Coupling of mental concepts to a reactive layer: incremental approach in system design

Inna Mikhailova **Martin Heracles** **Bram Bolder** **Herbert Janssen**
Holger Brandl **Jens Schmüdderich** **Christian Goerick**
Honda Research Institute Europe GmbH,
Carl-Legien-Strasse 30,
63073 Offenbach / Main, Germany
inna.mikhailova@honda-ri.de

Abstract

The design of a system that bootstraps an open-ended development is one of the most intriguing questions in Developmental Robotics. Inspired by evolution we propose an incremental design. We start with a reactive layer that provides task-unspecific interaction with the environment. We extend this initial layer by a layer of multi-modal expectation generation. The two layers are coupled by means of an active resolution of expectation mismatches. Such an extension allows for the transition from reactive behavior to hypothesis testing and goal-directed behavior. The expectations can also be used as a teaching signal. The proposed architecture is validated on the example of multi-modal learning and evaluation of auditory labels tested on the humanoid robot ASIMO.

1 Introduction

In the last years research in Developmental Robotics moved from a "no-predesign" philosophy to a "minimal design" philosophy. The first one aims at emergence of abilities from "tabula rasa". The second one aims at a careful design of a system that can use already existing abilities for open-ended acquisition and integration of new abilities, (Prince et al., 2005). In other words the latest research is looking for a system sufficient for bootstrapping of the development.

In the case of a simple agent, an evolutionary process, e.g. (Schembri et al., 2007), can provide a bootstrapping system suited to learning in its ecological niche. Unfortunately, the interaction between a humanoid robot and its environment has such a high complexity that a purely evolutionary approach can not be used yet. Still, we can take our inspiration from evolution and build our bootstrapping system incrementally. In the incremental design the layers added later to the system do not block the previous layers so that the overall system behavior is more robust. A further advantage of incremental system building is that we are not pressed to implement short-cut solutions to get to our goal. Instead, it is possible to carefully design the interfaces for later extensions.

Here we start from a reactive layer for task-unspecific interaction with the environment implemented earlier (Bolder et al., 2007). On top of this layer we propose to build a layer that allows for an expectation-driven behavior (section 2). In section 3 we concretize the general idea on the example of the system with speech driven expectations. Section 4 provides more details about the mechanism of expectation generation. Finally, in section 5 we analyze our system: which parts require further work of the designer in the sense of incremental building and which parts support further steps of the system itself in the sense of bootstrapping the development.

2 Coupling of reactive and abstraction layers

A robot, which uses internal world-models, runs the risk that the models may be out of sync with the reality. The behavior-based robotics and subsumption control architectures attempt to solve this well-known AI problem by designing simple reactive behaviors that do not need complex world models. This works well for simple scenarios but not for complex ones. It is still an open question how an anticipative layer that uses internal models and a reactive layer can be integrated together in order to efficiently control the behavior (Butz et al., 2007).

2.1 Related work

Some recent approaches in Reinforcement Learning use a reactive layer for the description of the state of the system-environment interaction as well as for execution of plans, e.g. (Hart et al., 2006). These approaches do not switch between reactive and anticipative modes; they are forced to always evaluate

the future reward and to always plan ahead.

The behavior-based approaches provide a possibility for the planning layer to manipulate the action selection in the reactive layer (Ulam and Arkin, 2008). However, the anticipation does not influence the perception that is separated into a 'symbol converter'.

Common to all discussed above approaches is the fact that the extension of the reactive behavior aims directly at planning. However, from the evolutionary perspective, the anticipation may first be used simply to detect an inappropriate behavior. The model described in (Balkenius, 1995) goes in this direction. The focus is set on expectancy learning and the interplay between the expectancy system, the perception system, and the control that does not require extensive planning (e.g. conditioning, habituation, behavior suspension in case of the expectation mismatch). The planning is seen as a next incremental step. Our approach, described below, shows some parallels to this work. One of our original contributions is the active resolution of mismatch situations in a way that has not been proposed before.

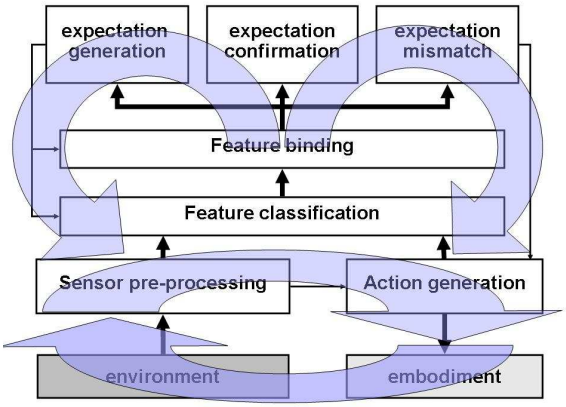

Figure 1: Coupling of mental concepts to a reactive layer. The loop on the lower part shows the reactive layer. The upper part generates and evaluates expectations with help of the associative memory. In this upper part two loops are active: to the left the loop of expectation-based perception and to the right a loop of expectation-based action.

2.2 Abstraction layer

There exist a number of notions: abstraction, semantic knowledge, model, etc., that denote an entity used by non-reactive behavior generation. We use an expression 'mental concept'. It is not used in the sense of a passive world model, but as a potential trigger of an expectation-driven behavior.

We say that the system learns a concept if it can bind different behaviorally relevant features into one entity. The decision to bind the features together into an entity can have different sources. Possible sources are correlation of features in time, in space, and correlation to the same reward. In such cases binding can be represented in form of an associative memory. Another source of binding can be e.g. usage of the same prediction model for some set of features.

The system should not only passively 'perceive' but actively evaluate and refine/relearn the concepts. We turn our reactive system into an active system by generation of expectations. One feature that is bound with other features to a concept generates expectation for the rest. These expectations are compared to the actual features. If there exist an overlap it can be used for disambiguation. If the difference is too high, the mismatch triggers a behavior that can potentially resolve the conflict. A resolving behavior can have different levels of complexity. For example, in order to meet the expectation of visual features the system can simply perform a random search or it can activate the search for the expected feature by modulating the saliency map. Similarly the mismatch in audio channel can either activate a simple request for the human to pronounce a new word or it can trigger the system to pronounce the right word by itself. The mismatch resolving reenacts the features that belong to the concept and brings the system again into the situation where it can re-experience the binding of features to the concept and check its correctness.

In this way our system achieves a coupling between sensing and acting not only on the reactive level but also on the level of expectations generated by mental concepts (see Figure 1). The loose coupling of abstractions to the reactive behavior has several advantages compared to inclusion of an action into the mental concept, as it is usually done with prediction of action outcome (see (Butz et al., 2007) for overview). First, it allows for a sufficient decoupling of the dynamics of the concept learning from the dynamics of the behavior. This leads to both stabilization and transparency of the overall behavior. Second, it allows to link a multitude of behaviors to the same concept. Finally, it provides a possibility to use the concepts as the subject of higher mental functions (attention, communication, planning, memorization) in the sense of symbol detachment (Pezzulo and Castelfranchi, 2007).

Below we compare our approach to other architectures that integrate anticipative and reactive control. Unlike subsumption architectures the abstraction layer not only inhibits the behavior generated on the lower layer, but also modulates the behavior. In difference to the behavior-based approach (Ulam and Arkin, 2008), the top-down influence by expectations is used for 'perception' as well. Further our system can switch from an expectation-driven mode to a reactive mode if it does not observe any features that are bound to the concepts and does not have internal sources for expectation generation

(e.g. planning). Different to the case of reinforcement learning, the expectation can be, but needs not to be generated by a planning routine. We believe that the control reorganization during the child development also uses the expectancy first for simple manipulation of a reactive layer before using it for an extensive planning. Different to (Balkenius, 1995) we use the mismatch in expectation not only to stop the executed behavior but also to resolve the mismatch. This mechanism can be used later for hypothesis testing and refinement of the concepts.

3 System instance: learning and evaluation of speech labels

One possible instantiation of our architecture is a system that generates expectations by using associations of speech labels to behaviorally relevant non-speech feature classes. For example the system learns that humans use the word 'table' for flat surfaces at the height of their waist. This is behaviorally relevant, because the naming of the 'right thing' with a 'right word' can trigger a primary social reward at the early stage of development and serve as a subgoal in later stages (e.g. asking for the table if you need to deposit something). We want to emphasize that we do not reduce the learning of concepts to the learning of words. We consider the binding of a word to behaviorally relevant non-speech features as one possible scenario for learning mental concepts.

We implemented our system on the humanoid robot ASIMO equipped with stereo vision cameras. The auditory signal for audio saliency was recorded by microphones mounted on ASIMO. The auditory signal for the speech recognition was recorded via a headset used by a human. Figure 2 shows the experimental setup while Figure 3 shows the overall system design. The extraction and classification of the visual features as well as the online learning of speech clustering are not in focus of this paper and will be described in detail elsewhere. Focus of this paper is a general system design that allows for the extension of a reactive system by expectation generation and evaluation.

3.1 Reactive layer for bootstrapping interaction

The basic reactive layer comprises saliency-driven gazing and tracking/reaching for proto-objects. The behavior selection is implemented by a competitive dynamics (arbiter) that allows several behaviors to run in parallel. The commands issued by different subsystems have different priorities so that the motion interface can resolve the resting conflicts. The robot's joints are controlled by whole-body-motion algorithms that includes walking. The self-collision avoidance prevents the robot to take dangerous postures. This reactive part of our system is shown on the lower part of Figure 3.

Figure 2: Experimental setup. The humanoid robot ASIMO interacts with a user in a reactive way by approaching and reaching for a proto-object. The user wears a headset for recording the speech signal. The user can teach the robot speech labels which describe behavior-relevant features of interaction: properties of proto-objects or the activity of the robot.

The saliency-driven gazing in spirit of (Itti et al., 1998) is implemented with the help of Dynamic Neural Fields. The saliency uses both audio and visual input. The audio saliency helps to attract the attention of the robot if the human is not yet in the robot's field of view. The motor command issued by this sub-system has the lowest priority and will be suppressed in the conflict resolution in case if the proto-object fixation issues a gazing command.

A proto-object is a coherent region or group of features in the field of view. Three different visual cues enter the system, each with their own short term memory. Depth proto-objects are based on contiguous regions of depth values in a restricted range we call the peripersonal space. This overlaps roughly with the manipulation range of both arms. The second kind of proto-objects are based on object proper motion, i.e. contiguous regions of image regions with similar movement relative to the robot (Schmüdderich et al.,). These proto-objects allow an interaction over a larger range. One can attract the robot for instance by waving. The third kind of proto-objects are based on textured or non-textured planar surfaces. Although the method can extract planar surfaces in arbitrary orientations, we restrict ourselves here to roughly horizontal surfaces. These proto-objects allow the robot to identify behaviorally relevant support surfaces like chairs, tables, and the floor. The proto-objects from the three sources are then merged, i.e. those that probably describe the same entity in the world are merged into one proto-object and assign a unique ID. The complete list of proto-objects is available for all interested subsystems. A second instance of short-term

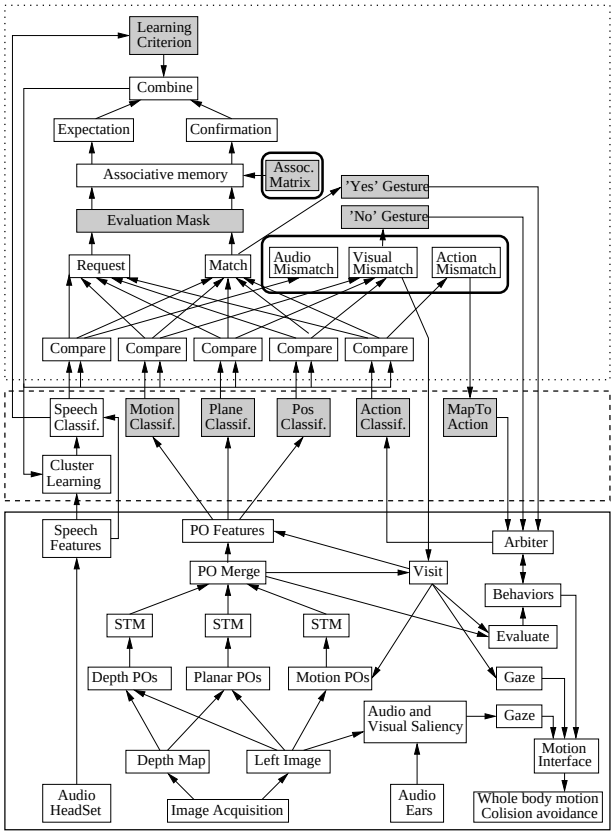

Figure 3: Incremental system design. The solid box on the bottom shows the implementation of the reactive part that comprises saliency-driven gazing and tracking/reaching for proto-objects. The dashed box displays the layer of feature classification. The dotted box shows the abstraction layer of expectation generation based on associative memory. The dark painted modules represent crucial predesign that leads to stabilization of the system. The rounded boxes show the modules that create potential for next steps (see section 5).

memory on the level of merged proto-object is used in order to increase the stability of the perceived 3D location.

A simple proto-object attention mechanism selects one of the currently available proto-objects. Its unique ID, the 'selected-ID' is also made available to the rest of the system. The selection mechanism stays on the same proto-object as long as it is available or the top down influence deselects the current ID. As long as there are proto-objects available (that weren't deselected), a next proto-object is selected based on an arbitrary metric based on status and distance to the robot. For example in the current instance the robot selects the closest proto-object. If no proto-object are available, the selected-ID is set to the value 'invalid'. The selection mechanism allows stable interaction with any single proto-object. The advantage of proto-objects is that we can start the interaction with environment without higher sensory processing (e.g. object recognition).

The reactive layer is implemented as a number of behaviors B_i, $i \in [1 \dots N]$, several of which can be active at any time, plus a competitive selection mechanism (arbiter) in form of a non-linear dynamical system (Bergener et al., 1999, Bolder et al., 2007). The state of the arbiter at any time can be described by the vector of current behavior activation values, i.e. the result of the competitive selection. This vector can be either stored to bias the reactive layer later on or be mapped bidirectionally to action classes such as 'forward' and 'return'.

If no top-down input is present (purely reactive case) the behavior selection is solely based on a fitness value F_i that is provided by each behavior. Top-down input - in our case from the expectation generation system - can act as an additional bias β_i to the fitness of each behavior so that the competitive advantage C_i provided as input to the selection mechanism is a sum $C_i = F_i + \beta_i$. This serves as both a way to trigger certain gestures - nod head, shake head, learning gesture - and a way to push the reactive layer in the direction of a memorized behavior activation state.

Since this external influence to the reactive layer always acts merely as a bias, the reactive system is still fully functional with respect to non-biased interaction behaviors while under the influence of top-down input. Practically this means that ASIMO will e.g. still fixate and reach for objects when a 'return' command was issued and ASIMO is thus walking backwards.

3.2 Extension by an abstraction layer

While a proto-object is stably tracked by the reactive layer, the system extracts the features of the proto-object, the state of behavior activation, and the speech features. In order to stabilize the learning we suppose that all non-speech features can be reliably classified. We accept this assumption for our first version of the system because the experiments on animals show that there exist indeed an ordering in the developmental of different sensory systems (Smith and Gasser, 2005). We use a mask that we call 'evaluation mask' for differentiating between reliable channels and channels to learn. This mask filters the input to the associative memory in such a way that data from a learning channel requests a confirmation, whereas the data from reliable channels sends a confirmation that can be used as a teaching signal. This mechanism will be explained in more detail in the next section.

An attention mechanism further restricts the channels that can generate teaching signals. In our framework attention is implemented in form of expectation for some particular feature channel. This expectation is raised by specific, predesigned user utterances

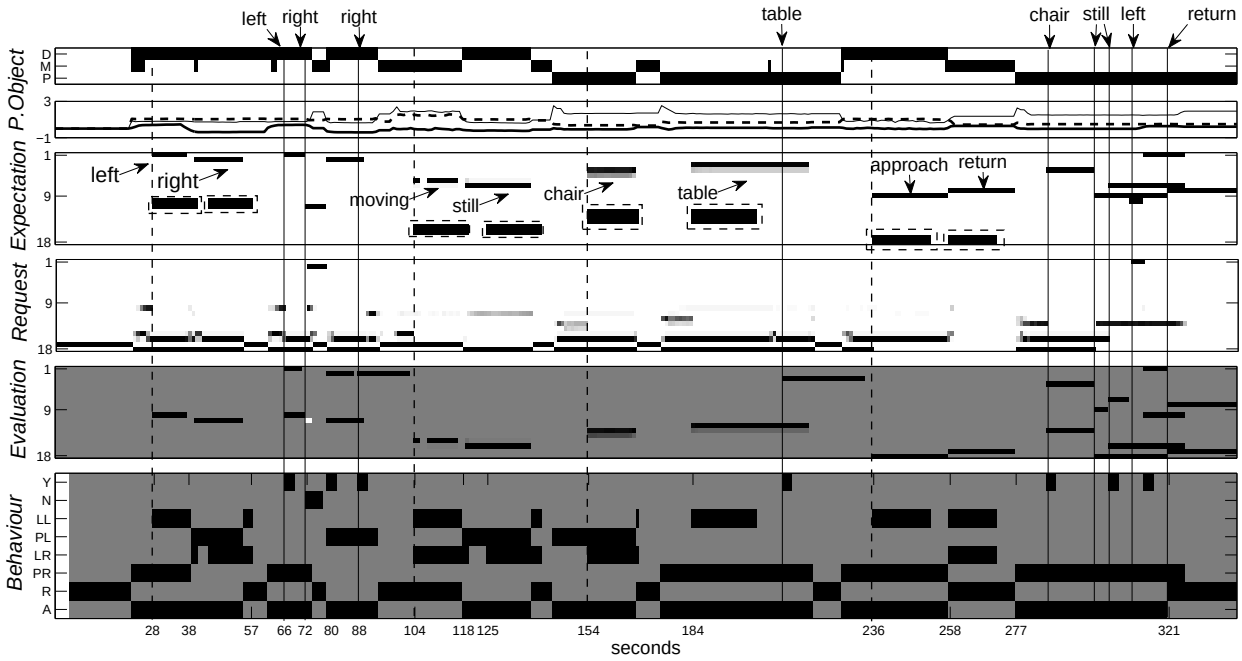

Figure 4: One run of the experiment: speech learning and evaluation. The text in the image shows the word learning sessions, the text on the top - the utterances for evaluation. The upper two plots display the proto object information. The labels stand for the source of proto object: 'D'- Depth, 'M'- Motion, 'P'- Plane; dark color shows high values. The second plot displays the object's position in the cylindric coordinates centered at robot's torso (waist): thick line shows the angle (rad), the dashed line - the hight, and the thin line - the depth. The plot on the very bottom shows the state of the behavior activations: 'Y'-node,'N'-shake, 'LL','LR'-learning gestures with left/right hand, 'PL','PR'-pointing with left/right hand, 'R'-return, 'A'-approach. The behaviors can run in parallel. Note the switch of the right and left hand doing pointing and learning gesture at 38 sec as the object moves from left to right. The middle 3 plots shows the state of the abstraction system. The first 9 values of expectation-, request-, and evaluation-vectors correspond to speech channel. The dashed boxes show the expectations in non-speech channels generated by learning criteria. The corresponding expectations in speech channel is used as a teaching signal. The evaluation show both the expectation match (dark) and mismatch (bright). At the 72 sec the user says 'right' while showing an object on the left. This creates a mismatch (white spot in Evaluation) that triggers head shaking (label 'N' in behavior activation plot) and stops tracking. The request stays active until the robot finds the object on the right (80 sec) and nodes (label 'Y').At the end of the experiment we evaluate some of the learned labels while Asimo is tracking a not yet seen object.

that we call 'learning criteria'. For example if the user says 'learn where this object is', an expectation of activity for any class in the position classifier is generated. Obviously a predesigned link from a known 'learning criterion' utterance to the channel expectation is a strong simplification. Still, this link is nothing but an association and thus it could be learned as well.

The fact that humans use words to name the features is also known to the system and is represented by a predesigned association matrix of the associative memory. It contains a non-zero element at location (i,j) if the feature classes (i) and (j) are associated. In contrast to detailed correlation, associations represent the general information that the features can be bound. With help of this associative memory the system can generate a teaching signal and learn the speech classes. The details of speech processing and learning can be found in (Brandl et al., 2008).

We use online learning, thus the learned classes can immediately be evaluated. By using the associative memory again the speech channel now generates expectations for non-speech-features. For example when the human says 'table' the system expects to see a flat surface at the height of the waist. If the currently tracked object doesn't have the expected features then the expectation mismatch inhibits the reactive tracking of this object. The system switches to the tracking of another object until the expected object properties are seen or a time-out cancels the expectation.

The expectation mismatch in the description of the action executed by the system can directly activate the respective actions via a bias to the competitive behavior selection mechanism. Thus the system can be taught 'command'-like utterances that influence

the action of the system without disabling the autonomy. Without expectations the system continues to interact with its environment in a reactive manner.

In order to make the behavior of the system understandable for the user, the robot communicates the state of expectation. In case of the expectation mismatch it shakes its head ('No' gesture) and in case of the match it nods ('Yes' gesture). The gestures are triggered via a bias vector in a way similar to the associated 'commands'. In the future we will extend the system by means to monitor the human reaction to the Yes/No gestures. Then the active evaluation of the expectation can be used for the refinement or relearning of the corresponding concept.

Figure 4 shows a run of the experiment. A typical learning session looks like follows (28-57 sec. in Figure 4): the user says 'learn where this object is!' (learn criteria) and then says 'left' a few times while presenting the object on the left side of the robot. If the user does not speak for 4 seconds the learning session is considered to be over. Newly learned clusters can be immediately evaluated: The user presents an object on the left side of the robot and says 'left' (66 sec.). Speech class 'left' now raises an expectation for the interaction on the left. This expectation is satisfied here since an object is shown on the left side, so the robot nods (Yes). In contrast, when the user presents an object on the left side but says 'right' (72 sec.), the expectation is not satisfied, so the robot shakes its head (No). Expectation is not only evaluated but it also influences the behavior: the robot stops tracking the object. The expectation continues to influence the behavior for a limited time (5 seconds). If an object on the right is presented within this time the robot nods (80 sec. in Figure 4).

4 Expectation generation and evaluation

In this section we discuss details of the expectation generation and evaluation that takes place in the bottom-up/top-down loop via the associative memory (see Figure 1).

As we already mentioned, all classifiers F_c for feature X_c, where c is the index of the feature channel, are treated in the same way. The output of every classifier at time step t is a vector of values $f_c^i(X_c)$ that represent the likelihood that X_c belongs to a class i. For motion the f_c^i are binary values: 1.0 if the proto-object is generated from a motion channel and 0.0 otherwise. For plane and position classification we use a population code: $f_c^i = d(X_c, X_c^i)$, where X_c^i is the center of the i-th cluster and $d(x, y)$ is a metric, e.g. $d(x, y) = \lambda \exp(-\frac{\|x-y\|^2}{\sigma^2})$. The clusters for non-speech features are predefined. The speech classifier is described in (Brandl et al., 2008). Every classifier output is the bottom-up input $\vec{I}_c^{bu}(t)$ to a module which we call 'compare' (see Figure 3).

If the expectation confirms the classification, then the bottom-up activity is memorized as the internal state $\vec{S}_c(t)$ of the compare module and is propagated to its output 'match' $\vec{M}_c(t)$.

If the classification contradicts the expectation, the module resets its internal state and sends the output 'mismatch' $\vec{H}_c(t)$.

If the expectations for the particular feature are negligibly small and the bottom-up input was not yet confirmed by the expectation so that the internal state is negligibly small as well, the compare module sends the 'request' output $\vec{R}_c(t)$.

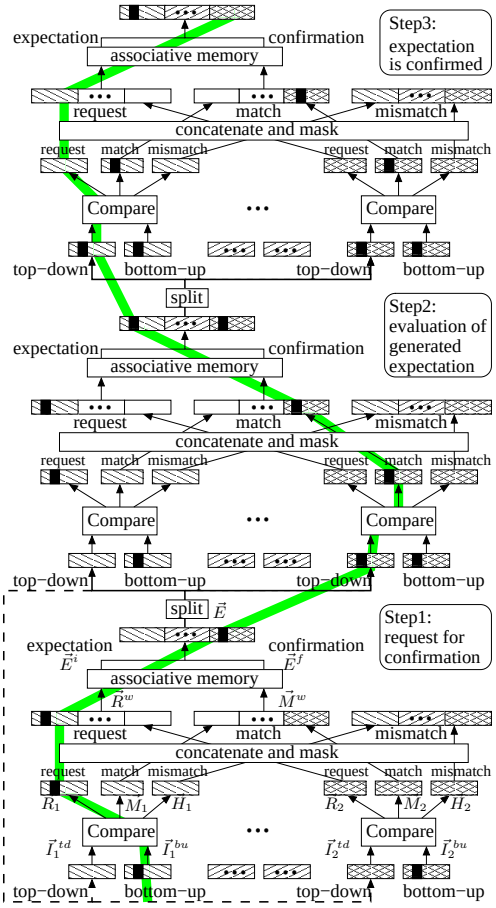

Figure 5: Unfolded example of 3 successive steps of the expectation generation and evaluation loop (the case of expectation match). The dashed line shows the feedback pathway if not unfolded. White spaces of request and match vectors show which parts are inhibited by the evaluation mask. For the sake of simplicity we show only two feature channels: one that generates the expectation and another one that can confirm the first one via the associative memory. The grey line in the background shows the progress of changes through the loop.

Processing of the 'mismatch' output is feature-specific. In contrast, the processing of the 'match'

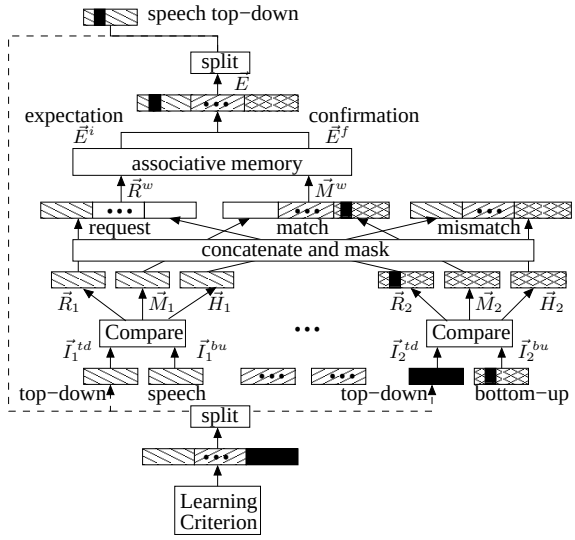

Figure 6: Generation of the teaching signal for the online-speech learning. At the bottom the expectation for one feature channel (e.g. position) is generated from a specific utterance (e.g. 'learn where this object is'). This top-down expectation is propagated through the loop in the standard way (see Figure 5) and generates the teaching signal for the speech shown at the top.

and 'request' outputs is not feature specific. The outputs of all channels are concatenated to a common 'match' $\vec{M}(t)$ and a common 'request' $\vec{R}(t)$. These two vectors are multiplied componentwise with the evaluation mask $\vec{W}$ that defines which channel should send a request and which channel a confirmation: $\vec{M}^w(t) = \vec{W} \bullet \vec{M}(t)$, $\vec{R}^w(t) = \vec{W} \bullet \vec{R}(t)$. The associations $\vec{E}^f$ and $\vec{E}^i$ to these vectors are generated by multiplication with the association matrix $\mathbf{A}$: $\vec{E}^f(t) = \mathbf{A}\,\vec{M}^w(t)$ and $\vec{E}^i(t) = \mathbf{A}\,\vec{R}^w(t)$.

We recall that the 'match' vector contains the features that were expected. Thus the association $\vec{E}^f$ to match can serves as a confirmation to not yet expected features. In terms of predictive models this is a 'forward model', whereas the association $\vec{E}^i$ to the 'request' is analog to an 'inverse model'. It shows which features can generate the confirmation for a request. These two outputs of the associative memory are combined with the expectation $\vec{E}^l$ generated by the attention system (learning criterion) and sent back through the loop as an expectation vector $\vec{E}(t) = \vec{E}^f(t) + \vec{E}^i(t) + \vec{E}^l(t)$. This vector is then split according to the used feature channel c (we denote this operation by $[\,]_c$) and every 'compare' module receives its corresponding part as top-down expectation $\vec{I}_c^{td}(t + \Delta t) = [\vec{E}(t)]_c$. This step closes the loop of expectation-based perception. In Figure 5 we schematically unfold an example of 3 successive steps of the loop. In Figure 6 we show how the general expectation mechanism can be used to generate a teaching signal.

5 Analysis: distribution of learning and predesign

Our long-term goal is the design of a bootstrapping system for open-ended autonomous development. For this reason we would like to carefully analyze which parts of our system can be considered necessary for bootstrapping; which parts help the designer to make further steps in incremental building; and which parts would enable the system to make further developmental steps.

Many approaches in Developmental Robotics learn models or concepts as correlations directly on the level of the features using only the statistics of the data. The problem for these bottom-up approaches is how to find the right level of generalization and how to update the models (stability-plasticity problem). As a side effect of these difficulties the research often stops at the level of correlation learning not coupling it to behavior at all or using it only for simple reactive behavior.

We do not learn in a bottom-up way, instead we stabilize the learning with help of a top-down teaching signal. As we explained in section 3, four predesigned parts in our system contribute to stabilization of learning at the level of a feature classifier: predesigned feature classifiers, an evaluation mask, the way how a user generates attention, and the associative memory.

The fixed classifiers provide initial hypotheses about possible clustering. The autonomous generation of hypotheses is necessary in order to learn more concepts, but it does not directly contribute to better learning behavior. We already have tested an unsupervised clustering instead of predesigned clusters. In contrast, the refining of the initial classifier hypothesis is crucial for the grounding of the concepts and it has to be considered together with the problem of concept representation by a fixed associative memory.

Similar arguments are valid for the predesign of learning criteria as a source of attention. Since we use a well defined interface in form of expectations, the source of these expectations can easily be replaced. We have preliminary results on how to raise the expectation by monitoring repeated changes in a particular feature channel. Replacement of predesign by emergent solutions would increase the number of things that can be taught to the system, without direct improvement of the learning behavior.

We believe that crucial steps towards more complex learning behavior and a truly complete, situated system are an integration of more complex behavior for resolving expectation mismatches and an integration of rewards as signals for building an associative memory.

6 Summary

We presented a way in which a reactive layer can be extended by an abstraction layer that generates expectations. The expectations can be used both as teaching signal and for generation of expectation-driven behavior. The expectations that are not met by sensory input activate mismatch resolution: a behavior that stops either if the expectation is met (with a positive answer) or after a time-out (with a negative answer). We see this behavior as a first step towards hypothesis testing and goal-directed behavior. We have shown the feasibility of our approach in a system working with the humanoid robot ASIMO. The system was tested extensively during interaction with different users and shows a stable and reliable performance.

In the framework of open-ended development we set a high value on task/scenario independent solutions, flexible interfaces, and the possibility of growing and scaling up. Below we summarize the design features that support the incremental building of an intelligent system:

- a generic concept of proto-objects allows for bootstrapping the development of the system with stable, task-unspecific reactive behavior;
- all feature channels are handled in a similar way for easy integration of additional channels;
- a mask decides which feature channel generates an expectation, it can be replaced by an internally generated mask without a system redesign;
- there is no explicitly coded teaching signal, instead the mechanism of expectation generation is used.

In summary, both the coupling of mental abstractions to the reactive layer and a flexible design allow for further steps in incremental building of autonomously developing systems.

Acknowledgements

We would like to thank our colleagues Mark Dunn, Achim Bendig, Michael Gienger, Benjamin Dittes, Marcus Stein, Antonello Ceravola, Martin Heckmann, Tobias Rodemann and Frank Joublin for the support and fruitful discussions.

References

Balkenius, C. (1995). *Natural Intelligence in Artificial Creatures*. Lund University Cognitive Studies 37.

Bergener, T., Bruckhoff, C., Dahm, P., Janßen, H., Joublin, F., Menzner, R., Steinhage, A., and von Seelen, W. (1999). Complex behavior by means of dynamical systems for an anthropomorphic robot. *Neural Networks*, 12(7-8):1087–1099.

Bolder, B., Dunn, M., Gienger, M., Janssen, H., Sugiura, H., and Goerick, C. (2007). Visually guided whole body interaction. In *IEEE International Conference on Robotics and Automation (ICRA 2007)*. IEEE.

Brandl, H., Joublin, F., and Goerick, C. (2008). Towards unsupervised online word clustering. In *Proceedings of International Conference on Acoustics Speech, and Signal Processing*, pages 5073–76.

Butz, M. V., Sigaud, O., Pezzulo, G., and Baldassarre, G. (2007). *Anticipatory Behavior in Adaptive Learning Systems: Advances in Anticipatory Processing*. Springer LNAI 4520.

Hart, S., Ou, S., Sweeney, J., and Grupen, R. (2006). A framework for learning declarative structure. In *Robotics: Science and Systems - Workshop on Manipulation for Human Environments*. Philadelphia, Pennsylvania.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.

Pezzulo, G. and Castelfranchi, C. (2007). The symbol detachment problem. *Cognitive Processing*, 8(2):115–131.

Prince, C. G., Helder, N. A., and Hollich, G. J. (2005). Ongoing emergence: A core concept in epigenetic robotics. In *Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Nara,Japan*.

Schembri, M., Mirolli, M., and Baldassarre, G. (2007). Evolution and learning in an intrinsically motivated reinforcement learning robot. *Advances in Artificial Life*, pages 294–303.

Schmüdderich, J., Willert, V., Eggert, J., Rebhan, S., Goerick, C., Sagerer, G., and Körner, E. Detecting objects proper motion using optical flow, kinematics and depth information. *IEEE Trans. Man Cybern. Part B – Accepted*.

Smith, L. and Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artif. Life*, 11(1-2):13–30.

Ulam, P. and Arkin, R. (2008). Biasing behavioral activation with intent. *to appear in Intelligent Service Robotics*.