7: Testing performances as a function of object velocity. The error bars indicate the standard deviation over a set of 200 testing sequences of constant object velocity. Only the pan axis is represented because results for tilt are similar.

smaller final velocity error than either of the two single-scale models. Indeed, the range of velocities that we use leads to maximum angular shifts of 3° per iteration for vertical motion and 5° per iteration for horizontal motion, while coarse (resp. fine) patches cover 7.1° (resp. 1.8°) (see Section 3.1). When confronted with larger shifts than it can encode, the fine scale model is not able to learn a good policy. On the other hand, the coarse scale model is capable of detecting large motion but it lacks accuracy. The use of the two-scale model allows our system to perceive the whole range of motions with higher accuracy.

Figure 9 represents for each action the norm of the weight vector $\|\theta_a\|_2$ from the actor networks for the fine scale and coarse scale bases. The results show that the fine scale bases are more strongly linked to the smaller actions, while the coarse bases have a stronger connection with the largest actions. This result validates our intuition that the multi-scale model is able to exploit the different scales, and link them to the corresponding actions.

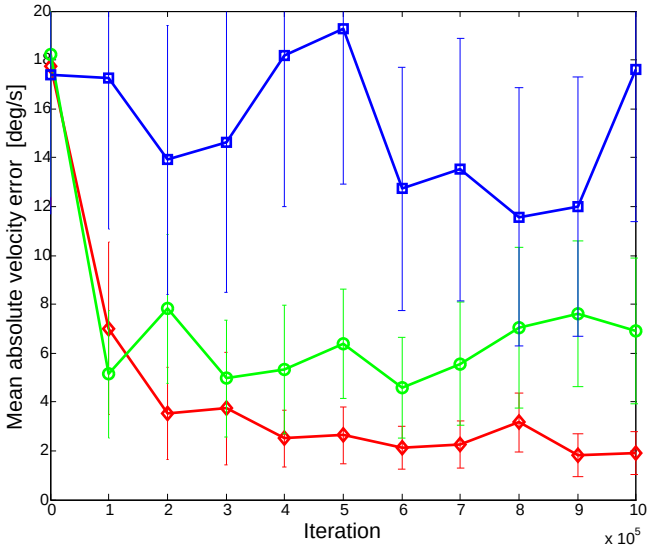


Figure 8: Testing results as a function of training iterations for three different models. The top (resp. middle) curve in blue (resp. green) represents the results for a model using fine (resp. coarse) scale only. The lower curve in red corresponds to the two-scale model. The error bars indicate half the standard deviation over a set of 200 testing sequences of constant object velocity. Only the pan axis is represented because results for tilt are very similar.

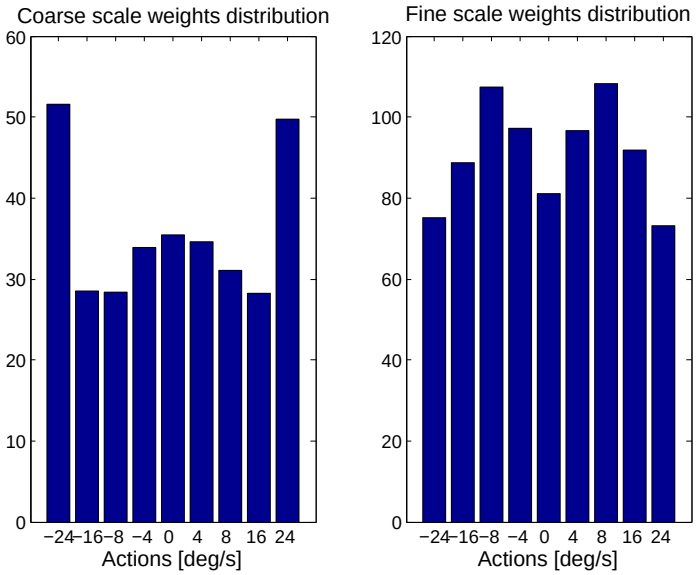


Figure 9: Weight norm $\|\theta_a\|_2$ of every action a for coarse (left) and fine (right) scales. Only the pan axis network is represented because results for tilt are very similar.

3.2.3. Robustness to perturbations

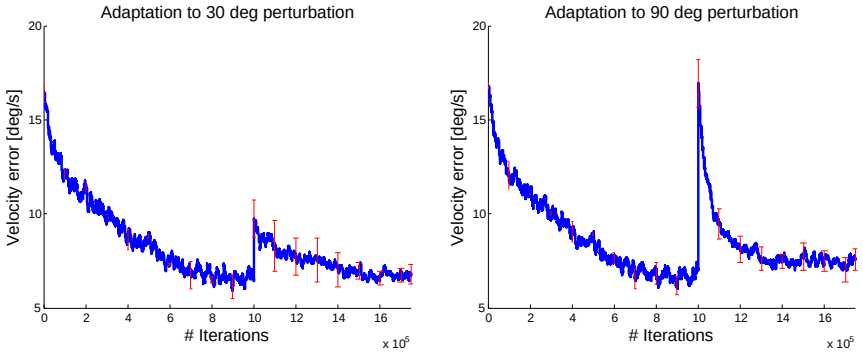
The experiments presented above demonstrate that our system is capable of autonomously learning a mapping between perception and action without requiring any calibration parameter. This property is particularly desirable for robotic applications.

In this section we address the question of whether the system can adapt to a change in the perception-action link occurring after the system has converged. Such a change could be the result of a physical impact to the system, introducing a perturbation of the geometric link between the camera and the actuators.

To assess the performance of our system in such conditions we artificially introduce a perturbation after 1 million iterations of training and keep training the system. During the full length of these experiments, both the bases and the RL weights are updated. The perturbation is simulated by rotating the input images. Figure 10 shows the MAE of the pan velocity for two different perturbations (30° and 90°). The velocity errors strongly increase right after

the perturbation is introduced, since the link between perception and action has been modified. Importantly, Figure 10 show that the system is able to largely recover from the perturbation.

Note that the error bars are computed with only four trials because of the long training time required (1 million iterations correspond to about 30h of training in our current implementation).



(a) (b)

Figure 10: Mean MAE of pan velocity during training, with a 30° (a) or 90° (b) rotation introduced after 1 million iterations. The mean and the error bars for one standard deviation are computed over 4 trials.

To illustrate the re-mapping that is learned we estimated which basis function orientations contribute more to the behaviour before and after the 90° perturbation. We used the dictionaries and RL models learned at 1 million (before the perturbation) and 1.5 million (after recovery) iterations. For each basis function of both coarse and fine scale, we used Gabor fitting to estimate the orientation it represents the most. We also computed the norm of the weight vector associated to this basis, *i.e.* the vector of the A weights that connect this basis to each action in $\mathcal{A}_{pan}$. Figure 11 represents the scatter plots of the weights norm with respect to the orientation. Each circle is a basis function of either coarse or fine scale. The density of basis functions with respect to the orientation is also illustrated with grayscale bars where the smallest density corresponds to the darkest areas. The norms have been scaled to $[0, 1]$.

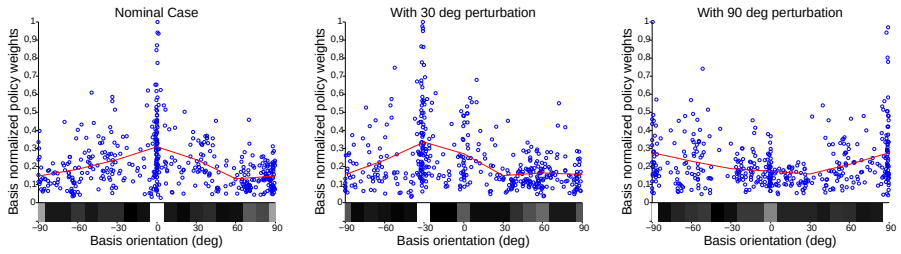


Figure 11: Distribution of the policy weights of the pan axis network, as a function of the main orientation selectivity (in degree) of each basis function. Each circle represents one basis from the dictionary. The red curve represents the mean value over bins of 30 degrees. (a) Nominal case, after training with no perturbation, (b) after training with a 30 deg perturbation, (c) after training with a 90 deg perturbation. The grayscale bars represent the density of basis functions where white (resp. black) indicates the maximal number of 107 bases (resp. 0) per bin of 10 degrees.

Importantly, this figure shows that both the basis functions and the policy weights have adapted to the perturbation. Indeed we can observe a shift of 30° (resp. 90°) of both the bases density and the weights repartition in (b) (resp. (c)) with respect to the nominal case (a). Before the perturbation, the pan movement is mostly driven by the basis selective to vertical orientation (angle = 0°). The orientation of the bases that drive most of the behaviour is shifted after the perturbation is introduced. This is the indication of the remapping between basis functions and actions in the RL policy network.

3.2.4. Robot experiment

In addition to the above evaluation in the simulation environment, we performed experiments on the real iCub robot. For these experiments, we used a model trained in simulation during 1 million iterations, with a greedy action selection. Without any ground truth on object velocity we empirically tested the model by moving objects in front of the robot with different velocities. Figure 12 and 13 show sample views from the camera. Figure 12 shows an example of testing with a rigid planar object. This setting is quite similar to the simulation setting although it can be seen that the images from the iCub camera

contain noise and distortions. Moreover, unlike the simulation setting the object is moved manually and its velocity is not constant. Although trained in simulation with basic dynamics, the system generalizes well to these situations. The motion of the robot's eyes can be seen from the background changes in Figure 12 while the object is smoothly tracked. For a better rendering of the behavior, a video of these experiments is available at `http://youtu.be/OyKKIWxo2aw`. Interestingly, the system also responds to the motion of a waving hand despite the presence of background in both coarse and fine scale windows (see Figure 13).

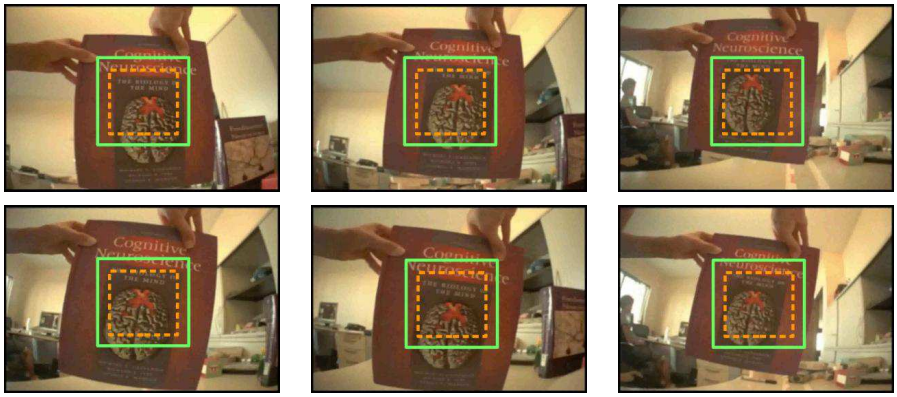


Figure 12: Samples from robot experiment with a flat object extracted every 3s from a short sequence. Green solid and orange dashed squares represent coarse and fine scale windows respectively.

4. Discussion

The work presented in this paper extends recent work on autonomous learning of active perception, and applies it to robot motion tracking. As mentioned in the introduction, there is a large literature in the computer vision and robotics communities on controlling a camera to track a moving object, using feature matching, tracking and visual servoing approaches. These techniques have proved to be efficient in many applications. However, they usually consider perception and action as separate blocks, for which some model is fixed

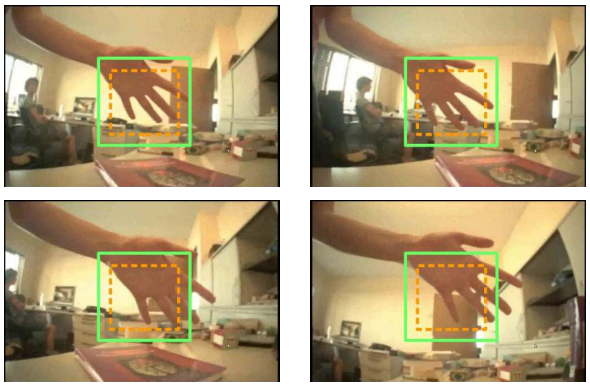


Figure 13: Samples from robot experiment with a moving hand extracted every 5s from a short sequence. Green solid and orange dashed squares represent coarse and fine scale windows respectively.

beforehand, and use them independently of the statistics of the environment. In this paper we consider a very different paradigm, where the active perception emerges with no explicit target as the result of an efficient encoding criterion. Our model has no prior knowledge of the geometric link between cameras and actuators, and the only expression of a task given to the system is the optimization of the reconstruction of the input data. Importantly, both sensory encoding and smooth pursuit behaviour develop in parallel, using this same criterion. We call this approach *active efficient coding*.

The experiments presented in this article show that the system autonomously learns to actively track object motion. Although the quantitative performance presented here does not reach the tracking accuracy of existing visual servoing approaches, we would like to point out four key characteristics of our approach. First, the system is fully self-calibrating, which is a desirable property for an autonomous system. Secondly, the visual encoding is tuned to best represent the input, and naturally captures the statistics of incoming visual information. This makes the system flexible in terms of the environment it can operate in. Third, the system can adapt to strong perturbations of the link between perception and action, as demonstrated in our experiments. Finally, the proposed approach makes use of very simple dynamics by considering a time history of two images

only and no modelling of delays. We believe that there is room for improvement here and future work will aim to integrate more complex dynamics. Also the finite set of actions we used can be replaced by continuous action selection by replacing the softmax policy by a Gaussian policy, as proposed in [13].

In its current form, our model requires textured objects. Note that some texture or at least object edges are necessary for both biological and artificial vision systems to perceive motion. When dealing with low textured objects, with a large plain area for instance, the activation of some central patches (or receptive fields in the biological case) will not vary with eye motion. For those patches our current system cannot determine a "best velocity" in terms of coding efficiency. Since the model equally considers every part of the image center, those patches would degrade the tracking performances. On the contrary, biological systems can make use of edges of uniform objects and successfully track them. In future work, a possible way to solve this problem can be to integrate some measurement of the texture as supplementary input to the reinforcement learning agent. Then, we believe that the system could learn to rely more on textured patches to estimate the velocity change that will provide better encoding, and thus correct smooth pursuit behaviour.

Finally, we believe that the active efficient coding has a larger scope than smooth pursuit behaviour learning. In future work, we will focus on the extension of this framework to account for other active perception behaviour. As a first step in this direction, we have recently proposed a model to learn vergence and smooth pursuit at the same time [26]. We have also argued how a reward signal for efficient sensory coding as used here may facilitate the emergence of imitation behaviours [33].

5. Acknowledgements

This work has partly received funding from the European Community's FP7/2007-2013, "Challenge 2-Cognitive Systems, Interaction, Robotics" under grant agreement No FP7-ICT-IP-231722, project IM-CLeVeR, from the Hong

Kong Research Grants Council under grant 619111 and from the BMBF Project "Bernstein Fokus: Neurotechnologie Frankfurt", FKZ 01GQ0840.

J. Triesch was supported by the Quandt foundation.

The authors want to thank Pramod Chandrashekhariah for his help during experiments.

References

[1] F. Attneave, Some informational aspects of visual perception, Psychological Review 61 (3) (1954) 183–193.

[2] H. B. Barlow, Possible principles underlying the transformation of sensory messages, Sensory communication (1961) 217–234.

[3] D. J. Field, What is the goal of sensory coding?, Neural computation 6 (4) (1994) 559–601.

[4] B. A. Olshausen, D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (6583) (1996) 607–609.

[5] J. Perez-Orive, O. Mazor, G. C. Turner, S. Cassenaer, R. I. Wilson, G. Laurent, Oscillations and sparsening of odor representations in the mushroom body, Science 297 (5580) (2002) 359–365.

[6] E. C. Smith, M. S. Lewicki, Efficient auditory coding, Nature 439 (7079) (2006) 978–982.

[7] J. Triesch, The role of a priori biases in unsupervised learning of visual representations: a robotics experiment, in: Workshop on Developmental Embodied Cognition (DECO), 2001.

[8] C. A. Rothkopf, T. H. Weisswange, J. Triesch, Learning independent causes in natural images explains the spacevariant oblique effect, in: Int. Conf. on Development and Learning (ICDL), IEEE, 2009, pp. 1–6.

[9] J. Ruesch, R. Ferreira, A. Bernardino, A computational approach on the co-development of artificial visual sensorimotor, Adaptive Behavior 21 (6) (2013) 452–464.

[10] Y. Zhao, C. A. Rothkopf, J. Triesch, B. E. Shi, A unified model of the joint development of disparity selectivity and vergence control, in: Int. Conf. on Development and Learning and Epigenetic Robotics (ICDL), IEEE, 2012, pp. 1–6.

[11] L. Lonini, Y. Zhao, C. P., B. Shi, J. Triesch, Autonomous learning of active multi-scale binocular vision, in: Int. Conf. on Development and Learning and Epigenetic Robotics, 2013.

[12] L. Lonini, S. Forestier, C. Teulière, Y. Zhao, B. Shi, J. Triesch, Robust active binocular vision through intrinsically motivated learning, Frontiers in Neurorobotics 7 (20) (2013) 1–10.

[13] C. Zhang, Y. Zhao, J. Triesch, B. Shi, Intrinsically motivated learning of visual motion perception and smooth pursuit, in: Int. Conf. on Robotics and Automation, 2014.

[14] C. Hofsten, K. Rosander, Development of smooth pursuit tracking in young infants., Vision Research 37 (13) (1997) 1799–1810.

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

[16] B. Horn, B. Schunck, Determining optical flow, Artificial Intelligence 17 (1-3) (1981) 185–203.

[17] J. L. Barron, D. J. Fleet, S. S. Beauchemin, Performance of optical flow techniques, Int. Journal of Computer Vision 12 (1994) 43–77.

[18] A. S. Jalal, V. Singh, The state-of-the-art in visual object tracking, Informatica: an International Journal of Computing and Informatics 36 (3) (2012) 227–248.

[19] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, P. Lima, Tracking objects with generic calibrated sensors: An algorithm based on color and 3d shape features, Robotics and Autonomous Systems 58 (6) (2010) 784 – 795.

[20] F. Chaumette, S. Hutchinson, Visual servo control, Part I: Basic approaches, IEEE Robotics and Automation Magazine 13 (4) (2006) 82–90.

[21] D. Coombs, C. Brown, Real-time smooth pursuit tracking for a moving binocular robot, in: IEEE Conference on Computer Vision and Pattern Recognition, 1992, pp. 23–28.

[22] T. Shibata, S. Schaal, Biomimetic smooth pursuit based on fast learning of the target dynamics, in: in Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, 2001.

[23] K. Hosoda, M. Asada, Versatile visual servoing without knowledge of true jacobian, in: IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems (IROS), Vol. 1, 1994, pp. 186–193 vol.1.

[24] M. Lopes, J. Santos-Victor, A developmental roadmap for learning by imitation in robots, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (2) (2007) 308–321.

[25] L. Jamone, L. Natale, F. Nori, G. Metta, G. Sandini, Autonomous online learning of reaching behavior in a humanoid robot, International Journal of Humanoid Robotics 9 (3) (2012) 1250017.1–1250017.26.

[26] T. N. Vikram, C. Teulière, C. Zhang, B. E. Shi, J. Triesch, Autonomous learning of smooth pursuit and vergence behaviour through efficient coding, in: Int. Conf. on Development and Learning and Epigenetic Robotics, 2014.

[27] S. G. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Transactions on Signal Processing 41 (12) (1993) 3397–3415.

[28] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, M. Lee, Natural actor-critic algorithms, Automatica 45 (11) (2009) 2471–2482.

[29] R. Beira, M. Lopes, M. Praga, J. Santos-Victor, A. Bernardino, G. Metta, F. Becchi, R. Saltaren, Design of the robot-cub (icub) head, in: IEEE Int. Conf. on Robotics and Automation, 2006, pp. 94–100.

[30] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator, in: Proceedings of IEEE Workshop on Performance Metrics for Intelligent Systems Workshop, ACM, 2008.

[31] Testimages.
URL `http://www.tecnick.com/public/code/cp_dpage.php?aiocp_dp=testimages`

[32] G. Metta, P. Fitzpatrick, L. Natale, Yarp: Yet another robot platform, International Journal on Advanced Robotics Systems (2006) 43–48.

[33] J. Triesch, Imitation learning based on an intrinsic motivation mechanism for efficient coding, Frontiers in Psychology 4 (800) (2013) 1–8.