Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Learning information acquisition for multi-tasking scenarios in dynamic environments

Cem Karaoguz, Tobias Rodemann, Britta Wrede, Christian Goerick

2013

Preprint:

This is an accepted article published in IEEE Transactions on Autonomous Mental Development. The final authenticated version is available online at: https://doi.org/[DOI not available]

Learning information acquisition for multi-tasking scenarios in dynamic environments

Cem Karaoguz, Tobias Rodemann, Britta Wrede, and Christian Goerick

***Abstract*—Real world environments are so dynamic and unpredictable that a goal-oriented autonomous system performing a set of tasks repeatedly never experiences the same situation even though the task routines are the same. Hence, manually designed solutions to execute such tasks are likely to fail due to such variations. Developmental approaches seek to solve this problem by implementing local learning mechanisms to the systems that can unfold capabilities to achieve a set of tasks through interactions with the environment. However, gathering all the information available in the environment for local learning mechanisms to process is hardly possible due to limited resources of the system. Thus, an information acquisition mechanism is necessary to find task-relevant information sources and applying a strategy to update the knowledge of the system about these sources efficiently in time. A modular systems approach may provide a useful structured and formalised basis for that. In such systems different modules may request access to the constrained system resources to acquire information they are tuned for. We propose a reward-based learning framework that achieves an efficient strategy for distributing the constrained system resources among modules to keep relevant environmental information up to date for higher level task learning and executing mechanisms in the system. We apply the proposed framework to a visual attention problem in a system using the iCub humanoid in simulation.**

***Index Terms*—Attention mechanisms, multi-task learning, robots with development and learning skills.**

I. Introduction

In order for a learning process to be developmental in character, that is to develop an emergent structure that goes beyond adapting parameters for a pre-programmed structure, it needs to be embedded in a system that is able to interact with its environment through perception and action. This allows the system to analyse information from the environment with respect to dependencies and to learn which information is important for which task. Such developing systems usually employ multiple learning mechanisms that are used to acquire capabilities for executing different tasks. Naturally each local learning mechanism may require information about a different set of environmental states. However, the amount of sensory information that can potentially be obtained is vast. Especially in the vision domain numerous cues may lead to a great number of candidate information sources. Processing all these information sources together is impractical due to the constraints the systems have on the information gathering and processing such as computational cost, energy consumption, memory restrictions, physical constraints, etc. This work focuses on the physical constraints: systems with sensors that cover a limited area (e.g. cameras with limited field of view) or having variable accuracy within their coverage must be directed towards different regions in the scene in order to collect information from the environment. This may introduce partial observability to the interaction of the system with its environment: when a particular element is attended, information about every other element in the scene is not accessible by the system until the time they are focused on. Such a partial observability has crucial effects on the local learning mechanisms the system employs as well as the task execution process. In order to be able to support the processes of learning and task execution a system has to discover important information sources in the scene and distribute its resources between them efficiently.

Pre-programming approaches for the information acquisition process (i.e. manual coding, prioritisation) may be insufficient for complex systems operating in dynamic environments. First of all, it could be difficult to provide the system with the information of modules, which deliver task-relevant information, beforehand. This is due to the fact that such knowledge may not exist before a task policy is learned or the scenario may be too complex to derive this knowledge manually as it depends on the properties of the scene elements (e.g. dynamic elements of the scene require more frequent updates than static ones). Furthermore, changing environmental conditions or degradations in module functionalities (e.g. when a module breaks down) may affect the operation of the system. Robustness is a crucial feature to cope with such situations. Moreover, multi-tasking situations require multiple targets to be monitored simultaneously. The human visual system resorts to time sharing in such situations i.e. learning strategies of employing sequential gaze-shifts between targets depending on tasks [1]. Additionally, it has been shown that humans when acquiring a motor task not only learn the motor behaviour and control, but they also learn the associated eye gaze pattern as shown by similar eye movements in complete darkness as in light conditions [2]. This indicates the necessity for a learning mechanism that moderates between motor and attention control.

The information acquisition process can also be learned for an artificial system so that an efficient support is provided for the local information processing mechanisms (such as task learning and task execution) in the system. We propose a system-level approach to realize this idea. A modular system architecture is adopted to structure the information acquisition

Cem Karaoguz and Britta Wrede are with the Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, 33501, Bielefeld, Germany. e-mail: {ckaraogu, bwrede}@cor-lab.uni-bielefeld.de

Tobias Rodemann and Christian Goerick are with the Honda Research Institute Europe GmbH, 63073, Offenbach, Germany. e-mail: {name.surname}@honda-ri.de

and processing in a formalised way. Fodor outlines a coarse level of formalism for modularity [3]. In the modular systems approach elementary functionalities of information gathering and processing (e.g. colour tracking, motion detection, object recognition) are encapsulated in modules. Using modules it is possible to construct a hierarchical system where learning mechanisms at higher levels are supported by information acquired and pre-processed by modules in lower levels. The representation of the environment can be done in a constructive way using these modules: a module can generate sensor-orienting motor commands to localise the information source it is interested in. Then, mid- and/or high-level representations of the scene can be built from the information source. Making associations between task and information at the module level rather than feature level can be more efficient since higher level representations such as proto-objects or objects are less redundant for tasks. The modularity approach translates the problem of finding task-relevant information sources to finding the modules delivering useful information. The arbitration problem then, becomes distributing the constrained system resources (e.g. sensor orienting actuators) between module demands (e.g. gaze commands). The proposed framework contains a module management learning (MML) mechanism to learn an efficient distribution of the constrained system resources among different modules. The learning is done using rewards indicating the outcomes of the actions the system takes. The functional description of the proposed system is illustrated in Fig. 1.

An earlier version of the framework was explained in [4] where only the prioritisation of modules was learned for a simple multi-tasking scenario. The contribution of this paper is two-fold: first, we extended the framework with learning temporal dynamics of the information gathering process. The basis of the module management mechanism implements a simple form of a dynamic neural fields (DNF) approach. The parameters of DNF associated with each module are adapted using rewards. To our knowledge combining rewards with DNF is a novel approach in the literature. Second, we investigate the capability of the framework to support other local learning mechanisms embedded in the system. Also, the scenario is modified to a more complex version with a more dynamic environment. The framework is applied to a multi-task scenario for the iCub humanoid platform in a simulation environment [5], [6]. The scenario requires touching one of the three objects moving on a table while keeping interaction with a partner present in the scene. The system employs policies for executing these tasks. Besides MML, the system also accomodates an additional learning mechanism for the object selection policy for reaching. The parameters of these two local learning mechanisms are initialized either with pre-defined values or randomly. In the beginning the system has no idea what it has to do (e.g. which object should be selected, reached and touched? Is it necessary to look at the parter? If yes for how long? etc.). During interactions of the system with its environment these parameters are learned and the system acquires knowledge about the tasks. Hence, the task performance of the system gradually improves. It is shown that the MML framework learns an efficient strategy to update the system's knowledge about environmental states in terms of arbitrating gaze commands from various modules to support learning and executing tasks.

The system is tested in 20 different experimental setups each of which defines a different environment and unique task constraints. For comparison additional methods were implemented (e.g. random gazing, saliency) and tested under the same experimental setups. We show that our framework can find better strategies than the compared methods and provide better support to the higher level policy learning mechanism. The proposed framework can adapt its parameters to the temporal properties of varying environments. Moreover, our framework separates the control mechanisms of information acquisition and policy execution for manipulation tasks. This reduces the complexity of task policies (designed or learned) since they do not have to deal with uncertainties in the information acquisition process. Our framework forms the bottom-up information acquisition process for these policies using task outcomes (i.e. rewards). Despite this functional separation both learning mechanisms co-evolve and end up in consistent strategies. This separation also speeds up the learning process. Compared to common reinforcement learning applications we report shorter learning times.

This paper is organised as follows: Sec. II presents several well established methods in the literature addressing whole or some part of the stated problem as well as representative state of the art modular systems using these methods. Sec. III explains the proposed module management mechanism and how to learn a module management strategy for efficient information acquisition. Sec. IV describes the modular system and the scenario to which the proposed framework is applied to. Sec. V presents the experimental results. The conclusion and a brief discussion is given in Sec. VI.

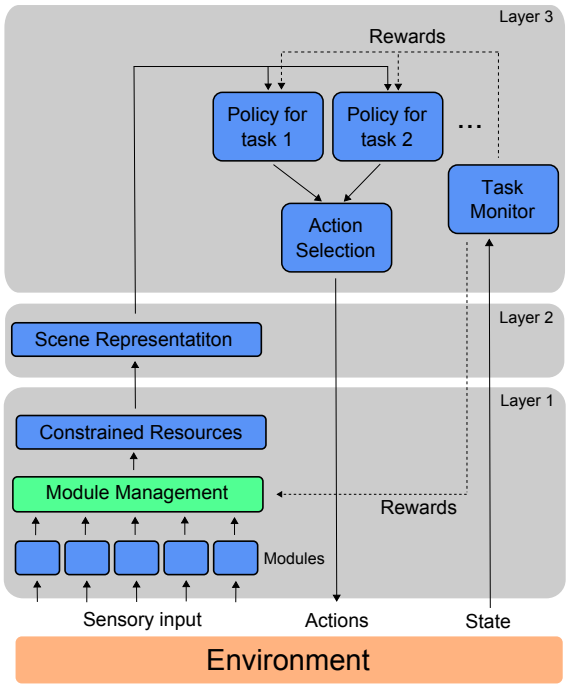


Fig. 1. The functional description of the proposed framework.

II. Related Work

The related work is evaluated in two main domains to facilitate the comparison with the proposed approach. The first domain is the structural organization of a system. This domain inspects the fundamental principles followed in the design of the system in relation to modularity. The second domain is the selection of actions for information acquisition since learning information gathering strategies is the novel idea our approach introduces.

An early formalisation for modular systems was introduced by Brooks where a layered control architecture for mobile robots was explained [7]. This approach, called subsumption, builds layers of control systems in a hierarchical manner at increasing levels of complexity. Layers are composed of modules each of which performs a certain computational functionality. Modules that belong to higher-level layers can affect lower levels by suppressing their outputs. The idea was later used in different applications with various extensions [8], [9], [10]. Although the subsumption approach facilitated engineering of formalised and successful control architectures for autonomous systems, it poses some limitations [11]. The arbitration problem was addressed by prioritisation: modules in higher layers have priority over the ones in lower layers for the access of motor resources. This may cause problems in varying environments. Priorities defined for one situation may not be valid for others. Moreover, as the complexity of the system grows it may be difficult to find optimum assignments of priorities.

Limitations of pre-programmed strategies motivated researchers to investigate adaptive solutions. Reinforcement learning (RL) has become a widely used framework in this domain because it facilitates learning through interactions with reward signals indicating the outcome of a system's actions [12]. Kwok et al. applied RL directly to the active sensing problem in a soccer playing robot without employing a modular architecture [13]. The state space is composed of a representation of objects relevant to the task (i.e. scoring goal), odometry and an explicit representation of uncertainties over information the robot has. The system learns which object to look at depending on the current state. Casting the active sensing problem directly into a RL framework may have some shortcomings. Different kinds of information may be necessary if the approach is desired to be implemented for additional tasks. Simply adding new representations in the model may cause the state space to grow beyond computational limits. This may only be avoided by re-designing the state space.

Another non-modular approach is the saliency-model with inhibition of return proposed by Itti et al. [14]. This scheme is based on computing conspicuity maps for various features and combining them to form a final saliency map. The most conspicuous location in the saliency map is selected as the gaze target. Deadlocks are avoided by the inhibition of return mechanism that suppresses the activity in previously visited regions. The saliency approach has successfully been used as the information gathering tool in active vision systems such as [15]. However, the mechanism this approach models is purely bottom-up. There are various works extending this scheme with top-down information to model the task influence on the attention process. A comprehensive one is presented in [16] where probabilistic models of object representations are learned from features and these models are used to define weights to bias feature maps so that the target object pops out. The application of this model is limited to simple object search tasks where target objects are already known. However, in developing systems where tasks are learned, task relevant objects should also be learned. The framework proposed in [17] goes more in this direction. This framework presents a model that can learn top-down attention skills incorporated by a bottom-up saliency mechanism . Task-specific knowledge on potential gaze targets was acquired using RL. The bottom-up saliency mechanism served as an exploration map to extract such knowledge. The learning is based on spatial relations of objects. This may cause problems with dynamic scenes.

While saliency based frameworks model the attention process in the spatial domain, a new line of research introduces object-based attention. Rather than using the image features independently for the saliency computation, the object-based approach uses these features to group parts of the visual field which are likely to correspond to parts of the same object in the real-world. These entities are referred to as 'proto-objects'. This concept is close to the modular approach and also compatible with our framework as the information collected from scene entities in the scene representation mechanism can be represented as proto-objects. Alternative examples of proto-object based visual attention models are presented in [18], [19] and several successful practices of proto-objects in large-scale complex intelligent systems are given in [20], [21]. The proto-object based attention models implement an inhibition of return mechanism as in the saliency model. However, the inhibition is done in object domain rather than spatial domain in this case. This requires an additional mechanism to track proto-objects. In our framework this requirement is eliminated since the system learns temporal sequences of gaze-shift between entities. Moreover, the task influence is realized in the same way as for the saliency-model and it is not learned unlike our framework.

Sprague et al. presented an approach based on modular RL for controlling behaviours incorporating visual information [22]. In this framework, small visual routines were handled as complete sensory-motor processes, called microbehaviours. From a functional point of view, microbehaviours can be regarded as modules. A microbehaviour is a RL agent: it has a pre-defined task, it gathers visual information, which predicts the current state of the environment relevant to the task and it takes actions based on this information to fulfil the task and receive rewards. A policy that maximises the acquired reward is learned during the course of such interactions. The problem of which microbehaviour gets access to the camera control is solved by reducing the cost of uncertainty in the internal scene representation over the microbehaviours. While the model proposed by Sprague et al. is robust and flexible, the definition of which microbehaviours should be used in which scenarios is still done in a pre-programmed fashion.

Busquets et al. designed a cooperative multi agent system (MAS) for a robot navigation task [23]. The interaction be-

tween different modules yields a complex trade-off mechanism where the resource allocation for different modules has to be managed. RL was used to optimise such a complex trade-off. It was shown that learned strategies performed better than pre-programmed ones. The model uses a mixed repertoire of looking and moving actions. However these can be separated since looking actions do not have direct effect on the environment, they only change the internal state of the system. Such a separation may result in simpler representations of state and action spaces and a more efficient learning strategy. Hence, the authors reported that an effective strategy is only learned with a careful design of state space.

Pezzulo et al. used the schema concept [24] to construct a behaviour coordination architecture based on the model of the praying mantis [25]. The architecture consists of four main parts: actuators realise a given motor command (e.g. looking or moving), perceptual and motor routines define sets of motor commands for primitive actions (e.g. detect colour, turn left, etc.), a behaviour repertoire defines a number of pre-programmed schemata (e.g. detect prey, chase) and inner states (e.g. hunger, fear, etc.) drive the behaviours in the repertoire. The model selects the appropriate schemata for a given context (i.e. state of the world and its drives) based on the activity levels of schemata. The parameters governing these activity levels are pre-programmed. Also, the coupling between high level behaviours and low level visual routines are fixed. For example a 'detect prey' schema assumes a prey is red and moving. Such assumptions may be relaxed by learning the relations between the visual routines and behaviours (i.e. which visual routines are useful for which task).

Compared to [7], [22], [25] our framework learns which modules are important for the tasks undertaken. This makes the system more robust against changing environments and may lead to easier design of the system. In contrary to [13], [23] in our framework action selection mechanisms for information acquisition (e.g. gazing) and for world manipulations (e.g. reaching, navigation, etc.) are separated. In this way, the information acquisition process can be optimised for parallel running task policies employed by the system. New tasks and policies can be introduced to the system easily without concern about the information acquisition process. As opposed to [17] the proposed method is not restricted to a spatial representation of information sources. Therefore, it may better cope with dynamic scene elements. Several examples of modular large-scale interacting systems were previously presented [26], [20], [21], [27], [28], [29]. The information acquisition processes in these systems follow a manually designed procedure like prioritisation, subsumption or state machines. Utilising our framework in these systems, various shortcomings of the above mentioned methods will be overcome.

III. METHODS

The general concept of our proposed framework is shown in Fig. 1. The system consists of a hierarchical organisation of three main layers. The first layer contains modules that are responsible for performing basic perceptual routines like colour tracking or motion detection. At any point in time, modules may generate motor commands for gaze shifts (solid arrows emerging from the modules in the figure) to localise the information source they process. The arbitration of these motor commands is done by the management mechanism residing in the same layer. The motor command from the selected module is sent to the camera system shown as 'constrained resources'. Perceptual information is acquired by a scene representation mechanism in the second layer. It is implicit (i.e. over the constrained resources) that modules deliver perceptual information about the environment. For example, if a red object is being tracked by a red colour tracking module, information about that specific object (e.g. position, velocity, etc.) can be extracted. The scene representation mechanism makes an interpretation of the scene and keeps this information available and temporally consistent. Hence, even if the system does not direct its sensors towards an object, it may still have an idea about where the object is. Third layer contains action-perception policies that use the scene representation for performing tasks. For example the position information of the red object in the scene representation can be used by a policy for grasping red objects. For task policies, an action selection mechanism may be necessary for arbitration if there are multiple tasks requiring the same actuator of the system. However, the tasks implemented in this work do not require this. A task monitoring mechanism at this layer generates reward signals depending on the outcome of the actions the system takes. A positive reward is given if the tasks are fulfilled successfully, negative otherwise. Rewards are used for learning done by various mechanisms throughout the system such as high-level policy learning mechanisms or MML.

A. Management of Modules

As stated before, the module management framework arbitrates motor commands for information acquisition (e.g. gaze-shifts). Arbitration should be done regarding the relevance of modules to the tasks appointed to the system and temporal constraints of the tasks (e.g. how often a module should be selected, in what order should modules be executed). Dynamic neural fields (DNF) formalised by Amari [30] is a well established and widely used approach that can achieve sequencing in the presence of numerous candidates for action (e.g. motor commands for gaze-shifts). However DNF do not directly address how to prioritise modules in relation to their task-relevance. We use similar dynamics with an extension to solve this problem for the module management process: a weighting mechanism determines priorities of modules while an inhibition mechanism similar to DNF achieves temporal scheduling between modules.

At time step t, any module i can generate a motor command $\overrightarrow{\theta}_i^t$ for a gaze shift. A motor command indicates the posture of the sensory device, which attends to the entity the module is interested in. There are no constraints on degrees of freedom and reference frame for the motor commands. In fact, different modules can use different qualities for these properties. This facilitates combining modules using different types of sensors such as audio and vision. In our application all modules use the head-centred reference frame and only one camera

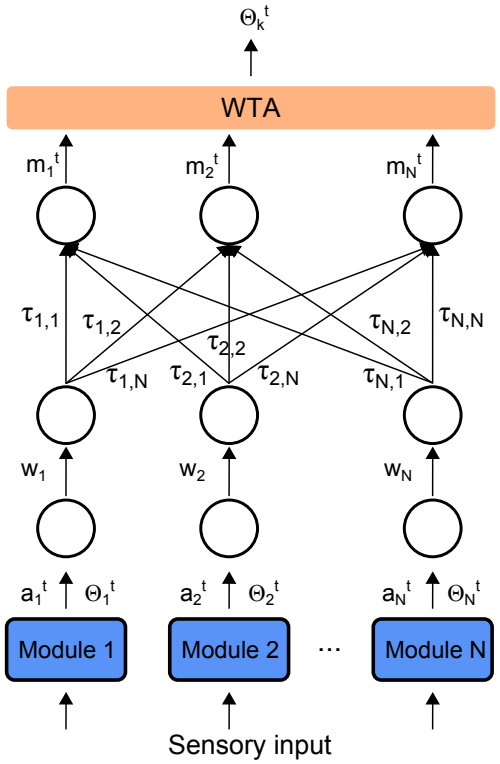


Fig. 2. The mechanism that performs the management of the module selection process. At every time step t modules can generate motor commands $\overrightarrow{\theta}_i^t$ and associated activity levels a_i^t. The activity levels are weighted by their corresponding weights w_i and inhibited by the currently selected module. The strength of inhibition is determined by an inhibition function defined over time domain and parametrised by inhibition parameters $\tau_{i,j}$. As a result, modified activity levels m_i^t are computed. A winner-take-all (WTA) mechanism determines the selected module index of which is referred to as k and motor commands from that module are executed.

(i.e. motor commands indicate pan and tilt angles of the left camera). Additionally, every module has an activity level a_i^t that is a value between 0 and 1. If a module generates a motor command, the activity is set to one. Modules, which do not generate any motor command will have zero activity. These activities are constantly modulated by the weighting and inhibition mechanism (Fig. 2). At every time step, the module with the highest modulated activity level is selected as the *active module* (referred to with index k). The motor command from this module $\overrightarrow{\theta}_k^t$ is executed to change the camera configuration for the next time step as $\overrightarrow{\theta}^{t+1} = \overrightarrow{\theta}_k^t$.

From a DNF point of view, the output activity of a module can be represented as activities on a neural field encoding the action space of the module (i.e. gaze space). Since in this work we are not interested in local interactions within the output space of modules, but global interactions of modules with each other, we represent the outputs as tuples of motor commands and activity levels for simplicity. Additionally no specific spatial topology exists between modules. Hence, neighbourhood interactions between modules do not vary with space in contrast to regular DNF frameworks.

The process of modulating module activity levels is formally defined as:

$$m_i^t = w_i y(t; \tau_{k,i}) a_i^t, \tag{1}$$

where a_i^t and m_i^t are the output activity and modulated output activity of module i at time step t respectively, w_i is the weight of module i, $y(t; \tau_{k,i})$ is the inhibition function parametrised by time constant $\tau_{k,i}$ and k is the index of the *active module* at time step t. From the modulated outputs, the *active module* for the next time step is selected following winner-take-all principle.

Weights w_i assigned to each module determine the importance of modules. The inhibition function $y(t; \tau_{i,j})$ determines how strongly module i inhibits module j at time t. This is defined as:

$$y(t; \tau_{i,j}) = \begin{cases} 1 - f(t; \tau_{i,j}) & \text{if } i = j, \\ f(t; \tau_{i,j}) & \text{if } i \neq j. \end{cases} \tag{2}$$

The case $i = j$ is called self-inhibition and the case $i \neq j$ is called lateral inhibition. The logistic function is used to model the temporal characteristics of inhibition:

$$f(t; \tau_{i,j}) = \frac{1}{1 + \exp\big(-\tau_{i,j}(t - t_c)\big)}, \tag{3}$$

where t_c is the time step when the currently *active module* was selected. In other words, $(t - t_c)$ keeps track of the *active module* runtime. This is like a clock that is set to zero every time a new module is selected. The logistic function limits the strength of inhibition between zero and one. This prevents instabilities caused by excessive inhibitions.

When a module is selected as the *active module* it inhibits the activity of other modules (lateral inhibition) and itself (self-inhibition). Self-inhibition ensures that the active module releases control at some point (a function similar to the inhibition-of-return), while lateral inhibition has a strong influence on the choice of the next active module. To gain an insight on how the inhibition mechanism works the inhibition functions for self-inhibition and lateral inhibition are shown in Fig. 3. When the module is first selected, the lateral inhibition is strong and the self-inhibition weak. The strengths of these inhibitions change during the time the module stays active: lateral inhibitions decrease while self-inhibition increases. Eventually another module is selected and the whole process repeats. The inhibition parameters $\tau_{i,j}$ control the profile of the inhibition function establishing temporal sequencing of modules.

B. Learning Management Parameters

The module management framework is influenced by weights w and inhibition parameters τ. Changing these parameters results in variations in gazing behaviour of the system. This subsection explains how these parameters are adapted so that an efficient gazing behaviour for specific scenarios is learned. A scenario consists of one or more tasks that have to be fulfilled in parallel. In our application the scenario involves reaching an object while staying in interaction with a partner. However, other relevant scenarios exist such as autonomous navigation with tasks like obstacle avoidance, following a path, etc. There are various information sources in the environment crucial to the tasks in the scenario. The environment is dynamic, i.e. the information sources can move

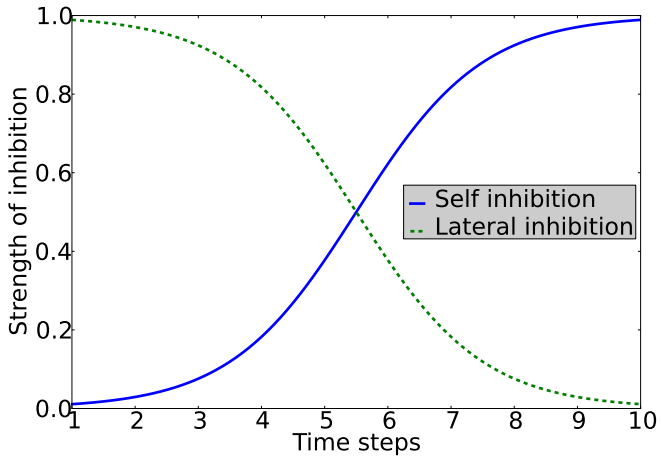


Fig. 3. Change of inhibition strength with time for self-inhibition and lateral inhibition. Computed using inhibition function $y(t;\tau)$ for $\tau = 0.5$ and $t_c = 0$.

in the world. The system is equipped with various modules that can acquire information about the environment. It has no a priori knowledge on which modules are task-relevant. We assume that there is a value set for parameters w and τ that results in an efficient way of updating relevant information sources to execute a scenario. Our learning algorithm attempts to find such values.

Conventional supervised learning algorithms may posses limitations in solving the problem at hand: explicit knowledge on how relevant the information source is and how often it should be updated is required. We follow a reward based strategy where such knowledge is inferred from the consequences of the actions the system takes. Basically, the system behaves under the scenario with a set of module management parameters. Depending on the outcome of the scenario a reward is generated. If all tasks were fulfilled successfully a positive reward is given whereas, if one or more failed the reward is negative. The reward is used to adapt the parameters and a new instance of the scenario is executed. The execution time of a scenario is called an epoch.

The learning mechanism has to deal with the credit assignment problem: for an epoch yielding a positive (or negative) outcome which internal decisions of the management mechanism were useful (or to blame)? To solve this problem for every epoch the indices of *active modules* in the last t_{hist} time steps were stored in a list and the reward acquired at the end of the epoch is propagated over the parameters related to these modules. This is a form of eligibility traces as explained in [12]. After a number of iterations the relevant modules will be revealed statistically in terms of gained positive rewards. The uncertainty of reward delay complicates finding an optimum value for t_{hist}. This is a common problem in RL applications [12]. Hence, there is no specific rule for selecting t_{hist}. Experiments showed that setting t_{hist} too low results in longer convergence times whereas too high values may lead to instabilities in the learning process. The credit assignment problem can further be addressed by a discount factor $0 < \gamma \leq 1.0$. This sets a heuristics for recency i.e. actions taken closer to the time a reward is acquired are valued more. In essence the value iteration computes the value of a state based on possible actions that can be taken in that state and the values of states possible to be reached by taking those actions. We would like to use such values as a metric of how relevant modules are to the scenario at hand. As a result, the following update rule is used:

$$\forall n \in \{1, 2, ..., t_{hist}\} :$$
$$w^{e+1}_{B(n)} \leftarrow w^{e}_{B(n)} + \alpha(r + \gamma^n w^{e}_{B(n)}), \quad (4)$$

where e denotes the epoch, r is the reward, α is the learning rate, $0 < \gamma \leq 1.0$ is the discount factor, $B(n)$ is a list that stores the indices of *active modules* in the last t_{hist} time steps of epoch e. Inhibition parameters control transitions between modules. In general we would like to reinforce a sequence (e.g. transition from module A to module B) that results in a rewarding outcome and avoid a sequence that yields a negative outcome. Therefore, adaptation of inhibition parameters is done in a Hebbian way [31] as:

$$\forall n \in \{1, 2, ..., t_{hist} - 1\} :$$
$$\tau^{e+1}_{B(n),B(n+1)} \leftarrow \tau^{e}_{B(n),B(n+1)} - \beta r, \quad (5)$$

where r is reward and β is the learning rate. Basically, a positive reward decreases the lateral inhibition parameter between modules $B(n)$ and $B(n+1)$, increasing the probability of selection of module $B(n+1)$ after $B(n)$ was active. If selecting module 1 after module 2 yielded a positive reward this scheme makes it more likely to keep this sequence. In the opposite case, likelihood of selecting module 1 after module 2 is reduced by a negative reward leading to exploration of sequences with other modules. The inhibition mechanism also helps to keep the overall system stable by preventing a strong module dominating the competition process. This is similar to the homoeostatic plasticity in the nervous system. Turrigiano and Nelson showed that synaptic strengths of neurons were adapted dynamically to prevent neural circuits from becoming hyper- or hypo active [32]. Changing the synaptic weights of neurons via a learning process will result in modification of the gain of the network to which the neurons belong to. This will cause too strong (for gains higher than one) or too weak (for gains lower than one) propagation of activities in the network. Local or global inhibitory mechanisms may regulate the gain of the network to provide stability. Similarly in our framework, self- and lateral inhibitions prevent modules with high weights from becoming overly-active and keep modules with low weights in competition. Weights and inhibition parameters are initialised randomly and kept between 0 and 1.

The learning process can be regulated in order to avoid changing the parameters after the learning is finished. For this purpose learning rates α and β are computed online at every epoch based on the performance of the system as:

$$\alpha = \alpha_b \exp(-\alpha_s(\overline{r}^e - r_{min})), \quad (6)$$
$$\beta = \beta_b \exp(-\beta_s(\overline{r}^e - r_{min})); \quad (7)$$

where α_b and β_b are base values, α_s and β_s are the sensitivity of the change of α and β in response to the change in the

average reward, r_{min} is the minimum reward that can be obtained and $\overline{r}^e$ is the smoothed average acquired reward at epoch e. This is computed as:

$$\overline{r}^e = \overline{r}^{e-1} + (r^e - \overline{r}^{e-1})/\epsilon_s, \tag{8}$$

where r^e is the reward acquired at epoch e and ϵ_s is the smoothing parameter. The initial value is set to zero $\overline{r}^0 = 0$. Performance based regulation also provides robustness to the learning framework. In case of drastic changes (e.g. scenario change) where learned configurations of parameters are no longer valid, the system will start to fail, leading to a decrease in the average reward. In this case learning rates will increase and the system will re-adapt itself to the new conditions. This automatic adaptation to environmental changes was also demonstrated for a visual sensory-motor learning task [33].

IV. System Implementation

The proposed framework was applied to a system capable of performing interaction tasks in a simulation environment. The iCub humanoid was used as the platform for the system and the experiments were conducted using the iCub Simulator [5], [6]. The general scheme of the system follows the concept shown in Fig. 1. In essence, modules perform elementary operations for information acquisition and pre-processing and may generate a motor command for sensor orientation depending on the result of the processing. A module is allowed to generate a single motor command at a time, any ambiguity in data (e.g. two similar objects in the scene) is assumed to be handled by the module itself. This section gives detailed information about the aspects of the system.

A. Modules for Information Acquisition

Different modules operating on different domains of modality can be plugged into the proposed framework. However, due to difficulties of reproducing stimuli other than visual ones in the simulation environment (e.g. audio signals) only visual processing modules were implemented. Methods that these modules apply are basic. State of the art approaches can be used for better performance. However, the implementation of modules is not the focus of this work. A snapshot of the system with module outputs is shown in Fig. 4. The following modules are used in the system:

1) Colour Tracking Module (CTM): extracts a specific colour feature from images and generates gaze commands to keep this feature in the centre of the image. Three CTMs are implemented for red, green and blue and indicated as CTM-R, CTM-G and CTM-B, respectively. At every time step an image $I^t(x, y)$ is acquired where t denotes the time step and (x, y) is the horizontal and vertical pixel positions. Colour channels $I^t_c(x, y)$ where $c \in \{R, G, B\}$ are extracted and sent to the corresponding CTM. The module computes conspicuity maps using the colour opponency method [14]:

$$C^t_c(x, y) = \sum_{c'} I^t_c(x, y) - (\eta I^t_{c'}(x, y) + \vartheta), \tag{9}$$

where C^t_c is the resulting conspicuity map for colour c, c' denotes the colours other than c, η and ϑ are threshold parameters. The resulting conspicuity map indicates the positions of the detected colour features the CTM is assigned to. For the image shown in Fig. 4(b) conspicuity maps computed by CTM-R, CTM-G, CTM-B are shown in Fig. 4(c), 4(d) and 4(e), respectively. If the colour feature is detected, a motor command for a gaze-shift towards its position is generated. The exact target for the motor command is the centre of mass of the positions where the feature is detected. The objects in the environment are unique in colour. Thus, no arbitration between multiple objects of the same colour is necessary. Conversion from image coordinates to the eye-motor coordinates is done using the kinematics of the robot head [5].

2) Motion Detection Module (MDM): detects motion in the current field of view by computing a simple temporal difference between acquired images as:

$$M^t(x, y) = I^t(x, y) - I^{t-1}(x, y), \tag{10}$$

where M^t is the motion map in the image coordinates at time step t, I^t and I^{t-1} are the images taken at time steps t and $t - 1$, respectively. Since this method is susceptible to ego-motion, the module does not process any information whenever the cameras are moving (i.e. during gaze-shifts). The resulting motion map for the case shown in Fig. 4(b) is shown in Fig. 4(f). The only moving object was the blue one in this case. If a motion is detected in the image, MDM generates gaze-shift command to bring the region where motion is detected (i.e. centre of mass of the pixels with motion) to the centre of the image. As in CTM, conversion from image coordinates to the eye-motor coordinates is done using the kinematics of the robot head. Since all three objects can move (depending on the parameters of the experimental setup), it is possible that motion from multiple objects is detected in one time step. In such cases the resulting motor command will be towards the centroid of all detected motion. Such an implementation aims to keep all moving objects in the field of view.

3) Visual Exploration Module (VEM): keeps track of locations, which were not gazed at for a long time and proposes gaze-shift towards these areas. To achieve this an exploration map E^t covering the whole head motor space in two dimensions (i.e. pan and tilt) is initialised and updated at every time step t in the following way:

$$E^t(\theta) \leftarrow P^t_{inh}(\theta)(E^{t-1}(\theta) + \xi), \tag{11}$$

where $\theta \in \Re^2$ denotes the camera pan and tilt positions in the whole head-centred reference frame. This implies an increment of all values by an amount given by parameter ξ and an inhibition of the values around the gaze position in the current time step $\overrightarrow{\theta^t}$ done by:

$$P^t_{inh}(\theta) = 1 - G(\theta; \overrightarrow{\theta^t}, \Sigma_{inh}), \tag{12}$$

where $G(x; x_0, \Sigma)$ is an n dimensional Gaussian function defined in the space $x \in \Re^n$ parametrised by mean $x_0 \in \Re^n$ and covariance matrix $\Sigma \in \Re^{n\times n}$. In the application $n = 2$ (i.e. pan and tilt dimensions) and the covariance matrix is in the form of $\Sigma_{inh} = \left(\begin{smallmatrix} \sigma_{inh} & 0 \\ 0 & \sigma_{inh} \end{smallmatrix}\right)$. The exploration map is normalised as it sums to one to be used as a probability

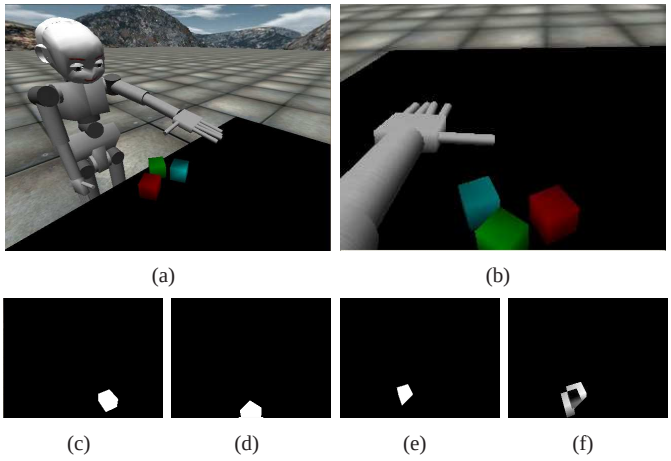


Fig. 4. A snapshot of the system during a scenario trial. (a) An external view of the robot and the environment from the simulator, (b) view from the left camera, the colour of the objects are blue, green and red from left to right respectively, (c)-(e) colour conspicuity maps for red, green and blue colours, (f) result of motion detection computation, only the blue object was in motion at this instance.

distribution. At every time step, a single motor command is sampled randomly using the resulting distribution.

4) Human Interaction Module (HIM): generates motor commands to gaze at humans whenever they are detected. Different methods can be used to detect humans in the scene. For instance face detection and sound localisation are two operations that can achieve this with different cues. However, due to difficulties in generating both auditory (e.g. voice) and detailed visual (e.g. face) elements in the simulation environment, such cues could not be used. Instead, the assumed position of the interaction partner is given directly to the module. HIM generates motor commands for gaze-shifts towards this position at every time step.

B. Scene Representation

A scene representation mechanism can keep and integrate various properties of elements in the scene over time. Even a coarse level of scene understanding can boost the performance of higher level functions in a system [34]. The scene representation is achieved by implementing a simple working memory (WM) in this work. The WM is a list of tuples keeping various properties of the scene elements. In our scenario those scene elements are the objects on the table (red, green and blue) and the interaction partner. The following properties of those elements are stored in the WM: an id for each element id, the estimated position of the element in the world p_{id} and the age of the entity in the WM a_{id}. The id is simply an integer number and distinguishes scene elements in the WM from each other. Position of the element is used by the reaching task. The age property makes it possible to delete obsolete elements from the WM.

The WM has no elements in the beginning of an epoch. At every time step an update is done. First, the age of the elements in the WM are updated:

$$\forall id \in WM : a_{id} \leftarrow a_{id} + 1. \tag{13}$$

Next, all elements with an age greater than the maximum age a_{max} are removed from the WM. The age constraint reflects the limited nature of the WM. Since our whole framework is motivated by constrained resources (computational, physical or memory) it is consistent to employ this constraint even though the memory constraints are trivial for our application. Subsequent to the age update, scene elements are updated. If an element is close to the centre of the field of view, position of the element is stored in the WM (or updated if it is already in the WM) and the age is set to zero. This is motivated from the evidence that active vision systems are likely to estimate more accurate 3D reconstructions of an object if the object resides in the centre of the field of view [35]. CTMs (R,G and B) and the HIM module are used to distinguish the scene elements for the WM. For example if the CTM-R reports that the red object is at the centre of the field of view, the id of that element is used for storing or update.

C. Tasks and Policies

The scenario designed for the system consists of two tasks that have to be executed in parallel: a reaching task and a human interaction task. The robot has to reach and touch one of the three differently coloured objects moving on a table while keeping interaction with a partner by gazing at him/her from time to time. Despite its simplicity, our scenario reflects various elements of human-robot interaction where a robot's gazing behaviour is crucial. Recent studies show that robot's gazing feedback shapes the human-robot interaction in tutoring scenarios [36], [37]. The scenario designed in this work is considered as a simplified version of a game between a robot and a child. Since the two tasks do not require a common actuator no task-level arbitration for actions is necessary. The way the system implements a solution for a task is defined as policy. There are no constraints on the implementation of policies. It can be defined manually or learned. We used both types of policies. As stated before, one of the aspects we would like to investigate is the ability of our framework to support other high-level learning mechanisms in the system. Therefore, in one instance of system implementation the reaching policy is learned via a RL algorithm. This case is called *RL Policy* (RLP). Since this case involves two separate learning mechanisms (i.e. learning in module management and learning in reaching task) it is also necessary to evaluate the MML individually. For this reason a second instance of the system was implemented with a fixed reaching policy. This second case is called *Programmed Policy* (PP). The human interaction task is the same and manually designed for both cases.

1) Reaching: In this task the system has to reach one of the three objects on the table. The objects are coloured in red, green and blue and they yield different amounts of base rewards upon reaching. These values are set to different values for every experiment setup. For that, a random object is assigned with a reward of 1.0 and the values for the other objects are set randomly between 0 and 1. Successfully reaching an object yields the base reward associated to that object. Additionally, the objects are moving on the table. The

probability of an object to make a movement in a time step is defined beforehand. This parameter is different for each object, and also set to different values in each experiment setup. This makes the differently coloured objects more or less attractive as grasping targets, depending on the relative amount of reward they provide and the difficulty of catching them.

In *Programmed Policy* the object selection is done randomly: At every time step the system samples one of the three objects randomly from a uniform distribution and starts/continues the reaching process with whatever information it has about the object in its working memory. In *RL Policy* case the policy for deciding which object to be selected for reaching is learned via a RL algorithm. For this purpose a simple and well established algorithm, Q-Learning, is used [12]. Here an action-value function $Q(s,a)$ that gives the expected utility of taking a given action a in a given state s is learned. Learning is done online using rewards r acquired from the scenario outcomes. Given state s_t and action a_t taken at time step t and observing state s_{t+1} and reward r_{t+1} in the next time step $t+1$, the action-value function $Q(s_t,a_t)$ is updated as $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \Delta Q$ with the following value:

$$\Delta Q = \alpha_q[r_{t+1} + \gamma_q \max_a(Q(s_{t+1},a)) - Q(s_t,a_t)],$$

where α_q is the learning rate, γ_q is the discount factor and $\max$ operation gives the value of the function $Q(s_t,a)$ for the action a, which gives the maximum value of the function. The state space is formed by the following five entities: distance of objects (red, green and blue) to the robot, distance of the partner to the robot, position of the hand. Since the distance/position information are real valued variables they have to be discretized to be used as a representation in the state space of the Q-Learning algorithm. This is done in the following way:

$$s = int\Big(s_{min} + \frac{(z - z_{min})(s_{max} - s_{min})}{z_{max} - z_{min}}\Big), \qquad (14)$$

where s is the encoded value, z is the value to be encoded (e.g. distance of the red object to the robot), $z_{min} = 0.1$ and $z_{max} = 0.5$ are the minimum and maximum distances that can occur in the application respectively and $s_{min} = 1$ and $s_{max} = 5$ are the minimum and maximum values for the state encoding respectively. The state value $s = 0$ is used for the case where no information is available about the entity.

Information about the scene elements (i.e. objects and the interaction partner) is gathered from the scene representation. The position of the hand is acquired directly from the simulator (i.e. proprioception). The action space of the policy determines which object to be reached. Therefore, the number of actions M is equal to the number of objects which is in our application three. Softmax action selection is used for this purpose [12]. This method computes the probability of selecting an action a as:

$$p(a|s) = \frac{e^{Q(s,a)/Z}}{\sum_{j=1}^{M} e^{Q(s,j)/Z}}, \qquad (15)$$

where Z is a positive parameter called temperature. Low temperatures cause a greater difference in selection probability for actions that differ in their value estimates, while the actions become more equi-probable for higher temperatures. Note that any noise that would emerge from uncertainties in the scene representation affects the RL algorithm. Entities in the scene representation are updated via gaze shifts. At every time step a new selection is done.

Once the object is selected, the reaching process is realised by minimising the distance between the position of the left hand and perceived object position (i.e. from the scene representation). If no information is available in the scene representation yet, no action for reaching is taken. The reaching task is fulfilled if this distance is below the minimum reaching error e_{reach}. The hand is moved in three dimensions in space using velocity control. Velocity commands are generated/updated in a ballistic fashion (i.e. no feedback control is used) using the errors between the hand position and the perceived object position. For hand movement, two degrees of freedom from the arm group of joints and one degree of freedom from the waist group of joints are used. Although blocking poses are avoided, collisions with the table may occur if the position of the reached object assumed by the system is different from the actual position of the object. In the case of a collision, the hand is moved back to a safe location and the reaching process continues. If the system fails to update the position information after the collision, the target position could be the same and a further collision may occur. This is not bad for the system as it provides data for the learning process as well, since collisions delay the reaching process which decreases the amount of reward acquired at the end.

2) Human interaction: For this task the system has to stay in interaction with a human partner present in the environment. The partner is modelled with a single variable called boredom. This variable is updated at every time step: if the partner is gazed at by the robot the boredom variable is set to zero; otherwise, it is increased by a value $0 < \tau_b < 1$ called the boredom rate. If the boredom variable exceeds the maximum value of 1.0 the partner is bored and the task is unsuccessful. Different values were generated for the boredom rate for each experimental setup.

D. Task Evaluation and Rewards

The task monitor can be thought as a general value mechanism, observing tasks, internal states and actions of the system and generating reward signals accordingly. It may be expected that it also receives an additional input from the scene representation mechanism as it reflects the knowledge of the system about its environment. However, in the current application direct sensory feedback suffices to generate reward signals indicating the task outcomes e.g. a touch sensor can inform about the grasping process, a simple auditory or visual signal can inform that the partner is bored. Such triggers are given directly in the application and the task monitor generates rewards depending on the outcome of the scenario. A scenario is unsuccessful if any of the ongoing tasks failed. For example if the human interaction task fails due to boredom of the partner, the scenario is unsuccessful yielding a negative reward. There is also a time limit t_{max} in which the tasks

has to be fulfilled. If this is exceeded the scenario is also unsuccessful. If all tasks of a scenario are achieved at time step t where $t < t_{max}$, the scenario concludes successfully and a positive reward r is generated as:

$$r = r_b + \frac{t_{max} - t}{t_{max}}, \tag{16}$$

where r_b is the base reward appointed to the object. This gives a time bonus to solutions yielding shorter task run time. In our scenario, if the object is reached without the partner being bored and time-out not being reached, the scenario outcome is a success.

V. Results and Discussion

A. Outline of Experiments

For comparison, a number of additional gaze control methods and information acquisition settings were implemented:

1) *Module Management Learning (MML)* is the proposed method.
2) *Saliency* is a simple application of a conspicuity-based bottom-up attention mechanism presented by Itti et al. [14]. For this method a saliency map covering the whole camera motor space is generated. In the beginning the map has no activity. At every time step, motor commands from modules are represented in the map as peaks of activity at corresponding positions. The peak values are set to 1.0. The most conspicuous location in the resulting saliency map was selected as the motor command for a gaze-shift if its value exceeds a previously defined threshold. Activity of previously visited positions are inhibited via an inhibition of return procedure. Inhibitions on the map weaken in every time step. A detailed explanation of saliency based attention dynamics is given in [14].
3) *Random* is a gazing strategy where at every time step a random module is selected and the gaze motor command from the selected module is executed.
4) *Perfect World Knowledge (PWK)* is the setting where the system is given the precise information about the position of objects at every time step. This means the system can perform the reaching task blindly without employing any gaze on the objects. This method was implemented for RLP to determine how well the policy learning mechanism can do in the absence of sensory uncertainties. Therefore, it can be a reference to asses the performance of the other methods. The gaze is directed to the interaction partner in this setting, so that the second task is always performed successfully.

For statistical assessment 20 different experimental setups were generated and all methods were tested in each of these setups. To achieve these experimental setups a number of environmental variables are defined:

- One object is selected randomly and its base reward r_b was set to 1. Base rewards of the other objects are sampled randomly between 0 and 0.99 from a uniform distribution.
- The probability of an object to make a random movement in a time step p_m is sampled randomly between 0 and 1 from a uniform distribution for each object.
- The boredom rate τ_b of the interaction partner was set to a random value between 0.1 and 0.3 that is sampled from a Gaussian distribution ($\mu = 0.18$).
- The maximum hand velocities $v_{max,x}$ and $v_{max,y}$ are sampled randomly between 20-40 deg/sec^2 and 10-20 deg/sec^2 from a uniform distribution for X and Y axes respectively.

This allowed us to evaluate and compare the performance of the methods in different environmental conditions.

TABLE I
Parameter values used in the experiments.

General		**MML**	
Image size	160×120	α_b	0.05
τ_b	0.2	γ	1.0
t_{max}	25	t_{hist}	10
e_{reach}	0.12	β_b	0.05
		α_s	3.0
Modules		β_s	3.0
η	1.2	ϵ_s	100
ϑ	10		
ξ	0.01	**Q-Learning**	
σ_{inh}	2	α_q	0.01
		γ_q	0.99
WM		Z	0.05
a_{max}	20		

The values of parameters used in the experiments are shown in Tab. I. Parameters for the MML and Q-Learning are determined empirically with best effort. Every module has different utility and their operations are independent from each other. For this reason, module parameters are selected in the way that each module delivers the maximum utility. Other parameters are selected with regard to computational efficiency (e.g. image size) and scenario execution time.

B. Overall System Performance

An assessment of the performance of the overall system can be made by observing the acquired rewards in time. This can give an idea on how different information acquisition methods can support the multi-task scenario. Fig. 5 shows average acquired rewards for four methods. Rewards are averaged over all experiments for each method and smoothed using Eq. 8 with $\epsilon_s = 100$. Similarly Fig. 6 shows means and standard deviations of acquired rewards for every 50 epochs over all experiments. The plots show that the performance of MML approaches the reference PWK in terms of acquired rewards as the learning takes place. Compared to MML, Saliency and Random gazing strategies were not efficient for acquiring information in the current scenario. The implementation of MML reaches its top performance in about 400 epochs. This is better than common reward-based learning methods [12] where the learning time is in the order of 1000 epochs.

Another metric that can be used to evaluate the performance of the methods is the task statistics that is shown in Tab. II. The data in the table indicates how many times on average did the scenario end with the corresponding outcome shown in the first column for the given epoch interval. Two epoch intervals

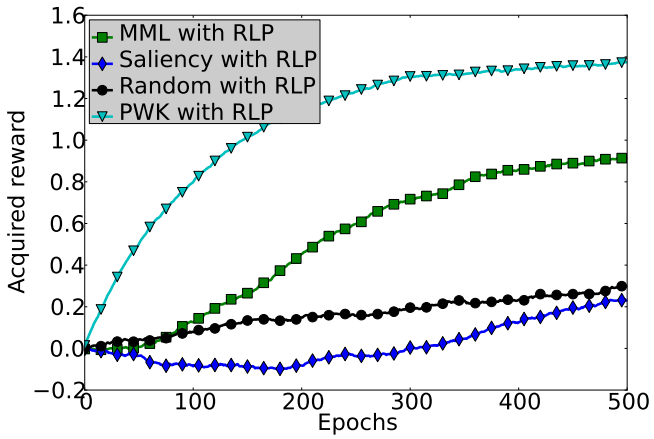


Fig. 5. Acquired rewards for four different methods used in RLP. The data for the learning method is averaged over the last 100 epochs.

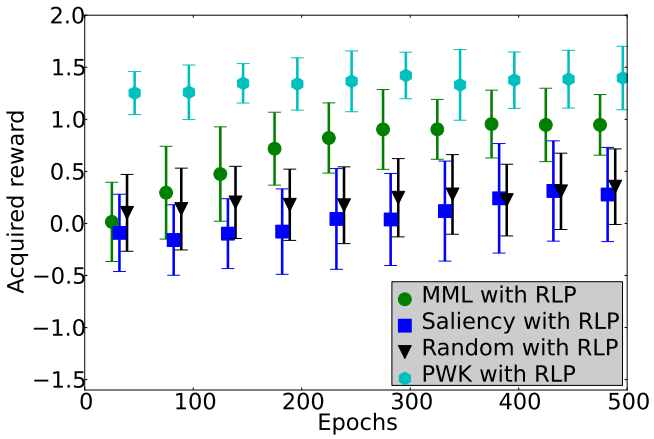


Fig. 6. Mean and standard deviations of acquired rewards for every 50 epochs over all experiments for four different methods used in RLP. Every data point represents means of acquired rewards in preceding 50 epochs. Data points of different methods are shifted slightly away from each other in the horizontal axis for better visualisation.

are shown: between epochs 0 and 100 (i.e. before learning or BL) and between epochs 400 and 500 (i.e. after learning or AL). The data is averaged over all experiments. With learning, our method (shown as MML with RLP) increased its success rate from 49% to 81%. Compared to the reference (PWK), MML reached 87% of the performance of the case where all information about the objects was available to the system all the time. The reductions in the negative outcomes (i.e. time-out and bored) are also indicators of the learning process. The significant decrease in the cases where the interaction partner is bored clearly shows that MML learned to cope with the human interaction task as well. Success rates are lower for the methods Saliency (45%) and Random (54%). Compared to their values before learning, there is slight improvement in the performance of these methods. This is due to the Q-Learning algorithm: as the policy for reaching is improved via learning, the system succeeded to grasp objects before the human interaction task fails in some cases. This indicates that

TABLE II
TASK STATISTICS AVERAGED OVER ALL EXPERIMENTS (IN %). BEFORE LEARNING (BL) COVERS EPOCHS BETWEEN 0-100, AFTER LEARNING (AL) COVERS EPOCHS 400-500.

Scenario out-come	Saliency	Random	PWK	MML with RLP	MML with PP
Success (BL)	31.3	46.1	91.8	48.6	44.3
Success (AL)	45.4	53.8	93.4	81.2	67.5
Time-out (BL)	1.1	11.8	8.1	20.8	20.4
Time-out (AL)	0.6	9.4	6.5	14.9	25.6
Bored (BL)	67.6	42.0	0.0	30.4	35.2
Bored (AL)	53.9	36.6	0.0	3.7	6.7

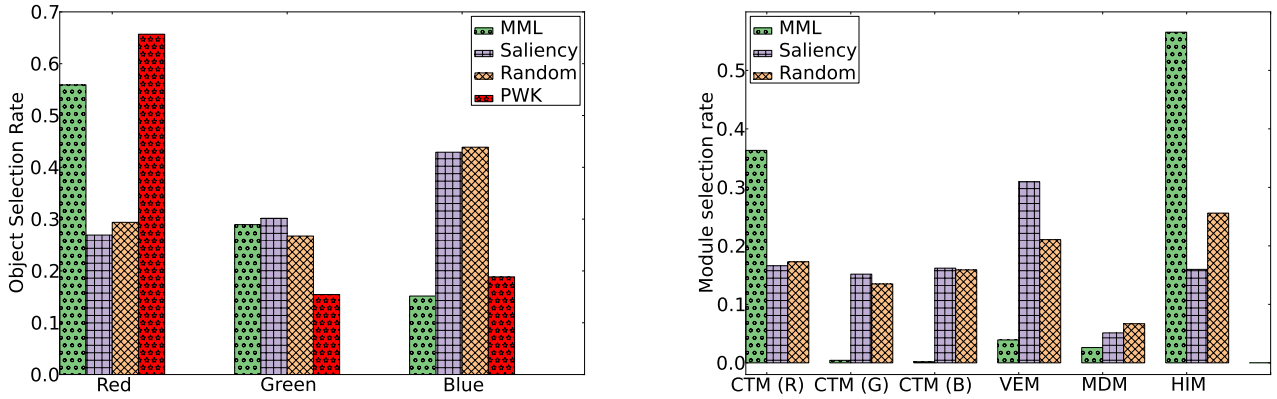
those methods can still provide information for the reaching task up to some degree. However, since there is no inherent support for human interaction task the system could not cope with multi-tasking.

C. Support for the Reaching Task

A convenient way of analysing the support of the compared methods to the reaching task could be to evaluate the learned task policies under the influence of each methods. However, due to the influence of varying environmental parameters to the scenario it is difficult to derive a baseline policy and compare the learned ones with that. The optimum policy is usually not as straightforward as 'select the object yielding the maximum reward' and differs from one experiment settings to another. For example, since objects have different dynamic properties it can be more rewarding in the long term to select a less rewarding but also less dynamic object. However, this also depends on the hand velocity of the robot and the boredom rate. Therefore, a quantitative assessment of the system behaviour can only be made by evaluating the experiments individually.

Fig. 7 and Fig. 8 show results from two representative experiments. In experiment no. 14 (Fig.7) red, green and blue objects yielded rewards of 1.0, 0.02 and 0.6 and had movement probabilities of 0.5, 0.48 and 0.28 respectively. When used with MML the resulting object selection policy learned by the Q-Learning algorithm most frequently selected the red object that was the most rewarding (Fig. 7(a)). The blue object that was less rewarding but also less mobile was not preferred in this case. This is because the hand velocity of the system was able to cope with the speed of the red object. Also, a similar strategy was learned in the case of the perfect world knowledge. The gazing strategy learned by MML is also consistent with the learned object selection policy: among the three colour tracking modules, the one appointed to red was selected dominantly (Fig. 7(b)).

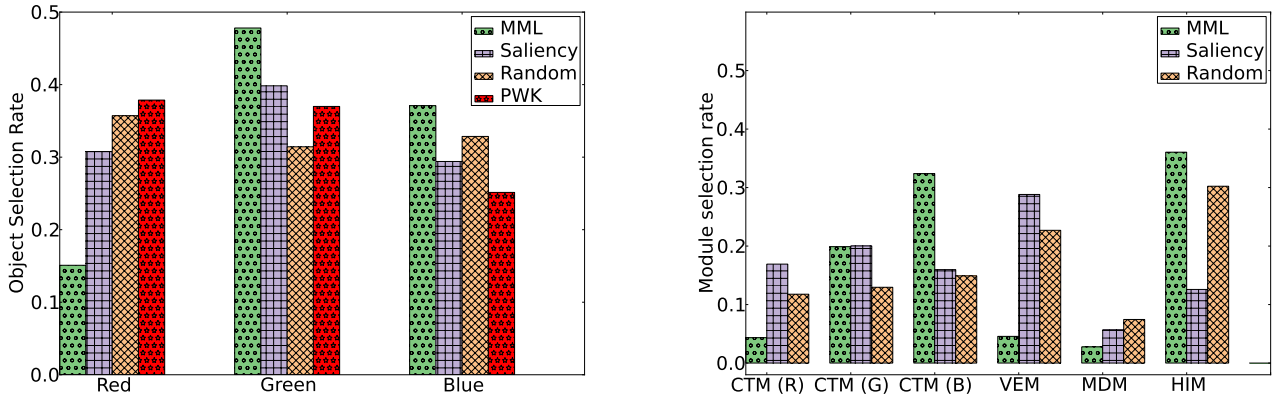
In experiment no. 9 the system with MML selected the green object most often with the blue and red objects following in that order (Fig.8(a)). This is the reverse order of base rewards yielded by those objects (base rewards are 0.51, 0.88 and 1.0 for green, blue and red objects respectively) but proportional to the stability of the objects (movement probabilities are 0.7, 0.76, 0.88 for green, blue and red objects respectively). Probably the system was not able to cope with more dynamic objects since the maximum hand velocity was lower compared to experiment no. 14. This is clearly an object selection policy where a less rewarding but more predictable object was



(a) Object selection rates (after learning)

(b) Module selection rates (after learning)

Fig. 7. Experiment no. 14: Module selection rates and object reaching rates, averaged over the last 100 epochs (i.e. after learning). The experiment settings: r_b is 1.0, 0.02, 0.6 and p_m is 0.5, 0.48, 0.28 for the red, green and blue objects respectively, $\tau_b = 0.18$, $v_{max,x} = 15.75$ deg/sec, $v_{max,y} = 39.26$ deg/sec.



(a) Object selection rates (after learning)

(b) Module selection rates (after learning)

Fig. 8. Experiment no. 9: Module selection rates and object reaching rates, averaged over the last 100 epochs (i.e. after learning). The experiment settings: r_b is 1.0, 0.51, 0.88 and p_m is 0.88, 0.7, 0.76 for the red, green and blue objects respectively, $\tau_b = 0.26$, $v_{max,x} = 12.13$ deg/sec, $v_{max,y} = 37.29$ deg/sec.

favoured. On the other hand, although the green object was selected more often than the blue one, the gazing strategy learned by MML selected CTM-B more often than CTM-G (Fig.8(b)). This is probably because the blue object was more dynamic than the green one and needed more updates for an efficient information acquisition.

The object selection policies learned by the systems using the methods Saliency and Random differ from the policies learned in MML case in both experiments. This signifies that the information acquisition policy has an effect on the learning process of the policy. The learned object selection policies supported by these methods can be sub-optimal as in experiment 14. On the other hand, MML can adapt its parameters in accordance with the dynamics of the environment and can support the policy learning process efficiently.

D. Support for the Human Interaction Task

The human interaction task is less constrained by the experimental setup. The major influence on this task comes from the boredom rate. As the boredom rate increases the partner becomes more impatient. This may lead to an expectation that the selection rate of the HIM should be proportional to the boredom rate. Fig. 9 shows the relation between the HIM selection rate and the boredom rate for all experiments. Linear regression on the data shows the HIM selection rate has tendency to increase with increasing boredom rate for MML. Deviations from the tendency is probably because of the constraints from the reaching task (e.g. in more dynamic scenes the objects need more frequent updates, leaving the HIM less time to stay active).

E. Emerging Gazing Behaviours

It was previously shown in Fig. 7(b), 8(b) that MML can develop a gaze strategy where selection of colour tracking modules are in accordance with dynamics of the environment. Similarly MML learned to use the Human Interaction Module (HIM) in order to accomplish the interaction task and this was reflected in the module selection rates of the HIM. The motion

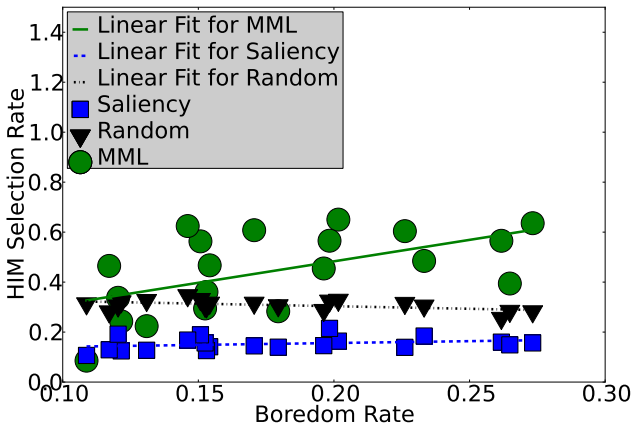


Fig. 9. Selection rate of HIM with respect to changing boredom rate in 20 experiments.

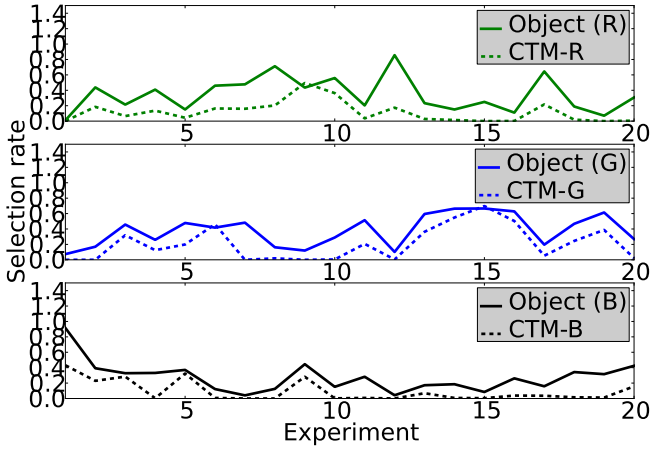


Fig. 10. Average module selection rates (solid lines) and average object selection rates (dashed lines) for the last 100 epochs (i.e. after learning) in RLP. R, G and B stands for red, green and blue respectively.

detection module (MDM) was not favoured by MML in both experiments. This is probably because it delivers ambiguous information in cases where multiple objects are in motion. The visual exploration module (VEM) was also not used by MML in both experiments as it may not deliver useful information every time. These examples demonstrate how the MML discards modules delivering ineffective information and selects the task relevant ones for efficiency. It is natural to think that the MDM is a vital information source since the objects are dynamic in the environment. It can be highly prioritised in a manually designed framework. However, better strategies than intuitive ones can be learned with the proposed framework.

An analysis of the relation between the object selection policy for reaching and gazing strategy reveals the co-development of both mechanisms. Fig. 10 shows average object selection rates and CTM selection rates after learning (i.e. in the last 100 epochs) in each experiment for the system with MML. The results show correlations between the colour of the selected object and the CTM for the same colour in selection ratings. The correlation coefficients[1] are computed as 0.67, 0.85 and 0.80 for the red, green and blue cases respectively. A similar correlation is not found for the systems using saliency method and random gazing method. The correlation coefficients are 0.25, 0.34 and 0.11 in saliency and 0.14, 0.44 and 0.20 in random gazing for the red, green and blue colours respectively. The results indicate that a basic form of eye-hand coordination may have formed by using MML even though the underlying control mechanisms are fairly separated.

F. Individual Contributions of Learning Mechanisms

In order to evaluate the individual contribution of the MML mechanism to the overall performance of the system a second instance of the system is realised. In this instance the reinforcement learning policy (RLP) for object selection

[1]Normalised cross correlation is used for the computation of all correlation coefficients in this work.

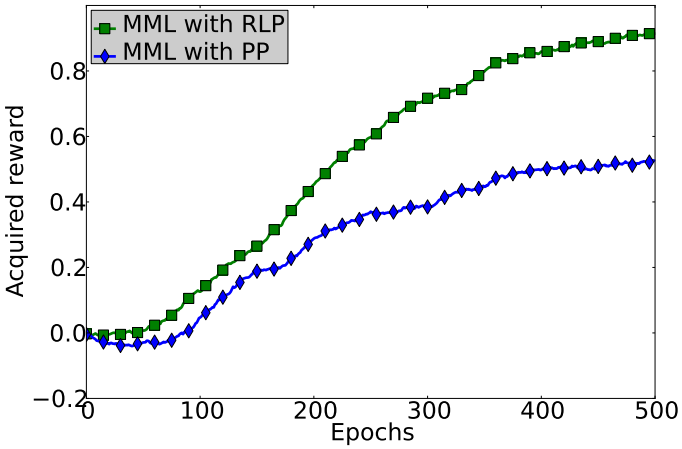


Fig. 11. Acquired rewards in cases RLP and PP with learning and no-learning settings. The data is averaged over all experiments and smoothed by averaging over the last 100 epochs.

is replaced with a programmed policy (PP). In this second case the system chooses a random object to reach at every time step. In summary, in the case of MML with RLP the system employs two learning mechanisms (MML and reaching policy learning) whereas in the case of MML with PP the system employs one learning mechanism (MML). Fig. 11 shows the acquired rewards in the experiments conducted with each instance of the system. The data is averaged over all experiments and smoothed using Eq. 8 with $\epsilon_s = 100$. The system using two learning mechanisms achieved a superior performance compared to the system that employs only one learning mechanism in terms of acquired rewards.

Task statistics for the case of programmed policy is given in Tab. II labelled as MML with PP. Compared to the case of reinforcement learning policy indicated as MML with RLP, both systems have similar initial performance statistics (i.e. success rate, boredom and time-out before learning). The success rate increases 68% for MML with PP and to 81% for

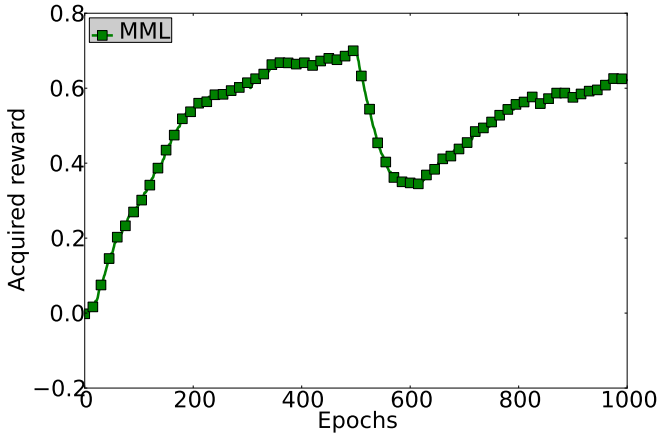


Fig. 12. Acquired rewards in the adaptation experiments conducted with the system using MML. The data for the learning method is averaged over the last 100 epochs.

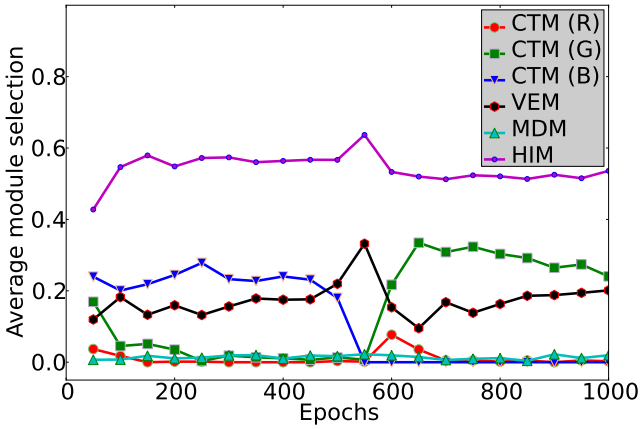


Fig. 13. Average number of selections for each module averaged over bins of 50 epochs using MML in experiment no. 2. The settings: r_b is 0.26, 0.88, 1.0 and p_m is 0.73, 0.02, 0.22 for the red, green and blue objects respectively, $\tau_b = 0.13$, $v_{max,x} = 19.33$ deg/sec., $v_{max,y} = 39.09$ deg/sec.

MML with RLP. This shows that the reinforcement learning policy is superior to the programmed one for the reaching task. Having that in mind, if we compare the post-learning success rates of the case MML with PP (68%) with Saliency (45%) and Random (54%) strategies we see that the system with MML still performs better than the other two methods even if it is equipped with an inferior algorithm for one of the task policies (programmed policy). It should be noted that Saliency and Random methods are used in systems with the reinforcement learning policy. This result shows that the MML framework can boost the overall performance of the system even if the system employs an inferior policy.

G. Adaptation to Changes

It is already shown that the framework can learn efficient gazing strategies in various scenario and environment conditions. To further investigate the flexibility of the proposed framework against perceptual changes in the environment new experiments are conducted (20 in total). In these experiments after 500 epochs, at which point a gazing strategy and a task policy are learned, one randomly selected module amongst the following five is turned off: CTMs for red, green and blue, VEM, MDM. This simulates the case where the stopped module is broken down and cannot deliver any information. Fig. 12 shows that the performance of the system reduces after the change. However, it can recover adapting its gazing strategy. The final value of average acquired reward after the adaptation (i.e. after 1000 epochs) is slightly lower than the value reached after the initial learning process (i.e. after 500 epochs). This is due to the fact that the adaptation of the Q-Learning algorithm used for the object selection policy was not sufficient in some of the experiments. The adaptation process is apparent in Fig. 13 where module selection rates during 1000 epochs from a representative experiment (experiment no. 2) is shown. In the beginning, the system learned to attend the most rewarding blue object by selecting CTM-B. The selection rate of HIM was also high due to the human interaction task. After 500 epochs CTM-B was shut down. The attempt of MML to replace this module can be seen in the variations of module selection rates between epochs 500 and 700. The remaining modules were not able to deliver information about the blue object reliably: the information from MDM was redundant since all objects are moving and points suggested by VEM for gaze-shifts may lie in irrelevant areas in the scene. Thus, the system began to select and attend to the green object that was the second most rewarding by selecting CTM-G. The selection rate of HIM goes back to the level it was after the first 500 epochs, after the adaptation happens. These results indicate that the MML mechanism bestows robustness to the system. However, such a flexibility is also limited by the capabilities of the high-level learning mechanisms in the system. The MML framework can address the changes in the perceptual elements of the scenario or environment. The overall flexibility of the system to changes of a larger extent may vary depending on which algorithm is used for other learning problems in the system.

VI. Conclusion

We propose a reward-based learning framework for information acquisition in resource constrained modular systems performing multi-task scenarios. The scenario applied in this work required two tasks to be executed in parallel: selecting and reaching one of the three objects moving on a table and keeping interaction with a human. The scenario seems rather simple, but it is at a level of complexity that an optimum policy is difficult to be derived manually. This is due to the fact that a number of environmental parameters are varied. In doing so, a number of instances of different environments are generated. The system has no prior knowledge about the environment (e.g. reward distribution of objects). Hence, a manual design of the module selection process may work in some of these instances but will fail in most. We show that the framework can learn efficient gazing strategies for the scenario. This is demonstrated both in terms of acquired rewards in time and the success rate of the system executing the tasks that the

scenario deploys.

Learning gazing strategies as our approach suggests introduces felxibility to the system, making it possible to cope with changes in the scenario or environment. This is demonstrated in two aspects. First, the system is able to learn efficient information gathering settings for various scenario and environmental conditions. Second, in cases of module malfunctions during the operation MML can find alternative solutions. However, the overall flexibility of the system to such changes is also limited by the other local learning mechanisms present in the system.

It is also shown individually how each of the two tasks are supported by the MML. One of the main topics we are interested in is whether the proposed information acquisition framework is able to support other higher level learning mechanisms. To investigate this a RL mechanism is used in the system to learn an object selection policy for the reaching task. It is shown that the framework can distinguish the modules delivering task relevant information and learn how to schedule them. It is also shown in general that a form of eye-hand coordination is formed even though the control mechanisms for information acquisition (i.e. gazing) and task execution are fairly separated. The relevant module for the human interaction task is also found in every experiment even though no prior information about this task was given to the system. The system eventually managed to utilise this acquired knowledge about the tasks in module scheduling.

For the applied scenario learning takes about 400 epochs which is a fairly short time for learning approaches using rewards (e.g. RL [12]). One epoch takes about 10 seconds on average in the simulations. Hence, the whole learning process takes about an hour to complete. This may be seem like a long time for a human-robot interaction task but it should be noted that through the learning process the system gains re-usable knowledge about the environment shaped by the scenario. From a developmental perspective the learning time is reasonable.

In essence, the framework shapes a bottom-up information acquisition process with the requirements from high-level learning and task execution processes. It can be argued that this task-oriented character of the framework could constrain the reactive capacities of the system. The information acquisition behaviour is governed by the parameters of the module management mechanism. In the absence of tasks or when sudden changes occur in the environment these parameters can be set to pre-defined values that would generate the default reactive behaviour, so the reactive capacities of the system will be retained. Detecting changes can be done using a performance based metric (e.g. average of the recently acquired rewards). Such a method was successfully implemented in a sensory-motor learning task to regulate the learning process [33].

Finally, the proposed framework can facilitate the design process of modular systems that needs arbitration and scheduling of motor commands from various information acquisition modules. It requires only a coarse feedback of the actions the system takes (i.e. rewards). Therefore, employment of the framework requires minimal effort to formalise tasks for the system. Moreover, implementing task policies (designed or learned) does not have to deal with the information acquisition process since these two mechanisms are divided. Using this feature the system can rapidly be extended with additional task policies and utilised for different purposes.

ACKNOWLEDGEMENTS

The work described was supported by the Honda Research Institute Europe.

REFERENCES

[1] M. F. Land and B. W. Tatler. *Looking and Acting*. Oxford University Press, 2009.

[2] R. M. Foerster, E. Carbone, H. Koesling, and W. X. Schneider. Saccadic eye movements in the dark while performing an automatized sequential high-speed sensorimotor task. *Journal of Vision*, 12(2), 2012.

[3] J. A. Fodor. *The Modularity of Mind*. The MIT Press, April 1983.

[4] C. Karaoguz, T. Rodemann, and B. Wrede. Optimisation of gaze movement for multitasking using rewards. In *Proceedings of Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[5] R. Beira, M. Lopes, M. Praga, J. Santos-Victor, A. Bernardino, G. Metta, F. Becchi, and R. Saltaren. Design of the robot-cub (iCub) head. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 94–100, June 2006.

[6] V. Tikhanoff, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, and A. Cangelosi. An open source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator. In *Proc. of the Workshop on Performance Metrics for Intelligent Systems, National Institute of Standards and Technology, Washington DC, USA*, August 2008.

[7] R. A. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14 – 23, March 1986.

[8] R. A. Brooks. Intelligence without representation. *Artif. Intell.*, 47(1-3):139–159, February 1991.

[9] E. B. Pacis, Hobart R. Everett, N. Farrington, and D. J. Bruemmer. Enhancing functionality and autonomy in man-portable robots. In *Defense & Security Symposium 2004*, volume 5422, pages 355+. The International Society for Optical Engineering., September 2004.

[10] J. Simpson, C. L. Jacobsen, and M. C. Jadud. Mobile Robot Control: The Subsumption Architecture and occam-pi. In Frederick R. M. Barnes, Jon M. Kerridge, and Peter H. Welch, editors, *Communicating Process Architectures 2006*, pages 225–236. IOS Press, sep 2006.

[11] H. Nakashima and I. Noda. Dynamic subsumption architecture for programming intelligent agents. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 190–197. IEEE, July 1998.

[12] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[13] C. Kwok and D. Fox. Reinforcement learning for sensing strategies. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3158–3163. IEEE, 2004.

[14] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, Nov 1998.

[15] M. Hülse, S. McBride, J. Law, and M. Lee. Integration of active vision and reaching from a developmental robotics perspective. *IEEE Trans. on Auton. Ment. Dev.*, 2(4):355–367, December 2010.

[16] V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 45(2):205–231, January 2005.

[17] D. Ognibene, G. Pezzulo, and G. Baldassarre. How can bottom-up information shape learning of top-down attention-control skills? In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pages 231–237. IEEE, August 2010.

[18] A. J. Palomino, R. Marfil, J. P. Bandera, and A. Bandera. A novel biologically inspired attention mechanism for a social robot. *EURASIP J. Adv. Signal Process*, 2011:4:1–4:10, January 2011.

[19] F. Orabona, G. Metta, and G. Sandini. Attention in cognitive systems: theories and systems from an interdisciplinary viewpoint. chapter A Proto-object Based Visual Attention Model, pages 198–215. Springer-Verlag, Berlin, Heidelberg, 2008.

[20] B. Bolder, H. Brandl, M. Heracles, H. Janssen, I. Mikhailova, J. Schmüdderich, and C. Goerick. Expectation-driven autonomous learning and interaction system. In *IEEE-RAS International Conference on Humanoid Robots*, 2008.

[21] C. Goerick, J. Schmüdderich, B. Bolder, H. Janssen, M. Gienger, A. Bendig, M. Heckmann, T. Rodemann, H. Brandl, X. Domont, and I. Mikhailova. Interactive online multimodal association for internal concept building in humanoids. In *IEEE-RAS International Conference on Humanoids 2009*. IEEE, 2009.
[22] N. Sprague, D. Ballard, and A. Robinson. Modeling embodied visual behaviors. *ACM Trans. Appl. Percept.*, 4(2):11+, 2007.
[23] D. Busquets, R. L. de Màntaras, R. L. Opez De, M. Antaras, C. Sierra, and T. G. Dietterich. Reinforcement learning for landmark-based robot navigation. In *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS*, pages 841–843, 2002.
[24] M.A. Arbib. *Schema Theory*, pages 1427–1443. Wiley, second edition, 1992.
[25] G. Pezzulo and G. Calvi. A schema based model of the praying mantis. In *Proceedings of the Ninth International Conference on Simulation of Adaptive Behaviour, volume LNAI 4095*, pages 211–223. Springer Verlag, 2006.
[26] A. Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J. Tsotsos, and E. Körner. Active 3D object localization using a humanoid robot. In *IEEE Transactions on Robotics*. IEEE, 2011.
[27] D. Vernon, G. Metta, and G. Sandini. The iCub cognitive architecture: Interactive development in a humanoid robot. In *Development and Learning, 2007. IEEE 6th International Conference on*, pages 122–127. IEEE, July 2007.
[28] T. J. Prescott, P. Redgrave, and K. Gurney. Layered control architectures in robots and vertebrates. *Adaptive Behavior*, 7:99–127, 1998.
[29] H. Jacobsson, N. Hawes, G. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. *Proceedings of the 3rd international conference on Human robot interaction HRI 08*, page 81, 2008.
[30] Shun-Ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27(2):77–87, June 1977.
[31] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, new edition, June 2002.
[32] G. G. Turrigiano and S. B. Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews. Neuroscience*, 5(2):97–107, February 2004.
[33] C. Karaoguz, M. Dunn, T. Rodemann, and C. Goerick. Online adaptation of gaze fixation for a stereo-vergence system with foveated vision. In *Proceedings of International Conference on Advanced Robotics 2009*, pages 1–6, July 2009.
[34] R. Yan, T. Rodemann, and B. Wrede. Learning of audiovisual integration. In *Proceedings of ICDL/EPIROB 2011*, Frankfurt, Germany, 2011. Springer.
[35] C. Karaoguz, A. Dankers, T. Rodemann, and M. Dunn. An analysis of depth estimation within interaction range. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3207–3212. IEEE, October 2010.
[36] A.-L. Vollmer. *Measurement and analysis of interactive behavior in tutoring action with children and robots*. PhD thesis, 2011. University of Bielefeld.
[37] Y. Nagai, A. Nakatani, and M. Asada. How a robot's attention shapes the way people teach. In *Proceedings of the 10th International Conference on Epigenetic Robotics*, pages 81–88, November 2010.



Cem Karaoguz is a research associate pursuing his Ph.D. degree at the Research Institute for Cognition and Robotics (CoR-Lab) in Bielefeld University and a guest scientist at the Honda Research Institute Europe (HRI-EU) since 2009. He received his B.Sc. degree in Electrical Engineering at Istanbul Technical University in 2005 and M.Sc. degree in Electrical Engineering/Automation and Control at Darmstadt University of Applied Sciences in 2007. He joined the HRI-EU in 2006 for his internship and master thesis. Afterwards, he continued working at the HRI-EU as a scientist for two years. His research interests are machine learning, biologically inspired robots, systems engineering and learning in large scale systems.



Tobias Rodemann studied physics and neuroinformatics at the Ruhr Universität Bochum, Germany, and received his Dipl.-Phys. degree from the Universität Bochum in 1998 and a Ph.D. degree from the Technische Universität Bielefeld, Germany in 2003. Since 1998 he is working at the Honda Research Institute Europe, Offenbach, Germany. Previous research fields were evolutionary algorithms, biologically-inspired vision systems, information processing with spiking neurons and learning of sensory-motor maps. Since 2003 he is working as a senior scientist on sound localization, auditory scene analysis and audio-visual interaction.



Britta Wrede is the head of the research group Hybrid Society within the Research Institute for Cognition and Robotics (CoR-Lab) at Bielefeld University. She received her Masters degree in Computational Linguistics and the Ph.D. degree (Dr.-Ing.) in computer science from Bielefeld University in 1999 and 2002, respectively. From 2002 till 2003 she pursued a PostDoc program of the DAAD at the speech group of the International Computer Science Institute (ICSI) in Berkeley, USA. In 2003 she rejoined the Applied Informatics Group at Bielefeld University and was involved in several EU and national (DFG, BMBF) projects. Since 2008 she is heading her own research group at the CoR-Lab.



Christian Goerick received the Diploma degree in electrical engineering and the Ph.D. degree in electrical engineering and information processing from Ruhr-Universitt Bochum, Bochum, Germany. During his time in Bochum, he was Research Assistant, Doctoral Worker, Project Leader, and Lecturer in the Institute for Neural Computation and Chair for Theoretical Biology. The research was concerned with biologically motivated computer vision for autonomous systems and learning theory of neural networks. He is currently a Chief Scientist with the Honda Research Institute Europe GmbH, Offenbach, Germany. His research interests are behaviour-based vision, audition, behaviour generation, cognitive robotics, advanced driver assistance systems, system architecture, and hard- and software environments.