Predictive Action Selector for Generating Meaningful Robot Behaviour from Minimum Amount of Samples

Erhard Wieser and Gordon Cheng

***Abstract*— Our aim is to better understand the action selection process of intelligent systems by looking at their ability of internal prediction. In robotic systems, one problem is to generate meaningful robot behaviour with a very small and simple set of trained motions. An additional problem is to compensate for incomplete sensory data while generating behaviour. We propose a new predictive action selector to contribute to the solution of these problems. Our action selector predicts task–relevant feature and motion sequences, and uses the prediction results to select the robot action. We validate our implemented model on a humanoid robot. The robot generates meaningful behaviour composed out of very simple and few trained motions, and at the same time it compensates for incomplete sensory data such as temporary loss of task–relevant visual features.**

Keywords: action selection, internal prediction, emergent behaviour

I. INTRODUCTION

A. Generation of Meaningful Robot Behaviour from Minimum Amount of Samples

Inspired by the human ability of internal prediction, our aim is to contribute to the understanding of the action selection process. For this purpose, the ability of internal prediction can play a key role [1], [2], [3]. At a higher level of abstraction for example, humans can predict the consequences of their actions. This ability enables them to choose the correct action in a given situation. At a lower level, prediction helps to compensate for incomplete sensory data or for delays in the sensory–motor system [2].

We propose a new predictive action selector which can generate meaningful behaviour from minimum amount of training samples. Our proposed action selector (see fig. 1) integrates the ability to predict the effect of the robot's own action and the ability to compensate for incomplete sensory data. We present the design, realization, and evaluation of our predictive action selector which is a key module of our future cognitive architecture. Our proposed action selector receives task–relevant visual features, goal states, and motor states as input, and delivers new motor commands as output. Our action selector consists of neural network–based multi–purpose modules which are integrated and linked together in a way that the overall system is *not* specific to a single task or behaviour, an important fact in developmental robotics [4]. We fully implemented our proposed model and validated it on a NAO humanoid robot. Our system controlled the head of the robot and we tested the emergence of basic behaviour such as object tracking and object evasion.

Erhard Wieser and Gordon Cheng are with the Institute for Cognitive Systems, Technische Universität München, München, Germany, http://www.ics.ei.tum.de. E-mail: erhard.wieser@tum.de, gordon@tum.de.
Video to this paper: http://web.ics.ei.tum.de/~erhard/videos/PAS.mp4

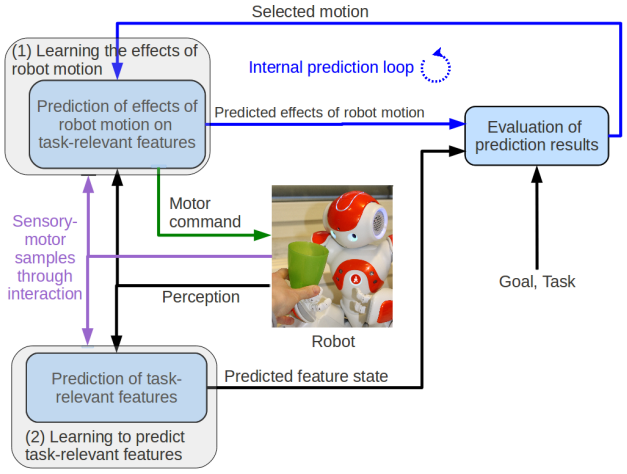


Fig. 1. Our proposed predictive action selector: The system forms a memory by learning from sensory–motor samples through interaction, see grey blocks (1), (2). The formed memory enables the system to predict features. In the execution stage (blue blocks), the predicted effects of robot motion are combined with the predicted feature state and evaluated according to a goal and a selected task. Our system evaluates possible actions along with their corresponding effects within an internal prediction loop before sending the appropriate motor commands to the robot. At the same time, prediction of task–relevant features allows to compensate for temporary loss of features.

Our contribution can be best described by the following capabilities of our proposed system:

- Behaviour generation: Our system generates meaningful robot behaviour with emergent robot motions which are composed out of very simple and few trained sequences (minimum number of samples). The behaviour is emergent, it is *not* engineered explicitly into the system. The system supports the emergence of different types of behaviour.
- Task or behaviour switching: Our system can switch immediately between different types of robot behaviour.
- Prediction of task–relevant features: Our system can deal with temporal losses of the perceived task–relevant features. Normally, traditional control systems will either stop moving the robot, or perform a search of sensory space (*e.g.* visual space), if the required sensory input is not present any more. Nevertheless, our system keeps on generating motor commands even in cases of temporal loss of sensory features. This is realized by using internal prediction of the sensory features.

B. Related Work

Infants learn about their environment by interacting with objects [5]. Inspired by this insight, Noda *et al.* [5] propose a model which enables an agent to estimate appearance and motion models based on visuomotor experience. Sensory–motor experience can lead to robot behaviour which is rather emergent than specifically engineered. The benefits of sensory–motor experience and the benefits of self–organization of internal representations are described in [6] and [7].

The iCub cognitive architecture tries to generate infant behaviour based on insights of cognitive science and developmental psychology [8], [9]. The action selection process of the state–of–art iCub cognitive architecture is based on homeostatic self–regulation governed by the affective state [2]. On the implementation level, the action selection of the iCub cognitive architecture is a function of the levels of curiosity and experimentation, both levels provided by the affective state [9]. The affective state module is similar to a goal module. It provides the motives which influence the action selection. The motives are curiosity, experimentation, and social engagement, and they are implemented as temporal series of event–related spikes [9]. The iCub architecture also has an anticipatory circuit for perception–action simulation, using motor–sensor and sensor–motor hetero–associative memories [8]. However, experimental results showing the anticipation or prediction ability of the architecture are missing in [8], [2], [9].

In the domain of object interaction, the tracking and avoidance of an object of interest form the most basic behaviour, *e.g.* visual servoing is one of the earliest human skill [9]. Gaussier *et al.* [10] realize an object tracking skill on a mobile robot. Their concept bases on the imitation of motion sequences. They propose a Perception-Action (PerAc) architecture [10] which learns sensory–motor associations with a delayed reward. However, they have to apply an active filtering mechanism to compensate for drawbacks in feature detection, since their model is sensitive to noise in sensory data [10]. They tested their model on a mobile robot, it is not clear how it performs on a humanoid robot. For a humanoid robot, Berthouze and Kuniyoshi [11] propose a system supporting gaze fixation and saccadic motion. These skills develop from a context–free control to a context–dependent control [11]. Berthouze and Kuniyoshi use two CCD cameras, position and velocity control, and a self–organizing map (SOM) for the association of the observed motion and the corresponding motor command. The SOM realizes the emergent categorization, *i.e.* it can respond to new sensory–motor patterns based on previous experiences. However, it is not clear how well a SOM–based concept scales up with increasing task or skill complexity. Shibata *et al.* [12] propose a model of smooth pursuit in primates based on learning the target dynamics. They use a recurrent neural network (RNN) in order to emulate the medial superior temporal area of the brain. They realize a forward model of target motion by using online learning. Their model predicts the eye velocity and uses the retinal slip as error signal for learning. However, it is not clear whether their model can alter the robot's behaviour towards related skills, *e.g.* target evasion, instead of target tracking.

Our review on related work shows that the systems presented in [10], [11], [12] can perform well, but they seem to be specifically engineered towards a single task (*e.g.* object tracking only). Compared to the described related work, our approach provides a systematic way to bootstrap different types of meaningful robot behaviour by re–using a minimum amount of taught samples.

C. Our Approach

In contrast to traditional robotics, we do not require to determine an analytical relation between parameters of robot limb motion and observed motion of features, *e.g.* a Jacobian matrix. Instead, our approach relies strongly on sequence prediction involved in both perception and action generation. A sequence is defined as a spatio–temporal pattern of sensory–motor data. Our proposed action selection system consists of two main modules or building blocks (see fig. 1 and fig. 2): A self–motion predictor and a feature predictor. The self–motion predictor predicts the effect of a selected robot motion (self–motion or ego–motion) on a perceived task–relevant feature of interest. The feature predictor predicts the future state of a task–relevant feature. The results of these two main modules are evaluated according to a goal pattern. An internal prediction loop determines the best possible robot motion before the corresponding motor command is computed. The memory of the self–motion predictor contains only a small amount of simple motor sequences obtained through either kinesthetic teaching or autonomous exploration of degrees of freedom (*DOFs*). More complex motor sequences emerge during robot–world interaction by continuously selecting/switching between these simple motor sequences depending on the environmental situation and the goal state. Our approach also includes a mechanism to alter the robot behaviour to switch immediately between different tasks (*e.g.* tracking an object of interest, or evading it).

We do not claim that our proposed model is biologically–plausible. In fact, our proposed model is plausible from a cognitive science point of view, since our model encompasses learning, prediction, and action, which are important characteristics of a cognitive system [1], [2].

II. SYSTEM DESCRIPTION

A. System Overview, Input and Output Data

The functional diagram (fig. 2) shows our action selection system which works with the following input data: The vector $\mathbf{p}_{in}$ contains the normalized joint positions of the robotic limb to control. The normalization of joint positions to values between 0 and 1 is necessary because these are directly processed by the neural network of the self–motion predictor. The vector $\mathbf{v}_g$ is a two–dimensional vector in the robot's field of view (normalized image plane of the robot camera) and represents the goal state of the visual feature of interest, *i.e.* here the goal state is the desired position

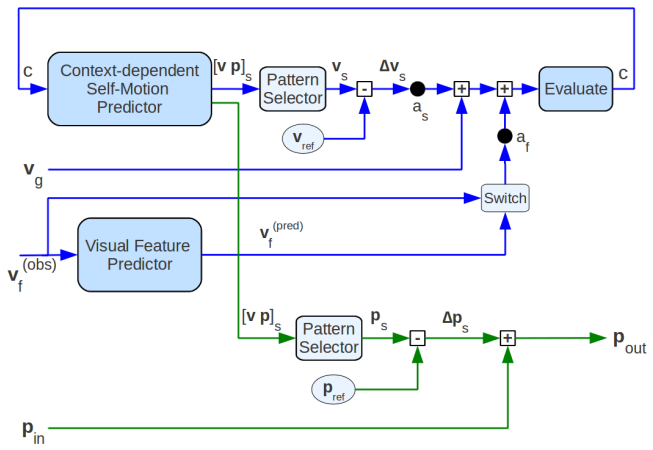


Fig. 2. Functional diagram of our predictive action selector. System inputs are the current proprioceptive state (sensed joint positions $\mathbf{p}_{in}$), the observed visual feature of interest ($\mathbf{v}_f$), and the goal position of the visual feature of interest ($\mathbf{v}_g$). System output is the new proprioceptive state sent to the robot joints ($\mathbf{p}_{out}$). The blue arrows represent the data flow of the first computation step. In the first computation step, the system finds the *optimal* context representing the optimal basic action. In the second computation step (green arrows), the system uses the optimal context in order to calculate the new joint positions $\mathbf{p}_{out}$.

of a blob center in the image plane. In this paper, the goal $\mathbf{v}_g$ is kept constant. The vector $\mathbf{v}_f$ is a two–dimensional vector in the robot's field of view. It represents the current task–relevant visual feature of interest in the image frame. Applied to a robot's interaction with a coloured object, *e.g.* a green cup, the vector $\mathbf{v}_f$ represents the position of the object in normalized image coordinates. We implemented visual feature cells for the extraction of $\mathbf{v}_f$ encoding the position of the largest colour blob. Note that $\mathbf{v}_f$ can represent another task–relevant feature as well, and $\mathbf{v}_f$ can also be delivered by an abstraction layer [13], for example an attention selection module proposed in [14]. Note that both $\mathbf{v}_g$ and $\mathbf{v}_f$ can be three–dimensional as well, if additional depth information is provided as third position coordinate. The vector $\mathbf{p}_{out}$ contains the new normalized joint positions to be de–normalized and sent to the robot limb at each computation cycle of the system.

B. Modules

1) Context–dependent Self–Motion Predictor: Basic actions are defined as simple short motion sequences of the robot's body parts, encoded by a sequence of joint position vectors $\mathbf{p}$. For example, basic actions of the head are left, right, up, down motion of the head, as well as no motion at all (termed as idle motion). Complex actions emerge by the combination of these basic actions over time, by continuously switching between them depending on the situation. For each computation cycle of our overall system (computation of $\mathbf{p}_{out}$ from the given input data), the self–motion predictor first predicts the changes of the visual feature of interest ($\Delta\mathbf{v}_s$ in fig. 2) for all possible basic actions (*i.e.* only the changes caused by self–motion). In fig. 2, the internal prediction circuitry, which is created by a closed loop between the evaluation step and the self–motion predictor module, determines the best basic action. A basic action is internally represented by the context index c. The self–motion predictor uses the selected context (representing the selected basic action) to predict the new final visual–proprioceptive state as a result of the selected robot motion (see the green arrow starting from the self–motion predictor module in fig. 2). This state is then used by other components to compute $\mathbf{p}_{out}$. The learning and generation of these sensory–motor sequences is realized by the continuous time recurrent neural network (*CTRNN*) proposed in [6]. We implemented the network structure proposed in [6]. In our implementation, each individual neuron uses the sigmoid function as activation function. Each sample $\mathbf{s}_i$ of a sequence S has the following format, eq. (1):

$$\mathbf{s}_i^T = \begin{bmatrix}\mathbf{v}_f^T & \mathbf{p}_{in}^T\end{bmatrix}_i \tag{1}$$

We do not use any pre– or postprocessor attached to the CTRNN, since the dimension of the sample vector $\mathbf{s}_i$ is low and the trained sequences are short and do not overlap. In our system, each robot DOF is normalized to values between 0 and 1, and mapped directly to one of the input/output neurons of the CTRNN. The CTRNN learns sequences by using the standard backpropagation through time (*BPTT*) [15], [16]. In addition to the weights, the initial potentials of the context neurons also self–organize during the learning phase by using the BPTT method. At the end of the training, each of the stored sequences is represented by the initial potential of the context neurons, denoted by the vector $\mathbf{c}^{(init)}$.

2) Visual Feature Predictor: Our system normally uses the observed position delivered by the visual feature cells. However, the visual feature of interest is not present all the time. For example, a typical interaction scenario would be a moving object of interest which is occluded temporally by an other object. This leads to a temporal loss of the observed $\mathbf{v}_f$. In such a case, the visual feature predictor delivers a prediction of $\mathbf{v}_f$ based on the last observed samples of positions. Thus, each computation cycle of the overall system uses either the *observed* $\mathbf{v}_f$, or the *predicted* $\mathbf{v}_f$. The system first relies on the observed $\mathbf{v}_f$. If the observed $\mathbf{v}_f$ gets lost, the system will use the predicted $\mathbf{v}_f$ until the feature will be observable again. The algorithm of the visual feature predictor module uses a CTRNN [6] to predict next samples of $\mathbf{v}_f$. The vector $\mathbf{v}_f$ is directly represented by two input/output neurons, pre– or postprocessing is not required. The network is trained with sequences encoding the observed trajectory of $\mathbf{v}_f$. During runtime, the visual feature predictor recognizes an observed trajectory of $\mathbf{v}_f$, predicts that trajectory, and delivers the predicted values as output.

3) Pattern Selector: This module delivers a sub–pattern of the given input pattern, representing either the visual pattern, or the proprioceptive pattern. The formal description of the pattern selector is $patternSelector(\mathbf{s}, m)$ with the input pattern $\mathbf{s}^T = \begin{bmatrix}\mathbf{v}^T & \mathbf{p}^T\end{bmatrix}$ and the modality selector m. If $m = v$, the pattern selector returns the visual pattern $\mathbf{v}$. If $m = p$, it returns the proprioceptive pattern $\mathbf{p}$.

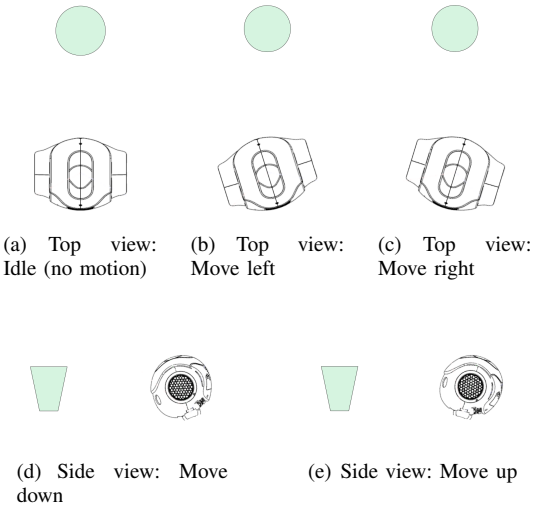
C. Stages of Processing

In the training stage, the memory of the self–motion predictor and the memory of the visual feature predictor are formed from a small amount of samples (data flow represented by the purple arrows in fig. 1). In the execution stage, the system relies on these memories to generate the robot behaviour (data flow represented by the black, blue, and green arrows in fig. 1).

1) Training of the Self–Motion Predictor: We trained the context–dependent self–motion predictor with a minimum amount of sequences encoding the previously described basic actions. The minimum amount of sequences depends on the DOFs of the robot limb our system should control. For each DOF, two sequences exist, one per direction of DOF. We tested our system on the head of the NAO robot which has two DOFs (yaw and pitch), therefore we trained four sequences and one additional sequence when the head does not move (idle sequence). During the training of the self–motion predictor, only that part or limb of the robot is allowed to move which our system controls, in this case the head. We put an object of interest (green cup) in front of the robot's head. During training, the object of interest is in a fixed position (not moving), because the self–motion predictor needs a spatio–temporal representation of the effects of self–motion only. Here, this effect is represented by gradual shifts of the object position in the robot's field of view during head motion. Note that two possible methods exist for the training of the self–motion predictor. One training method is to acquire the training data by moving the robot limb manually (similar to kinesthetic teaching). The other training method is to let the robot move its limb by itself, performing an exploration of DOFs. We first recorded the so–called idle sequence where no robot motion exists. Then we recorded the remaining four sequences by moving the robot head manually. For each of these remaining sequences, only one DOF of the head moves at a time, while the object is in fixed position. At the beginning of the motion, the object position is in the middle of the robot's field of view. While the head is moving, the object position in the robot's field of view shifts gradually to either right, left, up, or down, depending on the selected head DOF and its direction of motion. The training of the five basic sequences is depicted in fig. 3. Each of these sequences for the self–motion predictor should contain at least four samples (determined empirically).

The CTRNN of the self–motion predictor had four input/output neurons (two neurons encoding $\mathbf{v}_f$ and two neurons encoding $\mathbf{p}$) and ten context neurons. Although a small number of neurons already suffices to learn these short sequences, the CTRNN [6] is necessary when our model will be scaled up in near future. Thus, the CTRNN [6] can be regarded as a component for a more scalable architecture.

2) Training of the Visual Feature Predictor: We trained the CTRNN of the visual feature predictor with 46 sequences of $\mathbf{v}_f$ (a rough estimate determined empirically), each sequence contained about 30 to 60 samples. These training sequences consisted of various horizontal, vertical, and idle motions of the object of interest (here, a cup) in the field of view of the robot. The object of interest was always visible for the robot during the training, and the head was in resting state. The CTRNN of the visual feature predictor had two input/output neurons and ten context neurons. Note that training of the visual feature predictor with less samples (*e.g.* less than ten sequences with each a couple of samples) makes it less capable to predict visual features well enough, but this will not affect the robot behaviour if the features are observable.



(a) Top view: Idle (no motion) (b) Top view: Move left (c) Top view: Move right

(d) Side view: Move down (e) Side view: Move up

Fig. 3. The five basic motion sequences used for training the self–motion predictor. During training phase, the object of interest (cup, light green) is always in fixed position. The head of the NAO robot is depicted, facing the object of interest. For each sequence, the sampling of the sensory–motor data starts in the home position (fig. 3(a)) and stops in the end position (fig. 3(a)–3(e)). The first sequence is the so–called idle sequence where the robot head is in home position (HP) and not moving (fig. 3(a)). The second sequence is the turn left motion: From the HP, the robot slowly moves its head to left until it reaches the end position shown in fig. 3(b). The third sequence is the turn right motion: From the HP, the robot slowly moves its head to right (end position in fig. 3(c)). The fourth sequence is the move down motion: From the HP, the robot slowly moves its head down (end position in fig. 3(d)). The fifth sequence is the move up motion: From the HP, the robot slowly moves its head up (end position in fig. 3(e)). The taught sequences 3(a), 3(b), 3(c), 3(d), 3(e) are internally represented by the self–organized initial context states $\mathbf{c}_0^{(init)}$, $\mathbf{c}_1^{(init)}$, $\mathbf{c}_2^{(init)}$, $\mathbf{c}_3^{(init)}$, $\mathbf{c}_4^{(init)}$, respectively.

3) Execution: In each computation cycle of the execution stage, our action selection system works with two main steps, also visualized by the two different colours for the data flow in fig. 2. In the first step, the action selection system determines the *optimal* context index c_{opt}. The optimal context index is an internal representation of a simple basic action leading to a desired perceptual state. The system determines the optimal context index by evaluating a value function $V(c)$ for each possible context index c. The goal is to find the optimal context c_{opt} leading to the smallest value of V, see eq. (2).

$$\min V(c) = \min \|\mathbf{v}_g + a_s \cdot \Delta\mathbf{v}_s(c, M_1) + a_f \cdot \mathbf{v}_f\| \quad (2)$$

The visual feature of interest $\mathbf{v}_f$ is either the observed, or the predicted one. If $a_s = -1$ and $a_f = -1$ (be-

haviour alteration parameters), the value function can also be interpreted as an error function describing the *predicted* deviation of the current visual state (feature of interest $\mathbf{v}_f$) from the visual goal state ($\mathbf{v}_g$). The pattern $\Delta\mathbf{v}_s(c, M1)$ describes the predicted change of state of the feature of interest $\mathbf{v}_f$ due to the selected self–motion. The self–motion is characterized by the selected action c and the prediction length M_1. The prediction length M_1 determines how far ahead in time the self–motion predictor should predict a sequence. Besides the behaviour alteration parameters a_s and a_f, the predicted deviation mainly depends on the selected action c and on the length M_1 of the prediction of self–motion. In fig. 2, the box *Evaluate* symbolizes the calculation of the value function $V(c)$ for each recalled sequence c. Once the action selection system computes all $V(c)$ values, it selects the smallest value $V_{min} = V(c_{opt})$ and determines the corresponding optimal context c_{opt}. In the second step, the system calculates a new joint position vector $\mathbf{p}_{out}$ by using the optimal context index determined in the first step. It calculates the proprioceptive difference $\Delta\mathbf{p}$ (depending on the prediction length M_2) between the new predicted proprioceptive sample $\mathbf{p}$ and the reference position $\mathbf{p}_{ref}$ (the patterns $\mathbf{v}_{ref}$ and $\mathbf{p}_{ref}$ were initially obtained from the first sample $\mathbf{s}_0$ of the trained idle sequence). This difference is added to the latest proprioceptive sample $\mathbf{p}_{in}$. The new joint positions are sent immediately to the robot limb which our system controls. Algorithm 1 describes the computation cycle of our system during the execution stage.

Algorithm 1 The Computation Cycle of our Predictive Action Selector (fig. 2).

Input: Current joint positions $\mathbf{p}_{in}$, either the observed or the predicted visual feature of interest $\mathbf{v}_f$, visual goal state $\mathbf{v}_g$

Output: New joint positions $\mathbf{p}_{out}$

Parameters: Behaviour alterations a_s and a_f, prediction length M_1, prediction length M_2

Get the number N_c of context indexes from the self–motion predictor;
Initialize empty array V with size N_c;
for $(c = 0; \quad c < N_c; \quad c++)$ **do**
  $\mathbf{s} \leftarrow selfMotionPredictor(c)$; // Using prediction length M_1 //
  $\mathbf{v}_s \leftarrow patternSelector(\mathbf{s}, v)$;
  $\Delta\mathbf{v} \leftarrow \mathbf{v}_s - \mathbf{v}_{ref}$;
  $V(c) \leftarrow \|\mathbf{v}_g + a_s \cdot \Delta\mathbf{v}_s + a_f \cdot \mathbf{v}_f\|$;
end for
Find the smallest value $V(c_{opt})$ and its corresponding index c_{opt};
$\mathbf{s} \leftarrow selfMotionPredictor(c_{opt})$; // Using prediction length M_2 //
$\mathbf{p} \leftarrow patternSelector(\mathbf{s}, p)$;
$\Delta\mathbf{p} \leftarrow \mathbf{p} - \mathbf{p}_{ref}$;
$\mathbf{p}_{out} \leftarrow \mathbf{p}_{in} + \Delta\mathbf{p}$;
return $\mathbf{p}_{out}$;

The behaviour alteration parameters allow an integrative system design [17], because they ensure that each subsystem delivering feature vectors can be integrated into the action selection process by setting the strength and direction of a particular input. This is important when our system will be scaled up in near future, in order to realize arm control along with head control.

III. EXPERIMENTS

A. Setup

Robot experiments served as an assessment of our predictor modules. In the training stage, the memory of the self–motion predictor and the memory of the visual feature predictor were formed (see sections II-C.1 and II-C.2). In all experiments, the joint positions were sampled with a rate of 7 Hz. The goal pattern $\mathbf{v}_g$ was constant and set to $\mathbf{v}_g = \begin{bmatrix}0.5 & 0.5\end{bmatrix}^T$ (corresponding to the middle of the robot's field of view). The extracted object position (visual feature of interest) is represented by $\mathbf{v}_f = \begin{bmatrix}x & y\end{bmatrix}^T$ in the normalized image plane of the camera. The measured visual error is given by $e_v = \|\mathbf{v}_g - \mathbf{v}_f^{(obs)}\|$, where $\mathbf{v}_f^{(obs)}$ denotes the observed object position in the normalized image plane of the camera. The prediction parameters were $M_1 = 4$ (controlling $\Delta\mathbf{v}_s$) and $M_2 = 3$ (controlling $\Delta\mathbf{p}$), their values were determined empirically.

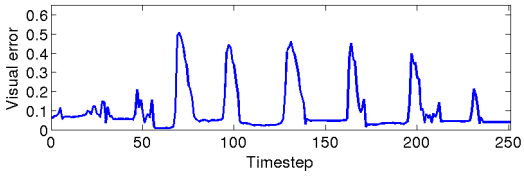
B. Results

During the execution stage, we observed the emergence of two different sorts of meaningful robot behaviour: object tracking and object evasion. The switching between these different types of behaviour is accomplished by changing the sign of the alteration parameter a_s.

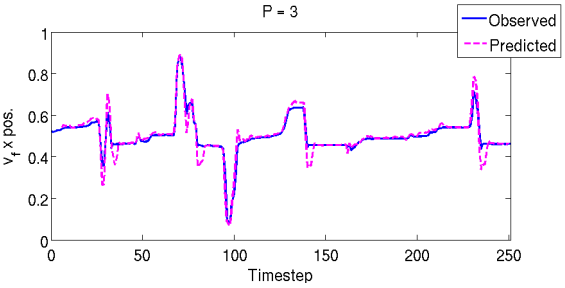
1) Tracking an Object of Interest: When the behaviour alteration parameters are set to $a_s = -1$ and $a_f = -1$, a tracking behaviour of the visual feature of interest emerges. We tested this behaviour by moving an object in front of the robot. The object was moved fast from one location to the other. The robot tracked the object and performed a saccade to it. Tracking results are shown in fig. 4.

2) Composing Complex Robot Motion from Simple Motions: The generated robot motion of the previous tracking scenario (fig. 4) is regarded more closely to show how complex robot motion sequences emerge by switching continuously between the taught simple sequences. In each computation cycle, our system computes a new joint position. The contexts $\mathbf{c}_0^{(init)}$, $\mathbf{c}_1^{(init)}$, $\mathbf{c}_2^{(init)}$, $\mathbf{c}_3^{(init)}$, $\mathbf{c}_4^{(init)}$ were self–organized after learning the simple action sequences and represent these taught actions *idle*, *left*, *right*, *down*, *up*, respectively, along with their corresponding effect on the perceived visual feature $\mathbf{v}_f$ (section II-C.1). These results are shown in fig. 5.

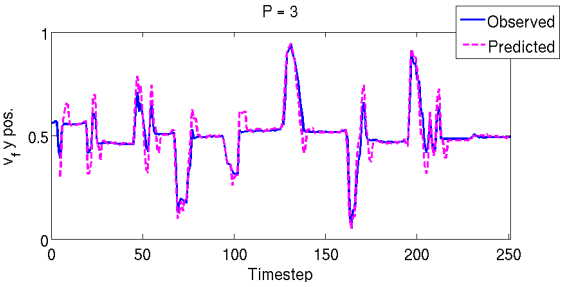
3) Prediction of Visual Features: The timesteps from 50 to 150 of the previous interaction (fig. 4) are regarded more closely and the effect of the prediction parameter P on the prediction of the visual feature is analysed, see fig. 6.



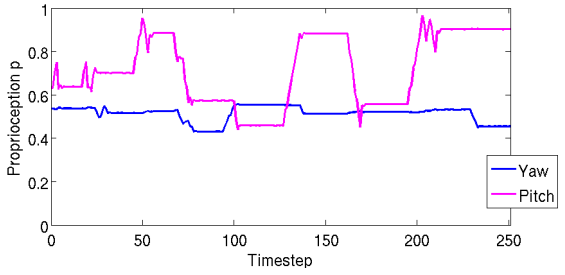
(a) Visual error



(b) Visual feature: x–position



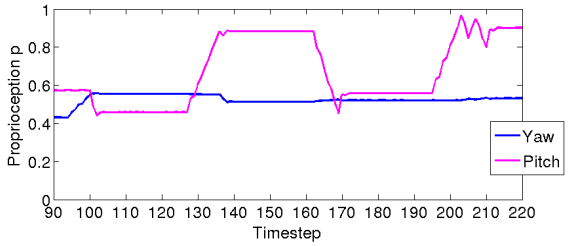
(c) Visual feature: y–position



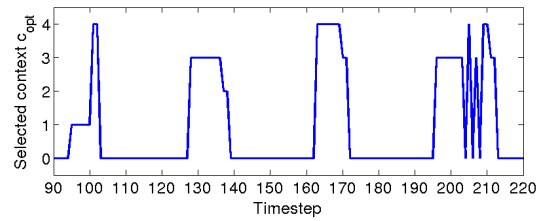
(d) Proprioception $\mathbf{p}_{out}$

Fig. 4. Tracking an object of interest. Each large peak in the visual error (fig. 4(a)) indicates a big and fast movement of the object in the robot's field of view. The robot reacts with a saccade in order to compensate for the visual error. The generated robot motion (fig. 4(d)) is composed out of simple and few motion primitives (represented by context indexes, fig. 5(b)). After a saccade, the remaining visual error is smaller than 4 - 5 %.

4) Dealing with Temporal Losses of Visual Features: In the previous interaction scenario, the visual feature was always observable. However, there are situations with temporal loss of the visual feature of interest due to occlusion or fast motions. In those situations, traditional tracking systems either stop moving entirely, or perform a search of the entire visual field of view and it takes time for them until they have re–found the object. In contrast to those traditional systems, our system works with the predicted object position $\mathbf{v}_f^{(pred)}$ in cases of object loss. Relying on the predicted feature helps the robot to move more smoothly, since it does not need to re–search the visual field. Figure 7 shows an example of the temporal loss of features.



(a) Proprioception $\mathbf{p}_{out}$



(b) Selected action per timestep, represented by the selected optimal context index i per timestep: The context indexes are numbered from 0 to 4, representing idle motion, left, right, down, and up motion, respectively.

Fig. 5. Action selection for composing complex robot motion: The motion sequence of the robot head (fig. 4(d)) seems complex. It was never taught to the robot before. Our system generates that sequences by continuously selecting and switching between the taught simple sequences (basic actions), represented by the context indexes 0 to 4. For example, consider the timesteps 128 to 169. One can observe the selected action $c_{opt} = 3$ (*down*) between timesteps 128 and 136. It leads to an increase of the pitch angle (in case of the NAO robot, the head tilts downwards when the angle of the head pitch DOF is increased). A slight correction of the yaw angle at timesteps 137 and 138 is achieved by selecting the action $c_{opt} = 2$ (*right*). The constant yaw and pitch angle from timestep 139 to 162 result from selecting the action $c_{opt} = 0$ (*idle*), *i.e.* $\Delta\mathbf{p} = \mathbf{0}$. The decrease in the pitch angle from timestep 163 to 169 results from selecting the action $c_{opt} = 4$ (*up*). See section II-C.1 for further description of the trained basic actions.

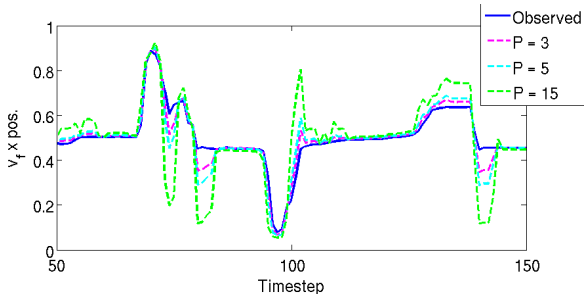


Fig. 6. Analysis of the feature prediction (the x–position is taken as example, same principle applies to the y–position, or any other predicted feature): The timesteps 50 to 150 of the previous interaction (fig. 4) are regarded more closely. In addition to the observed $\mathbf{v}_f$ (blue), a set of predicted $\mathbf{v}_f$ is plotted. The prediction is dependent on the parameter P of the visual feature predictor. The bigger the value for P, the more far ahead in time the prediction is. Thus, for small values of P (*e.g.* 1, 2, or 3), the predicted sequence is close to the observed one. For large values of P (*e.g.* 15), the predicted sequence shows deviations from the observed one, especially for rapid changes of the observed feature position.

5) Switching between Different Types of Behaviour: When the alteration parameters are set to $a_s = +1$ and $a_f = -1$, evasive movements emerge, *i.e.* when the object is in the

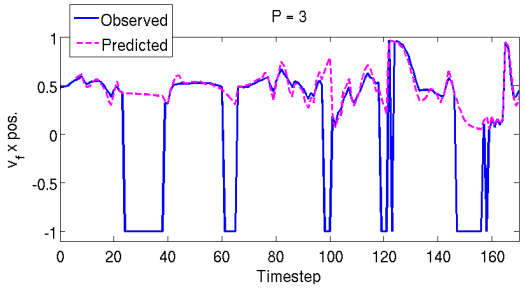
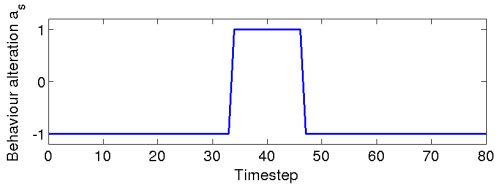
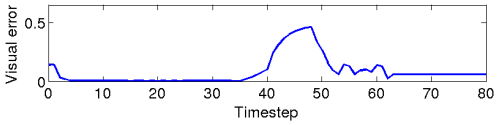


Fig. 7. Temporal loss of object: The object of interest (green cup) was moved behind an other object (a piece of paper) in front of the robot. Consider the x–position of the observed visual feature of interest $\mathbf{v}_f^{(obs)}$ (same principle applies to the y–position). The loss of the observed visual feature due to occlusion is represented by $\mathbf{v}_f^{(obs)} = [-1 \quad -1]^T$. In such situations, the system relies on the predicted feature $\mathbf{v}_f^{(pred)}$ (magenta graph). While the object of interest is occluded, the system anticipates its position ($\mathbf{v}_f^{(pred)}$) and generates the motor commands to follow it. Once the object of interest gets visible (observable) again, the system relies on $\mathbf{v}_f^{(obs)}$.

robot's field of view, the robot turns its head away from it. The robot can switch from the tracking movements to evasive movements and vice versa by simply touching its head sensor, changing the sign of a_s. Thus, the selection of tracking or evasive behaviour is done by changing only the sign (direction) of the alteration parameter, but not its scalar absolute value. A typical scenario is shown in fig. 8.



(a) Behaviour alteration parameter a_s



(b) Visual error

Fig. 8. Switching from tracking behaviour to evasive behaviour and vice versa. At the beginning of the interaction, there is a certain amount of visual error and $a_s = -1$. The robot compensates for this error by saccading to the object. At timestep 34, parameter $a_s = 1$, changing the robot's behaviour from tracking to evasive movements. This is indicated by the increasing visual error. At timestep 47, parameter $a_s = -1$, changing back the robot's behaviour from evasion mode to tracking mode. It again tries to compensate for the large visual error by moving accordingly. Thus, the visual error decreases.

IV. CONCLUSION

We proposed a predictive action selector which generates meaningful robot behaviour from a minimum amount of training data. Its operating principle is based on internal prediction and evaluation of learned sequences. We tested our system on a NAO humanoid robot. We showed how complex robot motion sequences can be composed out of very simple and few taught motion sequences. We showed that different behaviour (tracking and evading an object) can be accomplished by the same system, without the need for behaviour–specific re–training. We showed how a search of the visual space can be avoided in case of temporary feature loss by using internal prediction of the visual features. In near future, our proposed system will also control the robot arms in addition to the head, in order to realize behaviour involving hand–eye coordination.

REFERENCES

[1] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 151–180, 2007.

[2] D. Vernon, "Enaction as a conceptual framework for developmental cognitive robotics," *Paladyn Journal of Behavioral Robotics*, vol. 1, no. 2, pp. 89–98, 2010.

[3] D. George and J. Hawkins, "Towards a mathematical theory of cortical micro-circuits," *PLoS Computational Biology*, vol. 5, no. 10, pp. e1 000 532, 1–26, 2009.

[4] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, no. 5504, pp. 599–600, 2001.

[5] K. Noda, K. Kawamoto, T. Hasuo, and K. Sabe, "A generative model for developmental understanding of visuomotor experience," in *IEEE International Conference on Development and Learning (ICDL)*, vol. 2, 2011, pp. 1–7.

[6] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment," *PLoS Computational Biology*, vol. 4, no. 11, pp. e1 000 220, 1–18, 2008.

[7] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 721–736, 2014.

[8] D. Vernon, G. Metta, and G. Sandini, "The icub cognitive architecture: Interactive development in a humanoid robot," in *IEEE International Conference on Development and Learning (ICDL)*, 2007, pp. 122–127.

[9] D. Vernon, C. v. Hofsten, and L. Fadiga, *A roadmap for cognitive development in humanoid robots*, R. Dillmann, Y. Nakamura, S. Schaal, and D. Vernon, Eds. Springer-Verlag Berlin Heidelberg, 2010, vol. 11.

[10] P. Gaussier, S. Moga, M. Quoy, and J.-P. Banquet, "From perception-action loops to imitation processes: A bottom-up approach of learning by imitation," *Applied Artificial Intelligence*, vol. 12, no. 7-8, pp. 701–727, 1998.

[11] L. Berthouze and Y. Kuniyoshi, "Emergence and categorization of coordinated visual behavior through embodied interaction," *Autonomous Robots*, vol. 5, no. 3-4, pp. 369–379, 1998.

[12] T. Shibata, H. Tabata, S. Schaal, and M. Kawato, "A model of smooth pursuit in primates based on learning the target dynamics," *Neural Networks*, vol. 18, no. 3, pp. 213–224, 2005.

[13] D. Blank, D. Kumar, L. Meeden, and J. B. Marshall, "Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture," *Cybernetics and Systems: An International Journal*, vol. 36, no. 2, pp. 125–150, 2005.

[14] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel distributed processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland, Eds. MIT Press, Cambridge, 1986.

[16] ——, *Cognitive Modeling, ch. Learning representations by back-propagating errors*, T. A. Polk and C. M. Seifert, Eds. MIT Press, Cambridge, 2002.

[17] G. Cheng, A. Nagakubo, and Y. Kuniyoshi, "Continuous humanoid interaction: An integrated perspective – gaining adaptivity, redundancy, flexibility – in one," *Robotics and Autonomous Systems*, vol. 37, no. 2, pp. 161–183, 2001.