EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 5.5: Combining basic cross-modal concepts into novel concepts

Danijel Skočaj, Alen Vrečko, Barry Ridge, Peter Uršič, Matej Kristan, Aleš Leonardis, Sergio Roa, Geert-Jan Kruijff, and Miroslav Janíček

University of Ljubljana, DFKI Saarbrücken
⟨`danijel.skocaj@fri.uni-lj.si`⟩

Due date of deliverable:	30 June 2012
Actual submission date:	30 May 2012
Lead partner:	UL
Revision:	final
Dissemination level:	PU

Cross-modal learning is an important characteristic of a system that is supposed to be capable of self-extension. The system should exploit different modalities and extend its current knowledge based on the information obtained from different sources and based on the previously learned concept models. In this deliverable we address the cross-modal learning in different domains, ranging from self-supervised learning of object affordances through hierarchical learning of representation of space to combining perception from different modalities to facilitate high-level cross-modal learning.

Executive Summary

An important characteristic of a robot that operates in a real-life environment is the ability to continuously expand its current knowledge, in a life-long manner. The system has to create concepts by observing the environment and also to extend these concepts and create novel concepts on top of them while interacting with the environment as well as with other cognitive agents and humans. Interactive continuous cross-modal learning, which is the main research topic of Workpackage 5, is therefore an essential characteristic of a self-extending cognitive system.

Different types of cross-modality are addressed in this deliverable. Firstly, we present cross-modal learning of object affordances and action effects; here the information arising from the visual subsystem is combined with the information from the manipulation subsystem. Also, different derived modalities, or cues, from the visual subsystem are taken into account: colour, depth (which is converted in 3D point cloud), and motion.

Then we present a hierarchical approach to building the representation of space. Range data captured by the robot is used to learn a hierarchy of so called parts. The parts represent concepts about spatial shape primitives, which are very simple on the lowest layer, and are then combined into more complex parts in the upper layers of the hierarchy.

And finally, we also address the problem of binding of modal concepts from different modalities that facilitates high-level cross-modal learning.

Some of the work presented in this deliverable is a continuation of the work performed in the previous years and mostly presented in the deliverables DR.5.1. to DR.5.4. The work about hierarchical learning of space is also highly related to Workpackage 3 on spatial cognition and the work on learning action effects is very related to Workpackage 2 and the deliverable DR.2.5 on models of object behaviour. There is also an overlap between the work on cross-modal binding and learning presented in this deliverable and the Workpackage 1 deliverable DR.1.5 on representations of gaps in knowledge, since it is about beliefs, which play an important role in both processes, cross-modal information fusion and learning, and knowledge gap representation and management. This work is also highly related to Workpackage 7, since it presents the main principle for binding and reference resolution implemented in the George system.

Role of Combining basic cross-modal concepts into novel concepts in CogX

In the process of continuous interactive cross-modal learning, the system tries to understand what it does know and what it does not, and act accordingly with the goal of updating the current concepts and building novel concepts on top of them. Therefore, the main research topic fits very well with the main motto of the project: to self-understand to be able to self-extend.

Contribution to the CogX scenarios and prototypes

In order to monitor and show progress on active and interactive continuous learning, we have designed the George scenario (Interactive cross-modal learning scenario) [37] (see also deliverable DR.7.5). This scenario has been designed as a use case for guiding and testing the system-wide research and for demonstrating methods developed in WP 5 and in some other workpackages in a complex integrated working system. The management of beliefs and cross-modal binding presented in this deliverable form the central part of the George system, crucial for a consistent fusion of information from different subsystems and for enabling consistent behaviour of the very heterogeneous system.

1 Tasks, objectives, results

1.1 Planned work

This deliverable mainly tackles the problems addressed in Task 5.4 of Workpackage 5:

> *Task 5.4: Combining concepts into novel concepts. Develop a system that is able to combine concepts learned in the previous tasks into novel concepts; to learn complex concepts and hierarchies of concepts.*

As such, it is addressing the following objectives as specified in the Technical annex:

1. *A unified framework for representing beliefs about representations of action effects, observation models, incomplete information and categorical knowledge. [WPs 1,4,5]*

5. *A theory of how to use these representations to identify learning opportunities, plan and execute plans in order to learn so as to perform future tasks more effectively and efficiently. [WPs 4,5]*

8. *New representations and algorithms to allow a robot to extend its categorical knowledge by identifying gaps and learning the relationships between different modalities (e.g. vision and language). [WP 5]*

We will structure this deliverable according to several research lines that have been addressed. First, let us look at our plans and goals that we had set:

- **Self-supervised learning of object affordances.**

  We planned to develop an algorithm for inducing causal relationships of action/object complexes, in terms of the trajectory of objects, represented as a sequence of object poses, given some motor action (here a pushing action). For this purpose, a quantization algorithm needed to be developed which can discretize the sensorimotor space. Additionally, an algorithm that can extract causal relationships in form of probabilistic transitions among discrete states had to be devised, based on an algorithm for extracting substochastic sequential machines (CrySSMEx) [15] from dynamic systems.

  The CrySSMEx algorithm can find qualitative states depending on the output function given by the output space. Thus, we planned to cluster states that represent more abstract concepts from the sensorimotor space of a pushing scenario.

In addition to this, we also planned on improving our previously proposed algorithm for self-supervised cross-modal learning [31] by using additional mechanisms to enhance its performance at acquiring novel affordance concepts over short-term training periods. Specifically, we were interested in developing feature relevance determination algorithms that could rapidly find the most discriminative feature dimensions in the input space for predicting the naturally occurring categories in the output space.

- **Learning hierarchical representation of space.**

  The goal for this year was to develop an algorithm for learning a compositional hierarchical representation of space based on data obtained with a range sensor. We aimed to extend the existing Learning the Hierarchy of Parts algorithm [8], by incorporating rotational invariance of parts into the model. We planned to learn as many layers of the hierarchy as possible using only range scans as input data. We sought to evaluate the performance of the learned concepts through the room classification problem, and to validate our model in comparison with other, state-of-the-art, approaches.

- **Cross-modal binding and learning.**

  In Year 3 the cross-modal learning and binding concepts were used in parallel to the belief structures. A Markov Logic Network engine component was used for reference resolution. The aim for this year was to enhance the dialogue between the robot and the tutor, which in turn requires new enhanced reference resolution MLN. We also wanted to bring MLN reasoning into the belief structure itself. MLNs should have an important role in propagation of information between various types of beliefs.

1.2 Actual work performed

1.2.1 Self-supervised learning of object affordances

We developed an online learning algorithm for quantization of spaces in the pushing affordances scenario. After the robot performs a pushing action, a density estimation (quantization) algorithm runs for the current sequence of effector and object poses obtained, in order to estimate the density of this sensorimotor space. Additionally, an output space is also discretized, which corresponds to changes in rotation of the object. The output space is needed to split the state space, which are representations of object poses at the next time step. The quantization algorithm is based on the incremental Growing Neural Gas (GNG) algorithm and the Minimum Description Length principle for evaluation of clustering performance [33]. When a GNG network or graph reaches a stable state, the algorithm stops and a new action is performed by the robot to continue learning.

After the quantization is performed, a state representing a set of instances is split when the output or the next state differs (so that the entropy is high). The transitions among states and its probabilities are obtained to construct probabilistic machines that represent the behavior of these action/object dynamical systems.

By using the extracted substochastic sequential machines (SSMs), we were able to predict the object behavior accurately.

When the sensorimotor space is huge, it is appropriate to split it in different regions where different learning machines (in this case quantizers) can be employed. We used these divide-and-conquer approaches to accelerate the learning process and make it more efficient. The regions are split after some time step and by employing a measure of variance in the sensorimotor data sets.

The CrySSMEx algorithm also splits the state space in a similar way but using other criteria, as explained above. However, it is in principle a similar process. At the end, the state space is represented by a tree of Vector quantizers, resembling a hierarchical clustering.

Once we obtained a quantization of the input space, we use these quantizers with other representations for the output space. We then discretized the output space in such a way that we can distinguish among abstract object behaviors like sliding, flipping over and tilting. The state space quantizer then groups the states in a different way, splitting the space according to this new output function. Thus, these new state space regions might be viewed as components of a joint distribution.

Once again by using these new SSMs, we can predict with high accuracy the classifications of object behavior (see Annex 2.1).

In our other work on self-supervised cross-modal learning described in last year's deliverable [31], we have developed methods for feature relevance determination [32] that serve to augment the original algorithm. These methods stem from ideas originally touched upon in the previous year's work, but have been more thoroughly developed, investigated and evaluated this year. They are based on the idea of applying the Fisher criterion score to learning vector quantization algorithms for online feature relevance determination (see Annex 2.2).

In the attached paper [32], two new algorithms for LVQ-based relevance determination are presented. Both methods exploit the positioning of the prototype vectors in the input feature space to inform estimates of the Fisher criterion score along the input dimensions, which are then used to form online estimates of the relevance of the input dimensions with respect to the classifier output. Both methods provide online updates that may be used alongside regular LVQ updates or within the broader context of our self-supervised cross-modal learning framework and neither method requires the specification of a learning rate, as in stochastic gradient descent. Performance advantages are demonstrated in experiments on various popular classification datasets, as well as on data from our object push affordance learning experiments.

1.2.2 Learning hierarchical representation of space

In work described in Annex 2.3 we propose a new compositional hierarchical representation of space, which is learned based on statistically significant observations. We have focused on a two dimensional space, since many robots perceive their surroundings in two dimensions using range sensors. Range data is transformed into images and then a hierarchy of so called parts is learned from those images. Parts, which are rotationally invariant, represent concepts about spatial shape primitives. They are very simple on the lowest layer, while their complexity and size increases with respect to the height of the corresponding layer of the hierarchy. At the bottom, concepts are represented as small fragments of lines in several different orientations. On higher layers compositions of lower layer concepts are learned, forming more and more complex shapes. Only shapes that have been observed most frequently in the images used for learning are memorized, and then used to model the environment. Only a few lower layers of the proposed hierarchy are currently being learned. In the future, the image formation step will be omitted and information from other modalities, like odometry, will be used to combine the information from separate range scans into a unified map, which will provide a more complete view of the environment. Based on these maps even more complex shapes will be formed, which will introduce the abstraction needed to learn higher level concepts. These will provide good scalability of the model through sharing of same concepts between different room categories. A cognitive system using our representation of space would be able to make use of a large quantity of information, that has been obtained in past observations, to extend it's knowledge about general characteristics of space, and then use this knowledge as a compact and expressive description of it's surroundings.

In this work we also propose a new low-level image descriptor, by which we demonstrate the performance of our representation in the context of the room classification problem based only on data obtained with a laser range finder. Using only the lower layers of the hierarchy, we obtain state-of-the-art classification results on demanding datasets. Room classification methods, which are based on data obtained with range sensors have a potential to work faster than other, for example, vision based, approaches, since the input information is of lower dimensionality, while on the other hand, the stinginess of the data makes this approach much more demanding. Such approach could therefore provide a cognitive system with a quick first impression about room type, which could then serve as a reliable basis for the use of temporally more costly classifiers to verify the proposed hypothesis.

1.2.3 Cross-modal binding and learning

In work described in Annex 2.4 we devised a new belief scheme that now also supports MLN reasoning. The beliefs form a cognitive layer where multi-modal and multi-agent information is associated and merged to a-modal representations. In general a belief can be regarded as a high-level representation of an element of the physical reality, grounded in one or more sensory inputs, attributed to a specific agent or a combination of both. The new belief scheme distinguishes five distinct belief categories. *Private* beliefs reflect the robot perceptions of the environment based on its sensory input. *Assumed* beliefs are used to establish cross-agent or cross-modal common ground; they are created from private beliefs by translating the modal symbols to the a-modal ones. *Attributed* beliefs contain information that a robot attributes to another agent. *Verified* beliefs are created from attributed beliefs; they essentially contain the acknowledged information from the attributed beliefs. *Merged* beliefs combine information from verified and assumed beliefs and represent the final a-modal situated knowledge, ready to be used by the higher level cognitive processes (e.g. motivation, planning). They contain as reliable information as possible and as much information as available.

MLN components have a triple role in this Belief scheme: (i) They are used for binding — the binding process associates between beliefs from different modalities or different epistemic origins (in George the binding principles are used in reference resolution), (ii) as a translator between modal and a-modal symbols and (iii) for information fusion. In the information flow from sensory data to higher cognition, the information fusion can be regarded as a next step after the binding.

1.3 Relation to the state-of-the-art

In this section we discuss how our work is related to, and goes beyond the current state-of-the-art.

1.3.1 Self-supervised learning of object affordances

The problem of object prediction has already been tackled by using offline learning algorithms [20], which lack incremental ways of learning when new data sequences are added. In this new approach, we can estimate the density of sensorimotor spaces in an online way.

We also extract probabilistic machines that can be used in planning tasks a posteriori, since they are essentially graphs on which some reasoning methods could be applied. They also encode an entropy based representation of causal relationships that can be used for active learning.

We obtain qualitative representations of temporal sequences of action/object complexes, taking into account trajectory information like object and robot poses. Traditionally, learning algorithms with abilities of temporal processing like Hidden Markov Models (HMMs) [10], Recurrent Neural Networks (RNNs) [34] and Dynamic Bayesian Networks [26] have been used in robotic learning tasks. Substochastic sequential machines have been extracted from RNNs to extract qualitative information learned by the RNNs [15, 11]. Substochastic Sequential Machines are similar to HMMs, in that they are probabilistic finite state representations. Some characteristics of SSMs like entropy based representation of uncertainty might be advantageous when designing information-theoretic active learning methods.

Learning vector quantization (LVQ) [18] provides an intuitive, and often highly effective, means for discriminative learning where prototype vectors are used to quantize the input feature space and given labels to form piecewise-linear classifiers using the nearest neighbour rule. Since their introduction, LVQ algorithms have undergone various analyses and seen various improvements to their design and much attention has also been paid in recent years to the role that the distance metric plays in the effectiveness of LVQ methods, which was the focus of our investigation in [32]. LVQ ordinarily relies on the Euclidean metric to measure the distance between data points, which provides equal weighting to all input dimensions. Many of the input dimensions, however, may have little relevance when considering the desired output function and may even have a detrimental effect on the output if considered with equal weighting in the metric to the more important dimensions. One standard approach to this issue is to pre-process the data using some form of feature selection or dimensionality reduction, but this can be infeasible in many learning scenarios where the training data are not available in advance, e.g. autonomous robotics.

One early adaptation of LVQ3 known as *distinction sensitive learning vector quantization (DSLVQ)* [28] achieves this by using a heuristic to adjust weights along each of the input dimensions to modify the Euclidean metric. An adaptation of LVQ1 known as *relevance learning vector quantization (RLVQ)* [3] uses Hebbian learning to do similar, by adjusting weights for each of the input dimensions at every training step depending on whether they contributed to the correct or incorrect classification of a training sample. RLVQ was subsequently adapted for use with GLVQ producing a method known as *generalized relevance learning vector quantization (GRLVQ)* [13] such that the dimensional weight updates also adhere to gradient descent dynamics in a similar way to the prototype updates. Another modified version of GLVQ [43] uses Fisher's discriminant analysis to create an alternative metric to the weighted Euclidean distance that employs a matrix transformation to reduce the feature space dimensionality. More recently, an adaptive metric was used in combination with training data selection for LVQ [27].

By comparison, in our work described in Annex 2.2, an advantage provided by the proposed methods over other metric-adaptive LVQ methods based on gradient descent, is that they do not require a learning rate or other parameters to be specified. Moreover, they provide incremental update rules that operate alongside regular LVQ update rules and can therefore be applied to any algorithms based on the general LVQ paradigm. Experimental evaluations were provided under various stress conditions and over various datasets and the proposed methods were shown to perform competitively against various other LVQ-based methods, and against SVM.

1.3.2 Learning hierarchical representation of space

Numerous spatial models have already been proposed. Metric representations use sensory information to accurately describe the geometry of space to some desired extent [6, 1], topological representations use graphs to model space [40, 5], hybrid approaches combine both of the above paradigms [41, 42], while combining of one or even both of the approaches, metric and topological, on multiple levels of abstraction results in hierarchical representations [30, 21, 25, 44]. Perhaps the closest to our work is the work presented by Mozos [25]. He generates a topology of the environment for room classification based on laser scans. A major difference between his and our idea is that he uses occupancy grids under the topological level, which are not suitable for modelling large environments, since they scale poorly. In our approach, rooms will be represented with parts, which will be shared between different categories, and thus requiring less memory. Despite several existing approaches, to the best of our knowledge, our work is the first attempt of using a hierarchical compositional model for the representation of space on the lowest semantic level, at which range sensors are usually used to observe the environment.

However, compositional hierarchies have been used for some time by the computer vision community [8, 9, 19, 7]. In this work we adapt the hierarchical model from [9] to develop a description suitable for representation of space. It turns out that rotational invarance of parts is crucial for obtaining a compact and expressive hierarchy for spatial representation. This property is not present in the model of [9], therefore, we extended the model to satisfy the above condition.

Various systems performing topological localization have been developed for room classification. In [30] very accurate room classification is achieved using multimodal information. Approaches using less information available for classification have also been considered. Laser range data combined with vision was used for classification in [24], and many approaches that use vision only for the accomplishment of this task have also been presented [29, 46, 2, 48]. The most related to our work are the approaches performing room classification based only on data obtained with range sensors. In [39]

3D Time-of-Flight infrared sensor was used for acquiring 3D information, which allowed the distinction between three types of rooms (office, meeting room and hall). Only laser range data was used in [23]. Their robot was equipped with a 360 degree field of view range sensor and they were able to distinguish between four classes (rooms, corridors, doorways and hallways). The classification was performed with AdaBoost and it was based only on a single scan. Laser range data was also used for classification in [12], where Voronoi random fields (VRFs) were employed to label different places in the environment, providing the distinction between four classes (rooms, hallways, junctions and doorways). Their approach uses a state-of-the-art SLAM technique to generate a metric occupancy grid map of an environment, while the Voronoi graph is then extracted of this map. For each point on the Voronoi graph, VRFs then estimate the type of place it belongs to. Our approach to room classification is based on the proposed hierarchical model. We have taken into consideration a set of room types (living room, corridor, bathroom, and bedroom), which are in our opinion more demanding than the ones presented in the related work [23, 12].

1.3.3 Cross-modal binding and learning

Many of the past attempts at binding information within cognitive systems were restricted to associating linguistic information to lower level perceptual information. Roy et al. tried to ground the linguistic descriptions of objects and actions in visual and sound perceptions and to generate descriptions of previously unseen scenes based on the previously accumulated knowledge [35, 36]. This is essentially a *symbol grounding problem* first defined by Harnad [14]. Chella et al. proposed a three-layered cognitive architecture around the visual system with the middle, *conceptual layer* bridging the gap between linguistic and sub-symbolic (visual) layers [4]. Related problems were also often addressed by Steels [38].

Jacobsson et al. approached the binding problem in a more general way [17] [16] developing a cross-modal binding system that could form associations between multiple modalities and could be part of a wider cognitive architecture. The cross-modal knowledge was represented as a set of binary functions comparing binding attributes in pair-wise fashion. A cognitive architecture using this system for linguistic reference resolution was presented in [45]. This system was capable of learning visual concepts in interaction with a human tutor. A probabilistic binding system was developed within the same group that encodes cross-modal knowledge into a Bayesian graphical model [47]. In [22] a framework for constructing high-level cognitive representations of the environment, called beliefs, was presented. Markov logic was used as the main framework for various types of inference over beliefs, including perceptual grouping, which comes very close to our definition of binding. All these systems ([17] – [22]) assumed static cross-modal knowledge.

2 Annexes

2.1 Roa et al. "Online Density Estimation in a Robotic Manipulation Scenario and its application to Learning of Temporal Action/Object Models and Concepts"

Bibliography S. Roa and G.-J. Kruijff: "Online Density Estimation in a Robotic Manipulation Scenario and its application to Learning of Temporal Action/Object Models and Concepts". Technical Report, 2012.

Abstract Cognitive Robotics implies the ability of robots to learn from the environment by interacting with it and learning causal relations and associations stemming from these interactions. In this paper, we address the particular problem of interacting by manipulating objects, specifically robot arm pushes. To solve this problem we come up with models which can describe the behaviour of objects given some action. For a learning robot it is essential to learn in an incremental way, after new information is coming, without losing generalization and avoiding overfitting. We tackle this problem firstly by estimating the density of a sensorimotor space after a robot performs a new action by using a modification of the incremental Growing Neural Gas (RobustGNG) algorithm. RobustGNG performs a quantization of the space which is robust to noise and overfitting issues. Subsequently, we infer models useful for prediction of object trajectories in terms of object poses. The same machinery is useful for obtaining more coarse-grained predictions, for instance categorizations of object behaviours. Last, but not least, these prediction models should provide a qualitative temporal description of the state space, so that they can eventually be used in planning tasks. Thus, we infer cause-effect models by using a new version of the CrySSMEx algorithm for extraction of substochastic finite-state machines given the quantization obtained by means of RobustGNG.

Relation to WP This work is directly related to continuous learning of cross-modal concepts, where crossmodality comes from sources like manipulation and vision (in this case simulated). It also explores the problem of deriving categorical knowledge from previously learned tasks, i.e. from density estimation of the sensorimotor space.

2.2 Ridge et al. "Relevance Determination for Learning Vector Quantization using the Fisher Criterion Score"

Bibliography B. Ridge, A. Leonardis, and D. Skočaj: "Relevance Determination for Learning Vector Quantization using the Fisher Criterion Score". 17th Computer Vision Winter Workshop, Mala Nedelja, Slovenia,February 1-3, 2012.

Abstract Two new feature relevance determination algorithms are proposed for learning vector quantization. The algorithms exploit the positioning of the prototype vectors in the input feature space to estimate Fisher criterion scores for the input dimensions during training. These scores are used to form online estimates of weighting factors for an adaptive metric that accounts for dimensional relevance with respect to classifier output. The methods offer theoretical advantages over previously proposed LVQ relevance determination techniques based on gradient descent, as well as performance advantages as demonstrated in experiments on various datasets including a visual dataset from a cognitive robotics object affordance learning experiment.

Relation to WP The two new algorithms were proposed in order to augment the short-term training discriminative capacity of our previously proposed self-supervised cross-modal learning algorithm [31] which is capable of generating novel object affordance concepts autonomously.

2.3 Uršič et al. "Room Classification using a Hierarchical Representation of Space"

Bibliography P. Uršič, M. Kristan, D. Skočaj, A. Leonardis. "Room Classification using a Hierarchical Representation of Space". Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2012, 2012.

Abstract Mobile robots need an effective spatial model for the successful operation in real-world environment. The model should be compact and simultaneously possess large expressive power. Moreover, it should scale well. In this work we propose a new hierarchical representation of space, whose compositional structure is learned based on statistically significant observations. We have focused on a two dimensional space, since many robots perceive their surroundings in two dimensions with the use of a laser range finder or a sonar. We also propose a new low-level image descriptor, by which we demonstrate the performance of our representation in the context of room classification problem. Using only the lower layers of the hierarchy, we obtain state-of-the-art classification results on demanding datasets.

Relation to WP This work proposes a new hierarchical model of space. Spatial shape primitives are being learned by combining simple concepts into more complex ones, forming the hierarchical representation. Based on previous observations, the hierarchy containing most frequently detected shapes is learned and then used to derive new abstract concepts, like room categories. Therefore, the work is related to Task 5.4.

2.4 Vrečko et al. "Associating and merging multi-modal and multi-agent information in a cognitive system"

Bibliography A. Vrečko, A. Leonardis and D. Skočaj: "Associating and merging multi-modal and multi-agent information in a cognitive system". TR-LUVSS-02/2012, University of Ljubljana, Faculty of Computer and information science, May 2012

Abstract A critical ability for every cognitive system operating in a complex environment is the ability to combine several representations of the same physical reality into a single shared representation. Such combined, a-modal representations are then ready to be used by higher level cognitive processes, like motivation and planning. In this work we describe a cognitive layer where multi-modal and multi-agent information is associated and merged to a-modal representations. Furthermore we describe the application of cross-modal binding principles to a specific problem of reference resolution.

Relation to WP The technical report addresses the problem of cross-modal binding and learning, as defined in WP 5. It describes the application of these principles to a concrete problem of reference resolution. Furthermore it describes the belief schema where multi-modal information is associated and merged to a-modal representations.

References

[1] Kai Oliver Arras. *Feature-based robot navigation in known and unknown environments*. PhD thesis, Lausanne, 2003.

[2] B. Ayers and M. Boutell. Home interior classification using sift keypoint histograms. *CVPR*, pages 1–6, 2007.

[3] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz. Relevance determination in learning vector quantization. In *European Symposium on Artificial Neural Networks*, pages 271–276, 2001.

[4] A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artif. Intell.*, 89(1-2):73–111, 1997.

[5] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125 –137, apr 2001.

[6] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 136–146, New York, NY, 1990. Elsevier Science.

[7] Gil J. Ettinger. Hierarchical object recognition using libraries of parameterized model sub-parts. Technical report, Cambridge, MA, USA, 1987.

[8] S. Fidler, M. Boben, and A. Leonardis. *Object Categorization: Computer and Human Vision Perspectives*, chapter Learning Hierarchical Compositional Representations of Object Structure. Cambridge University Press, 2009.

[9] Sanja Fidler, Marko Boben, and Ales Leonardis. Evaluating multi-class learning strategies in a generative hierarchical framework for object detection. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 531–539. 2009.

[10] M Fox, M Ghallab, G Infantes, and D Long. Robot introspection through learned hidden markov models. *Artificial Intelligence*, 170(2):59–113, 2006.

[11] Stefan L. Frank and Henrik Jacobsson. Sentence-processing in echo state networks: a qualitative analysis by finite state machine extraction. *Connection Science*, 22(2):135–155, 2010.

[12] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: extracting the topological structure of indoor environments via place labeling. *IJCAI*, pages 2109–2114, 2007.

[13] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[14] S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.

[15] H. Jacobsson. The crystallizing substochastic sequential machine extractor - `CrySSMEx`. *Neural Computation*, 18(9):2211–2255, 2006.

[16] H. Jacobsson, N. Hawes, G-J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *Proc. of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, Amsterdam, March 2008.

[17] H. Jacobsson, N. Hawes, D. Skočaj, and G-J. Kruijff. Interactive learning and cross-modal binding - a combined approach. In *Symposium on Language and Robots*, Aveiro, Portugal, 2007.

[18] T. Kohonen. *Self-organizing maps.* Springer, 1997.

[19] Iasonas Kokkinos and Alan Yuille. Inference and learning with hierarchical shape models. *Int. J. Comput. Vision*, 93(2):201–225, June 2011.

[20] M. Kopicki, J. Wyatt, and R. Stolkin. Prediction learning in robotic pushing manipulation. In *Proceedings of the 14th IEEE International Conference on Advanced Robotics (ICAR 2009)*, Munich, Germany, June 2009.

[21] B. Kuipers. The spatial semantic hierarchy. *Artificial intelligence*, 119(1–2):191–233, 2000.

[22] P. Lison, C. Ehrler, and G.-J. Kruijff. Belief modelling for situation awareness in human-robot interaction. In *Proceedings of the 19th IEEE International Symposium in Robot and Human Interactive Communication.* IEEE, 2010.

[23] O. M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. *ICRA*, pages 1730–1735, 2005.

[24] O. M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[25] Oscar Martinez Mozos. *Semantic Labeling of Places with Mobile Robots.* PhD thesis, Springer Berlin Heidelberg, 2008.

[26] Jonathan Mugan and Benjamin Kuipers. Autonomously learning an action hierarchy using a learned qualitative state representation. In *In Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

[27] C. E Pedreira. Learning vector quantization with training data selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):157–162, 2006.

[28] M. Pregenzer, G. Pfurtscheller, and D. Flotzinger. Automated feature selection with a distinction sensitive learning vector quantizer. *Neurocomputing*, 11(1):19–29, 1996.

[29] A. Pronobis, B. Caputo, P. Jensfelt, and H. Christensen. A discriminative approach to robust visual place recognition. *IROS*, pages 3829–3836, 2006.

[30] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA'12)*, Saint Paul, MN, USA, May 2012.

[31] B. Ridge, A. Leonardis, and D. Skočaj. Self-Supervised Cross-Modal relevance learning vector quantization. Submitted for journal publication, 2011.

[32] B. Ridge, A. Leonardis, and D. Skočaj. Relevance determination for learning vector quantization using the fisher criterion score. In *Proceedings of the Seventeenth Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, February 2012.

[33] S. Roa and G.-J. Kruijff. Online Density Estimation in a Robotic Manipulation Scenario and its application to Learning of Temporal Action/Object Models and Concepts. Technical report, DFKI GmbH, 2012.

[34] S. Roa and G.-J.M. Kruijff. Offline and active gradient-based learning strategies in a pushing scenario. In *International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems 2010. ERLARS 2010*, pages 29–34, Lisboa, Portugal, 2010.

[35] D. Roy. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3-4):353–385, 2002.

[36] D. Roy. Grounding words in perception and action: computational insights. *TRENDS in Cognitive Sciences*, 9(8):389–396, 2005.

[37] Danijel Skočaj, Matej Kristan, Alen Vrečko, Marko Mahnič, Miroslav Janíček, Geert-Jan M. Kruijff, Marc Hanheide, Nick Hawes, Thomas Keller, Michael Zillich, and Kai Zhou. A system for interactive learning in dialogue with a tutor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*, San Francisco, CA, USA, 25-30 September 2011.

[38] L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings.* Laboratorium, Antwerpen, 1999.

[39] A. Swadzba and S. Wachsmuth. Categorizing perceptions of indoor rooms using 3d features. *Lecture Notes in Computer Science: Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–744, 2008.

[40] A. Tapus. *Topological SLAM: Simultaneous Localization and Mapping with Fingerprints of Places.* 2005.

[41] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[42] Sebastian Thrun, J.-S. Gutmann, Dieter Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[43] M. K Tsay, K. H Shyu, and P. C Chang. Feature transformation with generalized learning vector quantization for hand-written chinese character recognition. *IEICE Transactions on Information and Systems*, E82-D(3):687–692, 1999.

[44] Shrihari Vasudevan, Stefan Gächter, Ahad Harati, and Roland Siegwart. 50 years of artificial intelligence. chapter A hierarchical concept oriented representation for spatial cognition in mobile robots, pages 243–256. Springer-Verlag, Berlin, Heidelberg, 2007.

[45] A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *Proc. of the 2009 IEEE/RSJ Int. Conf. on Intelligent RObots and Systems*, pages 3140–3147, St. Louis, Oct. 2009.

[46] J. Wu, H.I. Christensen, and J. M. Rehg. Visual place categorization: Problem, dataset, and algorithm. *IROS*, pages 4763–4770, 2009.

[47] Jeremy L. Wyatt, Alper Aydemir, Michael Brenner, Marc Hanhiede, Nick Hawes, Patric Jensfelt, Matej Kristan, Geert-Jan M. Kruijff, Pierre Lison, Andrzej Pronobis, Kristoffer Sjöö, Danijel Skočaj, and Alen Vrečko. Self-understanding and self-extension: a systems and representational approach. *IEEE Transactions on Autonomous Mental Development*, 2(4):282–303, 2010.

[48] Z. Zivkovic, O. Booij, and B. Kröse. From images to rooms. *Robotic and Autonomous Systems*, 55(5):411–418, 2007.

Online Density Estimation in a Robotic Manipulation Scenario and its application to Learning of Temporal Action/Object Models and Concepts

Sergio Roa
Geert-Jan Kruijff
{sergio.roa, gj}@dfki.de
German Research Center for Artificial Intelligence (DFKI GmbH)

May 29, 2012

Abstract

We need robots to learn from the environment by interacting with it, and learn models of causal relations and associations from these interactions. In this paper, we address the particular problem of interacting by manipulating objects, using robot arm pushes. To solve this problem we define models which can describe the behaviour of objects given some action. For a learning robot it is essential to learn in an incremental way, after new information is coming, without losing generalization and avoiding overfitting. We tackle this problem firstly by estimating the density of a sensorimotor space after a robot performs a new action by using a modification of the incremental Growing Neural Gas (RobustGNG) algorithm. RobustGNG performs a quantization of the space which is robust to noise and overfitting issues. Subsequently, we infer models useful for prediction of object trajectories in terms of object poses. The same machinery is useful for obtaining more coarse-grained predictions, for instance categorizations of object behaviours. Last, but not least, these prediction models should provide a qualitative temporal description of the state space, so that they can eventually be used in planning tasks. Thus, we infer cause-effect models by using a new version of the CrySSMEx algorithm for extraction of substochastic finite-state machines given the quantization obtained by means of RobustGNG.

1 Introduction

Robots need to extend their knowledge in an incremental and online fashion. This aspect is important to allow robots to plan actions on the basis of new experience gathered. In the case of robotic learning, the problem of incremental and online learning is challenging, because one needs to address issues like robustness to noisy environments, overtraining and high dimensionality. Online Density Estimation methods, e.g. based on kernel methods, have been studied in the context of discriminative models for object classification [8]. In this paper, we tackle the problem of online density estimation of sensorimotor spaces in an interactive environment. Here, the robot interacts with simple geometrical objects via a pushing action. After performing an action, a quantization function for density estimation is applied to a representation of the sensorimotor space in an online manner. Basically, sequences of finger and object poses (rigid body transformations) are stored as instances of the sensorimotor space. The functions are gradually refined, after new training sequences are gathered. Moreover, the quantized space is split into two subspaces or regions after some iteration. This divide-and-conquer approach accelerates the convergence. We use a modification of the Growing Neural Gas algorithm [3, 11, 12, 14] for quantization which is robust for finding the right clusters in the presence of noise.

When the quantization process finishes, we employ an offline mechanism for constructing probabilistic models of action/object complexes that makes use of the quantization functions. This method is based on the CrySSMEx algorithm [6, 14] for extracting substochastic sequential machines (SSMs) from dynamical systems. The robot predicts the object behaviour in terms of trajectory information contained in sequences of object poses, given an arm pose and a certain action, represented as the target goal of the robot finger. The probabilistic models that we obtain are substochastic finite-state models. They have the potential to be used for planning, since we obtain a qualitative representation of the sensorimotor space which encodes action/object instances. Moreover, their graph-based nature encodes the probabilistic transitions that lead to subsequent states which is particularly useful in planning. In the experiments we present here, the state space encompasses a geometrical representation of object poses.

The substochastic machines are represented as a probabilistic case of Mealy or Moore machines. These machines allow us not only to encode a quantization of the state space but also to encode input functions and output functions, which are quantization functions. In this problem, input functions are a quantization mechanism applied to the representation of the sensorimotor space as mentioned above. They encode a combination of object/finger pose information. On the other hand, output functions may be used in different manners, depending of the discretization degree that we expect in the probabilistic finite-state machine. More precisely, if the objective would be to predict accurately the trajectories of objects given some action and some object pose, likewise in terms of object poses, we could apply an output function that acts as a probabilistic regression, as long as we represent object poses at subsequent states in this output function. In our work, we did not use this kind of output function, but a quantization mechanism that acts on vectors of object rotation change (object transformations), to further reduce the space complexity of output quantizers. This output representation takes also advantage of the learning properties of CrySSMEx. On the contrary, if we want to group state clusters on the basis of more coarse-grained sets (more abstract patterns), we might want to use other quantization functions. In our experiments, we obtained classifications for three different types of object behaviours by means of an alternative output quantization. A general learning schema is presented in Fig. 1.

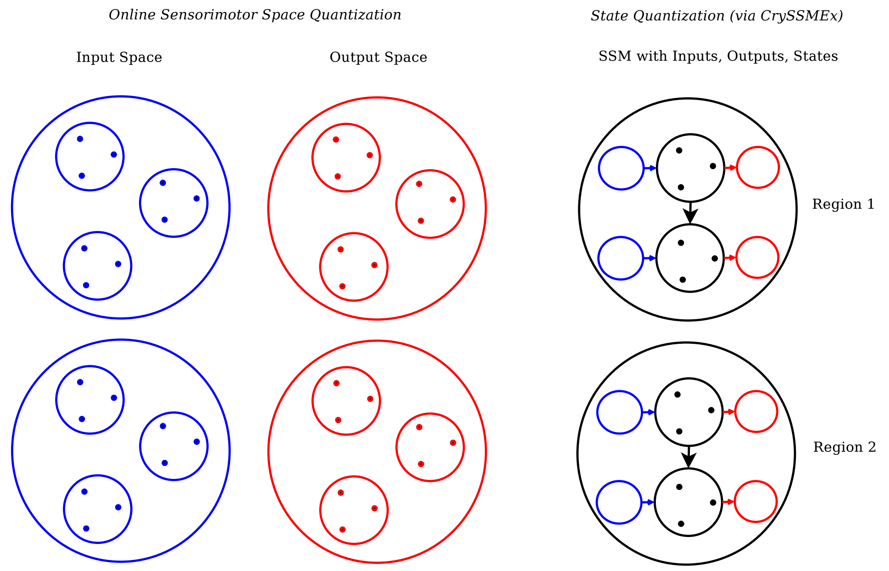

Online Sensorimotor Space Quantization
State Quantization (via CrySSMEx)
Input Space
Output Space
SSM with Inputs, Outputs, States
Region 1
Region 2

Figure 1: Learning schema. Quantization of input, output and state spaces and division of the spaces into regions.

The problem of predicting object trajectories after a rigid body pushes it has already been addressed in previous works. In [7], the authors applied offline probabilistic models for density estimation, specifically Gaussian kernel density methods, in contrast to the online methods explained later here. Moreover, the construction of qualitative models which can keep track of the temporal causality for long data sequences is an additional gain of our work. We also addressed this problem before by employing a Recurrent Neural Network (RNN) for prediction [15]. Here, we focus on extracting machines which are potentially useful for planning, as stated above. However, extraction of planning knowledge from RNNs is also possible [18]. Actually, CrySSMEx was originally conceived for rule extraction from RNNs [6, 2], which is closely related to planning.

The problem we consider in this paper is closely related to previous works on affordances learning [17, 10, 19, 13, 1], among others. The theory of affordances is a cognitive development theory, which attempts to explain how creatures are able to acquire sensorimotor skills when they are faced with the different features found in the environment [4]. In summary, given some object properties which afford some behaviour, humans or animals are able to employ these objects as instruments or ways to attain a specific goal. For instance, certain objects can afford a sliding, a flipping over or a rolling behaviour. In [17, 19], the robots (arms) perform some random actions and some features are extracted and used for learning. Then, this can be used for solving some task, by implementing a planning framework. In [10], the robot is more autonomous and enters into different developmental stages on the basis of a learning progress measure, which allows that the robot decides what action to perform next in the presence of different objects, emulating a curious agent. The authors demonstrated that the agent "discovers" affordances while it starts to perform certain actions repeatedly. In [13], the objective was to extract visual features from random pushing actions which can be used for obtaining categorizations of affordances via unsupervised learning. In the field of Cognitive Robotics, a related theory has also been developed in recent years, called Object-Action Complexes (OACs) [9]. OACs aims for a universal representation enabling efficient planning and execution of actions in different levels of a cognitive architecture [9]. OACs combine search capabilities based on theorem proving rules and the object and situation oriented concept of affordance, plus the logical framework of the situation calculus for planning [9].

Affordances are also strongly related to the emergence of concepts [1]. A new behaviour observed in an object or a new way to adapt the body and apply it to an object to achieve some goal might lead to new linguistic descriptions of these situations. As we mentioned above, we use probabilistic methods to discriminate among different object behaviours given some actions.

For evaluation purposes, we discuss a learning scenario where a simulated robotic arm interacts with a polyflap. A polyflap is a polygon (concave or convex) cut out of a flat sheet of some material (e.g. cardboard) and folded once (anywhere) to produce a 3-D object [16], cf. Fig. 2. In the implementation we use the NVidia® PhysX$^{\mathrm{TM}}$ library which allows us to perform realistic physical simulations and to obtain 3-dimensional feature vectors, so that we can easily re-adapt our algorithms to real scenarios. Although providing an idealized scenario, these experiments are necessary to establish a base line from which we can start facing noisy information.

This paper is organized as follows. First, we describe the online quantization algorithm and its application to sensorimotor spaces and specifically to the prediction problem. Then, we explain the induction of substochastic sequential machines for prediction and classification. Finally, we present experimental results and draw conclusions.

2 Online Quantization of Sensorimotor Spaces

2.1 Features for Learning about Sensorimotor Spaces in a Pushing Scenario

The learning scenario is shown in Fig. 3. The simulated arm corresponds to a Neuronics® Katana 6M$^{\mathrm{TM}}$ arm with a ball as a simple finger. In order to simulate a pushing action we apply a linear trajectory over a specified time period until the finger reaches a desired pose. The arm has 6 joints, including

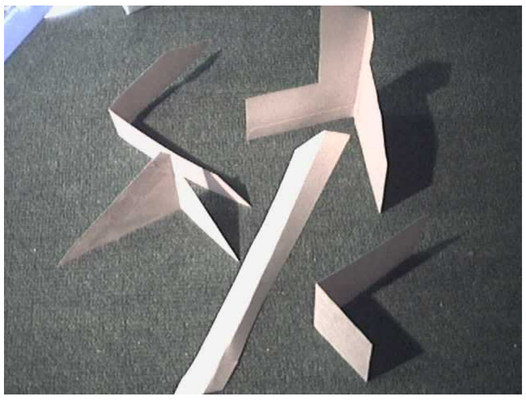


Figure 2: Polyflaps, `http://www.cs.bham.ac.uk/~axs/polyflaps/`. Used here are polyflaps of the shape (bottom-right corner)

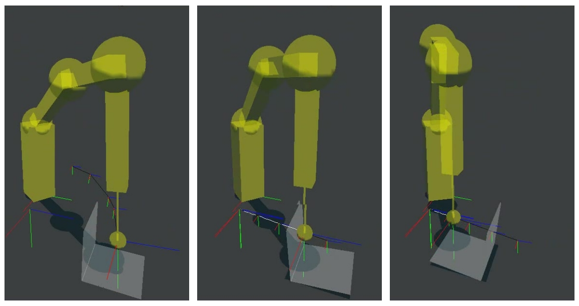


Figure 3: Learning scenario with a polyflap

the last joint for the finger which is static. The representation of object poses are in Euler angles with respect to a reference frame which is the origin in the scene (6-D pose).

[7] provides a complete description of the rigid body transformations involved in a pushing scenario, assuming the physical properties and net forces are constant in time. In summary, if two rigid bodies are present, where a time frame A corresponds to one object, a frame B to a second one, and a frame O for some fixed environment, rigid body transformations T between these frames at subsequent time steps can be used for describing the current system state. For instance, $T^{A_t,A_{t+1}}$ represents the transformation of the first object from time t to $t+1$. The prediction problem is stated as [7]: given we know the starting states and the motion of the pusher $T^{A_t,A_{t+1}}$, predict the resulting motion of the object $T^{B_t,B_{t+1}}$.

In our work, we assume rigid body transformations only with respect to a fixed frame O, although alternative representations are also possible. Rigid body transformations are encoded as rotation matrices which represent poses and can be transformed to 6-D poses consisting of position and orientation in Euler angles with respect to O.

Definition 1. *Let us denote $T^E, T^{A(t),O}$ and $T^{B(t),O}$ variables representing rigid body transformations with respect to a fixed frame O, where $T^{E,O}$ denotes the target pose representing the final expected pose of the finger at the end of a linear trajectory (desired pose), $T^{A(t),O}$ represents the pose of the finger at time step t and $T^{B(t),O}$ the pose of the object at t. We represent the poses by using a 6-D representation encoding 3-D position and 3-D orientation in Euler angles, so that we have the vectors $\mathbf{e}, \mathbf{a}(t), \mathbf{b}(t)$ to denote finger target pose, finger pose and object pose respectively. We call $\mathbf{e}$ also a motor command. The values are all normalized to obtain vectors with mean 0 and standard deviation 1.0.*

We want to model a situated discrete time dynamical system (SDTDS) [6] representing the interaction among these objects in the scenario. In an SDTDS, an input space $\mathcal{I} \subseteq \mathbb{R}^{n_i}$, a state space $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ and an output space $\mathcal{O} \subseteq \mathbb{R}^{n_o}$ are defined, where n_i, n_s, and n_o are the dimensionalities of the spaces. In an SDTDS, a transition function $\gamma : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S} \times \mathcal{O}$ is also defined, which allows to model the state of the system and its output, given some input. In Section 3, we explain how probabilistic finite-state machines can be constructed from $\mathcal{I}, \mathcal{S}, \mathcal{O}$ and γ. From now, we explain the process of collecting data for quantization.

Definition 2. *An SDTDS transition event at a time t, $\omega(t)$ is a quadruple $\langle \mathbf{s}(t), \mathbf{i}(t), \mathbf{o}(t-1), \mathbf{o}(t), \mathbf{s}(t+1) \rangle \in$, where $\mathbf{s}(t+1)$ is the state vector reached after the SDTDS received input $\mathbf{i}(t)$ while occupying state $\mathbf{s}(t)$, and $\mathbf{o}(t)$ is the output generated in the transition [6]. Here, we also take into account previous output states (e.g. $\mathbf{o}(t-1)$) to consider a longer history of past events, with the intention to avoid the Markov property.*

In the following paragraph we propose a definition of the input space suitable for the pushing scenario.

Definition 3. *We define the input space $\mathcal{I}$ as a sensorimotor space where a set of tuples of vectors concatenated $\langle \mathbf{e}, \mathbf{a}, \mathbf{b} \rangle$ represent motor commands, finger poses and object poses respectively. For a time step t, $\mathbf{i}(t) = \langle \mathbf{e}, \mathbf{a}(t), \mathbf{b}(t-1) \rangle$. $\mathbf{b}(t-1)$ denotes the pose of the object at the previous time step.*

Likewise, we can define the state space as follows.

Definition 4. *A state space $\mathcal{S}$ is defined as a set of vectors representing object poses. For a time step t, $\mathbf{s}(t) = \mathbf{b}(t)$.*

As mentioned in Section 1, we can define different output functions to quantize an output space. Considering the problem of object trajectory prediction, we can encode an output space as a set of object transformations representing the rotation and translation of the object. In Section 3 we will clarify why this representation is useful for the purpose of object motion prediction.

Definition 5. *Let us define an output space $\mathcal{O}$ as a set of rotation vectors representing object motions. For time step t, $\mathbf{o}(t)$ is calculated as follows. Let* *us define a transformation* $T^{B(t),B(t+1)}$ *to denote the rotation and translation of an object, with corresponding rotation matrix* $\mathbf{R}_{B(t),B(t+1)} = \mathbf{R}_{B(t),O}^{\top}\mathbf{R}_{B(t+1),O}$ *and translation vector* $\mathbf{p}_{B(t),B(t+1)} = \mathbf{p}_{B(t+1),O} - \mathbf{p}_{B(t),O}$. *Then,* $\mathbf{o}(t)$ *is the corresponding rotation vector.*

In Fig. 4, we show a diagram of the features involved in the learning process.

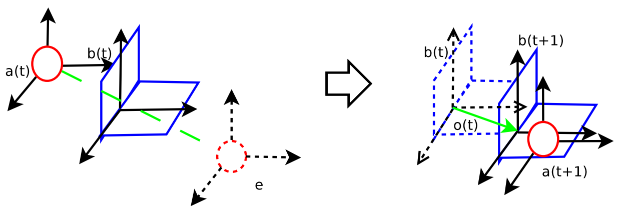


Figure 4: Features for prediction

Alternatively, we can define an output space $\mathcal{O}$ as a set of symbols representing a class or type of rough object behaviour. We carried out these experiments with simulated polyflaps. We consider here a discretization of possible object motions. For instance, $\mathcal{O} = \{-1, 1, 0, -0.5, 0.5\}$ is a set of possible values for an output symbol $o(t)$, denoting respectively:

- $o(t) = -1$ when θ object angle (with respect to Z axis) decreases (this happens when the object tilts but does not completely flips over, so that it returns to the original angle).
- $o(t) = 1$ when θ angle increases (object falling down)
- $o(t) = 0$ when the object does not move
- $o(t) = -0.5$ when the object moves backwards (negative direction along X axis).
- $o(t) = 0.5$ when the object moves forwards.

Additionally, we designed the output space to consider the final resulting motion, in abstract terms. In this case, we distinguished three possible behaviours, namely a slidding, a flipping over and a tilting behaviour. Combined with the discretization presented above, we have 15 possible output symbols.

2.2 Vector Quantization

The ability to estimate online a probability density function is important for robots to learn in an incremental way, after new data are available for learning. Before we start explaining our method, it is important to clarify that we use the term “online” here in contrast to batch, meaning that during a learning iteration where a new data source is present, the learning algorithm can be immediately updated. However, we will not use here the term online in the strict sense when a previous used data set is not anymore available for learning but only the new data items.

Our approach for density estimation is based on Vector Quantization [5] to partition probability distributions. Quantization divides the probability distribution in sets or clusters, by mapping a set of high-dimensional vectors to a prototype vector which is statistically similar. We have developed an efficient algorithm for Robust Quantization based on previous implementations of the Growing Neural Gas algorithm [3, 11, 12].

Definition 6. *A Growing Neural Gas (GNG) network [3] is a graph $\mathcal{G} = \langle A, \mathcal{C} \rangle$ where A is a set of M nodes, in which each node $c \in A$ has an associated weight $\mathbf{w}_c \in \mathbb{R}^d$. There exists a set of neighborhood connections $\mathcal{C}$ which are unweighted and symmetric. For each node c, there exists a possibly empty set of neighborhood connections $N_c = \{i \in A \mid (c, i) \in \mathcal{C}\}$. A GNG network can be used as a quantizer, by using the weights and indices associated to nodes in A as the prototype vectors and indices respectively.*

In the original algorithm [3], the network starts with 2 nodes and gradually adds more nodes in regions where a global error measure is high, after it reaches a given maximum number of nodes. The algorithm has issues related to proper quantization and overfitting. In [11, 12], some techniques were implemented to improve the convergence of the algorithm for efficient quantization and noise robustness. The algorithm presented here builds upon this work. We improve the efficiency in terms of computational complexity, and propose new strategies which are more suitable for online learning.

In [11, 12], the authors propose strategies to overcome the problems already mentioned. We applied some of these strategies and adapted it to a new algorithm. One of these methods is an outlier resistant strategy that reduces the influence of outliers in the node weight adaptation process. Additionally, they propose an information-theoretic method for evaluating the model complexity, namely the Minimum Description Length (MDL) criterion. In this way, models with a minimal MDL can be selected which optimize the quantization.

In the algorithms presented in [11, 12] parameters like the maximum number of nodes and the maximum number of iterations for a training epoch are still used. In this article, we propose a new algorithm that is less dependent on a priori parameters. It incorporates also an online calculation of error suitable for setting learning rates and to decide locations for new nodes to insert. The procedure can also decide when to stop according to the MDL principle and a notion of network stability. For the purposes of online estimation of a probability density function, we are interested in updating the quantizer after a new training instance is available. These training instances are actually sequences of vectors containing sensorimotor information. The stopping decision based on MDL and graph stability is here specially useful.

In this work, we incorporated modifications in learning rates calculation, insertion criteria for nodes, and efficient implementation of nodes deletion, and MDL-based and network stability stopping criteria.

In the following sections, we describe and explain the calculation of the different parameters involved in the learning process: the learning rates, the weight updates, the node insertion and deletion criteria and the Minimum Description Length calculation.

2.3 Learning Parameters

2.3.1 Learning rates and weights update

In this work, we propose a new method for obtaining learning rates which is dependent on an instantaneous calculation of mean error difference.

We obtain a harmonic mean error rate. This way we obtain error measures that are not strongly influenced by outliers. Assuming a data point $\mathbf{x}(t)$ is presented to the network and the corresponding winner weight $\mathbf{w}_{s_1}(t)$ is activated, the inverse error $e_{s_1}(t)$ associated to the node s_1 is:

$$e_{s_1}(t) = \|\mathbf{x}(t) - \mathbf{w}_{s_1}(t)\|^{-1} \tag{1}$$

In this case, we assume $\|\cdot\|$ to be the Euclidean norm. Given a time window parameter τ during which a node s_1 was selected as winner node, a harmonic smoothed mean error rate for the node s_1 at the current time step t and at a previous time step $t-\tau$ is calculated as follows:

$$\begin{aligned} \langle e_{s_1}(t)\rangle &= \left(\tfrac{1}{\theta+1}\textstyle\sum_{i=0}^{\theta} e_{s_1}(t-i)\right)^{-1} \\ \langle e_{s_1}(t-\tau)\rangle &= \left(\tfrac{1}{\theta+1}\textstyle\sum_{i=0}^{\theta} e_{s_1}(t-i-\tau)\right)^{-1}, \end{aligned} \tag{2}$$

where θ is a smoothing parameter.

For every node $k \in N_{s_1} \cup \{s_1\}$, the learning rate $\eta_k(t)$ is obtained in the following way:

$$\eta_k(t) = \begin{cases} \eta_k & \text{if } -(\log\langle e_k(t)\rangle - \log\langle e_k(t-\tau)\rangle) > 1 \\ -(\log\langle e_k(t)\rangle - \log\langle e_k(t-\tau)\rangle)\eta_k & \text{if } -(\log\langle e_k(t)\rangle - \log\langle e_k(t-\tau)\rangle) > 0.1 \\ 0.1\eta_k & \text{otherwise,} \end{cases} \tag{3}$$

whereby the default learning rate η_k is modulated by the the error difference $-(\log\langle e_k(t)\rangle - \log\langle e_k(t-\tau)\rangle)$. We call this value a learning quality measure. This function is depicted in Figure 5 for $\eta_k = 0.5$ and serves to moderate the impact of the default learning rate for weights update.

In the weight adaptation rule we also incorporated some strategies for outlier resistance based on the work described in [11]. The new proposed rule has the form:

$$\Delta\mathbf{w}_k(t) = \eta_k(t)\sigma_k(t)\frac{\mathbf{x}(t) - \mathbf{w}_k(t)}{\|\mathbf{x}(t) - \mathbf{w}_k(t)\|}, \tag{4}$$

where $\eta_k(t)$ is the learning rate in Eq. 3 and $\sigma_k(t)$ is the parameter for outlier resistance. From now on, we assume the time variable $t = 0$ at the beginning

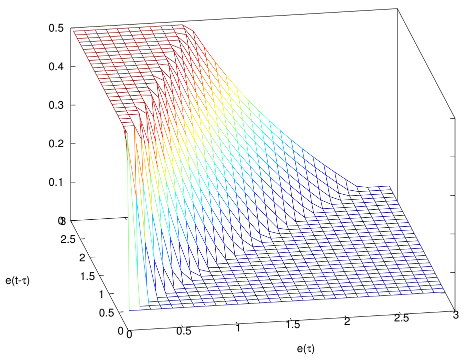


Figure 5: Learning quality function.

of a growth stage, that is, when a new node is added. The factor $\sigma_k(t)$ is used instead of the absolute distance information $\|\mathbf{x}(t) - \mathbf{w}_k(t)\|$, to mitigate the influence of outliers in the weight adaptation process. $\sigma_k(t)$ is calculated by using a historical restricting distance information $d_k(t)$ [11]:

$$\sigma_k(t) = \begin{cases} d_k(t) & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| \geq d_k(t-1) \\ \|\mathbf{x}(t) - \mathbf{w}_k(t)\| & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| < d_k(t-1) \end{cases} \tag{5}$$

where $d_k(t)$ serves as a restricting distance for $\mathbf{w}_k(t)$. $d_k(t)$ is updated when k becomes the winner node, i.e., when $k(t) = s_1(t)$:

$$d_k(t) = \begin{cases} \left(\frac{1}{2}\left(d_k(t-1)^{-1} + \|\mathbf{x}(t) - \mathbf{w}_k(t)\|^{-1}\right)\right)^{-1} & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| \geq d_k(t-1) \\ \frac{1}{2}\left(d_k(t-1) + \|\mathbf{x}(t) - \mathbf{w}_k(t)\|\right) & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| < d_k(t-1), \end{cases} \tag{6}$$

and

$$d_k(0) = \left(\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{w}_k^0\|^{-1}\right)^{-1}, \tag{7}$$

where $\mathbf{w}_k^0$ is the initial weight at the beginning of the growth stage and $N = |\mathcal{D}|$. Thus, $d_k(t)$ is initialized at the beginning and reinitialized when a new node is added to the network or when one or more nodes are deleted. We added a value $\varepsilon = 10^{-2}$ to Euclidean distances in order to avoid very big restricting distance values. The final updating rule is then:

$$\Delta\mathbf{w}_k(t) = \eta_k(t)\sigma_k(t)\frac{\mathbf{x}(t)-\mathbf{w}_k(t)}{\|\mathbf{x}(t)-\mathbf{w}_k(t)\|} \tag{8}$$

2.3.2 Criterion for node insertion

Definition 7. *The node insertion requirement is fulfilled if for all nodes $c \in A$ the mean error rate e_c is not reduced after T_e learning epochs.*

A new node is added when the node insertion requirement is satisfied. The node is inserted in the proximity of a node q with maximal insertion criterion and its neighbor f with highest insertion criterion. An insertion criterion K_c is defined in our work simply as the highest mean error rate:

$$\begin{aligned} K_q(t) &= \arg\max_{c \in A}(\langle e_c(t) \rangle) \\ K_f(t) &= \arg\max_{c \in N_q}(\langle e_c(t) \rangle) \end{aligned} \tag{9}$$

in our method the location of the new prototype r is calculated from the location of q and the direction of the mean average error vector $\langle \mathbf{e_q} \rangle$ associated to this node. This quantity is calculated in a similar way as in Eq. 2. When the data point $\mathbf{x}(t)$ is presented to the network, the mean error rate vector of the winner node s_1 is calculated:

$$\langle \mathbf{e}_{s_1}(t) \rangle = \left(\frac{1}{\theta + 1} \sum_{i=0}^{\theta} \mathbf{e}_{s_1}(t - i) \right)^{-1}, \tag{10}$$

with error vector $\mathbf{e}_{s_1}(t) = (\mathbf{x}(t) - \mathbf{w}_{s_1}(t))^{-1}$, where each component of $\mathbf{a}^{-1}$ is the inverse of the corresponding component of some vector $\mathbf{a}$. Thus, we set the weight for an inserted node r whose parent node is q as:

$$\mathbf{w}_r(t) = \mathbf{w}_q(t) + 2\langle \mathbf{e}_q(t) \rangle. \tag{11}$$

In some cases, a dislocated node will be deleted according to a criterion explained in Section 2.3.4. Moreover, nodes are occassionally deleted at the end of the learning stage.

2.3.3 Minimum Description Length Principle

We use a Minimum Description Length criterion as proposed in [12] to determine the optimal number of clusters. It is an information-theoretic measure that balances the complexity of the graph and its error.

Definition 8. *Given a data set $\mathcal{D}$ and the set of prototype node weights $\mathcal{W}$, the MDL is defined as [12]:*

$$MDL(\mathcal{D}) = modelL(\mathcal{D}, \mathcal{W}) + errorL(\mathcal{D}, \mathcal{W}), \tag{12}$$

where $errorL(\mathcal{D}, \mathcal{W})$ is the total encoding length or model efficiency and $modelL(\mathcal{D}, \mathcal{W})$ is the model complexity.

The total encoding length and the model complexity for a network $\mathcal{G}$ with M nodes are calculated as follows:

$$\begin{aligned} \text{errorL}(\mathcal{D}, \mathcal{W}) &= \kappa \sum_{i=1}^{M} \sum_{\mathbf{x} \in S_i} \sum_{k=1}^{d} \max \left(\log_2 \left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon} \right), 1 \right) \\ \text{modelL}(\mathcal{D}, \mathcal{W}) &= KM + N \log_2 M, \end{aligned} \tag{13}$$

where $N = |\mathcal{D}|$, d is the dimension of input vectors, ε is a data accuracy constant usually set to 10^{-4}. K is the number of bits needed to encode a single data vector, which is obtained according to the average value range ν and data accuracy ε: $K = \lceil \log_2 \left(\frac{\nu}{\varepsilon}\right) \rceil$. Finally, κ is a parameter to balance the contribution of the model complexity and model efficiency, which is here usually set to 1.3. By setting $\kappa > 1$, we give more weight to the network accuracy in terms of error. The value range is calculated by obtaining the average value in the data set and substracting its lower limit.

2.3.4 Criteria for node deletion

In our work, a node is called inactive and can be deleted if at the end of a growth stage it is not a representative for any vector in the dataset. Usually, new prototypes adapt quickly so that they approach regions containing data. However, in some cases clusters might be already represented by other prototypes and the node might stay isolated from clusters in some cases.

We use the MDL criterion to assess whether node are dislocated at the end of the learning process. Assuming that one prototype f is removed, if the MDL value calculated based on the set of nodes $\mathcal{C} \setminus \{f\}$ is smaller than that of $\mathcal{C}$, i.e., if:

$$\begin{aligned}\Delta\mathrm{MDL}(\mathcal{D}, f) \quad = \quad & -K + N(\log_2(M-1) - \log_2(M) \\ & +\kappa \left(\sum_{i=1, i\neq f}^{M} \sum_{\mathbf{x}\in S_i} \sum_{k=1}^{d} \max\left(\log_2\left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon}\right), 1\right) \right. \\ & \left. - \sum_{i=1}^{M} \sum_{\mathbf{x}\in S_i} \sum_{k=1}^{d} \max\left(\log_2\left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon}\right), 1\right) \right) < 0,\end{aligned} \tag{14}$$

we regard the removed prototype f as dislocated. This process is performed recursively until after there are no more nodes dislocated.

2.3.5 Robust GNG Algorithm

We implemented the learning algorithm in such a way that it can decide when to stop. This stopping criterion is based on evaluating the MDL after a number of T_m learning epochs after which no minimal graph has been found, in terms of MDL. The learning process is described in Algorithm 1.

Definition 9. *A graph is stable if the MDL criterion is not reduced after T_m learning epochs.*

Algorithm 1: RobustGNG($\mathcal{D}, \mathcal{G}$)

```
Data: A data set $\mathcal{D}$ and a set of 2 not connected nodes $\{c_1, c_2\} \in A$, whose weights are
      initialized randomly considering the data set bounds. Set the constants defined in
      Sections 2.3.1 and 2.3.3. Initialize restricting distances as explained in Eq. 7.
      Initialize smoothed mean error rates with the highest distance among the data set.
Result: A graph $\mathcal{G}$ which is stable
begin
    while Graph $\mathcal{G}$ is not stable do
        for $t = 1$ to $N$ do
            Randomly draw a vector $\mathbf{x}(t) \in \mathcal{D}$;
            Determine winner $s_1$ and second winner $s_2$, where
            $s_1 = \arg\min_{i \subset A} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$ and $s_2 = \arg\min_{i \subset A \setminus \{s_1\}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$;
            Update the weights of nodes $k \in N_{s_1} \cup \{s_1\}$ by using Eq. 8;
            if $t \mod \lambda = 0$ where $\lambda$ is a time window constant then
                Calculate MDL using Eqs. 12,13;
                if current MDL is minimal then
                    Store current graph $\mathcal{G}$ as the graph with minimal MDL $\mathcal{G}_{\min}$;
                if current graph is stable then
                    Store $\mathcal{G}_{\min}$ as the resulting stable graph;
                    Finish the algorithm here;
                if insertion requirement is fulfilled (Sec. 2.3.2) then
                    Delete any nodes if necessary according to Section 2.3.4. If a node is
                    deleted, initialize restricting distances $d_k$ and recalculate MDL.
                    Check if MDL is minimal and store it accordingly. Check if the graph
                    is stable, in which case the algorithm finishes;
                    Determine nodes $q$ and $f$ with maximal insertion criterions $K_q$ and
                    $K_f$ according to Eq. 9;
                    Insert a new prototype $r$ and set its reference vector as in Eq. 11. Set
                    initial smoothed mean error for $r, q$, and $f$: $e_r = e_q = e_f = 0$;
                    Insert edges connecting the new prototype $r$ with prototypes $q$ and $f$,
                    removing the original one: $\mathcal{C} = \mathcal{C} \cup \{(r,q),(r,f)\}$, $\mathcal{C} = \mathcal{C} \setminus \{(q,f)\}$;
                    Initialize restricting distances as explained in Eq. 7;
            Update the smoothed harmonic mean error rates according to Eq. 2;
            Update the restricting distances $d_k$ as in Eq. 6;
            if a connection between $s_1$ and $s_2$ does not exist already and $s_1$ or $s_2$ has
            not been deleted then
                Create connection: $\mathcal{C} = \mathcal{C} \cup \{s_1, s_2\}$;
                Set the age of the connection $\{s_1, s_2\}$ to 0: $\text{age}_{\{s_1,s_2\}} = 0$;
                Increment the age of all edges emanating from $s_1$:
                $\text{age}_{\{s_1,i\}} = \text{age}_{\{s_1,i\}} + 1, \forall i \in N_{s_1}$;
            Remove edges with age values greater than a constant $\alpha$;
            Remove all nodes without any edge and in such case initialize restricting
            distances;
    Check dislocated nodes according to 2.3.4;
    return $\mathcal{G}$
end
```

2.4 Online Learning

After a new traning instance is available, RobustGNG is employed for quantization of input and output spaces. However, to make this process efficient, the algorithm divides the spaces into subregions. This process is based on previous works [10, 15].

Starting with one region, successive regions are obtained by splitting the sensorimotor space depending on a measure of variance in the dataset $\mathcal{D}_r$ (exemplars used for Region $\mathcal{R}_r$). This division is performed after $|\mathcal{D}_r|$ achieves a certain threshold κ. A dataset $\mathcal{D}_r$ for a Region R_r is split in two datasets

$\mathcal{D}_{r+1}, \mathcal{D}_{r+2}$ (for regions $\mathcal{R}_{r+1}, \mathcal{R}_{r+2}$). Let us denote

$$\mathcal{D}_r = \{S_i\}$$

the set of instances in region $\mathcal{R}_r$. Then the split of $\mathcal{D}_r$ defined by the index c with value v_c is performed when the following criterion (Γ) is met:

- all the instances S_i of $\mathcal{D}_{r+1}$ have the cth component of their motor command vector $\mathbf{e}_i$ smaller than v_c.
- all the instances S_i of $\mathcal{D}_{r+2}$ have the cth component of their motor command vector $\mathbf{e}_i$ greater than v_c.
- the quantity $|\mathcal{D}_{r+1}| \cdot \sigma(\{[\mathbf{a}_{ij}\ \mathbf{b}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+1}\}) + |\mathcal{D}_{r+2}| \cdot \sigma(\{[\mathbf{a}_{ij}\ \mathbf{b}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+2}\})$ is minimal, where

  $$\sigma(\mathcal{S}) = \frac{\sum_{v \in \mathcal{S}} \|v - \frac{\sum_{v \in \mathcal{S}} v}{|\mathcal{S}|}\|^2}{|\mathcal{S}|}$$

  where $\mathcal{S}$ is a set of vectors.

Each region stores all cutting dimension and values that were used in its generation as well as in the generation of its parent regions. For the region $\mathcal{R}_r$ a quantizer Λ_r is stored, and this machine is inherited by the child regions. The learning process is described in the Algorithm 2.

Algorithm 2: Online learning process

```
Data: An initial region $\mathcal{R}_0$ which encompasses the whole sensorimotor
      space.
Result: A set of regions $\{\mathcal{R}_r\}$ with corresponding input and output
        quantizers $\{\Lambda_{i,o}\}$.
for $i$=1 to $I$ do
    Choose a motor command action $\mathbf{e}_{r,i}$ among all current regions $\{\mathcal{R}_r\}$.
    if $\kappa$ then
        Split region $\mathcal{R}_r$ into $\mathcal{R}_{r+1}$ and $\mathcal{R}_{r+2}$ according to $\Gamma$.
    end
    Update quantizers with current training sequence $S_{r,i}$.
end
```

3 Induction of Substochastic Sequential Machines for Prediction and Classification

3.1 Substochastic Sequential Machines

Definition 10. *A substochastic sequential machine (SSM) is a quadruple $\langle Q, X, Y, \mathcal{P} = \{p(q_j, y_l | q_i, x_k)\}\rangle$ where Q is a finite set of state elements (SEs), X is a fi-*

nite set of input symbols, Y is a finite set of output symbols, and $\mathcal{P}$ is a finite set of conditional probabilities (cf. explanation in [6] and eqs.15-17) where $q_i, q_j \in Q, x_k \in X$ and $y_l \in Y$.

We use the vector quantization method described in Section 2 for state discretization. A SSM models a situated discrete time dynamical system (SDTDS). A stochastic dynamical model of such a system is a joint probability mass function p_Ω induced from a transition event set Ω, and quantizer functions Λ_o, Λ_i and Λ_s for output, input and state spaces respectively. Ω consists of selected transition events recorded from a given set of input sequences. Thus, the joint probabilities of observed and quantized transitions (p_Ω) are translated into joint probabilities of SSM transitions according to $\mathcal{P}$. As already mentioned, we define $\Lambda_i(\mathbf{i}(t))$ and $\Lambda_o(\mathbf{o}(t))$ according to the discretization described in Section 2, and $\Lambda_s(\mathbf{s}(t))$ using a modified version of the original state space quantization method (Crystalline Vector Quantizer - CVQ) explained in [6]. Thus, we have:

$$\begin{gathered} p(q_i, x_k, y_l, q_j) = \\ p_\Omega(\Lambda_s(\mathbf{s}(t)) = i, \Lambda_i(\mathbf{i}(t)) = k, \Lambda_o(\mathbf{o}(t)) = l, \Lambda_s(\mathbf{s}(t+1)) = j) \end{gathered} \tag{15}$$

The conditional probability is calculated with:

$$p(q_i, x_k) = \sum_{j=1}^{|Q|} \sum_{l=1}^{|Y|} p(q_i, x_k, y_l, q_j) \tag{16}$$

$$p(q_j, y_l | q_i, x_k) = \begin{cases} \frac{p(q_i, x_k, y_l, q_j)}{p(q_i, x_k)} & \text{if } p(q_i, x_k) > 0 \\ 0 & \text{if } p(q_i, x_k) = 0 \end{cases} \tag{17}$$

Definition 11. *The translation procedure from Ω to an SDTDS and then into an SSM will be called create_machine$(\Omega, \Lambda_s, \Lambda_i, \Lambda_o)$.*

The substochasticity of the extracted machines is due to the possibility that the sample of input sequences in Ω will not necessarily provide examples of all possible input symbols in all possible enumerations of the quantized space of the dynamical system. As a consequence, the probability distributions can become substochastic [6]. The details of the procedure for extracting substochastic sequential machines is described in [6]. In summary, there is a recursive state splitting, starting from only one SE. Then, a decision to split data into different SEs is based primarily on the maximal output entropy $\arg\max H(Y|Q = q_i, X = x_k) = H(\mathcal{P}_y(q_i, x_k))$ and then on the maximal next state entropy $\arg\max H(Q|Q = q_i, X = x_k) = H(\mathcal{P}_q(q_i, x_k))$. This yields that state vectors that convey the most information (i.e., highly indeterministic) are used for splitting [6]. Here, $H(\mathcal{P}) = -\sum_{i=1}^{n} p_i \log p_i$ and $p(q(t+1)) = \mathcal{P}_q(q_i, x_k)$ and $p(y(t)) = \mathcal{P}_y(q_i, x_k)$ are marginal distributions of $\mathcal{P}$. Each split node has associated model vectors that point to other split states, merged ones, or leaf nodes. The model vectors are calculated from the average of the vectors which they represent. Additionally, states are possibly merged if there exists an equivalence relation between two states based on determining when two SEs are not equivalent if they, in their outgoing transitions, share some input symbols and transitions that lead to discrepancies in the future output. The procedure finishes when the machine is deterministic, i.e., when the entropies for all states equal to 0.

3.2 An improved CVQ

As mentioned above, the quantization procedure for the state space is based on the CVQ quantizer. This method has similarities with hierarchical decision trees [6]. In this work, we redefined the quantization procedure of a CVQ with respect to [6], in order to solve issues regarding symmetrical properties present in the evolution of some dynamical systems (see Fig. 6 discussed below). [14] provides a detailed description of these issues. A CVQ is defined as follows [6]:

Definition 12. *A CVQ Graph is a quadruple $CVQ = \langle N_{leaf}, N_{VQ}, N_{Merged}, n_{root}\rangle$ where n_{root} is the root node of the CVQ Graph and the constituents are defined below.*

Definition 13. *A leaf node $n \in N_{leaf}$ has only one constituent, $n = \langle i\rangle$, where $i \in \mathbb{N}$ is an enumeration of the node within the CVQ and $1 \leq i \leq |N_{leaf}|$.*

Definition 14. *A Vector Quantizer (VQ) node $n \in N_{VQ}$ is a tuple $n = \langle \mathcal{M}, \mathcal{H}\rangle$ where $\mathcal{M}$ is a list of L model vectors $[\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_L]$, where $\mathbf{m}_i \in \mathbb{R}^d$ and $\mathcal{H}$ is a nonrepetitive list of child nodes $[h_1, h_2, \dots, h_L]$ where $h_i \in N_{Leaf} \cup N_{VQ} \cup N_{Merged}$ and $d \in \mathbb{N}$ is the dimensionality of the vector space.*

Definition 15. *A merged node in a CVQ graph, $n \in N_{Merged}$, contains only a "link", $n = \langle n_{group}\rangle$, where $n_{group} \in N_{Leaf} \cup N_{VQ} \cup N_{Merged}$.*

The model vectors in a VQ-Node have associated a list $[\ell_1, \ell_2, \dots, \ell_L]$ of classifications (labels). An element ℓ_i is a tuple $\langle y(t_i), y(t_i + 1)\rangle$ which are the output symbols observed at some time step t_i and the subsequent $t_i + 1$. In this work, we redefine a CVQ quantizer function Λ_{cvq} in terms of a function winner : $N_{Leaf} \cup N_{VQ} \cup N_{Merged} \times \mathbb{R}^d \times \langle \mathbb{N}, \mathbb{N}\rangle \to \{1, 2, \dots, M\}$:

$$\Lambda_{cvq}(\mathbf{s}(t)) = \text{winner}(n_{root}, \mathbf{s}(t), \langle y(t), y(t+1)\rangle), \tag{18}$$

which in turn is recursively defined as:

$$\text{winner}(n, \mathbf{s}(t), \langle y(t), y(t+1)\rangle) = \begin{cases} ID & \text{if } n \in N_{Leaf} \\ \text{winner}(n_{group}, \mathbf{s}(t), \langle y(t), y(t+1)\rangle) & \\ & \text{if } n \in N_{Merged} \\ \text{winner}(h_w, \mathbf{s}(t), \langle y(t), y(t+1)\rangle) & \\ & \text{if } n \in N_{VQ}, \end{cases} \tag{19}$$

where we determine w, the index of the winning child of a VQ-node according to:

$$w = \arg\min \|\mathbf{s}(t) - \mathbf{m}_i\|, \quad \text{s.t. } \ell_i = \langle y(t), y(t+1)\rangle. \tag{20}$$

The symmetry problem can be visualized in Fig. 6, where two model vectors with identical geometrical locations have different associated outputs. Thus, depending on the context, one of these model vectors have to be selected to eliminate the ambiguity.

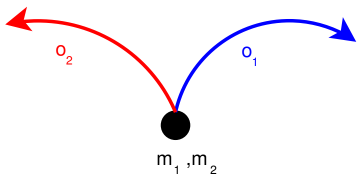


Figure 6: The symmetry issue in model vectors

We refer to [6] for more detailed explanations of CVQ training. When a CVQ leaf node is completely split [6], a recursive method is devised that splits data points in regions that separate them on the basis of labels for vectors as seen above. Model vectors are determined by averaging the data points in a region. Given a slightly different labeling present in this work, we reformulate the complete split in Def. 16. In Section 3.3 we describe the procedure for labeling vectors when splitting a data set.

Definition 16. *The complete split of several VQ nodes using several data sets at once is denoted $cvq = split\_cvq(cvq, D)$ where cvq is the CVQ to be split and $D = [D_1, D_2, \ldots, D_{|\Lambda_{cvq}|}]$ is a list of data sets where D_i is the data set for splitting the leaf node with $ID = i$ (if the node should not be split, then $D_i = \emptyset$). The elements of a data set are pairs $\langle \mathbf{s}, \ell \rangle$ where $\mathbf{s}$ is the data vector and $\ell \in \langle \mathbb{N}, \mathbb{N} \rangle$ is a label or class of the data vector. The leaf nodes are re-enumerated after the completion of all splits.*

3.3 CrySSMEx learning loop

The principal components of the CrySSMEx algorithm are listed below [6]:

- the SDTDS which represents the class of systems for CrySSMEx to analyse.
- the data set, i.e., the SDTDS transition event set Ω.
- SSMs, a subtype of SDTDSs.
- SDTDS transformation into SSM by quantizing input, output and state (cf. Sections 2 and 3.2).
- generation of UNDI-equivalence (universally not decisively inequivalent) sets in SSMs, which is the process that helps to determine when to merge states (function generate_UNDI_equivalence_sets).

- Use of CVQ as a state space quantizer (cf. Section 3.2).
- merging (function merge_cvq) and splitting of CVQ leaf nodes (cf. Definition 16).
- Selection and labeling of state vectors of Ω based on SSM information-theoretic properties (Algorithm 3).

Algorithm 3: collect_split_data($\Omega, \Lambda_i, \Lambda_s, \Lambda_o$)

```
Data: A transition event set, $\Omega$, an SSM, $ssm$, an input quantizer, $\Lambda_i$, a state
      quantizer, $\Lambda_s$ and an output quantizer, $\Lambda_o$.
Result: A list of data sets $D$, one data set per $q \in Q$. An element of each data set is
        described in Def. 16.
begin
    $D = [\emptyset, \emptyset, \dots, \emptyset]$;
    for $\forall \langle \mathbf{s}(t), \mathbf{i}(t), \mathbf{o}(t), \mathbf{s}(t+1) \in \Omega$ do
        $q_i = \Lambda_s(\mathbf{s}(t))$;
        $x_k = \Lambda_i(\mathbf{i}(t))$;
        $y_l = \Lambda_o(\mathbf{o}(t))$;
        $y_m = \Lambda_o(\mathbf{o}(t+1)$;
        $q_j = \Lambda_s(\mathbf{s}(t+1)$;
        if $\exists x_r : H_{ssm}(Y|Q = q_i, X = x_r) > 0$ then
            $x_{max} = \arg\max_{x_r \in X} H_{ssm}(Y|Q = q_i, X = x_r)$;
            if $x_k = x_{max}$ then
                $D_i = D_i \cup \langle \mathbf{s}(t), \langle y_l, y_m \rangle \rangle$;
        else if $\exists x_r : H_{ssm}(Q|Q = q_i, X = x_r) > 0$ then
            $x_{max} = \arg\max_{x_r \in X} H_{ssm}(Q|Q = q_i, X = x_r)$;
            if $x_k = x_{max}$ then
                $D_i = D_i \cup \langle \mathbf{s}(t), \langle y_l, y_m \rangle \rangle$;
    return $D$
end
```

In Algorithm 3, data sets are first split if output is indeterministic. If not, but the next state is not deterministic, data sets are then split and labeled accordingly. Finally, an improved version of the CrySSMEx main loop can be observed in Algorithm 4.

Algorithm 4: CrySSMEx($\Omega, \Lambda_i, \Lambda_o$)

```
Data: An SDTDS transition event set, $\Omega$, an input space quantizer $\Lambda_i$, an output
      space quantizer $\Lambda_o$.
Result: A deterministic SSM mimicking the SDTDS within the domain $\Omega$ as
        described by $\Lambda_o$.
begin
    $i = 0$;
    $ssm^0$ = create_machine($\Omega, \Lambda_i, \Lambda_{cvq^0}, \Lambda_o$) ($ssm^0$ has $Q = \{q_1\}$ with all transitions
    to itself);
    repeat
        $i = i + 1$;
        $D$ = collect_split_data($\Omega, ssm^{i-1}, \Lambda_i, \Lambda_{cvq^{i-1}}, \Lambda_o$);
        $cvq^i$ = split_cvq($cvq^{i-1}, D$);
        $ssm^i$ = create_machine($\Omega, \Lambda_i, \Lambda_{cvq^i}, \Lambda_o$);
        if $ssm^i$ has UNDI-equivalent states then
            $E$ = generate_UNDI_equivalence_sets($ssm^i$);
            $cvq^i$ = merge_cvq($cvq^i, E$);
            $ssm^i$ = create_machine($\Omega, \Lambda_i, \Lambda_{cvq^i}, \Lambda_o$);
    until $ssm^i$ is deterministic ;
    return $ssm^i$
end
```

For every Region $\mathcal{R}_r$, we executed the CrySSMEx algorithm, obtaining one SSM associated to some region.

4 Experimental Results

In the learning scenario, the arm starts a pushing action from 18 different poses. The robot applies a pushing angle ranging from 60 to 120 degrees, parallel to the ground plane in the direction to the object center, as shown in Fig. 7, where the spheres show the starting poses of the finger.

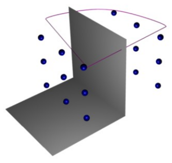


Figure 7: Pushing actions.

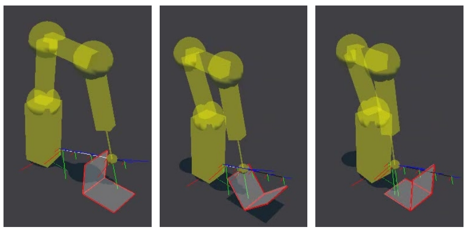
To test the prediction results, we applied the prediction process explained in [6], which is in principle a parsing process. In order to obtain the model vectors from the CVQ and calculate the normalized average error in prediction,

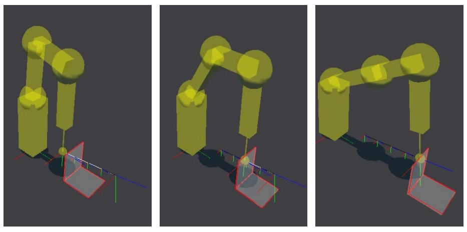
Table 1: Normalized average error in prediction.

Data size	Avg Norm. Error
100	0.0487653
200	0.0340824
500	0.0237009

we obtained a map of model vectors and corresponding quantization indices. In our experiments, the prediction is carried out for only one step ahead.

We performed training experiments with 100, 200 and 500 sequences. Table 1 shows the normalized average error in prediction for the three cases, by using a different set of 500 testing sequences. In Fig. 8 we illustrate the prediction results.


(a) Predicting the flipping over affordance


(b) Predicting the sliding affordance

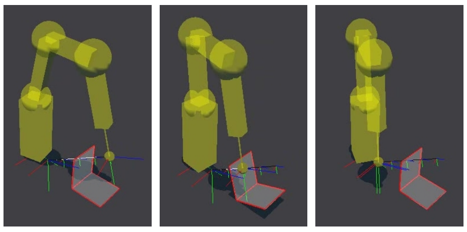

(c) Predicting the tilting affordance

Figure 8: Prediction of affordances. Red countour depicts last prediction.

We tested also the results for classification. For 100 sequences, the algorithm is already able to predict the correct object behaviour (sliding, flipping over, tilting) in all cases.

5 Conclusions

We investigated the problem of predicting the object behaviour after a robot pushes it in terms of trajectory estimation and behaviour categorization. We applied an online learning algorithm (RobustGNG) to quantize the sensorimotor space after a new training sequence is available. This algorithm is incremental and allows to add or delete prototypes representing clusters in the probability distributions of these spaces in an online manner. Stopping criteria are crucial to decide the right number of latent variables. Information-theoretic criteria like Minimum Description Length were used for these purposes. We used also a divide-and-conquer strategy to split the sensorimotor spaces in different regions, in order to accelerate convergence. Then, the quantized spaces were used for obtaining qualitative models of object behaviour and classification in form of substochastic finite-state machines, by applying the CrySSMEx algorithm. The prediction and classification ability of the learning algorithms were demonstrated experimentally.

References

[1] I Atil, N Dag, Sinan Kalkan, and E Sahin, *Affordances and Emergence of Concepts*, 10th International Conference on Epigenetic Robotics, 2010.

[2] Stefan L. Frank and Henrik Jacobsson, *Sentence-processing in echo state networks: a qualitative analysis by finite state machine extraction*, Connection Science **22** (2010), no. 2, 135–155.

[3] Bernd Fritzke, *A growing neural gas network learns topologies*, Advances in Neural Information Processing Systems 7, MIT Press, 1995, pp. 625–632.

[4] J. J. Gibson, *The theory of affordances*, Perceiving, Acting, and Knowing: Toward an Ecological Psychology (R. Shaw and J. Bransford, eds.), Lawrence Erlbaum, 1977, pp. 67–82.

[5] Siegfried Graf and Harald Luschgy, *Foundations of quantization for probability distributions*, Lecture notes in mathematics ; 1730, Springer, Berlin, 2000.

[6] H. Jacobsson, *The crystallizing substochastic sequential machine extractor - CrySSMEx*, Neural Computation **18** (2006), no. 9, 2211–2255.

[7] M. Kopicki, J. Wyatt, and R. Stolkin, *Prediction learning in robotic pushing manipulation*, Proceedings of the 14th IEEE International Conference on Advanced Robotics (ICAR 2009) (Munich, Germany), June 2009.

[8] Matej Kristan and Ales Leonardis, *Online discriminative kernel density estimation*, Pattern Recognition, International Conference on (Los Alamitos, CA, USA), vol. 0, IEEE Computer Society, 2010, pp. 581–584.

[9] Norbert Krueger, Justus Piater, Florentin Wörgötter, Christopher W Geib, Ronald P A Petrick, Mark Steedman, Ales Ude, Tamim Asfour, Dirk Kraft, Damir Omrcen, Bernhard Hommel, Alejandro Agostini, Danica Kragic, Jan-Olof Eklundh, Volker Krueger, Carme Torras, and Ruediger Dillmann, *A Formal Definition of Object-Action Complexes and Examples at Different Levels of the Processing Hierarchy*, Archives of clinical neuropsychology the official journal of the National Academy of Neuropsychologists **21** (2009), no. 1, 1–39.

[10] P-Y. Oudeyer, F. Kaplan, and V. V. Hafner, *Intrinsic motivation systems for autonomous mental development*, IEEE Transactions on Evolutionary Computation **11** (2007), no. 2, 265–286.

[11] A.K. Qin and P.N. Suganthan, *Robust growing neural gas algorithm with application in cluster analysis*, Neural Networks **17** (2004), no. 8-9, 1135 – 1148, New Developments in Self-Organizing Systems.

[12] K Qin and N Suganthan, *Enhanced neural gas network for prototype-based clustering*, Pattern Recognition **38** (2005), no. 8, 1275–1288.

[13] B. Ridge, D Skočaj, and A Leonardis, *Unsupervised learning of basic object affordances from object properties*, Proceedings of the Fourteenth Computer Vision Winter Workshop (CVWW) (Eibiswald, Austria), February, 4–6 2009, pp. 21–28.

[14] S. Roa and G.-J. Kruijff, *Robust Vector Quantization for Inference of Substochastic Sequential Machines*, Tech. report, DFKI GmbH, 2011.

[15] S. Roa and G.-J.M. Kruijff, *Offline and active gradient-based learning strategies in a pushing scenario*, 19th European Conference on Artificial Intelligence 2010: 3rd International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems (ERLARS 2010), 2010.

[16] Aaron Sloman, *Polyflaps as a domain for perceiving, acting and learning in a 3-D world*, Position Papers for 2006 AAAI Fellows Symposium (Menlo Park, CA), AAAI, 2006.

[17] Alexander Stoytchev, *Learning the affordances of tools using a behavior-grounded approach*, Proceedings of the 2006 international conference on Towards affordance-based robot control (Berlin, Heidelberg), Springer-Verlag, 2008, pp. 140–158.

[18] Ron Sun, *Beyond simple rule extraction: The extraction of planning knowledge from reinforcement learners*, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 2 - Volume 2 (Washington, DC, USA), IJCNN '00, IEEE Computer Society, 2000, pp. 105–110.

[19] Emre Ugur, Erol Sahin, and Erhan Oztop, *Affordance learning from range data for multi-step planning*, Ninth International Conference on Epigenetic Robotics (EpiRob09), 2009.

17th Computer Vision Winter Workshop
Matej Kristan, Rok Mandeljc, Luka Čehovin (eds.)
Mala Nedelja, Slovenia, February 1-3, 2012

Relevance Determination for Learning Vector Quantization using the Fisher Criterion Score [*]

Barry Ridge, Aleš Leonardis, and Danijel Skočaj.
Faculty of Computer and Information Science,
University of Ljubljana, Slovenia
{barry.ridge, danijel.skocaj, ales.leonardis}@fri.uni-lj.si

Abstract. *Two new feature relevance determination algorithms are proposed for learning vector quantization. The algorithms exploit the positioning of the prototype vectors in the input feature space to estimate Fisher criterion scores for the input dimensions during training. These scores are used to form online estimates of weighting factors for an adaptive metric that accounts for dimensional relevance with respect to classifier output. The methods offer theoretical advantages over previously proposed LVQ relevance determination techniques based on gradient descent, as well as performance advantages as demonstrated in experiments on various datasets including a visual dataset from a cognitive robotics object affordance learning experiment.*

1. Introduction

Learning vector quantization (LVQ) [9] provides an intuitive, and often highly effective, means for discriminative learning where prototype vectors are used to quantize the input feature space and given labels to form piecewise-linear classifiers using the nearest neighbour rule. Since their introduction, LVQ algorithms have undergone various analyses and seen various improvements to their design. The original formulations *(LVQ1, LVQ2, LVQ3)* [9] have been shown to be divergent, inspiring the *generalized learning vector quantization (GLVQ)* algorithm [14] where prototypes are updated such that a stochastic gradient descent is performed over an error function. LVQ algorithms have also been shown to be a family of maximum margin classifiers [3], thus providing excellent generalization for novel data with high-dimensional inputs. More recently, the nearest neighbour rule of LVQ has been modified to a k-nearest neighbours rule using a local subspace classifier [7].

Perhaps just as significantly, much attention has also been paid in recent years to the role that the distance metric plays in the effectiveness of LVQ methods. LVQ ordinarily relies on the Euclidean metric to measure the distance between data points, which provides equal weighting to all input dimensions. Many of the input dimensions, however, may have little relevance when considering the desired output function and may even have a detrimental effect on the output if considered with equal weighting in the metric to the more important dimensions. One standard approach to this issue is to pre-process the data using some form of feature selection or dimensionality reduction, but this can be infeasible in many learning scenarios where the training data are not available in advance, e.g. autonomous robotics. Various reformulations of LVQ have been proposed that can adjust the metric during training such that the impact of the individual input dimensions are dynamically re-weighted during training in accordance with the data under consideration. This can make a crucial difference, both during training for more efficient adjustment of the prototypes, and when classifying test samples where the undue consideration of irrelevant dimensions can mean the difference between a correct and incorrect classification.

One early adaptation of LVQ3 known as *distinction sensitive learning vector quantization (DSLVQ)* [11] achieves this by using a heuristic to adjust weights along each of the input dimensions to modify the Euclidean metric. An adaptation of LVQ1 known as *relevance learning vector quantization (RLVQ)* [1] uses Hebbian learning to do similar, by adjusting weights for each of the input dimensions at every

[*]This research has been supported by: EU FP7 project CogX (ICT-215181), and Research program P2-0214 Computer Vision (Republic of Slovenia).

training step depending on whether they contributed to the correct or incorrect classification of a training sample. RLVQ was subsequently adapted for use with GLVQ producing a method known as *generalized relevance learning vector quantization (GRLVQ)* [6] such that the dimensional weight updates also adhere to gradient descent dynamics in a similar way to the prototype updates. Another modified version of GLVQ [15] uses Fisher's discriminant analysis to create an alternative metric to the weighted Euclidean distance that employs a matrix transformation to reduce the feature space dimensionality. More recently, an adaptive metric was used in combination with training data selection for LVQ [10].

In this paper, two new algorithms for LVQ-based relevance determination are presented. Both methods exploit the positioning of the prototype vectors in the input feature space to inform estimates of the Fisher criterion score along the input dimensions, which are then used to form online estimates of the relevance of the input dimensions with respect to the classifier output. Both methods provide online updates that may be used alongside regular LVQ updates and neither method requires the specification of a learning rate, as in stochastic gradient descent. The remainder of the paper is organized as follows. In Section 2 the background theory and related algorithms are outlined. The new algorithms are described in Section 3. Experimental results are provided in Section 4 and concluding remarks are provided in Section 5.

2. Related Algorithms

Let $X = \{(\mathbf{x}^i, y^i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \,|\, i = 1, \ldots, N\}$ be a training set of n-dimensional vectors and corresponding class labels. Let $X^c = \left\{(\mathbf{x}^i, y^i) \in X \,\middle|\, y^i = c\right\}$ and $N^c = |X^c|$. Similarly, let $\mathcal{W} = \{(\mathbf{w}^i, c^i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \,|\, i = 1, \ldots, M\}$ be a set of prototype vectors with corresponding class labels, and let $\mathcal{W}^c = \left\{(\mathbf{w}^i, c^i) \in \mathcal{W} \,\middle|\, c^i = c\right\}$ and $M^c = |\mathcal{W}^c|$. Given a vector $\mathbf{x} \in \mathbb{R}^n$, denote its components as $(x_1, \ldots, x_n)$. Letting $\mathbf{x}$ be an n-dimensional data vector and $\mathbf{w}$ be an n-dimensional prototype vector, then a weighted squared Euclidean distance between both vectors may be defined as

$$d^2(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} \lambda_i (x_i - w_i)^2, \tag{1}$$

where the λ_i are weighting factors for each dimension. Adding such weights to the Euclidean metric allows for the possibility of re-scaling each of the input dimensions depending on their respective influences on the classification output. Moreover, it enables the metric to be made *adaptive* such that the weights are adjusted dynamically during training depending on the data.

Prototype vectors have associated receptive fields based on the metric and classification of samples is performed by determining which receptive fields those samples lie in, or alternatively, which prototype vectors are closest to the samples. The receptive field of prototype $\mathbf{w}^i$ is defined as: $R^i = \{\mathbf{x} \in X \,|\, \forall (\mathbf{w}^j, c^j) \in \mathcal{W},\, d^2(\mathbf{x}, \mathbf{w}^i) \leq d^2(\mathbf{x}, \mathbf{w}^j)\}$. Given a sample $(\mathbf{x}, y) \in X$, we denote by $g(\mathbf{x})$ a function that is negative if $\mathbf{x}$ is classified correctly, i.e. $\mathbf{x} \in R^i$ with $c^i = y$, and is positive if $\mathbf{x}$ is classified incorrectly, i.e. $\mathbf{x} \in R^i$ with $c^i \neq y$. We also let f be some monotonically increasing function.

The goal of GLVQ [14] is to minimize

$$E = \sum_{i=1}^{m} f(g(\mathbf{x}^i)) \tag{2}$$

via stochastic gradient descent. The update rules for GLVQ and many other LVQ algorithms can be derived using the above notation. In the following, the LVQ1 [9], RLVQ [1], GLVQ [14] and GRLVQ [6] algorithms will be reviewed, before introducing the proposed relevance determination methods.

2.1. LVQ1

Given a training sample $(\mathbf{x}, y) \in X$, by letting $f(x) = x$ and $g(\mathbf{x}) = \eta d_j$ where $d_j = d^2(\mathbf{x}, \mathbf{w}^j)$ with $\mathbf{w}^j$ being the closest prototype to $\mathbf{x}$ and $\{\lambda_i = 1\}_{i=1}^{m}$ (i.e. equal weights for regular Euclidean distance), with $\eta = 1$ if $\mathbf{x}$ is classified correctly (i.e. $c^j = y$) and $\eta = -1$ if $\mathbf{x}$ is classified incorrectly (i.e. $c^j \neq y$), the following stochastic gradient descent update rule may be derived for LVQ1 [9]:

$$\mathbf{w}_{t+1}^{j} = \begin{cases} \mathbf{w}_t^j + \alpha(\mathbf{x} - \mathbf{w}_t^j), & \text{if } c^j = y \\ \mathbf{w}_t^j - \alpha(\mathbf{x} - \mathbf{w}_t^j), & \textit{otherwise}, \end{cases} \tag{3}$$

where α is the learning rate and the t subscripts denote prototype states at different training steps. However, it should be noted that the error function as defined here is highly discontinuous, and thus can lead to instabilities in the algorithm. GLVQ, discussed next, was designed to resolve this issue.

2.2. GLVQ

Here, $d_j = d^2(\mathbf{x}, \mathbf{w}^j)$ is defined where $\mathbf{w}^j$ is the closest prototype to $\mathbf{x}$ with label $c^j = y$ and $d_k = d^2(\mathbf{x}, \mathbf{w}^k)$ where $\mathbf{w}^k$ is the closest prototype to $\mathbf{x}$ with some other label. By letting

$$g(\mathbf{x}) = \frac{d_j - d_k}{d_j + d_k} \tag{4}$$

and

$$f_t(g(\mathbf{x})) = \frac{1}{1 + \exp^{-g(\mathbf{x})t}}, \tag{5}$$

which is a sigmoidal function that redefines the error function (Eq. 2) such that it is continuous over borders between the receptive fields for $\mathbf{w}^j$ and $\mathbf{w}^k$. When minimized, the error function yields the following update rules for $\mathbf{w}^j$ and $\mathbf{w}^k$ [14]:

$$\mathbf{w}^j_{t+1} := \mathbf{w}^j_t + \alpha\nu \frac{d_k}{(d_j + d_k)^2}(\mathbf{x} - \mathbf{w}^j_t) \tag{6}$$

$$\mathbf{w}^k_{t+1} := \mathbf{w}^k_t + \alpha\nu \frac{d_j}{(d_j + d_k)^2}(\mathbf{x} - \mathbf{w}^k_t) \tag{7}$$

where

$$\nu = f'_t(g(\mathbf{x})) = f_t(g(\mathbf{x}))(1 - f_t(g(\mathbf{x})). \tag{8}$$

GLVQ, unlike LVQ1 or the rest of Kohonen's original LVQ formulations, has been shown to be convergent [14, 6], although it is sensitive to the initialization of the prototype vectors. This is demonstrated in the experimental results of Section 4.

2.3. RLVQ and GRLVQ

The LVQ prototype update equations can be accompanied by updates that also alter the λ_i in Eq. (1) dynamically during training, hence allowing for an adaptive Euclidean metric. In RLVQ [1], LVQ1 training is adjusted such that the following weighting factor update rule is applied alongside Eq. (3):

$$\lambda_l := \begin{cases} \lambda_l - \beta(x_l - w^j_l)^2 & \text{if } c^j = y \\ \lambda_l + \beta(x_l - w^j_l)^2 & \textit{otherwise}, \end{cases} \tag{9}$$

for each l-th dimension where $\beta \in (0, 1)$ is a learning rate for the weighting factor adjustments. The weights are normalized at each update such that $\sum_{i=1}^{n} \lambda_i = 1$. The motivation for the above comes from Hebbian learning, the idea being that when $\mathbf{w}^j$ classifies the sample $\mathbf{x}$ correctly, the weights for the dimensions that contributed to the classification the most are increased, whereas the weights of those that contributed the least are decreased. When $\mathbf{w}^j$ incorrectly classifies $\mathbf{x}$, the weights for dimensions that contributed most are decreased, whereas the weights for dimensions that contributed the least are increased. GRLVQ [6] is an application of the above idea to GLVQ, such that the updates for the weights for the metric also follow a stochastic gradient descent on the error function defined by GLVQ.

One disadvantage of both RLVQ and GRLVQ is that they require the specification of an additional learning rate, β, which can be difficult to specify appropriately with respect to its α counterpart in the prototype updates. Another disadvantage is that they fail to take into consideration the additional statistical information provided by the remaining prototypes other than the ones currently being updated at a given training step when making relevance estimates. These issues are addressed with the following two proposed LVQ relevance determination algorithms.

3. Proposed Algorithms

The Fisher criterion, while ordinarily associated with Fisher's discriminant analysis [4], can also serve as an effective means for relevance determination when applied across individual data dimensions. Letting $\overline{x}^A = \frac{1}{N}\sum_{x^i \in A} x^i$ be the mean of a set of points A with cardinality N, the Fisher criterion score for a given individual dimension l is defined as

$$F(l) = \frac{S_B(l)}{S_W(l)}, \tag{10}$$

where

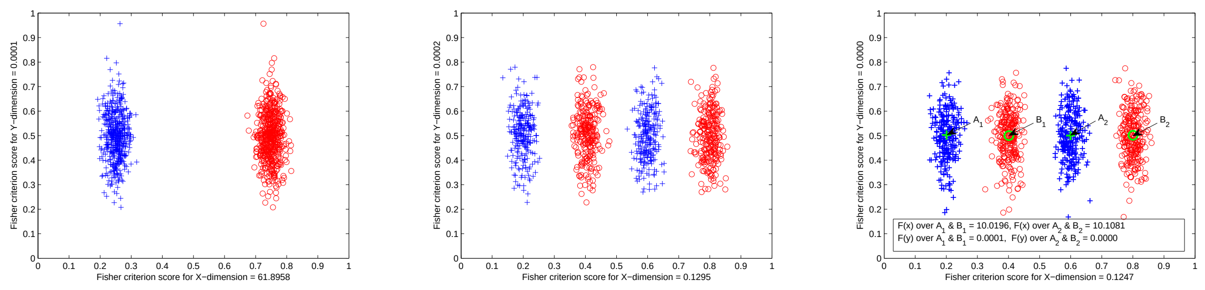
$$S_B(l) = \sum_{c=1}^{C} N^c \left(\overline{x}^{X^c}_l - \overline{x}^X_l\right)^2 \tag{11}$$

is the between-class variance and

$$S_W(l) = \sum_{c=1}^{C} \sum_{\mathbf{x} \in X^c} \left(x_l - \overline{x}^{X^c}_l\right)^2 \tag{12}$$

is the within-class variance over the l-th dimension.

With regard to relevance determination for LVQ, $F(l)$ could be calculated for each dimension over the entire training set X in advance of LVQ training and applied to the weighting factors in Eq. (1) by setting $\lambda_l = F(l)$ for all l to form a weighted metric. However, for many applications it is more desirable to have an online feature relevance training mechanism that is not reliant on having access to the entire training set at once. Two such online algorithms where estimation of the Fisher criterion score is integrated into the training scheme are presented next.



(a) (b) (c)

Figure 1. A simple 2D, 2-class example of how the Fisher criterion score (see Eq. (10)) can fail as a feature relevance metric over multi-modal distributions. (a) shows uni-modal class data distributions, linearly separable in the x-dimension, but with large overlap in the y-dimension. The score reflects the relevance of each dimension to class discrimination. (b) by comparison, shows the same number of data points, but with a multi-modal distribution (yet still linearly separable in x). The score is significantly lower for the x-dimension in this case. (c) shows the improvement provided by calculating the score between pairs of clusters with centers at points A_1, B_1, A_2 and B_2. See Section 3 for more details.

3.1. Algorithm 1

With the first algorithm, rather than calculating $F(l)$ over the data in X, at a given timestep t the score is estimated over the values of the prototype vectors in W. This is plausible since the distribution of the prototype vectors should approximate the distribution of the data over time. During training, certain prototypes will quantize more significant modes of the distribution than others, thus to account for this, weighted means and variances are calculated for each class based on the classification accuracy of each of the prototypes of that class, then the Fisher criterion score is calculated over the weighted means and variances for all classes. Firstly, the definition of $\mathcal{W}$ is altered to $\mathcal{W} = \{(\mathbf{w}^i, c^i, p^i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \times \mathbb{R} \mid i = 1, \ldots, M\ \}$ where, given random variable $(\mathbf{x}, y)$, $p^i = p(\mathbf{x} \in R^i | y = c^i)$ is the conditional probability of x lying in receptive field R^i of prototype $\mathbf{w}^i$ given that $\mathbf{w}^i$ correctly classifies x. The p^i form probability distributions over class prototypes such that $\sum_{p^i \in \mathcal{W}^c} p^i = 1$ for each class c. A definition of the estimated Fisher criterion score may now be formed as

$$F(l) \simeq \hat{F}(l) = \frac{\hat{S}_B(l)}{\hat{S}_W(l)}, \tag{13}$$

where

$$S_B(l) \simeq \hat{S}_B(l) = \sum_{c=1}^{C} \frac{N^c}{N} \left(\hat{w}_l^{\mathcal{W}^c} - \hat{w}_l^{\mathcal{W}}\right)^2 \tag{14}$$

is the estimated between-class variance over the l-th dimension,

$$S_W(l) \simeq \tag{15}$$

$$\hat{S}_W(l) = \sum_{c=1}^{C} \frac{N^c}{N} \sum_{(\mathbf{w}^i, c^i, p^i) \in \mathcal{W}^c} p^i \left(w_l - \hat{w}_l^{\mathcal{W}^c}\right)^2 \tag{16}$$

is the estimated within-class variance over the l-th dimension, and

$$\hat{w}_l^{\mathcal{W}^c} = \sum_{(\mathbf{w}^i, c^i, p^i) \in \mathcal{W}^c} p^i w_l^i \tag{17}$$

is a weighted mean over the l-th dimension of prototypes in a given set $\mathcal{W}^c \subseteq \mathcal{W}$.

The λ_m relevance factors may then be updated at each timestep by taking a running mean of the normalized estimated Fisher criterion score:

$$\lambda_{l,t+1} := \lambda_{l,t} + \frac{\frac{\hat{F}(l)}{\sum_{l=1}^{n} \hat{F}(l)} - \lambda_{l,t}}{t+1}. \tag{18}$$

While the Fisher criterion score is suitable for feature relevance determination in many cases, its main drawback is that it does not cope well with multimodal feature distributions. An example of this is shown in Figure 1. This problem remains in the estimation proposed above, since Eq. (14) and Eq. (16) are calculated over all class prototypes. The second proposed algorithm was designed to account for this.

3.2. Algorithm 2

The second proposed algorithm is based on the idea of calculating the Fisher criterion score between single prototype vectors of opposing classes, where the assumption is made that each class prototype vector may be quantizing different modes of the underlying class distribution. During training, Gaussian kernels are used to maintain estimates of the accuracies of each of the prototypes over the parts the data distribution accounted for by each of their receptive fields. At a given training step, the nearest single prototypes of each class to the training sample are found, and their Gaussian kernels are used to calculate an estimate of the Fisher criterion score for that local portion of the distribution, which is subsequently averaged over the entire training period.

The definition of $\mathcal{W}$ is this time altered to accommodate a Gaussian estimate of the accurate portion of the receptive field for each prototype, such that

$$\mathcal{W} = \{ \left(\mathbf{w}^i, c^i, \mathcal{N}(\mathbf{x}; \mu^i, \Sigma^i)\right) \subset \mathbb{R}^n \times \{1, \dots, C\} \times (\mathbb{R}^n \times \mathbb{R}^{n \times n}) \mid i = 1, \dots, M \}, \tag{19}$$

where $\mathcal{N}$ approximates $\tilde{R}^i = \{\mathbf{x} \in R^i | y = c\}$ with mean μ^i and covariance matrix $\Sigma^i = \mathrm{diag}([s_1^i, \dots, s_n^i])$ where the $\{s_l^i\}_{l=1}^n$ are variances along each l-th dimension. During LVQ training, given a random sample $(\mathbf{x}, y) \in X$ at training step t, if the closest prototype $\mathbf{w}^j$ classifies x correctly, i.e. $c^j = y$, then μ_l^j and s_l^j are updated in each l-th dimension as follows [8]:

$$\mu_{l,t}^j := \mu_{l,t-1}^j + \frac{x_l - \mu_{l,t-1}^j}{t} \tag{20}$$

$$\hat{s}_{l,t}^j := \hat{s}_{l,t-1}^j + (x_l - \mu_{l,t-1}^j)(x_l - \mu_{l,t}^j) \tag{21}$$

where $\mu_{l,t}^j$ is the running mean estimate and $s_{l,t}^j = \frac{\hat{s}_{l,t}^j}{t-1}$ is the running variance estimate for the l-th dimension at training step t. If $c^j \neq y$, then the above updates are not performed. Assuming a sufficient number of updates have been performed on the relevant prototypes up until step t, a Fisher criterion score estimate may be calculated between

$$\mathcal{W}' = \{\omega^k = \left(\mathbf{w}^k, c^k, \mathcal{N}(\mathbf{x}; \mu^k, \Sigma^k)\right) \in \mathcal{W} \mid \forall \mathbf{w}^i, c^i = c^k, d(\mathbf{x}, \mathbf{w}^k) \leq d(\mathbf{x}, \mathbf{w}^i) \}, \tag{22}$$

the closest prototypes of different classes (including $\mathbf{w}^j$), as follows:

$$F(l) \simeq \tilde{F}(l) = \frac{\tilde{S}_B(l)}{\tilde{S}_W(l)}, \tag{23}$$

where

$$S_B(l) \simeq \tilde{S}_B(l) = \frac{1}{C} \sum_{c=1}^{C} (\mu_l^c - \overline{\mu}_l)^2 \tag{24}$$

is the between-class variance estimate in the l-th dimension with

$$\overline{\mu}_l = \frac{1}{C} \sum_{\omega^k \in \mathcal{W}'} \mu_l^k, \tag{25}$$

and

$$S_W(l) \simeq \tilde{S}_W(l) = \sum_{\omega^k \in \mathcal{W}'} s_l^k \tag{26}$$

is the within-class variance estimate in the l-th dimension. The relevance factors may then be updated in a similar way to Eq. (18), this time using the new estimates:

$$\lambda_{l,t+1} := \lambda_{l,t} + \frac{\frac{\tilde{F}(l)}{\sum_{l=1}^n \tilde{F}(l)} - \lambda_{l,t}}{t+1}. \tag{27}$$

Since each prototype carries an accompanying Gaussian kernel that estimates its accuracy, it is now possible to estimate the Fisher criterion score using only single prototypes from each class, as opposed to the previous algorithm where multiple prototypes in each class have to be considered to achieve variance estimates. Though the model is made more complex, it is more capable of successfully handling the multi-modal distribution issue described in Fig. 1 as shown by the experimental results in the next section.

4. Experiments

The proposed algorithms were evaluated over simulated data, datasets from the UCI repository, and a real-world dataset from a cognitive robotics object affordance learning experiment. In the following, the datasets are described in more detail and experimental results are provided in Section 4.1. Two simulated datasets were proposed in [1, 6], the first of which was replicated for the experiments here. The data is composed of three classes, each separated into two clusters with some small overlap to form multi-modal class data distributions in the first two dimensions. Eight further dimensions are generated from the first two dimensions as follows: assuming (x_1, x_2) is one data point, $x_3 = x_1 + \eta_1, \dots, x_6 = x_1 + \eta_4$ is chosen where η_i comprises normally-distributed noise with variances $0.05, 0.1, 0.2$, and 0.5 respectively. The remaining $x_7, \dots, x_{10}$ components contain pure noise uniformly distributed in

$[-0.5, 0.5]$ and $[-0.2, 0.2]$. This dataset is multi-modal for each class in the two relevant dimensions and thus provides a good test for the potential difference between the two proposed algorithms.

Dataset	# Features	# Samples	# Classes
Simulated	10	90	3
Iris	4	150	3
Ionosphere	34	351	2
Wine	13	178	3
Soybean	35	47	4
WBC	30	569	2
Affordance	11	160	2

Table 1. An attribute list for the datasets in Section 4.

Five different datasets from the UCI repository [5] were tested: Fisher's Iris dataset, the ionosphere dataset, the wine dataset, the soybean dataset (small), and the Wisconsin breast cancer (WBC) dataset. A dataset from a cognitive robotics object affordance learning experiment [13] was also tested. It consists of eight household objects separated into two classes, four rolling objects and four non-rolling objects, and labeled as such, accompanied by eleven different shape features, two of which measure the curvature of 3D points from stereo images of the objects and the remainder of which were derived from 2D silhouettes of the objects.

4.1. Results

The primary goal of the investigation was to evaluate whether or not the new algorithms when applied to standard LVQ methods such as LVQ1 and GLVQ offer performance improvements over those methods in their original form, as well as over other relevance determination techniques for LVQ, such as RLVQ and GRLVQ. The results of these comparisons are outlined in Table 2 and are discussed in more detail in the following. In the results, the proposed Fisher criterion score-based relevance determination algorithms are referred to as FC1LVQ1 and FC2LVQ1 respectively when applied to LVQ1, and FC1GLVQ and FC2GLVQ when applied to GLVQ.

A secondary consideration was to test the methods under the duress of various different conditions. GLVQ, for example is known to perform poorly if the prototype vectors are not initialized within the data distribution [12], thus in our evaluations, both random prototype initializations as well as initializations where the prototypes are placed at the mean points of class clusters were considered. Note that random prototype initialization in this case refers to selecting random values for each prototype dimension scaled within the data range. K-means clustering was used to determine class clusters in the latter case.

The performance of LVQ algorithms over short training periods is not often considered in the literature, which tends to favour evaluations of the algorithms over several hundred training epochs until convergence is reached. Given that LVQ algorithms have online training mechanisms, and that the relevance determination techniques proposed above were explicitly developed to also function online, sample-by-sample without access to the rest of the training set, such short-term training evaluations are important if the methods are to be considered useful in real-world online settings, e.g. cognitive robotics [13], where the entire training set is often unavailable at any given point during training.

Thus, the results in Table 2 are divided into four main evaluations: both 1 epoch and 300 epochs of training from random initialization, and both 1 epoch and 300 epochs of training from class cluster mean initialization. The 300 epoch sessions used the relatively slow learning rates of $\alpha = 0.1$ for the prototype updates (cf. Eq. (3), Eq. (6) & Eq. (7)) and $\beta = 0.01$ for the dimensional relevance updates where required (cf. Eq. (9)), whereas the 1 epoch training sessions used the faster rates of $\alpha = 0.3$ and $\beta = 0.1$. Note that the FC1 and FC2 methods do not require the additional β learning rate. In each of the 1 epoch evaluations, 20 trials of ten-fold cross validation were performed with random data orderings in each trial, and results were averaged over test data performance, whereas in the 300 epoch evaluations, 5 trials were performed. 10 prototypes were used for every dataset and the data dimensions were scaled prior to training.

The results in Table 2 show that when trained over a single epoch from random initialization, of the algorithms tested FC2LVQ1 and FC2GLVQ achieved higher mean classification scores than their counterparts in many cases. Over long-term training of 300 epochs from random initialization, the results for all algorithms aside from GLVQ, tend to improve with FC2LVQ1 and FC2GLVQ again tending to be competitive with their counterparts. It is worth noting here the impact relevance determination has on improving the results of GLVQ when exposed to poor prototype initialization. When the prototypes are initialized optimally at the class cluster mean points the results tend to improve dramatically across all of the

Dataset	LVQ1	RLVQ1	**FC1LVQ1**	**FC2LVQ1**	GLVQ	GRLVQ	**FC1GLVQ**	**FC2GLVQ**
Random Initialization, 1 Epoch of Training, 20 Trials								
Sim	53± 18%	64± 22%	54± 19%	**69± 18%**	37± 17%	63± 22%	51± 20%	**70± 19%**
Iris	90± 8%	91± 9%	93± 9%	**95± 5%**	63± 24%	**89± 13%**	83± 19%	88± 15%
Iono	81± 8%	75± 11%	**85± 6%**	84± 7%	66± 13%	80± 9%	82± 7%	**84± 7%**
Wine	93± 6%	79± 13%	92± 9%	**94± 6%**	52± 19%	92± 8%	85± 14%	**94± 7%**
Soy	**89± 17%**	83± 24%	89± 18%	85± 21%	34± 27%	84± 22%	83± 21%	**85± 20%**
WBC	92± 4%	86± 8%	93± 4%	**93± 3%**	71± 19%	93± 5%	90± 10%	**94± 3%**
Afford	97± 7%	93± 10%	98± 4%	**99± 3%**	78± 22%	96± 9%	84± 20%	**98± 6%**
Random Initialization, 300 Epochs of Training, 5 Trials								
Sim	79± 14%	79± 13%	77± 16%	**87± 12%**	38± 17%	**96± 7%**	90± 12%	94± 9%
Iris	92± 7%	92± 8%	95± 5%	**96± 5%**	47± 24%	96± 5%	91± 16%	**96± 4%**
Iono	85± 7%	80± 10%	**86± 8%**	85± 7%	60± 16%	**90± 5%**	90± 6%	89± 6%
Wine	95± 5%	77± 11%	95± 5%	**96± 5%**	42± 18%	96± 5%	97± 4%	**98± 3%**
Soy	99± 6%	97± 10%	**100± 4%**	98± 7%	33± 26%	97± 8%	**97± 7%**	96± 9%
WBC	93± 3%	87± 7%	**94± 3%**	**94± 3%**	62± 20%	96± 3%	96± 3%	**96± 2%**
Afford	**99± 2%**	95± 7%	**99± 2%**	99± 3%	67± 24%	**99± 2%**	**99± 2%**	**99± 2%**
Class Cluster Mean Initialization, 1 Epoch of Training, 20 Trials								
Sim	82± 12%	**98± 5%**	78± 17%	93± 8%	90± 9%	91± 9%	85± 13%	**93± 8%**
Iris	**96± 5%**	**96± 5%**	**96± 5%**	**96± 5%**	95± 5%	95± 5%	95± 5%	**96± 5%**
Iono	87± 6%	80± 10%	**88± 6%**	**88± 6%**	**90± 5%**	88± 6%	89± 5%	**90± 5%**
Wine	95± 5%	86± 11%	**96± 5%**	**96± 5%**	**97± 4%**	97± 5%	97± 5%	97± 5%
Soy	**100± 2%**	95± 10%	100± 3%	99± 5%	**100± 2%**	99± 4%	**100± 2%**	99± 5%
WBC	**95± 3%**	88± 7%	94± 3%	94± 3%	96± 3%	96± 3%	**97± 3%**	95± 3%
Afford	**99± 2%**	98± 4%	**99± 2%**	**99± 2%**	**99± 2%**	**99± 2%**	**99± 2%**	**99± 2%**
Class Cluster Mean Initialization, 300 Epochs of Training, 5 Trials								
Sim	84± 11%	86± 16%	87± 12%	**91± 10%**	90± 9%	**97± 6%**	90± 10%	96± 8%
Iris	96± 5%	95± 6%	**96± 4%**	96± 5%	96± 6%	95± 5%	**97± 4%**	96± 4%
Iono	88± 5%	82± 9%	**89± 5%**	88± 5%	89± 5%	90± 5%	90± 5%	**91± 5%**
Wine	96± 5%	82± 12%	**97± 4%**	96± 5%	97± 4%	**98± 3%**	**98± 3%**	**98± 3%**
Soy	**100± 0%**	94± 11%	99± 6%	98± 7%	**100± 0%**	98± 8%	99± 5%	99± 6%
WBC	**96± 2%**	89± 5%	95± 3%	95± 3%	96± 3%	96± 3%	**97± 2%**	**97± 2%**
Afford	**99± 2%**	98± 3%	**99± 2%**	**99± 2%**	99± 3%	**99± 2%**	**99± 2%**	**99± 2%**

Table 2. 10-Fold cross validation, 10 prototypes. Highest scores for LVQ1 & GLVQ based algorithms are shown in bold.

classifiers in short-term training, with both FC1 and FC2 relevance determination doing well over both short-term and long-term training periods, with FC1 out-performing FC2 in some cases and vice versa. Over all the evaluations, FC1GLVQ and FC2GLVQ trained over 300 epochs with class cluster mean initialization tended to score well when compared with the other methods. It should also be noted that, when the class distribution in the data is multi-modal, as is the case with the simulated dataset, FC2-based methods tend to be a better choice than FC1-based methods, as predicted.

A third consideration was to compare the new methods to a state-of-the-art batch method such as the *support vector machine (SVM)*. Batch methods, as opposed to online methods that are trained sample-by-sample, have access to the entire training set during training, and therefore usually provide superior results. Table 3 shows the results of a comparison between FC1GLVQ, FC2GLVQ and a multi-class SVM trained with a radial basis function (RBF) kernel [2]. For this comparison, the results for FC1GLVQ and FC2GLVQ from the 300 epoch, class cluster mean-initialized evaluation described previously were used, while ten-fold cross validation over five trials was also used for the SVM, where the test data results were averaged over the five trials and SVM parameters were optimized using cross validation over the training data prior to training. The results show both FC1GLVQ and FC2GLVQ performing well when compared with SVM over the various datasets, particularly in the case of the simulated multi-modal dataset.

It is difficult to evaluate the performance of the algorithms with respect to the estimation of the λ_l weighting factors themselves, but examples of the

Dataset	FC1GLVQ	FC2GLVQ	SVM
Simulated	90±10%	**96±8%**	78±14%
Iris	**97±4%**	96±4%	96±6%
Ionosphere	90±5%	91±5%	**94±4%**
Wine	**98±3%**	**98±3%**	**98±3%**
Soybean	99±5%	99±6%	**100±0%**
WBC	97±2%	97±2%	**98±2%**
Affordance	**99±2%**	**99±2%**	99±3%

Table 3. FC1GLVQ & FC2GLVQ versus SVM. Highest mean scores are shown in bold.

mean values for certain datasets are provided here. For the simulated dataset, $\lambda_{\text{FC1GLVQ}} = \{0.10, 0.42, 0.07, 0.06, 0.10, 0.06, 0.04, 0.03, 0.07, 0.04\}$ and $\lambda_{\text{FC2GLVQ}} = \{0.40, 0.43, 0.06, 0.01, 0.01, 0, 0, 0, 0, 0\}$, thus demonstrating that FC2GLVQ does indeed do a better job of handling the multi-modal distribution. For the Iris dataset, $\lambda_{\text{FC1GLVQ}} = \{0.02, 0.02, 0.55, 0.40\}$ and $\lambda_{\text{FC2GLVQ}} = \{0.03, 0.07, 0.37, 0.53\}$. For the object affordance dataset, $\lambda_{\text{FC1GLVQ}} = \{0.04, 0.56, 0.05, 0.05, 0.03, 0.05, 0.04, 0.04, 0.01, 0.09, 0.05\}$ and $\lambda_{\text{FC2GLVQ}} = \{0.05, 0.34, 0.07, 0.07, 0.07, 0.08, 0.01, 0.08, 0.06, 0.12, 0.06\}$, where one of the 3D curvature features is favoured in each case.

5. Conclusion

In conclusion, two new relevance determination algorithms have been proposed for LVQ that exploit the positioning of prototypes in the input feature space to calculate Fisher criterion score estimates in the input dimensions for an adaptive metric. An advantage provided by these methods over other metric-adaptive LVQ methods based on gradient descent, is that they do not require a learning rate or other parameters to be specified. Moreover, they provide incremental update rules that operate alongside regular LVQ update rules and can therefore be applied to any algorithms based on the general LVQ paradigm. Experimental evaluations were provided under various stress conditions and over various datasets and the proposed methods were shown to perform competitively against various other LVQ-based methods, and against SVM. With regard to future work, it would be interesting to apply the proposed techniques to prototype-based methods other than LVQ, such as supervised neural gases.

References

[1] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz. Relevance determination in learning vector quantization. In *European Symposium on Artificial Neural Networks*, pages 271–276, 2001. 1, 2, 3, 5

[2] C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001. 7

[3] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin analysis of the LVQ algorithm. *Advances in Neural Information Processing Systems*, page 479–486, 2003. 1

[4] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936. 3

[5] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2010. 6

[6] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002. 2, 3, 5

[7] S. Hotta. Learning vector quantization with local subspace classifier. In *Proceedings of the 19th International Conference on Pattern Recognition*, page 1–4, 2008. 1

[8] D. E. Knuth. *The Art of Computer Programming (Volume 2)*. Addison–Wesley, 1981. 5

[9] T. Kohonen. *Self-organizing maps*. Springer, 1997. 1, 2

[10] C. E. Pedreira. Learning vector quantization with training data selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):157–162, 2006. 2

[11] M. Pregenzer, G. Pfurtscheller, and D. Flotzinger. Automated feature selection with a distinction sensitive learning vector quantizer. *Neurocomputing*, 11(1):19–29, 1996. 1

[12] A. Qin and P. Suganthan. Initialization insensitive LVQ algorithm based on cost-function adaptation. *Pattern Recognition*, 38(5):773–776, 2005. 6

[13] B. Ridge, D. Skočaj, and A. Leonardis. Self-Supervised Cross-Modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010. 6

[14] A. Sato and K. Yamada. Generalized learning vector quantization. In *Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference*, page 423–429. MIT Press, 1996. 1, 2, 3

[15] M. K. Tsay, K. H. Shyu, and P. C. Chang. Feature transformation with generalized learning vector quantization for hand-written chinese character recognition. *IEICE Transactions on Information and Systems*, E82-D(3):687–692, 1999. 2

Room Classification using a Hierarchical Representation of Space

Peter Uršič, Matej Kristan, Danijel Skočaj, Aleš Leonardis

Abstract— **Mobile robots need an effective spatial model for the successful operation in real-world environment. The model should be compact and simultaneously possess large expressive power. Moreover, it should scale well. In this paper we propose a new hierarchical representation of space, whose compositional structure is learned based on statistically significant observations. We have focused on a two dimensional space, since many robots perceive their surroundings in two dimensions with the use of a laser range finder or a sonar. We also propose a new low-level image descriptor, by which we demonstrate the performance of our representation in the context of room classification problem. Using only the lower layers of the hierarchy, we obtain state-of-the-art classification results on demanding datasets.**

I. INTRODUCTION

The choice of spatial representation plays an important role in the process of designing a cognitive mobile robot. The model has an influence on the performance of many spatially related tasks, like navigation, localization, and room recognition. Moreover, designing an efficient spatial model is a challenging problem. The model should be as compact as possible, while simultaneously it needs to be able to efficiently represent huge variety of the environment. Numerous spatial models have already been proposed [1], [2], [3], [4], but some challenges still remain. Our long-term goal is to address the scalability issue. We expect that if the robot has already been observing the environment for some time, it should have learned about the general characteristics of space from performed observations, and then use the gathered knowledge when it arrives into new environment. For example, if a robot has already seen a large number of apartments, when it arrives into the next one, it should already have some clue about what it will see. Current state-of-the-art robotic systems usually scan the entire never before seen room, before they achieve good performance [1]. We want to be able to provide some prior knowledge for a service robot, which would enable it to reason about the new environment, even without recording the whole place.

In recent years, hierarchical compositional models have been shown to possess many appealing properties, which have a potential to meet our goals. They are used in computer vision community for object class detection. A central point of these models is that their lowest layer is composed of elementary parts, which are combined to produce more complex parts on the next layer. This procedure may be recursively repeated over several layers which gradually increases the complexity of the vocabulary of parts. An appealing aspect of the compositional hierarchies is that, on the one hand, they offer sharing of object parts within object category, while on the other hand, they can also reuse the parts at multiple levels of granularity among different categories [5]. In fact, in the recent work Fidler et al. [5] have shown that a hierarchical compositional model allows incremental training and significant sharing of parts among many categories. The sharing reduces the storage requirements and at the same time makes inference efficient, since hypotheses of the shared parts are verified simultaneously for multiple categories.

In this paper we adapt the hierarchical model from [5] to develop a description suitable for representation of space. The representation is two dimensional, and based on data obtained with a laser range finder. To the best of our knowledge, this is the first attempt of using a hierarchical compositional model such as [5] for the representation of space on the lowest semantic level.

To demonstrate the suitability of our representation, we perform a series of experiments in the context of room classification problem, based only on laser scans. Our experiments on demanding datasets show that our method delivers comparable results to the state-of-the-art.

A. *Related Work*

The Spatial Semantic Hierarchy [2] is a model of knowledge of large-scale space consisting of multiple interacting representations. Their hierarchy is composed of a sensory, control, causal, topological and metrical level. Although some prior knowledge is provided for a robot by the model, this work is quite different from ours, since a semantically higher-level approach is used, and there is no attempt to address the scalability issue, like in our approach.

Various systems performing topological localization have been developed for room classification. In [1] very accurate room classification is achieved using multimodal information. Approaches using less information available for classification have also been considered. Laser range data combined with vision was used for classification in [6], and many approaches that use vision only for the accomplishment of this task have also been presented [7], [8], [9].

In [10] 3D Time-of-Flight infrared sensor was used for acquiring 3D information, which allowed the distinction between three types of rooms (office, meeting room and hall). Only laser range data was used in [11]. Their robot was equipped with a 360 degree field of view range sensor and they were able to distinguish between four classes (rooms,

The research leading to these results has received funding from the EU FP7 project CogX.

All authors are with Vicos Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia `peter.ursic@fri.uni-lj.si`

corridors, doorways and hallways). Their classification system could determine the type of the place with a recognition rate of more than 89 % in the environment that was similar to one on which it was trained, while transferring the classifiers to completely new environments resulted in 82 % success rate. The classification was performed with AdaBoost and it was based only on a single scan. Laser range data was also used for classification in [12], where Voronoi random fields (VRFs) were employed to label different places in the environment, providing the distinction between four classes (rooms, hallways, junctions and doorways). Their approach uses a state-of-the-art SLAM technique to generate a metric occupancy grid map of an environment, while the Voronoi graph is then extracted of this map. For each point on the Voronoi graph, VRFs then estimate the type of place it belongs to, achieving the average accuracy of leave one out place labelling above 91 %.

B. Our Approach

The contributions of our paper are two fold. First, we present a new hierarchical representation of space, along with an algorithm that is used to learn the hierarchy. The learning algorithm is an extended version of Learning the Hierarchy of Parts (lHoP) algorithm [13]. Our extended version is called the *sHoP* (Spatial Hierarchy of Parts) algorithm. As our second contribution, we propose a new low-level descriptor that can be used for room classification, which was inspired by the work of [14]. We use the descriptor not only to perform the classification, but also to simultaneously verify the reasonableness and effectiveness of our spatial model.

Our approach to room classification differs from others in several ways. We use a representation of space that is learned by the sHoP algorithm. In this paper, the elements of the hierarchy are used to solve a room classification problem with Support Vector Machine (SVM). Our classification problem is more challenging than the problems presented in [11], [12]. Our range sensor's field of view is only 240 degrees. Although several scans are obtained in every room, we do not yet use any information about how these scans are correlated to one another. We have also taken in consideration a more challenging set of room types (living room, corridor, bathroom and bedroom). Our goal in this paper is to show that a set of laser scans holds enough information to distinguish between these difficult room types, even with the use of relatively local geometrical shape descriptions, and to show that our representation can extract this information. We are able to distinguish between these four classes with reasonably high accuracy in a multiclass formulation of the problem.

The remainder of the paper is organized as follows: Section 2 provides an overview of the lHoP algorithm and our extended version. In Section 3, we make a leap to the room classification problem. We describe the steps leading from data acquisition to image descriptor formation and classification. In Section 4 we report experimental results, and present conclusions with future work in Section 5.

II. LEARNING A HIERARCHICAL REPRESENTATION OF SPACE

We start this section by describing the original lHoP algorithm designed for object categorization. In the following subsections we extend the algorithm to make it suitable for the representation of space.

A. The lHoP Algorithm

We are following the ideas of lHoP, which is an efficient, biologically-inspired, algorithm that was originally designed for object categorization. The algorithm learns a compositional hierarchy of so-called parts. On the lowest level of the hierarchy parts are represented as Gabor filters, corresponding to small fractions of oriented edges in the image. Parts in the upper layers are composed from the lower-layer parts, therefore, they are increasing in size and complexity with each following layer. Parts in the top-most layer are representations of the entire objects.

When learning object categories the procedure follows two important steps. First, the algorithm is given a large set of everyday images containing various objects. Based on these images, a few lower layers of the hierarchy are learned in an unsupervised fashion. All learned parts are the ones which occur most frequently in the images, thus giving the hierarchy compactness, while maintaining high representativity. Lower layers of the hierarchy are category-independent, therefore, they are common to all classes. This sharing of lower-layer parts is the basic advantage of the described representation. Second, for each specific class that needs to be learned, the algorithm is given a set of images containing only objects of that particular class. Upper layers of the hierarchy are then learned based on these images, again providing only statistically significant parts in terms of frequencies of appearance. Upper layers of the hierarchy are category-specific, therefore, characteristic of a specific class, and are learned with minimal supervision.

B. The sHoP Algorithm

Our goal is to create an effective representation of two dimensional space. We use a laser-range finder mounted on a mobile robot to observe the environment, which provides us with range data that is ground plan-like. Laser scans are afterwards transformed into images, which are the input for lHoP learning algorithm (Fig. 1).

The main drawback of lHoP is that it's library parts are not rotationally invariant, which is a crucial property for obtaining a compact and expressive hierarchy. For this reason, we extended lHoP to satisfy the rotational invariance condition. To augment the set of all detectable orientations, the extended algorithm, sHoP, uses 18 Gabor filters on the lowest layer in contrast to lHoP, which uses 6 of them. We are able to construct lower, category independent, layers of the hierarchy with an unsupervised learning algorithm, which learns the most frequent parts observed in the images by examination of a large amount of spatial data.

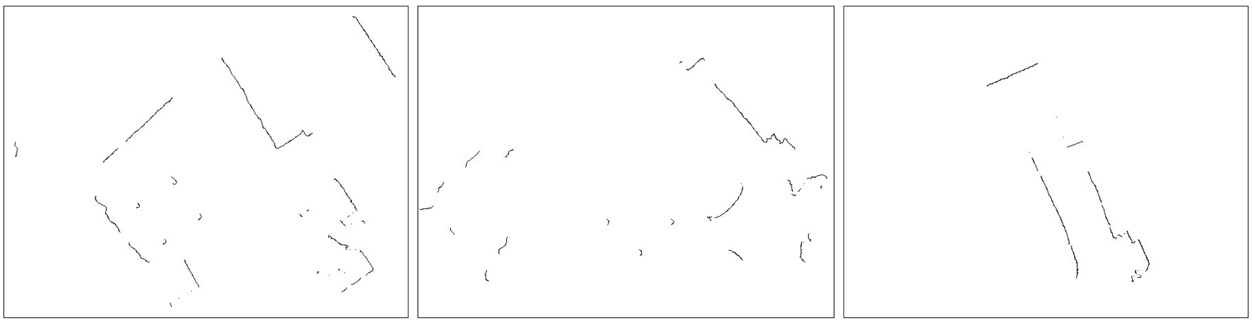


Fig. 1. Laser scans are transformed into images. Figure shows three example images acquired at different positions in one of the living rooms. The contents of these images are hard to recognize even for a human, which shows the difficulties of our approach. We might get some impression of the shape of the room from the first image. However, most of the lines that seem to form the walls of the room are actually sofas, while the walls are hidden behind them. To get some sense out of the other two images is even harder. In the last image the robot was directed to a passage between the couch and the wall.

C. The Library of Parts

As noted, the sHoP learning algorithm takes as input a set of images, where each image is a representation of a single range scan (Fig. 1). The output is a library of parts, representing the learned hierarchy, which is composed of several layers (Fig. 4). Learned rotationally invariant parts represent spatial shape primitives with a compositional structure.

As mentioned earlier, on the lowest layer there are 18 Gabor filters. These model 18 different detectable orientations, where two consecutive ones differ for 10 degrees. To clarify, orientations α and $180° + \alpha$ are equivalent in our edge detection. These filters could be considered as a single rotationally invariant first-layer part, but for better computational efficiency all of them are stored in the library.

On the higher layers, each part is a composition of two previous-layer parts. Each layer contains only those parts, which were observed most frequently in the input images, and each part is stored only in a single orientation - the one in which it was observed most frequently.

D. Part Structure and Rotational Invariance

The library serves as a knowledge base, which is used by the robot when it observes the environment. Parts are stored in the library in a single, reference, orientation. Each laser scan that is potentially observed by the robot during exploration is represented with a list of parts at certain locations relative to the robot. The creation of this list falls under the topic of inference process, which will be further explained in the next subsection. However, parts in this list are instantiated in their specific orientation, which is a necessity if we want to undertake any reconstruction of the environment. To assign them a correct type, they are linked to the reference parts in the library.

The structure of parts and their rotational invariance is defined in the following way. Suppose we already have a library containing at least k layers. Let us also suppose we have an image and somewhere on that image we found two Layer k parts, P_1 and P_2, positioned close to each other. Parts P_1 and P_2 are both in some specific known orientation and we also know to which part in the library they correspond to. Let us denote those parts in the library with L_{k_1} and L_{k_2}, where k_1 and k_2 are indices corresponding to the k-th layer of the library. Therefore, P_1 is some concrete instantiation of part L_{k_1}, and the orientation of P_1 can be the same or different from the orientation of L_{k_1}. An analogous statement holds for P_2 and L_{k_2}.

Now, suppose we want to construct a Layer $k+1$ part P from subparts P_1 and P_2 (see Fig. 2). The new part would be defined by the expression

$$P = \{k_1, k_2, p_{1pos}, P_1, P_2\}, \tag{1}$$

where $k_1 \leq k_2$ and p_{1pos} is the position of P_1 subpart relative to the geometric mean of both subpart positions. If $k_1 > k_2$ the order of indices and subparts is reversed, and p_{1pos} refers to the position of P_2.

Following this notation, the structure of subparts looks like

$$P_1 = \{(k-1)_{11}, (k-1)_{12}, p_{11pos}, P_{11}, P_{12}\}, \tag{2}$$

$$P_2 = \{(k-1)_{21}, (k-1)_{22}, p_{21pos}, P_{21}, P_{22}\}, \tag{3}$$

where P_{11}, P_{12}, P_{21} and P_{22} are some instantiations of Layer $k-1$ parts, which correspond to parts in the library, also on Layer $k-1$, with indices $(k-1)_{11}$, $(k-1)_{12}$, $(k-1)_{21}$ and $(k-1)_{22}$. If $k-1$ is the lowest layer, than $P_{ij} = (k-1)_{ij} \in \{1, 2, \ldots 18\}$ for $i, j \in \{1, 2\}$, where 18 corresponds to the number of Gabor filters. Positions of P_{11} and P_{21} subparts are here denoted with p_{11pos} and p_{21pos}.

We allow for some variance in part structure. This means that subpart positions can vary slightly, while still representing the same part. The same holds for rotational invariance. If we have two parts and we want to verify if they represent the same part in different orientations, we try to rotate one part into another. The result of rotation does not need to match perfectly with other part's orientation, rather there is some predefined threshold that defines how much the parts can differ. This simple process of rotation is used in learning of the hierarchy, as well as in inference, which will both be described in the next subsection. However, rotation of parts is performed in the following way:

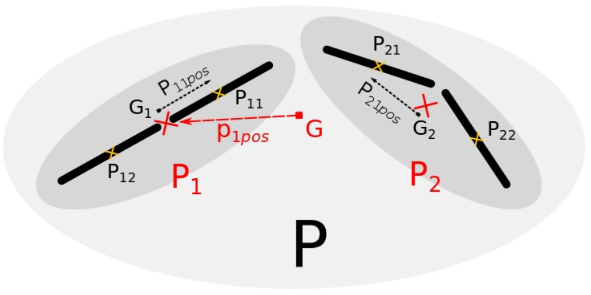


Fig. 2. Schematic view of the structure of an example part. A letter G is used to denote the geometric mean of subpart P_1 and P_2 positions. Geometric mean of P_{11} and P_{12} is denoted with G_1, while G_2 corresponds to geometric mean of P_{21} and P_{22}. Other quantities are defined in the text above.

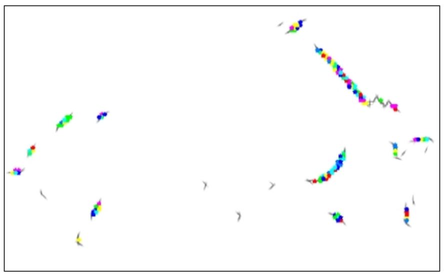


Fig. 3. Positions of Layer 3 parts inferred from an example image acquired in one of the living rooms (middle image in Fig. 1). Different colours correspond to different part types.

Let us assume we have another Layer $k+1$ part Q. This part has an analogous structure

$$Q = \{k_3, k_4, p_{3pos}, P_3, P_4\}, \tag{4}$$

where

$$P_3 = \{(k-1)_{31}, (k-1)_{32}, p_{31pos}, P_{31}, P_{32}\}, \tag{5}$$

$$P_4 = \{(k-1)_{41}, (k-1)_{42}, p_{41pos}, P_{41}, P_{42}\}. \tag{6}$$

To verify if parts P and Q are two different representations of the same part the following sequence of simple computations is performed:

- Index k_1 has to match k_3 and k_2 has to match k_4.
- We calculate the potential angle of rotation φ of a part P from p_{1pos} and p_{3pos}.
- Rotation of p_{11pos} for angle φ must result in p_{31pos}, and rotation of p_{21pos} for angle φ must result in p_{41pos}.

If the results of the above computations lie within the boundaries defined by thresholds we conclude that the parts are the same.

E. Learning Algorithm and Inference Process

In this subsection learning of the library of parts and inference process are described in more detail. As noted before, the input for the learning algorithm is a set of images obtained from range data, and the output is a hierarchy of rotationally invariant parts. While inference process is also used as an intermediate step in the learning procedure, it's main purpose is to detect and extract part information from input images, once we already have a learned library.

Learning starts with a library containing a single layer, which represents 18 Gabor filters. In the first step edge detection is used to find small fragments of oriented edges in the input images. Positions and response intensities of edge fragments, representing Layer 1 parts, are then used as an input for learning the second layer of the hierarchy.

In all of the following steps the sequential interchange of layer learning and inference process is performed. Precisely told, the hierarchy is being learned layer by layer by repeating the next two steps:

- *Layer learning:* Positions, confidence values, and orientation information of Layer k parts corresponding to every input image is used to learn Layer $k+1$ of the hierarchy. The notion of confidence values mentioned here will be addressed shortly in the context of inference.
- *Inference process:* Using the inferred image data from the previous, k-th, layer, and Layer $k+1$ of the library, positions, orientations and types of Layer $k+1$ parts in the input images are inferred (Fig. 3).

When inferring the positions and orientations of parts in the images each inferred part is assigned two confidence values. These values carry the information about how well a certain instantiation of a part in the image represents the corresponding part from the library. The first value is based on the response intensities of edge detection and the second is a measure of how precisely an instantiation of a part can be rotated into a corresponding part in the library.

Now let us take a closer look at learning of Layer $k+1$ from Layer k. This is performed in the following steps:

1) Finding Neighbourhoods: All of the inferred images, constructed from Layer k parts, are being inspected for pairs of nearby parts, while the following actions are being performed:

- For every part C that is observed in the images, and for every orientation o_C of part C that is observed assume, $\{C, o_C\}$ is the central part of the pair.
- For every part N, and every orientation o_N of part N that is observed in the proximity of $\{C, o_C\}$, assume $\{N, o_N\}$ is the neighbour part in the pair.
- A neighbourhood $Nb(\{C, o_C\}, \{N, o_N\})$ is maintained for every such pair.

These neighbourhoods are used to store the information about relative positions and position frequencies of parts. That is, each neighbourhood tells us in which positions has $\{N, o_N\}$ been observed relative to $\{C, o_C\}$ in the input images, and also how many times these positions were observed.

2) Finding Local Maxima: For every observed combination of part types and orientations, the most frequent relative positions of parts are obtained by searching for maximum values of relative position occurrences in the

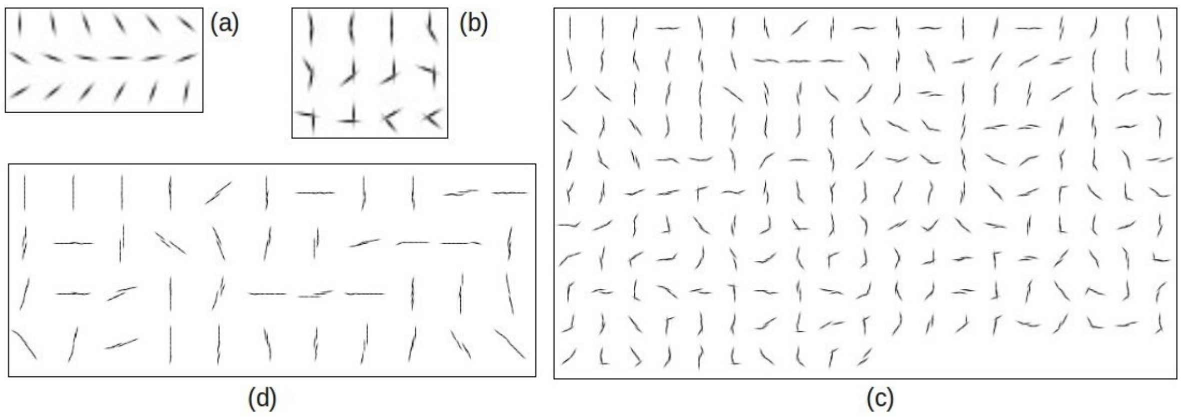


Fig. 4. (a) Orientation-specific Layer 1 parts. (b) Layer 2 parts. (c) The most frequent 200 parts on Layer 3. The image is shrunken to save space. (d) Only a few Layer 4 parts on a shrunken image. Other parts are not shown to save space.

neighbourhoods. Usually, one or two maxima are found for each neighbourhood.

3) Forming a Sequence of Rotationally Invariant Parts: Again, all of the inferred images containing Layer k parts are inspected. The images are being searched for pairs of parts in specific orientations, while considering only their relative positions defined by neighbourhood maxima. The number of occurrences of all of the observed pairs satisfying this condition is being counted, while the results representing the same pairs in different orientations are being summed together. This procedure thus constructs a list of pairs with their corresponding frequencies, which is at the end sorted according to the frequency of occurrence in descending order. In this way a list of rotationally invariant Layer $k + 1$ part candidates is obtained.

4) Adding Parts to Library: Sequence of part candidates constructed in the previous step contains the most frequent parts at the beginning and the least frequent ones at the end. So, depending on how large representation we want, a portion of parts at the beginning of the list is declared as Layer $k+1$ of the library, while the rest is discarded.

III. THE ROOM CLASSIFICATION PROBLEM

In this section we describe the steps that were taken from data acquisition to room classification. For the classification we propose a low-level image descriptor, which we name Histogram of Compositions (HoC).

A. Data Acquisition and Learning of the Hierarchy

Our recorded data consists of laser scans obtained with a mobile robot. Each laser scan was converted to an image in which nearby points were artificially connected. This step was necessary because pure range data provides images with a discrete set of points, on which edge detection is not efficient. Some example images, obtained at different locations in one of the living rooms, are shown in Fig. 1.

A separate dataset, containing data from several different room types, was used to learn the library of parts, which was constructed up to the fourth layer (Fig. 4). Concurrently, positions and orientations of parts were inferred from the input images for every layer of the hierarchy. Different layers of inferred data, with different number of part types, served then as an input for image descriptor formation step.

B. Image Descriptor Formation and Classification

To form our low-level descriptor, a single layer of the hierarchy was chosen. Inferred images corresponding to that layer were then sorted into groups, where each group represented a set of images acquired in the same room. From this point on, processing was the same for every group, and was performed in the following way:

1) Positions of parts in the inferred images were rotated into a reference position, with the use of principal component analysis (PCA).
2) Each image was divided into 24 regions as shown in Fig. 5. The robot was positioned in the center.

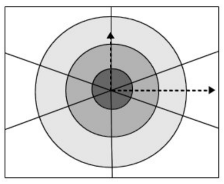


Fig. 5. The 24 regions of the image. The dashed arrows correspond to the principal axes.

3) For every image a sequence of 24 histograms was created, where each histogram corresponded to one region of the image. The number of bins in each histogram was equal to the number of parts in the considered layer of the hierarchy. Each bin corresponded to one part type from that layer, while the value corresponding to the height of the bin equalled to the sum of confidences of parts in that region. All of the histograms corresponding to one image were then concatenated into a single feature vector, forming our HoC descriptor.

4) For every room a single feature vector was created as a composition of an average and standard deviation of all the feature vectors in the group.

Input data for the classification algorithm were the feature vectors (one per room), obtained with the procedure described above. After normalization to zero mean and unit variance, LIBSVM library [15] was used for the classification with SVM.

IV. EXPERIMENTAL RESULTS

We have conducted two sets of experiments. The aim of the first experiment was to show that the lower layers of our hierarchy actually do hold useful information, and that HoC descriptor is an effective tool for room classification. We demonstrate this using our demanding dataset. The aim of the second experiment was to show that our approach is comparable to other, state-of-the-art, approaches. We demonstrate this using two publicly available datasets.

In addition to the dataset that will be described in the following subsection, we obtained another, separate, dataset that consists of a large number of rooms, which can be divided into 20 different categories. We used this dataset to learn our category independent layers of the hierarchy, which were then used in all of the following experiments.

A. *Experiments With Our Dataset*

We have gathered our data with a Pioneer P3-DX robot, which was manually driven through different rooms, while observing the space with a Hokuyo URG laser range-finder (see Fig. 6). The sensor was positioned approximately 30 cm above the floor. In this way 24 different homes have been scanned, consisting of houses and apartments from 4 different cities, which contribute to high variability of our data. Our dataset consists of 25 living rooms, 6 corridors, 35 bathrooms and 28 bedrooms. Typical corridors, which are long, straight, and narrow spaces, are not very common in homes, and this is the reason why we were only able to observe such a small number of them.

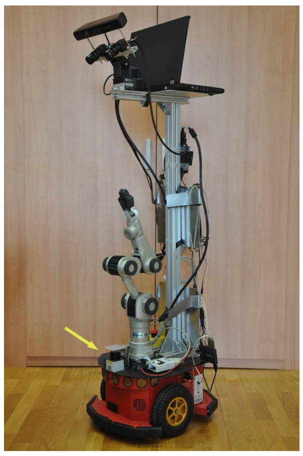


Fig. 6. Our robot. We used only the base and the laser range finder, which is indicated with an arrow, to gather our data.

Different number of laser scans were obtained in each room. Their number is dependent on room size and varies all the way from 100 to 600 scans. We have also posed some restrictions while gathering the data. There were no people in any of the rooms and all of the doors were closed, meaning that no other room was visible from any room, but only the one, which the robot was currently observing.

We performed a set of classification experiments, with different layers of the hierarchy and also without the hierarchy, to evaluate the suitability of our representation of space. At the beginning we tested with many different divisions of the image into regions for the formation of HoC image descriptor. We chose an optimal one, which enabled us to obtain very good classification results even without the use of our representation of space. The chosen regions are shown in Fig. 5 in the previous section. The radii that correspond to circles have the values of 0.8, 1.5, and 2.5 meters if scaled to the real world size, while the angles of the straight lines have the values of 20, 90, 160, 200, 270, and 340 degrees.

In the classification procedure we performed 1000 trials of each experiment, in which we tried to distinguish between 4 previously mentioned types of rooms. In each trial, training (80 %) and testing (20 %) data was randomly chosen from the set of all available features. Linear kernel was used for classification in every experiment. We used 4-fold crossvalidation on the training set to find the best parameter C for the classification model. The learned model's performance was then tested on the testing data.

The results of the classification are shown as confusion matrices in Table I. With Layer 0 we denoted the experiment without sHoP processing, but using the raw scans instead. In this case simply the number of scan points were counted in each region. We could say that this is analogous to having only a single part in the library. As mentioned earlier, on Layer 1 there is a fixed number of orientation specific parts. Our learning algorithm learned 12 parts on Layer 2. On Layer 3 a total of 13046 different parts were observed, from which only a few most frequent were used for the representation. We tested the libraries with 80, 200 and 500 parts on that layer. The confusion matrix is shown only for the case with 200 parts, because this one yielded the best results. We have also tested with Layer 4 of the library. Parts on that layer are relatively large and therefore cover the image quite poorly. A lot of information is lost in this way, which causes reduced classification performance. We therefore focus only on Layers 1, 2 and 3 of the hierarchy.

The accuracies of the classification, computed as a percentage of correctly classified examples, averaged over all trials of the experiment, are shown with their corresponding standard deviations in Fig. 7 (a). We have also performed a series of t-tests, with which we compared each two of the obtained results. We tested the null hypothesis, at $\alpha = 0.01$ significance level, that the two distributions of calculated accuracies have equal means. The results are shown as the above diagonal elements in Fig. 7 (c). The '+' signs denote a failure to reject the null hypothesis, which implies that the two observed accuracies are statistically equivalent. On

TABLE I

CONFUSION MATRICES OBTAINED WITH SVM CLASSIFICATION USING 4 CLASSES AND DIFFERENT LAYERS OF THE HIERARCHY. THERE WERE 200 PARTS USED ON LAYER 3. LABELS AT THE BORDERS OF THE TABLE DENOTE: LR - LIVING ROOM, CR - CORRIDOR, BA - BATHROOM, BE - BEDROOM. THE ENTRIES ARE IN PERCENTS. ROWS CORRESPOND TO GROUND-TRUTH, AND COLUMNS CORRESPOND TO PREDICTED CLASSES.

		Lr	Cr	Ba	Be
Layer 0	Lr	**76.96**	3.50	0.02	19.52
	Cr	6.80	**82.70**	0.20	10.30
	Ba	0.14	3.21	**87.50**	9.14
	Be	10.92	0.04	15.18	**73.86**
Layer 1	Lr	**79.68**	0.22	0.04	20.06
	Cr	0	**100.00**	0	0
	Ba	0.19	1.30	**83.50**	15.01
	Be	15.44	0	10.32	**74.24**
Layer 2	Lr	**75.16**	0	3.46	21.38
	Cr	0	**82.50**	0	17.50
	Ba	0.04	0.51	**92.19**	7.26
	Be	15.68	0.06	12.14	**72.12**
Layer 3	Lr	**82.74**	0	0.08	17.18
	Cr	0	**91.60**	0	8.40
	Ba	0.20	0.29	**92.96**	6.56
	Be	13.32	0.22	16.24	**70.22**

the contrary, the '-' signs denote the rejection of the null hypothesis at $\alpha = 0.01$ significance level, which means that there is a 99% probability that the two accuracies truly represent a different result. The results suggest that Layers 0, 1 and 2 show equal classification performance, which means that parts on the lowest layers of the hierarchy are to small to possess any significant information about the local structure of space. On the other hand, Layer 3 provides statistically significantly better results. In particular, the accuracy of 80.52% on Layer 0 is increased to 83.73%. The results of t-tests also suggest that the number of parts on Layer 3 is of no significant importance.

To evaluate the results from a different perspective we analysed another measure of classification performance. We call this measure the *mean success rate*, which is computed as a mean of the diagonal entries of the confusion matrix, averaged over all trials of the experiment. Mean values with standard deviations are displayed in Fig. 7 (b), while the corresponding t-test results are shown as the below diagonal entries in Fig. 7 (c). This view of the results confirms that Layer 0 and 2 perform equally well, but Layer 1 stands out, showing significantly better performance. The reason for this is the fact that on Layer 1 we actually perform the classification based on the orientation of an edge. A specific orientation, as expected, very well characterizes corridors, but not also the other rooms, which can be confirmed by examining the confusion matrices. On Layer 2 this orientation information is not incorporated any more, and therefore the performance drops back. This measure also clarifies that Layer 2 and Layer 3 with low number of parts show similar performance. A greater number of parts on Layer 3 significantly improves the classification performance, from 80.26% on Layer 0 to 84.38%. The results also suggest that continuously increasing the number of parts does not increase the mean success rate. Therefore, using only the most frequent parts, and also not too many of them, seems a reasonable thing to do.

Overall, bathrooms and corridors are the ones that are classified the most accurately, followed by living rooms and then bedrooms. We believe that what makes these classifications possible at all are general room shapes and properties. At least in homes from which our data was obtained, the characteristics of each class could be the following: Bathrooms are small rooms with square or slightly rectangular shape, bedrooms have a distinctive property of small passages between the bed and nearby walls, living rooms are usually larger and stuffed with all sorts of furniture and objects, while corridors represent long and narrow spaces.

B. Comparison With State-of-the-art

We tested our approach to room classification on two datasets from [11]. The first one was an office environment in Building 79 at the University of Freiburg, while the second one is Building 101 at the same University. Using the former one, [11] distinguished between three classes, which are corridor, room, and doorway, and managed to achieve the accuracy of 93.94 %. With the latter one, they distinguished between four classes, where hallway was added to previous ones. Their obtained accuracy in this case was 89.52 %.

In our experiments, first, all laser scans were transformed into images, and than HoC descriptors were created corresponding to those images. The descriptors were functionally the same as in the experiments with our database. Our laser range finder has a maximum range of less than 6 meters, whereas the laser used in [11] has a range of 80 meters. An extra circle was used in the formation of the descriptor in these experiments, corresponding to a circle around the robot with radius of 6 meters, which resulted in 6 additional regions of the descriptor. We used Layer 3 of the hierarchy with 200 parts.

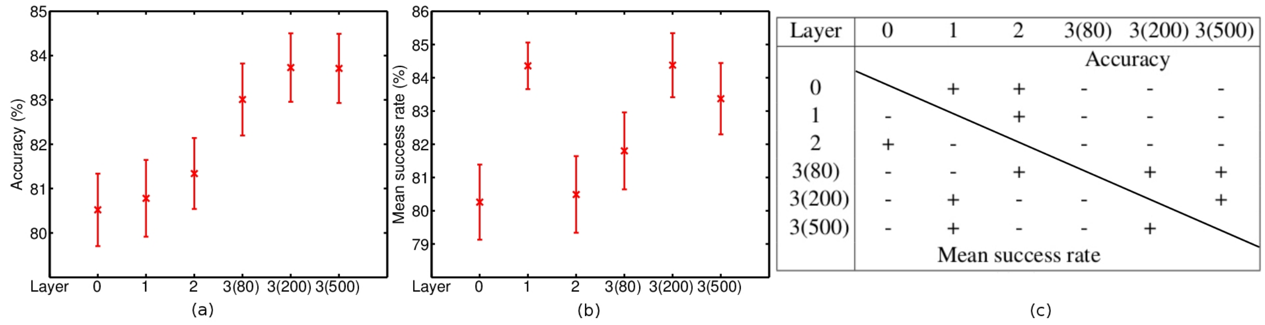
The experiments were analogous to the ones in [11]. Part of the environment specified for training was used to train the SVM, while the part specified for testing was used for testing. We did not perform any averaging of descriptors as we did in the experiments of previous subsection. Like [11], we performed the classification based on a single scan and using the same class definitions.

In the Building 79 experiment we obtained an accuracy of 98.32 %, while in the Building 101 experiment our result was 90.39 %. The results are summarized in Table II. We conclude that our approach delivers comparable results to the state-of-the-art [11].

TABLE II

ACCURACY OF THE CLASSIFICATION IN THE EXPERIMENTS WITH TWO PUBLICLY AVAILABLE DATASETS.

	[11]	Our approach
Building 79	93.94	**98.32**
Building 101	89.52	**90.39**



Layer	0	1	2	3(80)	3(200)	3(500)
	Accuracy					
0		+	+	-	-	-
1	-		+	-	-	-
2	+	-		-	-	-
3(80)	-	-	+		+	+
3(200)	-	+	-	-		+
3(500)	-	+	-	-	+	
	Mean success rate					

Fig. 7. (a) Classification accuracy, computed as a percentage of correctly classified examples, averaged over all trials of the experiment. (b) Mean success rate, computed as a mean of the diagonal entries of the confusion matrix, averaged over all trials of the experiment. In both cases, average values are shown with $\times$ symbol, while the vertical lines correspond to one standard deviation. (c) Results of t-tests of the null hypothesis that the two results are statistically equivalent, at $\alpha = 0.01$ significance level. Above diagonal elements correspond to accuracy, while the below diagonal elements correspond to mean success rate. The '+' sign denotes failure to reject the null hypothesis, and '-' denotes successful rejection. Numbers in the brackets at Layer 3 annotations denote the number of parts used on that layer.

V. CONCLUSION AND FUTURE WORK

We have presented a new hierarchical representation of space, which is learned using the sHoP algorithm. The hierarchy is constructed from parts that are rotationally invariant and statistically significant. To our knowledge this is the first application of a hierarchical compositional model to a spatial representation on the lowest level. We have also presented a low-level image descriptor, which was used for the validation of our spatial model. The performance of the lower layers of the hierarchy was evaluated through a room classification problem with four classes. We have obtained good classification results on a demanding dataset, which indicate that the proposed hierarchy of parts is suitable for representing space. We have also shown that the performance of our approach to room classification is comparable to the performance of the state-of-the-art approaches.

There are several research venues we intend to pursue in our future work. From current research we have determined that three is the maximum number of layers that are reasonable to be learned from separate images. In the future we plan to combine the images, obtained in a certain room, to a single map of that room. These maps will be built using a feature based SLAM algorithm, where Layer 3 parts will be used as input features. Therefore, the output maps will be represented as lists of Layer 3 parts at certain locations. These will provide us with a more complete view of the environment, which in contrast cannot be obtained using a single laser scan. The maps will serve as a basis for learning higher layers of the hierarchy. We plan to learn one or even more category-independent layers, and than switch to category-specific layers, which will be learned with minimal supervision according to room type. We expect that room recognition performance will be increased with the abstraction introduced by higher layers. Moreover, we also expect that we will be able to perform room classification without the use of HoC and SVM, but only using our category specific layers of the hierarchy.

The learned hierarchy could then be used as a prior knowledge for a service robot, providing the basis for recognition of new, never before seen, environments, and simultaneously ensure good scalability of the model. Sensors providing richer information could also be used afterwards to refine the belief about robot's surroundings.

REFERENCES

[1] A. Pronobis, P. Jensfelt: Understanding the real world: Combining objects, appearance, geometry and topology for semantic mapping, Technical Report TRITA-CSC-CV 2011:1 CVAP319, Kungliga Tekniska Högskolan, CVAP/CAS, May 2011
[2] B. Kuipers: The Spatial Semantic Hierarchy, Artificial intelligence, vol. 119, pp. 191-233, 2000
[3] H. Moravec: Sensor fusion in certainty grids for mobile robots. AI Magazine, vol. 9, no. 2, pp. 61-74, 1988
[4] S. Vasudevan, S. Gächter, A. Harati, R. Siegwart: A Hierarchical Concept Oriented Representation for Spatial Cognition in Mobile Robots. In: 50 years of artificial intelligence, Springer-Verlag, 2007
[5] S. Fidler, M. Boben, A. Leonardis: Evaluating multi-class learning strategies in a hierarchical framework for object detection. NIPS, 2009
[6] O. M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, W. Burgard: Supervised semantic labeling of places using information extracted from sensor data, Robotics and Autonomous Systems, Vol. 55, no. 5, pp. 391-402. May, 2007
[7] J. Wu, H.I. Christensen, J. M. Rehg: Visual Place Categorization: Problem, dataset, and algorithm, IROS, pp.4763-4770, 10.-15. October 2009, USA
[8] B. Ayers, M. Boutell, Home Interior Classification using SIFT Keypoint Histograms, CVPR, pp. 1-6, 17.-22. June 2007, USA
[9] Z. Zivkovic, O. Booij, B. Kröse: From images to rooms, Robotic and Autonomous Systems, vol.55, no.5, pp. 411-418, May 2007
[10] A. Swadzba, S. Wachsmuth: Categorizing Perceptions of Indoor Rooms Using 3D Features, Lecture Notes in Computer Science: Structural, Syntactic, and Statistical Pattern Recognition, pp. 734-744, 4. December 2008, USA
[11] O.M. Mozos, C. Stachniss, W. Burgard: Supervised Learning of Places from Range Data using AdaBoost, ICRA, pp. 1730- 1735, 18.-22. April 2005, Spain
[12] S. Friedman, H. Pasula, D. Fox: Voronoi random fields: extracting the topological structure of indoor environments via place labeling, IJCAI, pp. 2109-2114, 6.-12. January 2007, India
[13] S. Fidler, M. Boben, A. Leonardis: Learning Hierarchical Compositional Representations of Object Structure. In: Object Categorization: Computer and Human Vision Perspectives, CUP, 2009.
[14] S. Fidler and M. Boben and A. Leonardis: Similarity-based cross-layered hierarchical representation for object categorization, CVPR, pp. 1-8, 24.-26. June 2008, Alaska, USA
[15] C. C. Chang, C. J. Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

Associating and merging multi-modal and multi-agent information in a cognitive system
Technical report TR-LUVSS-02/2012

Alen Vrečko, Miroslav Janíček, Aleš Leonardis, Danijel Skočaj

Faculty of Computer and Information Science,
University of Ljubljana, Slovenia
German Research Centre for Artificial Intelligence (DFKI GmbH),
Saarbrücken, Germany

Abstract. A critical ability for every cognitive system operating in a complex environment is the ability to combine several representations of the same physical reality into a single shared representation. Such combined, a-modal representations are then ready to be used by higher level cognitive processes, like motivation and planning. In this work we describe a cognitive layer where multi-modal and multi-agent information is associated and merged into a-modal representations. In addition we describe the application of cross-modal binding principles to a specific problem of reference resolution.

1 Introduction

Two critical properties of any cognitive system operating in a complex environment are (i) the ability to sense, perceive and process complex information about physical reality and (ii) the ability to use this information to plan, manage and execute complex actions in such an environment. The complexity of the physical reality implies the ability to collect information from different sources, i. e. different sensor types and possibly also different agents (other than the cognitive system itself). This means that at least on lower levels the information is inherently multi-modal and multi-agent. On the other hand the higher cognition (e. g. motivation, planning,..) predominantly assumes a-modal information, only. Hence, an intermediate cognitive layer capable to relate and merge multi-modal and multi-agent information is needed to close the semantic gap that divides the lower and the higher cognition.

In this work we describe the intermediate cognitive layer of the George prototype cognitive system — the *belief layer*. The George scenario assumes a robot capable of making situated dialogue with a human tutor about the objects on a table. The robot is thus able to observe, track and recognize the objects on the table and through the dialogue with the tutor improve its knowledge about the object's properties (cross-modal learning). Such a scenario obviously relies on the ability of the robot to first associate and later merge multi-agent information. The resulting representations can be then used by the higher cognitive processes like motivation and planning.

Many of the past attempts at associating multi-modal or multi-agent information within cognitive systems were restricted to relating linguistic information to lower level perceptual information. Roy et al. tried to ground the linguistic descriptions of objects and actions in visual and sound perceptions and to generate descriptions of previously unseen scenes based on the previously accumulated knowledge [9, 10]. This is essentially a *symbol grounding problem* first defined by Harnad [4]. Chella et al. proposed a three-layered cognitive architecture around the visual system with the middle, *conceptual layer* bridging the gap between linguistic and sub-symbolic (visual) layers [1]. Related problems were also often addressed by Steels [11].

Jacobsson et al. approached the binding problem in a more general way [6] [5] developing a cross-modal binding system that could form associations between multiple modalities and could be part of a wider cognitive architecture. The cross-modal knowledge was represented as a set of binary functions comparing binding attributes in pair-wise fashion. A cognitive architecture using this system for linguistic reference resolution was presented in [13]. This system was capable of learning visual concepts in interaction with a human tutor. A probabilistic binding system was developed within the same group that encoded cross-modal knowledge into a Bayesian graphical model [14]. In [7] a framework for constructing high-level cognitive representations of the environment, called beliefs, was presented. Markov logic was used as the main framework for various types of inference over beliefs, including perceptual grouping, which comes very close to the definition of binding in [12]. The systems described in [6] – [7] assumed static cross-modal knowledge, while the binding model in [12] also included cross-modal learning as a means to gradually improve the binding ability of the system.

The process of determining the denotation of a referring expression is called *reference resolution* (Section 2). We base our implementation of this process on the general method of *cross-modal binding in Markov logic networks* described in [12]. The process of merging multi-agent (or multi-modal) information into a single a-modal representation is called *information fusion.* Both processes operate on the *belief layer*, described in Section 3. As we will see in Section 3.2, the difference between robot's own perceptions and information attributed to another agent — both encoded in beliefs — can be directly exploited for *implicit learning.*

2 Binding and Reference resolution

One of the most important abilities of any cognitive system operating in a real world environment is the ability to relate and merge information from different modalities. The process of combining two or more modal representations (grounded in different sensory inputs) of the same physical entity into a single multi-modal representation is called *binding.* In [12] we presented a model of cross-modal binding and learning system formulated in *Markov logic networks (MLN).* MLN [8, 2, 3] combine first-order logic and probabilistic graphical models in a single representation. An MLN knowledge base consists of a set of first-order logic formulae (rules) with a weight attached:

$$weight \quad first\text{-}order\_logic\_formula.$$

The weight is a real number, which determines how strong a constraint each rule is: the higher the weight — the less likely that rule is violated. MLN is used to encode the cross-modal knowledge, which is the base for the binding inference.

In general the MLN binding is applied to the belief cognitive layer, where the various beliefs represent perceived and assumed facts that are used to instantiate the rules from the cross-modal knowledge base to the *Markov network* graphical model. If MLN knowledge represents the general rules encoding relations between concepts (e.g. object properties as color, shape,...), the graphical model encodes the relations between concrete instances (objects) that are currently perceived by the system. A successful inference results in a shared multi-modal representation of a physical entity, also called *binding union*. Successful binding unions can be used as learning samples to improve cross-modal knowledge, i. e. *cross-modal learning*.

In George scenario the binding principles are used for *reference resolution*. Reference resolution is a process akin to binding that tries to associate multi-agent information. In our case the robot uses reference resolution to relate information attributed to a human tutor to its own perceptions, hence it is critical for its ability to make situated dialogue with the human.

MLN are implemented as special components that process information stored in beliefs. A *MLN engine component* maintains a Markov network graphical model, which makes continuous online inference (MCMC sampling) and can continuously adapt to the changes in the beliefs. MLN engines can also combine the information encoded in the current graphical model with the external information about the correct inference outcome to perform on-line weight learning. *MLN client components* filter the information stored in beliefs or other data structures and feed it to MLN engines. They can also read and process inference results and trigger weight learning in MLN engines.

The implementation of reference resolution features a single MLN engine and two MLN clients. One MLN client (*belief filter*) continuously filters the information in beliefs and forwards it to the MLN engine as evidence about the perceived entities. The other MLN client (*restrictor*) acts on request; triggered by the dialogue subsystem (when it recognizes a referring expression in the tutor's utterance) it first feeds the MLN engine with the referring (restrictive) information, then reads and forwards the inference result back to the dialogue subsystem and finally withdraws the referring information.

The inference result, which is a probability distribution over perceived entities represented as beliefs, is used by the dialogue subsystem to determine the interpretation of the tutor's utterance. This eventually results in additional beliefs related to the beliefs grounded in robot perceptions (see Section 3). As a result of successful reference resolution, the restrictor can also trigger weight learning in MLN engine. A successful reference resolution usually means that the resulting probability distribution favors with a suitable degree of reliability the reference resolution to one of the existing beliefs. In this case the restrictor first feeds the 'winning' resolution to the MLN engine as evidence and then triggers the learning. Afterwards, it withdraws both pieces of evidence, the referring information and the 'winning' resolution, from the MLN engine.

2.1 An example of reference resolution

The following is an example of reference resolution performed in an MLN engine component. For more detail about MLN, MLN Binding and its integration in a cognitive system please refer to [12].

We assume a small MLN reference resolution knowledge database that encodes associations between two visual color models (denoted as 'Color1' and 'Color2') and two linguistic color descriptions ('Red' and 'Blue'):

$$\begin{aligned} 2.5\ &\mathrm{percColor}(b, Color1) \wedge \mathrm{restrict}(Red) \Rightarrow \mathrm{resolveTo}(b) \\ -1.9\ &\mathrm{percColor}(b, Color1) \wedge \mathrm{restrict}(Blue) \Rightarrow \mathrm{resolveTo}(b) \\ -1.3\ &\mathrm{percColor}(b, Color2) \wedge \mathrm{restrict}(Red) \Rightarrow \mathrm{resolveTo}(b) \\ 2.0\ &\mathrm{percColor}(b, Color2) \wedge \mathrm{restrict}(Blue) \Rightarrow \mathrm{resolveTo}(b) \end{aligned}$$

The predicate percColor(b, Color1) denotes that the object represented by the belief b was perceived to be of modal color representation $Color1$ by the visual subsystem, the predicate restrict(Red) denotes the Red as restriction (referring information, see Section 3.1) given by the tutor, while the predicate resolveTo(b) denotes the reference resolution to the belief b. Variables (b) begin with a lowercase character, while constants (Color1, Red, etc.) begin with an uppercase character. We can see that the rules in the knowledge database instantiate concept (in our case the colors), but encode beliefs about objects as variables. The predicate resolveTo(b) is also the object of the MLN engine query. The inference can, for example, result in the following probability distribution:

$$\begin{aligned} &0.2\ \mathrm{resolveTo}(B1) \\ &0.1\ \mathrm{resolveTo}(B2) \\ &0.7\ \mathrm{resolveTo}(B3) \end{aligned}$$

In this case the real numbers denote the probabilities. In addition to the knowledge database the reference resolution system includes the following set of hard rules that regulate the inference process:

$$\mathrm{belief}(b) \wedge \mathrm{belief}(b') \wedge \mathrm{resolveTo}(b) \wedge \mathrm{resolveTo}(b) \Rightarrow b = b'. \tag{1}$$

$$\mathrm{resolveTo}(b) \Rightarrow \mathrm{belief}(b). \tag{2}$$

$$\mathrm{resolveTo}(b) \Rightarrow \exists f:\ \mathrm{restrict}(f). \tag{3}$$

The hard rules are rules with an infinite weight that can never be broken. The predicate belief(b) denotes the existence of belief b. The rule (1) restricts the reference resolution to exactly one belief, the rule (2) restricts the reference resolution to an existing belief and finally the rule (3) makes reference resolution possible when referring information exists, only.

Let's suppose the system perceives two objects on the desktop, one red (perceived as $Color1$) and one blue (perceived as $Color2$). The belief filter feeds the MLN engine with the following evidence:

$$\text{belief}(B1) \wedge \text{belief}(B2) \wedge \text{percColor}(B1, Color1) \wedge \text{percColor}(B2, Color2)$$

Based on this information the MLN engine builds the Markov network graphical model. First it instantiates the rules with both beliefs:

$$\begin{aligned}
2.5\ &\text{percColor}(B1, Color1) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B1)\\
-1.9\ &\text{percColor}(B1, Color1) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
-1.3\ &\text{percColor}B1, Color2) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B1)\\
2.0\ &\text{percColor}(B1, Color2) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
2.5\ &\text{percColor}(B2, Color1) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B2)\\
-1.9\ &\text{percColor}(B2, Color1) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
-1.3\ &\text{percColor}(B2, Color2) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B2)\\
2.0\ &\text{percColor}(B2, Color2) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
\infty\ &\neg\text{resolveTo}(B1) \vee \neg\text{resolveTo}(B2)\\
\infty\ &\neg\text{resolveTo}(B1) \wedge \neg\text{resolveTo}(B2)
\end{aligned}$$

Then it applies the evidence to the instantiated rules:

$$\begin{aligned}
2.5\ &\text{restrict}(Red) \Rightarrow \text{resolveTo}(B1)\\
-1.9\ &\text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
-1.3\ &\text{percColor}B1, Color2) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B1)\\
2.0\ &\text{percColor}(B1, Color2) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
2.5\ &\text{percColor}(B2, Color1) \wedge \text{restrict}(Red) \Rightarrow \text{resolveTo}(B2)\\
-1.9\ &\text{percColor}(B2, Color1) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
-1.3\ &\text{restrict}(Red) \Rightarrow \text{resolveTo}(B2)\\
2.0\ &\text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
\infty\ &\neg\text{resolveTo}(B1) \vee \neg\text{resolveTo}(B2)\\
\infty\ &\neg\text{resolveTo}(B1) \wedge \neg\text{resolveTo}(B2)
\end{aligned} \tag{4}$$

The instantiated rules above represents a Markov graphical model, where each fully instantiated predicate represents a sampling variable (atom). MLN engine performs a continuous inference that in the present case (because of the last instantiated rule, derived from the hard rule 3) does not yield any positive resolution.

Let's suppose that the tutor refers to a red object in his utterance. By the request of the dialogue subsystem, the restrictor component feeds the MLN engine with the predicate restrict(Red). This new piece of information modifies the graphical model as follows:

$$\begin{aligned}
2.5\ &\text{resolveTo}(B1)\\
-1.9\ &\text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
-1.3\ &\text{percColor}B1, Color2) \Rightarrow \text{resolveTo}(B1)\\
2.0\ &\text{percColor}(B1, Color2) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B1)\\
2.5\ &\text{percColor}(B2, Color1) \Rightarrow \text{resolveTo}(B2)\\
-1.9\ &\text{percColor}(B2, Color1) \wedge \text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
-1.3\ &\text{resolveTo}(B2)\\
2.0\ &\text{restrict}(Blue) \Rightarrow \text{resolveTo}(B2)\\
\infty\ &\neg\text{resolveTo}(B1) \vee \neg\text{resolveTo}(B2)
\end{aligned}$$

As we can see the referent information also removes the hard rule preventing any positive reference resolution. The inference result is now clear, the resulting probability distribution reliably indicates the belief $B1$ as the referent. The restrictor forwards this information to the dialogue subsystem and removes the referent information, which returns the graphical model to the state (4).

Saliency can be a very useful addition to the situated human-robot dialogue. An object on the desktop can become salient as a result of nonverbal communication, e. g. the robot or human pointing with his arm or directing his gaze to an object, or simply by being the only object on the desktop. The information about the saliency has to be part of the belief representing the object. The belief filter can feed this information to the MLN engine simply as the predicate salient($B1$). The human can then refer to that object with the word 'this', which the restrictor can represent with the predicate restrict($This$). The easiest way to implement this mechanism is to add another rule to the regulative set of hard rules:

$$\text{salient}(b) \wedge \text{restrict}(This) \Rightarrow \text{resolveTo}(b).$$

When the human refers to a salient object with the word 'this', the above (instantiated) hard rule simply overrules all the instantiated soft rules in the graphical model, resolving the reference to the salient object.

3 The belief layer

Beliefs form a cognitive layer where multi-modal and multi-agent information is associated and merged to a-modal representations. In general a belief can be regarded a high-level representation of an element of the physical reality, grounded in one or more sensory inputs, attributed to a specific agent or a combination of both. Our belief scheme distinguishes five distinct belief categories:

– *Private* beliefs reflect the robot perceptions of the environment based on its sensory input. Private beliefs are expressed in modal symbols and can form various associations with private beliefs stemming from other modalities or beliefs with other epistemic statuses (e. g. reference resolution).
– *Assumed* beliefs are used to establish cross-agent or cross-modal common ground. They are created from private beliefs by translating the modal symbols to the a-modal ones. Depending on complexity of the modal learners and their ability for autonomous unsupervised learning, this process can be as simple as one-to-one symbol mapping or much more complex (e. g. translating between two sets of symbols with overlapping meaning that consequently also modifies the original probability distribution). In cross-agent case the robot uses assumed beliefs to establish a common ground with another agent to facilitate communication. Thus the beliefs reflect the robot assumptions about the meaning of its perceived information for a particular agent (e. g. human). In cross-modal case the assumed beliefs establish a common ground between modalities. In both cases this process facilitates cross-belief information fusion in later stages.
– *Attributed* beliefs contain information that robot attributes to another agent (e. g. human). This kind of beliefs are the direct consequence of some kind of communication with another agent. The robot is in principle able to analyze and understand the information in such beliefs, but does not necessarily agree with it (especially, if it doesn't match the robot's own perception of the same reality).
– *Verified* beliefs are created from attributed beliefs. They basically contain the acknowledged information from the attributed beliefs. Acknowledgment (verification) does not necessarily mean that the agent's information in the belief is consistent with the robot's perception; it just means that that information was adequately processed by the robot and is now ready to be used in higher level cognition (e. g. in communication with the agent that issued it). After a successful reference resolution the restrictive information is stored in verified shared beliefs, while the asserted information is in attributed belief.
– *Merged* beliefs combine information from verified and assumed beliefs and represent the final a-modal situated knowledge, ready to be used by the higher level cognitive processes (e.g. motivation, planning). They contain as reliable information as possible and as much information as available. This process is also called *information fusion*. Information can be merged in different ways. E. g. the system can completely trust a certain agent (typically a tutor) so that the merged belief contains all information from

the verified belief and only uses the assumed belief to fill the information gaps left by the verified belief. A more complex solution for the information fusion involves merging probability distributions over feature values.

The private beliefs are created by mediator components using the information from the modal subsystems. The attributed and verified beliefs are created as results of successful resolution of another agent's reference. The changes in perception are propagated in real-time through the belief structure from private beliefs to the merged ones. In similar manner the progress in dialogue and dialogue processing (certain events in other subsystems can be treated as acknowledgments for the attributed information) are reflected in changes in attributed and verified beliefs. This means that the process of information fusion (belief merging) is repeated each time new information is propagated to the assumed belief or new attributed information is verified.

3.1 An example of information flow in beliefs

Figures 1, 2 and 3 illustrate how the belief representations of an object change with the activity of the system. The objects are described in terms of colors, shapes and affordances. The goal of the system is to use the new infomation provided by the human tutor for visual learning.

Figure 1 represents the belief state after the robot has processed the visual information about a physical object on the desktop. It reflect the robot's own perception of the object. We can see that the internal (modal) visual symbols (and the object's affordance, which is based on its shape) are translated to a-modal symbols (in our case the dialogue also operates with a-modal symbols). The translation can be performed by an MLN engine component. As described in Section 3, the translation can be more than just a simple symbol mapping; it can also have to re-calculate the probability distributions of the translated symbols. The merging process in this case just forwards the information to the merged belief.

Figure 2 represents the belief state after the system has processed a tutor's statement about the object ("The compact object is blue."). In this sentence the 'compact' represents the *referring* or *restrictive* information, which is used to determine (restrict) the entity in question. With the *assertive information* in the sentence ('blue') the human expressed a new quality about the refered entity (perhaps not known to the robot). The assertive information doesn't completely agree with the robot's perceptions. Fortunately, the restrictive part of the statement is consistent with the analogue information in the current merged belief, which guarantees the success of the reference resolution. We can see that the information attributed to the tutor is initially split in two parts: the restrictive part is already considered verified (since the reference resolution was successful) and goes to the verified belief, while the assertive part goes to the attributed belief since it is not yet clear, whether it represents a common ground between the robot's and tutor's perceptions (we can see that in our particular example

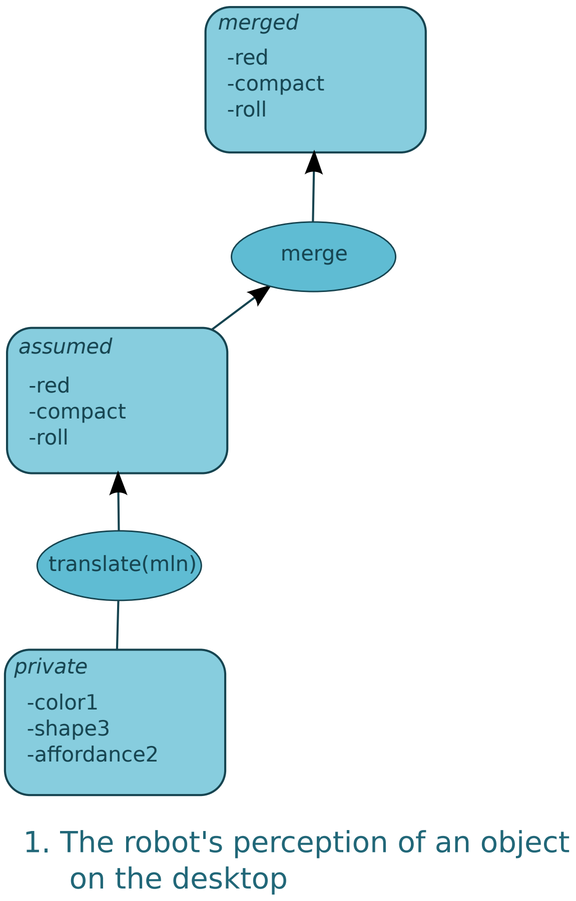


Fig. 1.

the doubt is justified). We can see that the merging process confirms the shape information in the merged belief.

Figure 3 illustrates what happens after a certain event in other parts of the system (in our case the visual learning) triggers the acknowledgement (of a portion) of the asserted information. The acknowledged attributed information is propagated to the verified belief and then merged. In our case the color property is replaced with its attributed version. The merged belief therefore contains one piece of information that is purely perceptual (roll); the information about the shape (compact) is shared by both, robot perception and human description; the color information (blue) is not shared, but since it is provided by the human it is treated as more reliable.

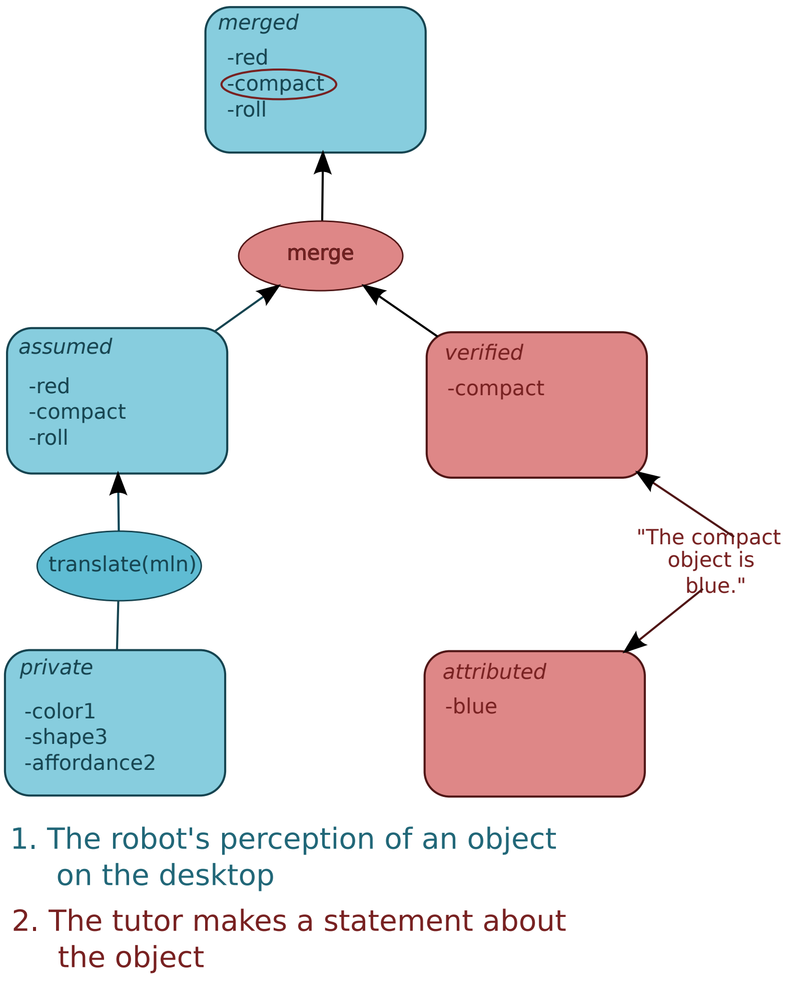


Fig. 2.

3.2 Implicit learning

The main purpose of the George scenario is to demonstrate several learning paradigms like tutor driven, tutor assisted, robot driven, autonomous, etc. All of these paradigms have the goal to obtain and use tutor's information about visible objects for learning basic visual concepts. In most cases the tutor provides this information explicitly, as we saw in the example in Section 3.1. In this case the learning act is executed as a deliberate action issued by the planner, hence in a goal-driven fashion. We can categorize such learning mechanism as *explicit learning*.

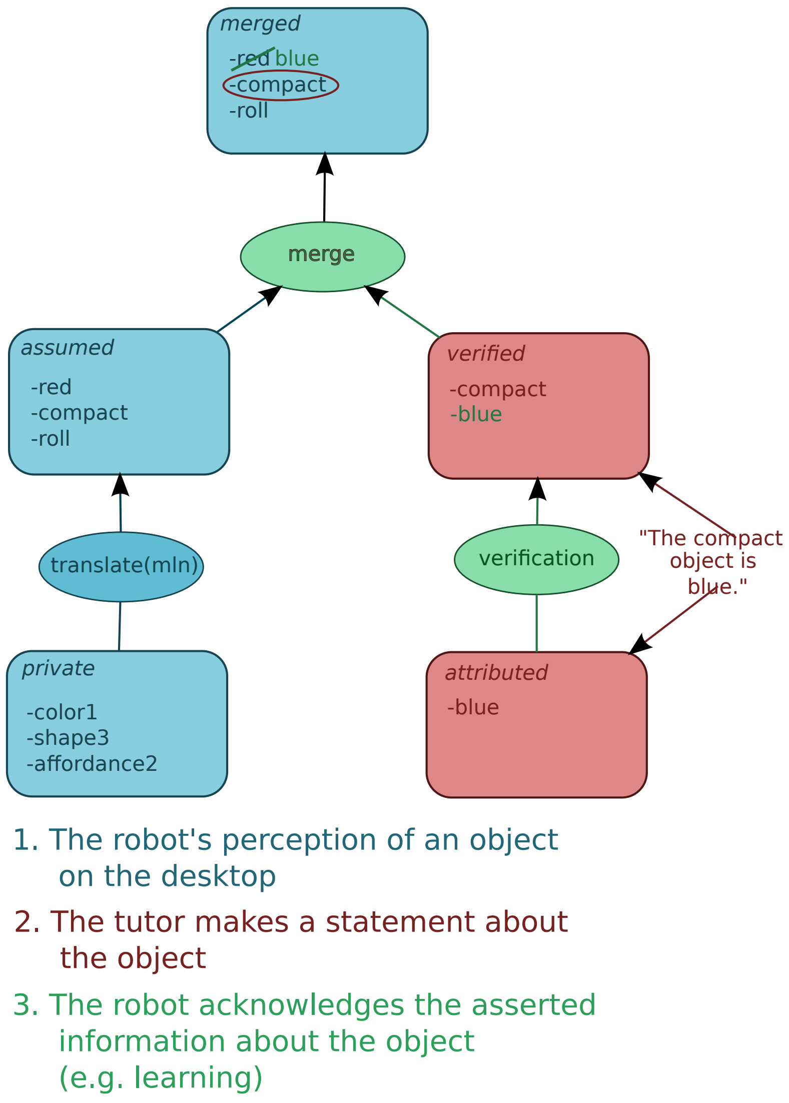

merged
-~~red~~ blue
-compact
-roll
merge
assumed
-red
-compact
-roll
verified
-compact
-blue
translate(mln)
verification
"The compact object is blue."
private
-color1
-shape3
-affordance2
attributed
-blue
1. The robot's perception of an object on the desktop
2. The tutor makes a statement about the object
3. The robot acknowledges the asserted information about the object (e.g. learning)

Fig. 3.

In contrast, *implicit learning* completely bypasses planing and motivation and occurs in a data-driven fashion (the difference between implicit and explicit learning is also explained in [13]). It exploits the difference in information between assumed and merged beliefs to update modal visual concepts. Depending on the information that was merged and the type of process that performs information merging (see Section 3) the difference can be (i) in the property confidence and (ii) in the property quality (e. g. as depicted in Figure 3). In the former case the system has to have a difference threshold that triggers learning. In the latter case the system can perform both, learning of the right concept in the merged belief and unlearning of the wrong concept in the assumed belief. Of course before the learning action the property information in the merged belief has to be translated back to modal symbols.

An important question when implementing implicit learning is when to trigger it. It would not be advisable to simply trigger the implicit learning after each merging, since this could result in learning the same information several times. A better strategy is to compare the new merged belief with the old one and react only when there is a change in the quality of merged information or if confidence of the new information raises the confidence difference above the threshold.

Another problem concerning implicit learning occurs when the implicit learning is combined with the explicit learning. After the assertive information is used for learning, it is verified and consequently merged to the merged belief. This can trigger the implicit learning, which once more means that the same information is used for learning twice. We can avoid this problem by simply restrict implicit learning to the restrictive information only (as is also the case in [13]). This means that the implicit learning is triggered after the first merging of the verified information, only. When used as a supplementary learning mechanism in combination with the explicit learning, it is important to adequately tune the effects of both learning mechanisms (the effect of implicit learning is usually less pronounced as in general happens more often and not as a deliberate action).

4 Conclusion

In this work we described the *belief cognitive layer*, which is where multi-modal and multi-agent information is associated and merged. As an important association mechanism in the belief layer we also described and exemplified the process of *reference resolution*, which relates robot's own perceptions of a physical entity to the human description of the same entity. *Implicit learning* is a learning mechanism that exploits the difference between merged and assumed information in beliefs. It can be used together with the explicit learning as a supplementary learning mechanism.

5 Acknowledgment

This work was supported by the EC FP7 IST project CogX-215181.

References

1. A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artif. Intell.*, 89(1-2):73–111, 1997.

2. Pedro Domingos. Toward knowledge-rich data mining. *Data Min. Knowl. Discov.*, 15:21–28, August 2007.
3. Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proc. of the ICML-2004 workshop on statistical relational learning and Iits connections to other fields*, pages 49–54, 2004.
4. S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.
5. H. Jacobsson, N. Hawes, G-J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *Proc. of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, Amsterdam, March 2008.
6. H. Jacobsson, N. Hawes, D. Skočaj, and G-J. Kruijff. Interactive learning and cross-modal binding - a combined approach. In *Symposium on Language and Robots*, Aveiro, Portugal, 2007.
7. P. Lison, C. Ehrler, and G.-J. Kruijff. Belief modelling for situation awareness in human-robot interaction. In *Proceedings of the 19th IEEE International Symposium in Robot and Human Interactive Communication.* IEEE, 2010.
8. M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.
9. D. Roy. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3-4):353–385, 2002.
10. D. Roy. Grounding words in perception and action: computational insights. *TRENDS in Cognitive Sciences*, 9(8):389–396, 2005.
11. L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings.* Laboratorium, Antwerpen, 1999.
12. A. Vrečko, A. Leonardis, and D. Skočaj. Modeling binding and cross-modal learning in markov logic networks. *Neurocomputing*, (0):In press, 2012.
13. A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *Proc. of the 2009 IEEE/RSJ Int. Conf. on Intelligent RObots and Systems*, pages 3140–3147, St. Louis, Oct. 2009.
14. J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G-J. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, A. Vrečko, H. Zender, and M. Zillich. Self-understanding & self-extension: A systems and representational approach, 2010. Accepted for publication.